UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE COMPUTACIÓN

TRABAJO DE GRADUACIÓN PARA OPTAR AL GRADO DE
INGENIERO EN CIENCIAS DE LA COMPUTACIÓN

APLICACIÓN DE METODOLOGÍAS DE DESARROLLO ÁGIL Y HERRAMIENTAS DE DESARROLLO OPEN SOURCE, PARA LA CREACIÓN DEL SISTEMA INFORMÁTICO DE GESTION DE LA PRODUCCIÓN EN EL DEPARTAMENTO DE ÓRTESIS Y PRÓTESIS DE LA UNIVERSIDAD DON BOSCO

PRESENTADO POR

RAFAEL EDUARDO PANIAGUA MORENO

ROBERTO CARLOS CALLES MONTERROSA

JUNIO 2005
SAN SALVADOR, EL SALVADOR CENTRO AMERICA

UNIVERSIDAD DON BOSCO

RECTOR
ING. FEDERICO HUGUET RIVERA

SECRETARIO GENERAL
LIC. MARIO RAFAEL OLMOS

DECANO DE LA FACULTAD DE INGENIERÍA
ING. ERNESTO GODOFREDO GÍRON

UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA

COMITÉ EVALUADOR DEL TRABAJO DE GRADUACIÓN

ING. MANUEL ORELLANA

Asesor

ING. ARNOLDO INOCENCIO RIVAS MOLINA

Jurado

ING. CARLOS ALBERTO REYES QUINTERO

Jurado

ING. HEINZ TREBBIN

Jurado

INDICE

CAPÍTULO III

RECURSOS METODOLOGICOS Y HERRAMIENTAS UTILIZADAS

CAPÍTULO IV

PROCESO DE DESARROLLO

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

INTRODUCCION

Los cambios tecnológicos han permitido el acceso a la información, facilitando la adquisición de conocimiento, y haciendo cada día más pequeñas las distancias, Internet es el mejor ejemplo de esto y la mayoría de empresas desea formar parte de este fenómeno global. Estos cambios también afectan la manera en que se desarrolla el software, no basta el clásico ciclo de vida de sistemas , ya que los requerimientos y los cambios de estos, ocurren con mayor frecuencia y rapidez durante el proceso de desarrollo; las empresas buscan la manera de satisfacer a sus clientes, y los programadores se ven abrumados por los tiempos de entrega cada vez mas cortos, y en el peor de los casos a un proceso de prueba y error para obtener algún tipo de resultado satisfactorio.

En El Salvador el desarrollo de software ha sido el trabajo de un grupo pequeño de personas, o una persona en particular dentro de la empresa. Hasta hace algún tiempo esto era una práctica muy común, pero ante la necesidad de mejores productos de software que den soporte a las funciones de la empresa, se opta por la compra de soluciones elaboradas por consultoras multinacionales o pequeñas consultoras locales de software. El éxito de los proyectos depende del proceso de desarrollo que le de a los clientes el software esperado y permita a los desarrolladores cumplir con los plazos establecidos.

Existen muchas prácticas para el desarrollo de software y muchos procesos de desarrollo. Algunos de ellos llamados "pesados" debido al uso de una metodología rígida y a los requerimientos de documentación del proceso. Otras reciben el

nombre de "ligeras" o Metodologías Ágiles pues se enfocan en el desarrollo del software y reduce la documentación innecesaria del proceso.

También existen un conjunto de herramientas de desarrollo: lenguajes, ambientes integrados de desarrollo, gestores de base de datos, compiladores, gestores de archivos y empaquetado, etc. Algunos de estos son vendidos bajo licencias de compañías propietarias que cobran por su utilización. Pero también existen herramientas conocidas como software Open Source, que se distribuyen bajo licencia de uso libre, y que pueden ser usados sin restricciones para el desarrollo de nuevo software. Estos presentan la ventaja de tener un costo relativamente mucho menor, ya que requieren solamente una conexión a Internet para ser descargados. Este proyecto de graduación analizará las características y la aplicación de un conjunto de metodologías conocido como Metodologías Ágiles. Con el propósito de ayudar en la elección de estas para el desarrollo de nuevos productos de software, y que puedan ser aplicadas por los programadores y equipos de desarrollo en proyectos donde los requerimientos no estén totalmente definidos y el numero de integrantes del equipo de desarrollo es pequeño.

Se incluye en este proyecto el uso de herramientas de desarrollo de software libre Open Source, para mostrar las ventajas de la utilización de estas, aprovechando que las metodologías a utilizar son independientes de una herramienta de software especifica y que las herramientas Open Source son de menor costo que el software propietario.

Para aplicar estas metodologías y software Open Source se ha seleccionado el desarrollo del sistema informático de gestión de la producción (SIPOP) para el Departamento de Órtesis y Prótesis de la Universidad Don Bosco (DOP).

CAPÍTULO I

PLANTEAMIENTO DEL PROBLEMA

1.1 SITUACION ACTUAL

Las pequeñas empresas de desarrollo se encuentran en desventaja frente a sus competidores que poseen mayores recursos para la adquisición de software de desarrollo y a su vez las pequeñas y medianas empresas que contratan programadores para desarrollo y mantenimiento de sus sistemas muchas veces deben adquirir software de costos elevados y no cuentan con otra alternativa.

De acuerdo a una consulta realizada en el mes de Abril de 2004[1] a 21 personas dedicadas profesionalmente a la programación y realización de sistemas informáticos determinamos que en El Salvador se desconoce de una alternativa adecuada y concreta que ayude a disminuir los costos y facilite el desarrollo de software (Horas de trabajo por proyecto, número de programadores, herramientas utilizadas) a las pequeñas empresas consultoras y a los que desarrollan software en general; que además reduzca los precios de venta del software para las empresas clientes que necesitan adquirir algún tipo de producto elaborado localmente, como alternativa a ofertas de software propietario. Los programadores expresan su deseo por cambiar a una metodología de desarrollo adecuada pero también expresan dudas y resistencia a este cambio ya que no conocen sobre casos prácticos o experiencias concretas y verídicas de la aplicación de Metodologías Ágiles en proyectos de software realizados en El Salvador.

[1] Ver anexo sobre consulta realizada.

1.2 ENUNCIADO DEL PROBLEMA

Ahora que se tiene una mejor perspectiva sobre la problemática en el desarrollo de software en El Salvador surge la interrogante:

¿Serán la aplicación de Metodologías Ágiles y el uso de Herramientas De Desarrollo Open Source las alternativas adecuadas para mejorar los procesos de desarrollo de software disminuyendo su costo, realizando entregas en tiempos más cortos y proporcionando al cliente exactamente lo que solicita en el momento que lo necesita?

1.3 JUSTIFICACION

La importancia de este proyecto de trabajo de graduación radica en que se dará a conocer una alternativa al desarrollo de software en El Salvador, a partir de otras formas de desarrollo de software y el uso de herramientas que no sean propietarias.

Este proyecto de trabajo de graduación ofrecerá una guía para el uso de Metodologías Ágiles y Herramientas De Desarrollo Open Source por medio de la elaboración de un sistema informático, permitiendo su aplicación en futuros proyectos de desarrollo de software logrando que estos sean competitivos frente a las soluciones de software propietario o creado por procesos tradicionales. Al realizar este proyecto podrá comprobarse que si se usa una Metodología Ágil adecuada en proyectos de software las probabilidades de éxito aumentarán.

1.3.1 FACTIBILIDAD TECNICA

Existe un grupo internacional que promueve el uso de las Metodologías Ágiles para el desarrollo de software (Agil Alliance) y muchos grupos de usuarios que las practican.

Existen varios libros publicados en idioma inglés que tratan sobre la aplicación de Metodologías Ágiles.

Se cuenta con computadoras para desarrollo de la aplicación, y para su futura implantación en el Departamento de Órtesis y Prótesis. Poseen conectividad por medio de una red LAN y acceso a Internet.

Existe un grupo de software para desarrollo orientado a objetos: Java, Python, C++. Disponibles en Internet con amplia documentación y entornos de desarrollo integrado (IDE) como: NETBEANS, DEVC++, WxPython. Todos estos del tipo Open Source.

En cuanto a las bases de datos disponibles se pueden mencionar: Firebird, Interbase, MySql, PostgreSQL. Igualmente disponibles en Internet, sin costo adicional.

1.3.2 FACTIBILIDAD OPERATIVA

El uso de las Metodologías Ágiles y Herramientas De Desarrollo Open Source se complementan en un proyecto de software, apoyando el trabajo de los programadores para la obtención de requerimientos y pruebas del software desarrollado.

El sistema a desarrollar es requerido como solicitud a la escuela de computación por parte del departamento de Órtesis y Prótesis de la UDB. Su implementación

esta contemplada dentro de sus planes de mejora. El acceso de datos y autorización para la etapa de investigación esta autorizado por la jefatura del departamento de Órtesis y Prótesis.

Tanto las Metodologías Ágiles como las Herramientas De Desarrollo Open Source que se utilizarán en el proyecto podrán ser reutilizadas en otros proyectos similares adaptándolas a sus propias características, esto podrá lograrse debido a la documentación que se proporcionará sobre las prácticas que se utilizarán en le proceso del desarrollo del software.

1.3.3 FACTIBILIDAD ECONOMICA

Para la etapa de desarrollo no se requiere la compra de nuevo hardware. Se utilizará el equipo disponible actualmente en el Departamento De Órtesis y Prótesis de la UDB. En cuanto a las herramientas a utilizar se han considerado productos del tipo Software Open Source, con amplia documentación y disponibles sin costo al obtenerlos vía Internet. En el caso de Java bajo licencia de Sun Microsystems y no de código abierto pero disponible sin costo para desarrollo.

1.3.4 BENEFICIARIOS DEL PROYECTO DE TRABAJO DE GRADUACION

En primer lugar los programadores, desarrolladores que buscan alternativas para ser competitivos a través de el uso de Herramientas Open Source y metodologías de desarrollo que faciliten la creación de sus productos de software.

Las empresas que requieran comprar un software o solución que no sea de costo elevado y satisfaga sus requerimientos.

En este proyecto se beneficia directamente al departamento de Órtesis y Prótesis de la Universidad Don Bosco, como cliente que solicita un sistema informático que se desarrolla bajo las Metodologías Ágiles y Herramientas de Desarrollo Open Source. Este software le ayudara en la gestión de la producción y la administración del departamento.

1.3.5 VENTAJAS DE LA APLICACIÓN DE METODOLOGIAS AGILES

- Se garantizará la disminución de los costos del proceso de desarrollo de software.
- Se incrementarán oportunidades para vender productos a precios competitivos al software propietario.
- Las Metodologías Ágiles están diseñadas para facilitar la obtención de requerimientos de software que satisfaga a la empresa cliente, por medio de un software que le ayude a ser productiva automatizando el registro de información que es vital para sus diversos procesos y funciones.
- Las Metodologías Ágiles no están restringidas a una herramienta o producto exclusivo y pueden ser adaptadas al proceso de desarrollo de software particular de cada proyecto.
- Las Metodologías Ágiles son de fácil aprendizaje porque son más flexibles que algunos procesos tradicionales, esto lo hace ideal para las empresas productoras de software y programadores que no poseen una metodología definida de desarrollo pero desean mejorar sus procesos y ofrecer a sus clientes mejores productos y cuentan con un número pequeño de personal como puede verse en la comparación hecha en la Tabla 1-1.

METODOLOGIAS ÁGILES	METODOLOGIAS TRADICIONALES
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.
El cliente es parte del equipo de desarrollo.	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes).	Grupos grandes y posiblemente distribuidos.
Pocos artefactos.(modelos ,prototipos)	Más artefactos.(modelos, prototipos)
Pocos Roles. (¿Quién hace que?)	Más roles. (¿Quién hace que?)

Tabla 1-1

1.3.6 VENTAJAS DEL USO DE HERRAMIENTAS DE DESARROLLO OPEN SOURCE

Las Herramientas De Desarrollo Open Source reducen los costos pues no requieren de pago por su uso y en algunos casos contienen documentación completa y una gran comunidad de usuarios que comparten experiencias e información.

El código fuente de las Herramientas Open Source esta disponible para poder ser modificado y de esta manera adaptar el software a nuestras necesidades obteniendo una Herramienta De Desarrollo personalizada a cada programador.

La seguridad con las Herramientas Open Source esta garantizada ya que generalmente son desarrolladas por grandes equipos de programadores a través de la Internet. Esto significa que cada línea de código es vista por muchas personas. Esto reduce drásticamente las posibilidades de que Bugs e imperfecciones pasen desapercibidos.

1.4 DELIMITACION

1.4.1 UBICACIÓN Y MAGNITUD DEL PROYECTO A REALIZAR

El presente proyecto trata sobre la aplicación de Metodologías Ágiles y Herramientas De Desarrollo Open Source para la creación de un sistema informático que gestionará la producción en el Departamento De Órtesis y Prótesis de la Universidad Don Bosco (DOP).

La propuesta incluye el diseño general del sistema y la implantación de algunos de sus módulos, concretamente el control de materiales en Bodega, préstamo de equipo y, la asignación de materiales por solicitud de servicio de producción.

Se aplicarán las Metodologías Ágiles y Herramientas De Desarrollo Open Source en el tiempo requerido por este proyecto. Así los módulos restantes podrán desarrollarse e implantarse en futuros proyectos o por personas interesadas en retomar el diseño presentado y las herramientas propuestas.

1.4.1.1 DEPARTAMENTO DE ORTESIS Y PROTESIS DE LA UNIVERSIDAD DON BOSCO

El Departamento de Órtesis y Prótesis es el resultado del trabajo conjunto que, desde 1996, la Universidad Don Bosco ha mantenido con la Cooperación Técnica Alemana (GTZ). Dicha colaboración se ha producido dentro del marco del Proyecto de "Mejoramiento de la Ortopedia Técnica en El Salvador", el cual ha sido impulsado en el país por el gobierno alemán desde el año 1993.

Las principales áreas de acción del Departamento de Órtesis y Prótesis son:

Formación de Técnicos en Órtesis y Prótesis, reconocidos por el Sistema de Educación Superior de El Salvador y acreditados Internacionalmente por la I.S.P.O. (Sociedad Internacional de Ortética y Protética) dentro de la Categoría II.

Producción de Ayudas en Órtesis y Prótesis a través del programa de Práctica Profesional, beneficiando a personas que necesitan este tipo de productos y que en muchas ocasiones vienen remitidos de instituciones .médico hospitalarias.

Programas educativos: Diplomado a Distancia en Órtesis y Prótesis, el cual se desarrolla completamente en línea, realizando un examen presencial al final del Módulo. Este programa está abierto a estudiantes de diversos países y es ampliamente reconocido en el continente americano.

1.4.1.2 ANALISIS DE LA SITUACIÓN ACTUAL DEL DEPARTAMENTO DE ORTESIS Y PROTESIS DELA UNIVERSIDAD DON BOSCO

El departamento. De Órtesis y Prótesis posee actualmente un sistema para la gestión de bodega (Desarrollado en alpha4), además se lleva un registro de las personas atendidas con los servicios de Órtesis y Prótesis, técnicos que elaboran

los distintos productos y las horas de trabajo asignadas, todo esto mediante hojas de calculo electrónicas.

Los usuarios que manejan el sistema actual expresan la necesidad de un sistema que contemple la seguridad y el control de usuarios, que no sea vulnerable a manipulación por cualquier usuario de sistema.

Otra aspecto requerido por DOP es una reorganización de los productos en bodega (espacios y ubicación), apoyado por el sistema de gestión. Se necesita obtener reportes de las existencias de los diferentes materiales, con un control sobre cantidades mínimas y máximas que se requieren.

En el área de producción se necesita un sistema que genere órdenes de producción para cada técnico, en la cual se detallen los materiales utilizados, el tiempo de elaboración y la persona beneficiaria del producto.

Un detalle particular en el desarrollo del futuro sistema, es que los técnicos involucrados en la producción, son alumnos del departamento. Estos alumnos son asignados cada año, de tal forma que la cantidad de técnicos puede cambiar año tras año. Esto se traduce en asignaciones de producción diferentes debido a la demanda de productos solicitados. Actualmente estas reasignaciones son difíciles administrar con las hojas de cálculo electrónicas.

1.4.2 METODOLOGIAS A UTILIZAR

Actualmente son varias las metodologías pertenecientes al grupo Metodologías Ágiles y cada una de ellas basa sus principios y sus valores en el Manifiesto Ágil creado por el grupo Alianza Ágil; pero para el desarrollo del software en el

Departamento De Órtesis y Prótesis de la Universidad Don Bosco se han seleccionado Programación Extrema y Crystal Clear.

Los valores y principios de estas metodologías son mencionados en la Tablas 2- 2 y 2-3 respectivamente, los cuales se han tomado en cuenta y han sido de gran influencia para la selección de estas metodologías ya que puede observarse con claridad que ambas metodologías tienen metas y objetivos concretos, sólidos y congruentes.

Un criterio fundamental en la selección de estas metodologías es que sus características se ajustan a proyectos con perfiles similares al que se realizará.

1.5 ALCANCES Y LIMITACIONES

1.5.1 ALCANCES

- Documentación del sistema en base a las metodologías aplicadas (manual del usuario).
- Se aplicara los elementos de metodologías Crystal Clear y prácticas de Extreme Programming.
- Uso de herramientas Open Source en la elaboración del proyecto de desarrollo.
- El sistema dará soporte a los procesos de bodega y producción del departamento Órtesis y Prótesis de la UDB.
- Soporte multiusuario y control de acceso al sistema (permisos de usuario), en red LAN.

1.5.2 LIMITACIONES

- El proyecto no cubre todas las metodologías de Ingeniería de Software reconocidas como Metodologías Ágiles, ni pretende agotar las posibilidades del uso de alguna en particular.
- Para esta versión de la aplicación no se contempla otras plataformas fuera de Windows.
- No se implantaran módulos gerenciales sino solamente los operativos que puedan alimentar con información a dichos módulos.

1.6 OBJETIVOS

1.6.1 OBJETIVO GENERAL

Aplicar Metodologías Ágiles del Desarrollo de Software y utilizar Herramientas de Desarrollo Open Source para la creación de un sistema informático de gestión de la producción en el Departamento de Órtesis y Prótesis de la UDB.

1.6.2 OBJETIVOS ESPECIFICOS

- Desarrollar el sistema informático al Departamento de Órtesis y Prótesis de la UDB de acuerdo a los principios de las metodologías y procesos: Extreme Programming, Cristal Clear, adaptándolas al proceso particular.
- Demostrar el uso de Herramientas De Desarrollo De Software y Base de Datos alternativos (software Open Source) al uso de software propietario, en cuanto a sus características y su bajo costo.
- Apoyar el desarrollo de las funciones del DOP en especial en La Producción de Ayudas (Órtesis y Prótesis); Manejo De Materiales En Bodega.

- Presentar un diseño completo del sistema solicitado que permita desarrollos o ampliaciones posteriores al proyecto (Diseño de la base de datos del DOP).
- Documentar el software SIPOP proporcionando al DOP lo siguiente:
    - Manual del usuario
    - Manual del programador
    - Diccionario de datos

CAPÍTULO II

MARCO TEORICO

2.1 MODELOS DE PROCESO DE SOFTWARE

El desarrollo del software se puede caracterizar como un ciclo de resolución de problemas en el que se encuentran cuatro etapas distintas: Estado actual, definición de problemas, desarrollo técnico e integración de soluciones.

Existen muchos modelos de desarrollo de software desde el clásico modelo lineal o en cascada hasta los modelos de desarrollo evolutivo: Incremental y espiral. En todos ellos cada tarea suele tener asociado un producto (documento o software). A partir de estos tipos base se han definido procesos específicos que detallan en mayor medida cómo deben llevarse a cabo las tareas del proceso de desarrollo.

Examinemos brevemente los modelos de proceso.

2.1.1 MODELO EN CASCADA

No tan de moda, pero es básico para comprender los otros. Este requiere la definición previa de los requerimientos para continuar el proceso.

Contempla las fases de: Análisis, Diseño, Código, Mantenimiento. Cada una debe completarse para continuar a la siguiente fase, estas pueden recibir una retroalimentación y continuar el proceso lineal.

2.1.2 MODELO DE PROTOTIPOS

Comienza con la recolección de requisitos, el cliente y el desarrollador definen objetivos globales, entonces aparece un diseño Rápido que lleva a la construcción del prototipo. Este se evalúa por el cliente y sirve para obtener más requerimientos y se va ajustando hasta lograr el sistema deseado.

2.1.3 MODELO DE DESARROLLO RAPIDO DE APLICACIONES

Es una versión adaptada del proceso lineal, pero utilizando componentes. Es decir piezas de software previamente creadas y que contribuyen a la generación de una aplicación de manera veloz. Enfatiza la reutilización ya que muchos componentes son comunes a diferentes sistemas, por ejemplo las interfaces de usuario (UI).

2.1.4 MODELOS EVOLUTIVOS

Cuando los requisitos cambian, las fechas de entrega son cercanas debido a las exigencias del mercado, y auque se comprenden los requisitos centrales del sistema se tienen que definir detalles de la aplicación, el modelo evolutivo se acomoda a este tipo de productos.

Los modelos evolutivos son iterativos. Permiten desarrollar versiones cada vez mas completas del software. Algunos de estos modelos son:

El modelo incremental: es un modelo evolutivo pero a diferencia de la construcción de prototipos, se centra en la entrega de un producto operacional en cada incremento. Es útil cuando la dotación de personal no está disponible para una

implementación completa en la fecha límite que se haya establecido, los primeros incrementos se pueden realizar con menos personas.

El modelo en Espiral. Es un proceso propuesto por Boehm, es un modelo evolutivo que conjuga la naturaleza iterativa de la construcción de prototipos con los aspectos controlados y sistemáticos de modelo lineal secuencial. Proporciona el potencial para el desarrollo rápido de versiones incrementales de software, en este modelo el software se desarrolla en una serie de versiones incrementales. Durante las primeras versiones podría ser un simple prototipo o modelo en papel y al final las últimas iteraciones producen un sistema mas completo conforme al diseño[2].

2.2 HERRAMIENTAS DE MODELADO

La Object Management Group OMG es una organización sin fines de lucro que produce y mantiene especificaciones en la industria para operabilidad entre las aplicaciones empresariales. Esta conformada por miembros individuales y por la mayoría de empresas de la industria del software representadas por medio del Board Of Directors.

OMG es propietaria de CORBA (Common Object Request Broker Architecture), arquitectura independiente de vendedor e infraestructura que usan la aplicaciones de computara para trabajar juntas sobre redes y las especificaciones de MDA Model Driven Arquitecture .OMG es responsable de las especificaciones de modelado XMI,CWM y UML.

[2] Ingeniería del Software un enfoque práctico, Quinta Adición McGraw Hill Cap 2 Roger Pressman

2.2.1 LENGUAJE UNIFICADO DE MODELADO UML

¿Qué es un modelo?; "es una representación abstracta de una especificación, un diseño o un sistema desde un punto de vista particular" [3] . Tomando esto en cuenta se puede decir que, el Lenguaje de Modelado Universal, UML por sus siglas en ingles ha evolucionado desde su primera versión desarrolla por Grady Booch, James Rumbaugh e Ivar Jacobson en 1996 hasta convertirse en un estándar para modelar mantenido por OMG y actualmente en su versión 1.5[4].

Un lenguaje de modelado es una manera de expresar distintos modelos que se producen en el proceso de desarrollo. Define una colección de elementos del modelo, que son aproximadamente análogos a la pronunciación en el lenguaje hablado. De esta forma UML ayuda ha especificar, visualizar y documentar modelos de los sistemas de software, incluyendo la estructura y diseño. Con UML se pueden analizar los requerimientos de la aplicación y diseñar la solución representando los resultados por medio de los veinte tipos de diagramas estándar que incluye.

UML es independiente de plataforma. UML no se apega a una especificación de hardware o software, se puede modelar diferentes tipos de sistemas, incluso proyectos que no sean de software. UML se puede usar en el análisis y diseño de cualquier tipo de aplicación corriendo en cualquier tipo de combinación de hardware, sistema operativo, lenguaje de programación y red.

---

[3] Utilización de UML en Ingeniería del Software con Objetos y Componentes, 54-55,Perdita Stevens.

[4] A finales del 2004 la versión actualizada formal/[03]-03-01 (Unified Modeling Language, v1.5).

Modelos y metodologías. El proceso para obtener y analizar los requerimientos para la aplicación, e incorporar estos en el diseño de un programa, es complejo y existen actualmente muchas metodologías que definen procesos formales de cómo hacerlo. Una característica de UML es que es independiente de la metodología que utilice para su proceso de desarrollo.

El UML incluye tres categorías de diagramas:

Structural Diagramas: incluye Diagramas de clase, diagramas de objeto, diagramas de componentes y diagramas de desarrollo.

Behavior Diagrams: incluye los diagramas de casos de uso (Utilizados para obtener requerimientos, en este proyecto el uso principal de UML); Diagrama de Secuencia, Diagrama de Actividad y Diagrama de Colaboración, y Diagramas de estado.

Model Management Diagrams: incluye Paquetes, Subsistemas, y Modelos.

2.3 METODOLOGIAS AGILES

2.3.1 HISTORIA

En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "Ágil" aplicado al desarrollo de software. En esta reunión participo un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto.

Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo "Ágil" de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía "Ágil"[5].

2.3.2 DEFINICION

Metodología Ágil es una solución para el desarrollo de software que se ajusta específicamente a proyectos actuales donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad y donde el entorno del sistema es muy cambiante, con pequeños equipos de desarrollo, requisitos que cambian fácilmente y nuevas tecnologías.

Según Alistair Cockburn una metodología se compone de muchos elementos una serie de valores que interactúan para lograr un Milestone o evento importante en el proyecto, esto es como un juego en que los participantes buscan colaboración y se comunican continuamente.

---

[5] Metodologías Ágiles en el Desarrollo de Software Universidad Politécnica de Valencia

José H. Canós, Patricio Letelier y Mª Carmen Penadés

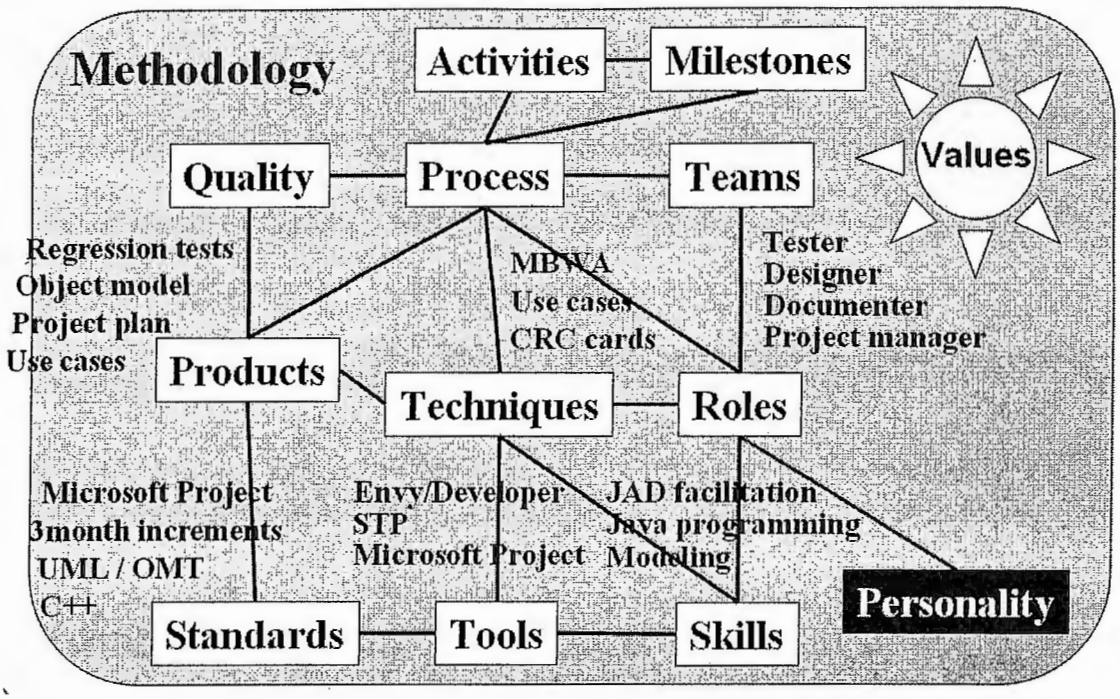


Figura 2-1[6]

Pero esta metodología "XYZ" adquiere una personalidad propia al ser asumida por un equipo de personas que la llevan a la practica, según Cockburn esto es un ecosistema, donde cada miembro contribuye en los logros del proyecto.

[6] What Is Agile Development & What does it Imply? Alistair Cockburn alistair.cockburn@acm.org http://Alistair.Cockburn.us

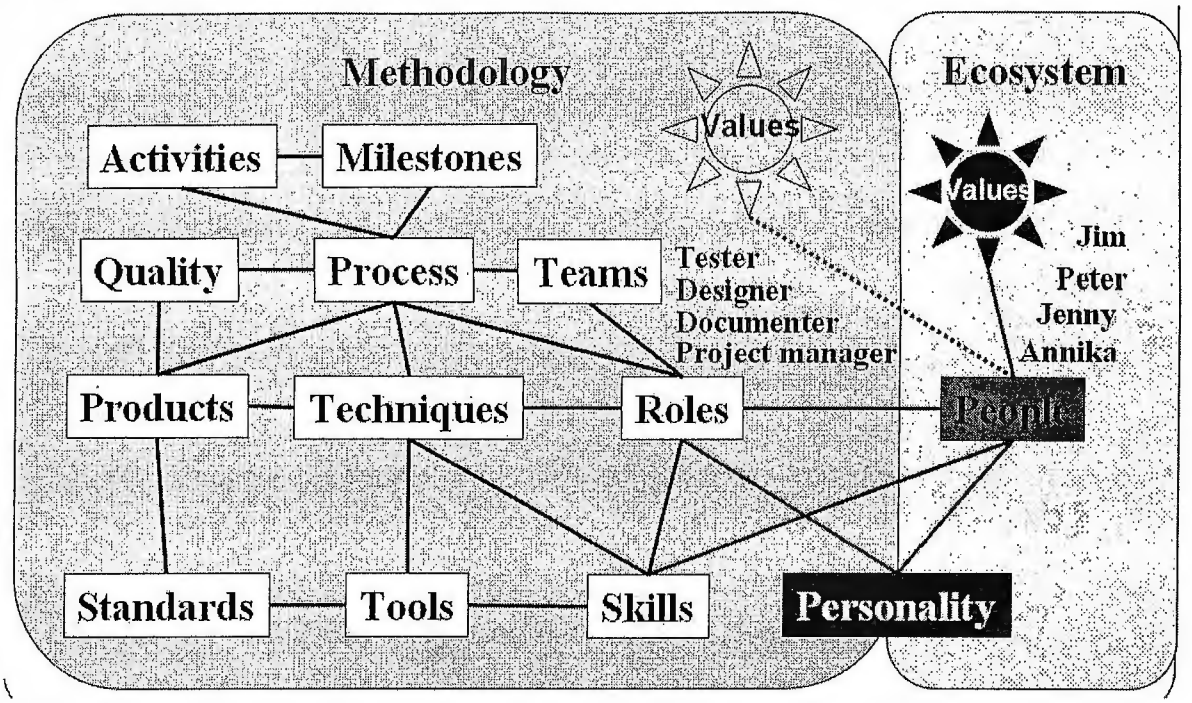


Figura 2-2[7]

2.3.3 MANIFIESTO ÁGIL

2.3.3.1 VALORES DEL MANIFIESTO ÁGIL[8]

Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.

La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte

[7] What Is Agile Development & What does it Imply? Alistair Cockburn alistair.cockburn@acm.org http://Alistair.Cockburn.us

[8] Metodologías Ágiles en el Desarrollo De Software. Taller realizado en el marco de las VIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2003. Alicante, del 12 al 14 de Noviembre de 2003

automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.

Desárrollar software que funciona más que conseguir una buena documentación. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental.

La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito. Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

2.3.3.2 PRINCIPIOS DEL MANIFIESTO ÀGIL

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso a seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.

- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

2.3.4 METODOLOGÍAS ÁGILES EXISTENTES

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil y coinciden con los principios enunciados anteriormente, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. A continuación se mencionan algunas de estas metodologías ágiles. La mayoría de ellas ya estaban siendo utilizadas con éxito en proyectos reales pero les faltaba una mayor difusión y reconocimiento.

- SCRUM
- Extreme Programming, XP
- Crystal Methodologies (Crystal Clear, Crystal Orange)
- Dynamic Systems Development Method (DSDM)
- Adaptive Software Development (ASD)
- Feature-Driven (FDD)
- Lean Development (LD)

2.4 PROGRAMACIÓN EXTREMA (EXTREME PROGRAMMING, XP)

Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico. XP se descompone en las siguientes perspectivas:

- Valores
- Principios
- Actividades
- Roles
- Prácticas

Los principios y prácticas son de sentido común pero llevadas al extremo, de ahí proviene su nombre. Kent Beck, el padre de XP, describe la filosofía de XP9 sin cubrir los detalles técnicos y de implantación de las prácticas. Posteriormente, otras publicaciones de experiencias se han encargado de dicha tarea.

2.4.1 ELEMENTOS DE LA METODOLOGÍA PROGRAMACIÓN EXTREMA (XP)

2.4.1.1 Valores

Es decir como debe de estar basado el desarrollo de software. XP se basa en cuatro valores que son:

- **Retroalimentación** (FEEDBACK): Retroalimentación continúa entre el cliente y el equipo de desarrollo.
- **Comunicación** (COMUNICATION): Comunicación fluida entre todos los participantes.
- **Simplicidad** (SIMPLICITY): Simplicidad en las soluciones implementadas.
- **Coraje** (COURAGE): Coraje para enfrentar los cambios.

[9] Beck, K.. "Extreme Programming Explained. Embrace Change", Pearson Education, 1999. Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio", Addison Wesley, 2000.

2.4.1.2 Principios

Usados para determinar que tanto una práctica puede ser utilizada exitosamente en un contexto XP. Los cinco principios básicos de XP son:

- Retroalimentación Rápida (Rapid feedback).
- Asumir Simplicidad (Assume simplicity).
- Cambio Incremental (Incremental change).
- Adoptar los cambios (Embracing change)
- Trabajo de alta calidad (Quality work)

2.4.1.3 Actividades

Se refiere al fundamento del desarrollo de software. Son cuatro las actividades básicas de XP:

- Planificación (PLANNING).
- Diseño (DESIGNING)
- Codificación (CODING).
- Prueba (TESTING).

2.4.1.4 Prácticas[10]

Son usadas para realizar exitosamente las ACTIVIDADES, con la finalidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto,

[10]Actas Metodologías ágiles en el desarrollo de software. Alicante España 12 de Noviembre 2003.Taller realizado en el marco de las VIII Jornadas de Ingeniería del Software y Bases de Datos, JISBD 2003.

lo suficiente para que el diseño evolutivo funcione. Las prácticas utilizadas en XP son:

1. El juego de la planificación. Hay una comunicación frecuente entre el cliente y los Programadores. Busca determinar rápidamente el alcance de la versión siguiente, combinando prioridades de negocio definidas por el cliente con las estimaciones técnicas de los programadores. Éstos estiman el esfuerzo necesario para implementar las historias del cliente y éste decide sobre el alcance y la agenda de las entregas. Las historias se escriben en pequeñas fichas.
2. Entregas pequeñas y frecuentes. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses. Pueden liberarse nuevas versiones diariamente (como es práctica en Microsoft), pero al menos se debe liberar una cada mes.

3. Metáforas del sistema. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo (managers y desarrolladores). Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema). Una metáfora puede interpretarse como una arquitectura simplificada. La concepción de metáfora

que se aplica en XP deriva de los estudios de Lakoff y Johnson, bien conocidos en lingüística y psicología cognitiva.

4. Diseño simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto. Se eliminan complejidades innecesarias y código extra, y se define la menor cantidad de clases posible. No debe duplicarse código. En un oxímoron deliberado, se urge a "decir todo una vez y una sola vez". Nadie en XP llega a prescribir que no haya diseño concreto, pero el diseño se limita a algunas tarjetas elaboradas en sesiones de diseño de 10 a 30 minutos. Esta es la práctica donde se impone el minimalismo de YAGNI: no implementar nada que no se necesite ahora; o bien, nunca implementar algo que vaya a necesitarse más adelante; minimizar diagramas y documentos.

5. Prueba continua. La producción de código está dirigida por las pruebas. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema. Esto es test-driven development. El propósito del código real no es cumplir un requerimiento, sino pasar las pruebas. Las pruebas y el código son escritas por el mismo programador, pero la prueba debería realizarse sin intervención humana, y es a todo o nada. Hay dos clases de prueba: la prueba unitaria, que verifica una sola clase, o un pequeño conjunto de clases; la prueba de aceptación verifica todo el sistema, o una gran parte.

6. Refactorización (Refactoring). Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo. El proceso consiste en una serie de pequeñas transformaciones que modifican la estructura interna preservando su conducta aparente. La práctica también se conoce como Mejora Continua de Código o Refactorización implacable. Se lo ha parafraseado diciendo: "Si funciona bien, arréglelo de todos modos". Se recomiendan herramientas automáticas. Ken Orr en sus comentarios recomienda GeneXus de ARTech,

7. Programación en parejas. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores). Dos personas escriben código en una computadora, turnándose en el uso del ratón y el teclado. El que no está escribiendo, piensa desde un punto de vista más estratégico y realiza lo que podría llamarse revisión de código en tiempo real. Los roles pueden cambiarse varias veces al día. Esta práctica no es en lo absoluto nueva. Hay antecedentes de programación en pares anteriores a XP. Steve McConnell la proponía en 1993 en su Code Complete.

8. Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento siempre que escriba antes la

prueba correspondiente. Algunas veces los practicantes aplican el patrón organizacional CodeOwnership de Coplien[11]

9. Integración continua. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día. Debe ejecutarse la prueba antes y después de la integración. Hay una máquina (solamente) dedicada a este propósito.

10. Ritmo sostenible, trabajando un máximo de 8 horas por día. Antes se llamaba a esta práctica Semana de 40 horas. Mientras en RAD las horas extras eran una "best practice"[12], en XP todo el mundo debe irse a casa a las cinco de la tarde. Dado que el desarrollo de software se considera un ejercicio creativo, se estima que hay que estar fresco y descansado para hacerlo eficientemente; con ello se motiva a los participantes, se evita la rotación del personal y se mejora la calidad del producto. Deben minimizarse los héroes y eliminar el "proceso neurótico". Aunque podrían admitirse excepciones, no se permiten dos semanas seguidas de tiempo adicional. Si esto sucede, se lo trata como problema a resolver.

---

[11] James O. Coplien y Douglas C. Schmidt, Pattern Languages of Program Design (A Generative Development-Process Pattern Language), Reading, Addison Wesley, 1995.

[12] Steve McConnell. Rapid Development. Taming wild software schedules. Redmond, Microsoft Press, 1996.

11. Todo el equipo en el mismo lugar. El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita. A esta práctica también se le conoce como "El Cliente en el Sitio", pero como esto no parecía cumplirse (si el cliente era muy "junior" no servía para gran cosa, y si era muy "senior" no deseaba estar allí), se especificó que el representante del cliente debe ser preferentemente un analista. (Tampoco se aclara analista de qué; seguramente se definirá en una próxima versión).

12. Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible. Según las discusiones en Wiki de Extreme Programming, algunos practicantes se desconciertan con esta regla, prefiriendo recurrir a la tradición oral. Otros la resuelven poniéndose de acuerdo en estilos de notación, indentación y nomenclatura, así como en un valor apreciado en la práctica, el llamado "código revelador de intenciones". Como en XP rige un cierto purismo de codificación, los comentarios no son bien vistos. Si el código es tan oscuro que necesita comentario, se lo debe reescribir o refactorizar.

13. Espacio abierto. Es preferible una sala grande con pequeños cubículos o, mejor todavía, sin divisiones. Los pares de programadores deben estar en el centro. En la periferia se ubican las máquinas privadas. En un encuentro de espacio abierto la agenda no se establece verticalmente.

14. Reglas justas. El equipo tiene sus propias reglas a seguir, pero se pueden cambiar en cualquier momento. En XP se piensa que no existe un proceso que sirva para todos los proyectos; lo que se hace habitualmente es adaptar un conjunto de prácticas simples a las características de cada proyecto.

Las prácticas se han ido modificando con el tiempo. Originariamente eran doce; de inmediato, trece. Las versiones más recientes enumeran diecinueve prácticas, agrupadas en cuatro clases, tal como se muestra en la Tabla 2-1.

Prácticas Conjuntas	• Iteraciones • Vocabulario Común – Reemplaza a Metáforas • Espacio de trabajo abierto • Retrospectivas
Prácticas de Programador	• Desarrollo orientado a pruebas • Programación en pares • Refactorización • Propiedad colectiva • Integración continua

	• YAGNI ("No habrás de necesitarlo") – Equivale a Diseño Simple
Prácticas de Management	• Responsabilidad aceptada • Cobertura aérea para el equipo • Revisión trimestral • Espejo – El manager debe comunicar un fiel reflejo del estado de cosas • Ritmo sostenible
Prácticas de Cliente	• Narración de historias • Planeamiento de entrega • Prueba de aceptación • Entregas frecuentes

Tabla 2-1.

Oficialmente las prácticas siguen siendo doce y su extensión se basa en documentos inéditos de Kent Beck.

2.4.1.5 Roles[13]

Existen diferentes roles en XP para diferentes propósitos y tareas durante el proceso y sus prácticas.

Los roles de acuerdo con la propuesta original de Beck son los siguientes:

[13] De los métodos Heterodoxos en la Construcción del Software. Carlos Reynoso. Willi.net

- **Programador (PROGRAMMER).** El programador escribe las pruebas unitarias y produce el código del sistema.
- **Cliente (CUSTOMER).** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
- **Encargado de pruebas (TESTER).** Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- **Encargado de seguimiento (TRACKER).** Proporciona realimentación al equipo. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.
- **Entrenador (COACH).** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor (CONSULTANT).** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (BIG BOSS).** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

2.4.1.6 Proceso

El ciclo de desarrollo o ciclo de vida ideal de XP consiste de seis fases[14]:

1. **Exploración**
2. **Planificación de la Entrega (Release)**
3. **Iteraciones**
4. **Producción**
5. **Mantenimiento**
6. **Muerte del Proyecto**

Fase I: 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.

Fase II: Planificación de la Entrega

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

[14] Beck, K.. "Extreme Programming Explained. Embrace Change", Pearson Education, 1999. Traducido al español como: "Una explicación de la programación extrema. Aceptar el cambio", Addison Wesley, 2000.

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 de 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.

Fase III: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se

implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores. Wake [15] proporciona algunas guías útiles para realizar la planificación de la entrega y de cada iteración.

Fase IV: 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).

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De

15 Wake, W.C. "Extreme Programming Explored". Addison-Wesley. 2002.

esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI: Muerte del Proyecto

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

El ciclo de vida es, naturalmente, iterativo. El siguiente diagrama describe su cuerpo principal:

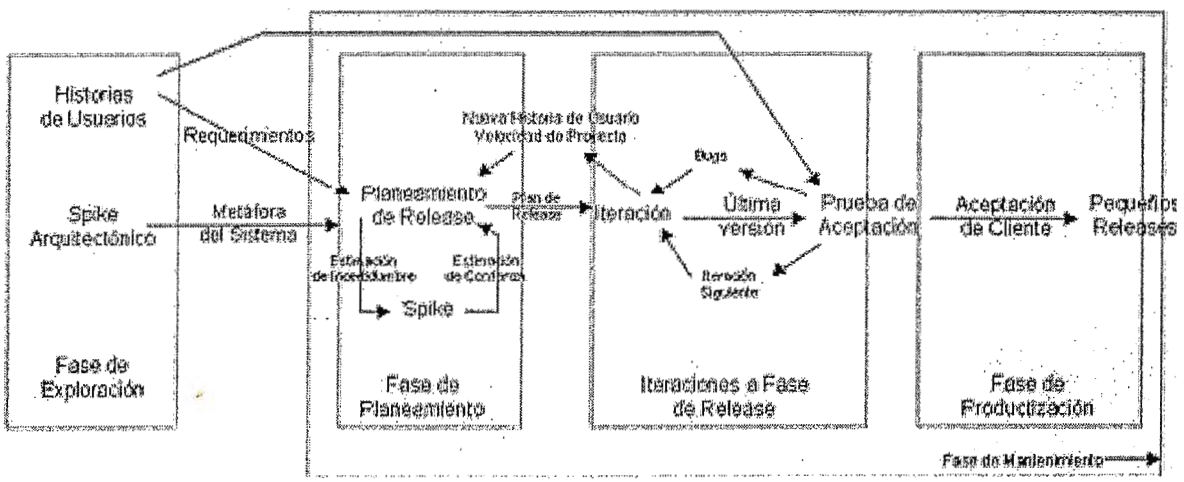


Figura 2-3.

2.5 METODOLOGIAS CRYSTAL

Se trata de un conjunto de metodologías para el desarrollo de software caracterizadas por estar centradas en las personas que componen el equipo y la

reducción al máximo del número de artefactos producidos. Han sido desarrolladas por Alistair Cockburn. El desarrollo de software se considera un juego cooperativo de invención y comunicación, limitado por los recursos a utilizar. El equipo de desarrollo es un factor clave, por lo que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo Crystal Clear (3 a 8 miembros) y Crystal Orange (25 a 50 miembros) que vienen dada por los factores : Confort(C), Discresionary Money(D), Essential Money (E) and Life(L).

Estos factores indican una potencial perdida causada por una falla en el sistema es decir el nivel critico del sistema. Así una falla en C indicara que un "crash" del sistema debido a defectos causará perdida de comodidad en el usuario, un "crash" del sistema critico de vida L causará la perdida de la vida[16].

[16] Agile sotware development methods Review and Analysis Pekka Abrahmasson, Outi Salo, Jussi ronkainen & Juhani Warsta . Espoo 2002 VIT publications. pag 36.

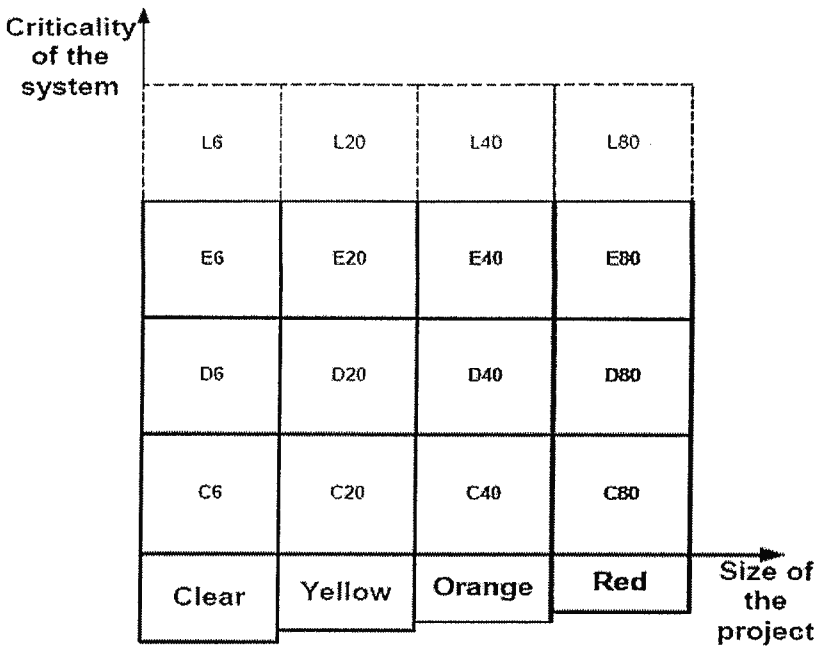


Figura 2-4

2.6 CRYSTAL CLEAR

Es una metodología de la familia Crystal donde el tamaño del equipo es hasta 6 personas y puede crecer hasta 12 si estos se están turnando. No se utiliza con equipos grandes. Tampoco se recomienda para sistemas de vida críticos, en los cuales se requiere verificación de la exactitud.

Los sistemas creados con Crystal Clear pueden ser cliente-servidor, basados en Web, usando cualquier tipo de base de datos central o distribuido. No apropiado para sistema a prueba de fallas.

Los valores de Crystal Clear incluyen una fortaleza en la comunicación, y agilidad en las entregas. Cortas e informales maneras de comunicación, que incluyen a patrocinadores y la comunidad de usuarios.

Frecuentes entregas con retroalimentación sobre productos y el proceso. Tolerancia en la variación del estilo de trabajo de las personas.

Posee políticas estándares que son obligatorias a menos que puedan ser sustituidas por un equivalente. Por ejemplo las técnicas de planeación de Programación Extrema pueden ser aceptadas en un proyecto Crystal Clear.

2.6.1 Elementos de la metodología Crystal Clear [17]

Crystal Clear como otras metodologías posee, propiedades o principios, técnicas, estrategias o prácticas, y un proceso incremental con entregas (releases) formadas por iteraciones.

2.6.1.1 Propiedades.

Crystal Clear posee un grupo de siete propiedades las tres primeras son necesarias para funcionar y las cuatro restantes incrementan la seguridad a los proyectos.

- Frecuentes Entregas(Frequent Delivery)
- Comunicación Osmótica (Osmotic Communication)
- Mejoramiento Reflexivo (Reflective Improvement)
- Seguridad Personal (Personal Safety)
- Enfoque (Focus)
- Fácil Acceso a los Usuarios Expertos (Easy Access to Expert Users)
- Ambiente Técnico (Technical Environment

Frecuentes Entregas:

[17]Crystal Clear reviewed Kevin Hykin http://xpwestmichigan.org/meeting041026.page

- Provee a los patrocinadores retroalimentación.
- Rápida retroalimentación de los desarrolladores a los usuarios.
- Rápida retroalimentación de los usuarios a los desarrolladores.
- Ayuda a los desarrolladores a mantener su enfoque.
- Permite continua depuración del proceso de desarrollo y de implementación.
- Estímulo moral a través de los logros.

Comunicación Osmótica:

- La información fluye en los antecedentes escuchados por los miembros del equipo.
- Los miembros del equipo adquieren información como si fuera por osmosis.
- Preguntas y respuestas fluyen naturalmente.
- Sorprendentemente hay pocos disturbios en el equipo.

Mejoramiento reflexivo:

- Hacer una lista de lo que está y no está trabajando.
- Discutir que cosas podrían trabajar mejor.
- Hacer cambios en la próxima iteración.

Seguridad Personal:

- Se refiere a sentirse capaz de hablar cuando algo le molesta sin miedo a represalias. Ejemplo:
- Decirle al Manager que la programación es poco realista.
- Decirle a un colega que sus diseños necesitan mejorar.

Enfoque:

- Saber en que ocuparse y tener el tiempo y la tranquilidad para trabajar en ello:

- Objetivos.
- Directrices.
- Prioridades.

Fácil acceso a los usuarios expertos:

- El continuo acceso a los usuarios expertos provee al equipo en:
- Un lugar para implementar y probar las entregas frecuentes.
- Rápida retroalimentación de la calidad de sus productos finalizados.
- Rápida retroalimentación sobre sus decisiones de diseño.
- Actualización de requerimientos.

Ambiente Técnico.

Esto Incluye:

- Test automatizados.
- Administración de la configuración.
- Integración Frecuente.
- Continúas construcciones (BUILD).

2.6.1.2 Roles.

Existen ocho roles nombrados en Crystal Clear, cuatro de estos probablemente tendrán que ser distintas personas. Los otros cuatro roles pueden ser adicionalmente asignados a las personas en el proyecto. Los primeros cuatro son.

- Patrocinador Ejecutivo(Executive Sponsor)
- Diseñador Lider(Lead Designer)
- Diseñador-Programador (Designer-Programmer)
- Usuario Experto (User Expert)

Los otros roles son

- Coordinador (Coordinator)
- Experto de Negocio (Busines Expert)
- Examinador (Tester)
- Escritor (Writer)

Patrocinador Ejecutivo

Es la persona que asigna el dinero en el proyecto. El Patrocinador Ejecutivo es el que mantiene la visión a largo plazo, balanceando las prioridades a corto plazo con las subsecuentes versiones, evolución del equipo y mantenimiento del sistema. Es la persona que creara una vista desde fuera del proyecto y proveerá al l equipo de decisiones gerenciales o de negocio.

Productos esperados:

- Declaración de la Misión
- Compromiso y Prioridades

Diseñador Lider(Lead Designer)

Es la persona técnica al frente, con supuesta experiencia en el desarrollo de software, capaz de hacer el mejor diseño de sistema, dice cuando el equipo esta en la pista o fuera de ella en la ruta de correcta del proyecto y sino dice como regresar. Es un experto en diseño y programación, como los otros design – programmers, generalmente pero no siempre maneja las mas difíciles asignaciones de programación. Asume el rol de Coordinador, arquitecto y

experimentado programador. El puede asumir todo o en parte el papel de Coordinador.

Productos esperados:

- Descripción de la Arquitectura.

Diseñador-Programador (Designer-Programmer)

El nombre indica que cada persona en este rol diseña y programa. El diseñar sin programar trae muchos problemas por falta de retroalimentación por lo que no se admite en un proyecto del tipo Crystal Clear. Programador involucra el diseño naturalmente. Por lo que se utiliza un solo rol Designer-Programmer.

Productos esperados:

- Diseños de pantallas(draft screen)
- Modelo Común de Objetos (Common Domain Model)
- Diseño de notas y bosquejos.
- Código fuente(Source Code)
- Código de migración
- Pruebas
- Empaquetado de Sistema(Package System)

Usuario Experto (User Expert)

Es la persona que se presupone familiarizado con el sistema en uso, que acciones son usadas frecuentemente, que modos de operación y cuales no, que short-cuts son necesarios, que información necesita estar visible al mismo tiempo en la pantalla.

Productos Esperados:

- Lista de Metas
- Casos uso
- Archivo de Requerimientos

Coordinador (Coordinator)

Probablemente es una ocupación parcial de alguien del equipo. En equipos pequeños rara vez se tiene alguien dedicado al manejo del proyecto. Alternativamente el Lead Designer puede jugar el rol de Coordinador, o incluso rotar entre los otros miembros del equipo. El Coordinador, toma notas de las sesiones de plantación y estatus del proyecto. El coordinador tiene que dar la visibilidad dentro del estatus del proyecto, facilita la discusión y reduce el antagonismo en el equipo, posee buenas relaciones interpersonales.

Productos Esperados:

- Mapa del Proyecto
- Plan de versión
- Estatus del proyecto
- Lista de Riesgos
- Plan de Iteración y Status
- Observación de cronograma

Experto de Negocio (Busines Expert)

Es la persona que conoce las reglas de negocio necesarias dentro del sistema, cuales políticas son estables, y cuales son las que comúnmente cambian, conoce como va el negocio, . No posee intima familiaridad con las actividades que los

usuarios realizan minuto a minuto como el Usuario Experto que si las conoce. En algunas ocasiones puede ser el mismo Usuario Experto o el Patrocinador.

Productos Esperados, en conjunto con el Usuario Experto elaboran.

- Lista de Metas
- Casos uso
- Archivo de Requerimientos

Examinador (Tester)

Este rol se asume por los programadores quienes obviamente tomaran turno para ocupar este rol, en otras ocasiones se asigna a una persona para probar el trabaja en conjunto con los programadores.

Productos esperados

- Reporte de Bugs a la fecha

Escritor (Writer)

Se utiliza en algunos periodos del proyecto para documentar.

Productos esperados:

- Manual de Usuario.

Role	Crystal Clear Work Product	PROJECT										
			DELIVERY 1						DELIVERY 2			etc.
			Iteration 1			Iteration 2			Iteration 3			
		Chartering	Delivery & Iteration Planning	Design, Code, Integrate	Reflection workshop	Iteration Planning	Design, Code, Integrate	Deliver, Reflection Workshop	Delivery & Iteration Planning	Design, Code, Integrate	Reflection workshop	
Sponsor	Mission statement	Write							update			
Coordinator	Project map	Write	update			update			update			
	Release/viewing plan	Write		update	update	(update)	update	update	update			
	(-> project status)											
	Iteration plan		Write			Write			Write			
	(-> iteration status)			update			update					
	Risk list	Write	update	update		update	update		update			
Team	Reflection output											
Business Expert	Actor-goal list	Write	update			update			update			etc.
	UCs &Requirements	Write		update			update		Write			
Lead Programmer	Team Conventions	Write			update							
	Architecture doc.			Write			update					
Designer Programmers	Screens, design docs domain model, source code, tests, migration code			continuous writing			continuous writing					
	Package			continuous			Write					
Tester	Test Reports			continuous			continuous					
Writer	User Manual			continuous			Write					

Write = major writing, c = continuous writing, update = updating

Figura 2-5

Relación de Roles y productos en un ciclo de desarrollo Cristal Clear.[18]

[18] Alistair Cockburn http://groups.yahoo.com/group/crystalclear/files/

2.6.1.3 Prácticas

Existen muchas prácticas documentadas de cómo realizar un proyecto de desarrollo de software. El autor de Crystal Clear describe cinco estrategias y seis técnicas que son aplicadas por los equipos de desarrollo ágil[19].

Las cinco estrategias son:

- Exploratory 360° .
- Early Victory.
- Walking Skeleton.
- Incremental Rearchitecture.
- Information Radiators.

Exploratory 360°.

Es parte de establecer el contrato del proyecto, la estrategia incluye las tareas y productos como:

- Valor de Negocio(Busines value).
- Requerimientos.
- Modelo del Dominio (Domain Model).
- Planes de Tecnología (Technology plans).
- Plan del proyecto (Project Plan).
- Crear el Equipo.
- Definición del proceso o metodología.

Victoria Tempranera (Early Victory)

Es una estrategia para la gestión o gerencia del proyecto. Los pasos a seguir son:

[19] Crystal Clear A Human-Powered Methodology for Small teams. Allistair Cockburn Capitulo 3

- Primero las cosas más fáciles. Desarrolle la primera pieza de código visiblemente ejecutable y probado.
- Segundo lo más difícil. Una vez el que quipo ha logrado su primer "victoria" se debe enfrentar el problema mas difícil.
- Secuencia de Valor de Negocio. Desarrolle ítems con mayor valor de negocio lo mas pronto posible, ítems de menor valor de negocio se desarrollan posteriormente.

Walking Skeleton

Es una estrategia para priorizar el trabajo, en tempranas iteraciones del proyecto. Usa una pequeña implantación del sistema que ejecute una pequeña función de extremo a extremo es decir:

- No necesita ser la arquitectura final pero debe enlazar los principales componentes de la arquitectura.
- No es completa o robusta.
- El código es permanente construido con hábitos de producción de código.

Incremental Rearchitecture

Es otra estrategia para priorizar el trabajo a realizar y consiste en:

- No desarrollar en forma cerrada.
- El equipo debe evolucionar la arquitectura en escenas.
- Siempre debe mantenerse el sistema con capacidad de ejecutarse.

Information Radiators

Mantiene un display publicado en un lugar donde las personas pueden ver como ellas trabajan o avanzan en sus tareas. Las características son:

- Es grande y fácilmente visible.
- Es entendible al echar una mirada.
- Cambia periódicamente.
- Es fácil de mantener actualizado.

Las seis técnicas utilizadas por Crystal Clear

- Reflection Workshop
- Process Miniature
- Project Interviews
- Blitz Planning
- Daily Stand-ups
- Burn Charts

Reflection Workshop

Es una reflexión sobre las pasadas iteraciones de trabajo, que lleva a comprometerse en esforzarse para optimizar la próxima iteración.

- Captura que cosas deberían ser mantenidas.
- Identifica hacia donde van los problemas existentes.
- Adiciona nuevas técnicas de Internet para probar en la próxima iteración.

Process Miniature

Crea un pequeño proyecto (90 minutos al día) para permitir al equipo probar su metodología. Le permite aprender al equipo una metodología rápidamente y ayuda a identificar deficiencias en el proceso.

Project Interviews

Construir un pequeño inventario de experiencias en su organización que permita mostrar las fortalezas y debilidades, y temas de su organización.

Blitz Planning

En ésta técnica el equipo trabaja junto para construir un mapa del proyecto y una línea de tiempo. Algunas actividades son:

- Lluvia de ideas.
- revisión de tareas
- extracción y etiquetado de tareas.
- Identificar otras versiones.
- Capturas de salida.
- Reunión con los asistentes a la sesión de Blitz planning
- Optimiza el plan para adecuarlo a las actividades

Daily Stand-ups

- Reuniones cortas
- Se usa para identificar problemas.
- No usar estas reuniones para resolver problemas.
- Cada persona participante responde las preguntas siguientes:
    - ¿Qué trabajo hice ayer?
    - ¿En que planeo trabajar hoy.?
    - ¿Que estoy logrando con mi manera de trabajo?

Burn Charts

Son herramientas para comunicar el estado del proyecto. Estas pueden ser.

Cárteles, gráficos, listas de comprobación del proceso realizado.

2.6.1.4 Proceso

El proceso de desarrollo en Cristal Clear es incremental y se puede representar como el grafico siguiente.

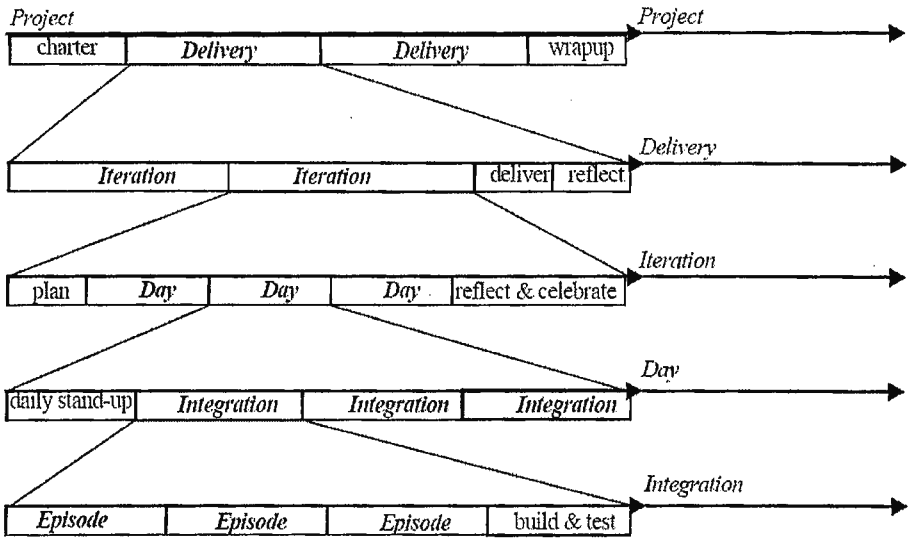


Figura 2-6

Cada proyecto esta compuesto de una serie de entregas (Delivery) o versiones (release) en estas entregas se realizan una serie de iteraciones para alcanzar la versión final. Cada iteración pude realizarse en días, cada iteración posee un periodo de planificación que es seguido por el periodo de desarrollo y uno de reflexión y celebración. Este periodo de reflexión motiva la mejora en la siguiente iteración.

En un día de trabajo de desarrollo se cuenta con una reunión de planificación y varios momentos para integrar el código producido. El código se produce en momentos durante el día llamados episodios que son seguidos de pruebas y generan productos ejecutables o builds. Así el proyecto de Cristal Clear no exige

muchas iteraciones pero si varias entregas (delivery) que entregan un producto que agrega valor al cliente , esta probado y se ejecuta adecuadamente.

2.7 INTEGRACION XP Y CRYSTAL CLEAR

Tanto Crystal Clear como XP son metodologías ágiles que comparten los valores del manifiesto ágil, sin embargo la diferencia entre ambas según Alistair Cockburn es que XP es una metodología más disciplinada siendo Crystal Clear más tolerante hasta el punto que permite utilizar o incorporar algunas de las prácticas de XP. En la tabla 2-3. Se presentan los roles de Crystal y sus respectivos productos, estos se asocian a practicas o productos en XP.

Metodología Ágil	Valores
Programación Extrema	• Se concentra en la satisfacción del cliente. • Posee rapidez y eficiencia en la entrega del software. • Está preparada para requerimientos cambiantes. • Hace énfasis al trabajo en equipo. • Mejora el proyecto de software en 4 aspectos: Comunicación, simplicidad, retro alimentación y confianza.
Crystal Clear	• Fuerza en las comunicaciones y agilidad en las entregas. • Mantener rutas cortas de comunicación • Entregas frecuentes • Reduce esfuerzos extras. Más entregas y mejor comunicación reduce la necesidad de intermediar en productos del trabajo.

Tabla 2-2

Metodología Ágil	Principios
Programación Extrema	• Rápida retroalimentación • Asumir simplicidad • Incrementar gradualmente los cambios • Calidad en el trabajo
Crystal Clear	• Cada proyecto merece sus propias metodologías y procesos adaptados. • Ágil y frente a frente son atributos deseables. • Las personas se comunican mejor frente a frente. • Variabilidad diaria e inconsistencia son debilidades a tomar en cuenta • Un proceso debe construirse sobre personas de buen comportamiento, la capacidad de las personas de observar a su alrededor, hablar con otros, y tomar la iniciativa según la fuerza natural de la persona.

Tabla 2-3

Metodología Ágil	Características
Programación Extrema	• Si los requerimientos del software cambian constantemente. • Si el nivel de problemas que arriesgan el proyecto es alto. • Si el grupo de desarrolladores de software es pequeño. Entre 2 y 12 programadores. • Si lo importante es entregar el software necesitado cuando es solicitado.

Crystal Clear	• Tamaño del proyecto: hasta 6 desarrolladores, típicamente de 3 a 4.
	• Riesgos potenciales del sistema:
	  ○ El dinero es esencial pero no vital.
	  ○ Podría ser ajustable a 8 personas máximo o a sistemas de responsabilidad legal pero no ambos.
	• Tipos de sistemas: Mainframe o Cliente/Servidor, cualquier base de datos, central o distribuida.

Tabla 2-4

ROLES	PRODUCTOS Ó PRACTICAS	
	CRYSTAL CLEAR	XP
PATROCINADOR (SPONSOR)	Declaración de la misión	No es requerido por XP
COORDINADOR (COORDINATOR)	Sucesión de versiones	
	Revisión y cronograma de versiones	El final de cada iteración es una revisión y una versión (Release)
	Lista de riesgos	Es usado como parte de la planificación de versiones, desde entonces ha sido rechazado como redundante
	Estado del proyecto	

Tabla 2-5

DISEÑADOR LIDER (LEAD DESIGNER)	Estructura del equipo	Manager, Coach, customer, developer, control de calidad.
	Diseño del sistema	Tradición-Oral, Crc Cards, propiedad colectiva.
EXPERTO DE NEGOCIO (BUSINESS EXPERT)	Lista de pares Actor-Metas	No está contemplado
	Casos de usos	Historias de Usuarios
	Archivo de requerimientos	Unit Test.
USUARIO EXPERTO (USER EXPERT)	Diseños de pantallas	Diseños de pantalla no son explícitamente requeridos, pero si comúnmente usados
DISEÑADOR-PROGRAMADOR	Diseño de bosquejos y notas	Diseño de bosquejos y notas no son explícitamente requeridos, pero sí comúnmente usados. Las CRC-CARDS son más asociadas con XP.
	Modelo común de objetos (Common Object Model)	Propiedad de código colectivo y código escrito en lenguaje objeto, se establecen como convenciones entre los programadores.
	Código fuente	
	Empaquetado del sistema	
	Migración de código	
	Casos de Prueba	
TESTER	Reportes de pruebas (Defectos)	La prueba fallada es el reporte
WRITTER	Manual del usuario	Los manuales de usuario no se contemplan dentro del marco de trabajo de XP. Un manual es producido como parte de los requerimientos del cliente

Tabla 2-6

2.8 HERRAMIENTAS OPEN SOURCE

Este proyecto utiliza herramientas Open Source traducido como código de fuente abierta, pero el término en ingles es más utilizado y aceptado. No significa solamente la posibilidad de ver el código origen del software a utilizar sino algunas condiciones que lo hacen realmente una alternativa de desarrollo a los productos propietarios o bajo licencias comerciales restrictivas.

Actualmente la Iniciativa Open Source[20] (institución sin fines de lucro) soporta las licencias y certifica los productos bajo el titulo de Open Source.

Bruce Perens escribió el primer bosquejo del documento "The Debian Free Software Guidelines", En 1997 creo "Open Source Definition" : "Definición de Open Source" , traducida por Diego Rodrigo de Open Source argentina[21].Esta definición incluye las características que un software Open Source debe incluir:

- Un producto Open Source debe incluir:
- Libre distribución
- Código fuente abierta
- Permite trabajos derivados del código original
- Integridad del código fuente del autor.
- La licencia no debe discriminar personas o grupos
- La licencia no debe restringir el campo de uso del software en un campo específico de aplicación.

[20] http://opensource.org/index.php

[21] http://www.opensource.org.ar/es-osd.html

- Los derechos concedidos deben ser aplicados a todas las personas a quienes se redistribuya el programa, sin necesidad de obtener una licencia adicional
- La licencia no debe ser específica para un producto.

La licencia no debe imponer restricciones sobre otro software que es distribuido junto con el. Por ejemplo, la licencia no debe insistir en que todos los demás programas distribuidos en el mismo medio deben ser software Open-Source.
Como ejemplo de licencias del tipo Open Source tenemos las siguientes: Las licencias GNU GPL, BSD, X Consortium, y Artistic ,MPL cumple con la definición.

2.8.1 COMPILADORES Y ENTORNOS DE DESARROLLO

Las herramientas Open Source son nativas de los entornos GNU/Linux, ya que es este sistema operativo en el cual se ha producido la mayor parte de programas Open Source. Se basan fundamentalmente en los compiladores GNU, la familia del gcc, compilador para C. Por otra parte, no solo se limita a un compilador de C / C++ / Objective C sino que también incluye compiladores de Pascal, Fortran, ADA y Java. El compilador de Java genera bytecodes o código nativo directamente.
Veamos ahora algunos entornos integrados para desarrollo en entornos Open Source[22]:

- KDevelop junto a Qt-Designer forman un entorno de desarrollo integrado (IDE) que permite desarrollar aplicaciones con interfaz gráfica de usuario de manera sencilla. Tiene todas las características que incluye Visual C para Windows, salvo que utiliza la librería Qt en lugar de MFC con lo que se pueden crear

[22] ver listado de software en el anexo C con los sitios web .

interfaces de usuario simplemente arrastrando y soltando los elementos de la librería sobre el entorno de diseño.

- El proyecto GNOME nació en 1997, apoyado por la Free Software Foundation, como parte de su proyecto GNU. En los últimos años, apoyado por grandes empresas de la industria del software, en su plataforma de desarrollo ha dado gran importancia a la creación de herramientas, bibliotecas o tecnologías de reutilización de componentes que facilitan el desarrollo de sistemas de última generación. Variedad de lenguajes para el desarrollo, desde C con el que se pueden sacar los mejores rendimientos a una máquina, de programación orientada a objetos como C++; para la programación de sistemas distribuidos de sofware real time como Erlang; lenguajes de futuro como el C# estandarizado por ECMA para la creación de una alternativa a Java; Python para la creación de prototipos.
- El IDE NetBeans es un entorno de desarrollo, una herramienta para programadores para escribir, compilar, corregir errores y para ejecutar programas. Está escrito en Java - pero puede servir de soporte a cualquier otro lenguaje de programación. Existe también un número enorme de módulos para extender el NetBeans IDE. El NetBeans IDE es un producto libre y gratuito sin restricciones de utilización, fue creado por Sun MicroSystems que fundó el proyecto Open Source NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos. El proyecto NetBeans incluye también la plataforma NetBeans una fundación modular y extensible usada como una estructura de base para crear aplicaciones de escritorio. Sociedades de desarrollo especializadas proporcionan plug-ins que se integran fácilmente en

la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones. El código fuente está disponible para la reutilización de acuerdo con la Licencia Pública de Sun (SPL).

- **Eclipse** es un entorno independiente de la plataforma, de código abierto, para crear aplicaciones clientes de cualquier tipo. La primera y más importante aplicación que ha sido realizada con este entorno es el afamado IDE Java y compilador, que fueron utilizados en el desarrollo propio de Eclipse. Eclipse fue creado originalmente por IBM. Ahora lo desarrolla la Fundación Eclipse, una organización independiente sin ánimo de lucro que fomenta una comunidad de código abierto y un conjunto de productos complementarios, capacidades y servicios. El entorno de desarrollo (IDE) de Eclipse emplea módulos (en ingles plug-in) para proporcionar toda su funcionalidad. El mecanismo de módulos permite que el entorno de desarrollo soporte otros lenguajes además de Java. Por ejemplo, C/C++. Existen módulos para añadir un poco de todo desde telnet hasta soporte a bases de datos.

2.8.2 BASE DE DATOS Y SOFTWARE AUXILIARES

A medida que la era de la información continúa creciendo, de igual modo lo hace la necesidad del almacenamiento de datos que facilite el acceso flexible y rápido a la información y como propuestas para solventar esta necesidad constantemente escuchamos a cerca sistemas de base de datos comerciales con licencia de uso de software propietario. Oracle, IBM y Microsoft son algunas de las empresas mas importantes que ofrecen este tipo de software y han captado el mayor porcentaje de usuarios en este mercado tal como lo muestra la figura 2-8, esto se debe a que

sus sistemas administradores de bases de datos relacional (RDBMS) son reconocidos por proporcionar muchas ventajas y soporte a los usuarios en cuanto a rapidez en las transacciones (INSERSERT, SELECT, UPDATE, DELETE), seguridad en el respaldo e integridad de los datos, facilidad en la importación y exportación para poder migrar los datos de un RDBMS a otro, sin embargo es importante mencionar que hay muchas empresas y usuarios que no pueden tener acceso a estos RDBMS ya que el costo monetario de sus licencias es muy elevado, y sin duda alguna se puede asegurar que en muchos países las empresas no adoptan un sistema informático debido a que no tienen los recursos para invertir en software propietario de alto costo; pero en realidad actualmente el poco recurso monetario ya no es una limitante para invertir en un eficiente RDBMS debido al surgimiento de diversas herramientas de software Open Source para RDBMS las cuales pueden llegar a proporcionar las ventajas y soporte ya antes mencionados del software propietario, incluso nuevas características y esencialmente sin costo alguno ya que en su mayoría son distribuidos bajo la licencia GPL y BSD.

Razones por las cuales usar un RDBMS Open Source:

- Bajo costo: Son una buena alternativa para no invertir miles de dólares en una licencia para un RDBMS.
- Flexibilidad: Existe la disponibilidad del código fuente para poder modificar un RDBMS y hacer mejoras y funcionamiento personalizado.
- Confiabilidad: existen organizaciones y comunidades de desarrolladores trabajando para que estos RDBMS puedan implementar la "ACID TEST"

- Fácil de usar: Estos RDBMS son simples, intuitivos y poseen abundante documentación.

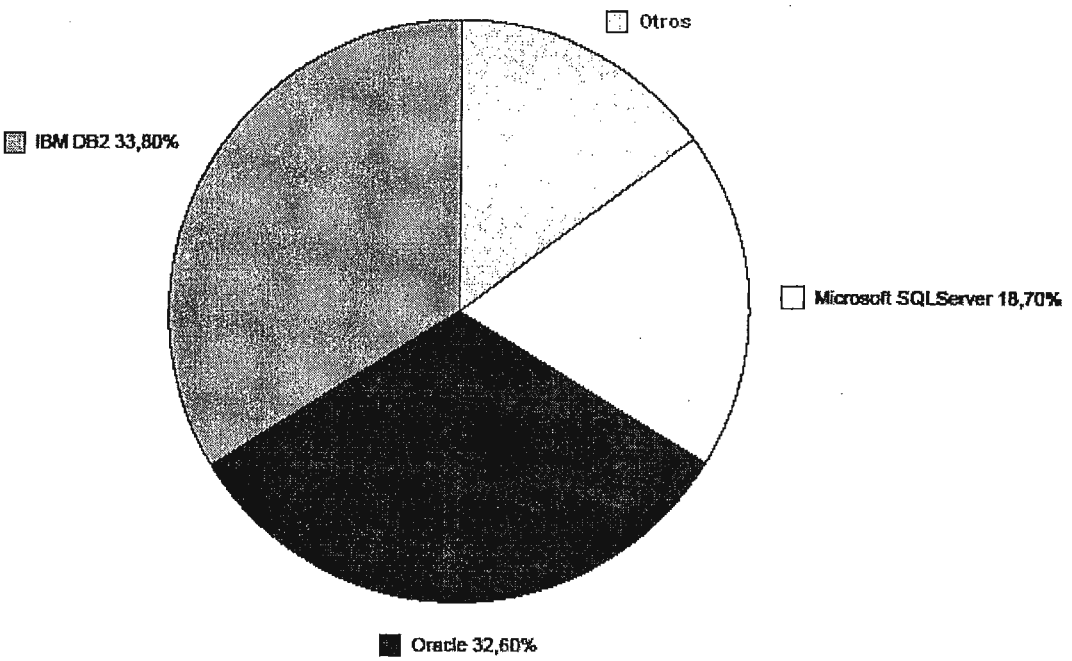


Figura 2-7[23]

Los RDBMS Open Source utilizados actualmente son:

- MySQL
- PostgreSQL
- Ingres
- Firebird

MySQL

MySQL es uno de los servidores de bases de datos relacionales más utilizado, desarrollado y proporcionado por MySQL AB. MySQL AB es una empresa cuyo

[23] FirebirdQL : Il DataBase opne e affidabile. Alessandro Carichini alexsoft@riminilug.it

negocio consiste en proporcionar servicios en torno al servidor de esta base de datos. MySQL Surge con la intención de utilizar mSQL para conectar tablas utilizando rutinas rápidas de bajo nivel (ISAM). Sin embargo después de algunas pruebas se llegó a la conclusión de que mSQL no era lo suficientemente rápido o lo suficientemente flexible para las necesidades de MySQL AB.

MySQL consiste de un sistema cliente/servidor que se compone de un servidor SQL multihilo, varios programas clientes y bibliotecas, herramientas administrativas, y una gran variedad de interfaces de programación (APIs). Se puede obtener también como una biblioteca multihilo que se puede enlazar dentro de otras aplicaciones para obtener un producto más pequeño, más rápido, y más fácil de manejar. Una de las razones para el rápido crecimiento de popularidad de MySQL, es que se trata de un producto Open Source, y por lo tanto, va de la mano con este movimiento.

MySQL es una parte clave de LAMP (Linux, Apache, MySQL, PHP / Perl / Python), un grupo de software empresarial Open Source de rápido crecimiento. Las compañías cada vez más están usando LAMP como una alternativa al grupo de software propietario debido a su bajo costo.

Ventajas que MySQL proporciona a las empresas:

Reduce los costos de licencia al 90%

Recorta el tiempo muerto del sistema al 60%.

Aminora el desgaste del hardware al 70%

Reduce los costos del soporte de administración e ingeniería al 50%.

Entregas soluciones menos complicados que complementan las bases de datos corporativas existentes.

Algunos de los clientes de MySQL son:

Alcatel	Lufthansa
AOL	NASA
The Associated Press	Nortel
Caterpillar	NYSE
Cox Communications	Omaha Steaks
DaimlerChrysler	Sabre Holdings
Dow Jones	Siemens
EarthLink	Suzuki
Enercon	Texas Instruments
Ericsson	Time Inc.
Google	UPS
Hoover's Online	US Census Bureau
Hewlett-Packard	Lufthansa
Lucent	NASA

Firebird

Firebird es una base de datos de código abierto, basada en la versión 6 de interbase cuyo código fue liberado por Borland y de la cual corrige múltiples errores. Además, su código fue totalmente reescrito en c++.

Borland compro Ashton-Tate en octubre de 1991, obteniendo tambien InterBase en ese momento en la version. Fue asi como Internase vino a ser un producto open source en Julio del 2000. A partir de entonces se desarrollo nuevos proyectos entre ellos Firebird el cual se encuentra en la versión 1.5.2.

Ventajas de Firebird

Arquitectura multigeneracional.- Aunque este nombre nos suene un poco raro y complicado, es la forma en que Firebird administra la concurrencia en las

actualizaciones en los datos, así como el manejo de las transacciones. Nos asegura que no habrán bloqueos a nivel de página de datos ni de registro, ya que cada vez que se abre una transacción, Firebird genera una copia de los datos para ese usuario. Esto asegura que las transacciones nunca necesitarán bloquear los registros en uso, por lo que los usuarios que están consultando datos no bloquearán a los que están escribiendo datos

Triggers, o disparadores.- Firebird cuenta con una de las implementaciones de triggers más completas comparada con otras bases de datos. Los triggers permiten la realización de acciones cada vez que se agrega, modifica o elimina un registro. De esta manera, podemos implantar reglas de negocio desde el nivel de la base de datos, por ejemplo, actualizar totales en una tabla cuando se modifican datos en otra, o llevar un registro de acciones mediante una bitácora. Todo esto al nivel de base de datos, lo que asegura que los triggers se ejecutarán siempre, no importando desde dónde se acceda a la base de datos.

Procedimientos almacenados.- Funcionan de manera similar a los triggers, con la diferencia de que pueden ser ejecutados de manera independiente a las acciones que se ejecuten sobre los registros. También permiten regresar los datos mediante una orden SELECT de SQL, como si fuera una tabla, de tal manera que se pueden realizar complejas consultas y devolverlas como una tabla, simplificando los procedimientos de consultas en los programas clientes hechos con Delphi/C Builder.

Integridad referencial.- Permite establecer reglas de integridad entre tablas, para que no violen los principios de las relaciones entre tablas maestro-detalle.

Seguridad integrada.- Firebird mantiene su lista de usuarios, y es necesario que se registre el usuario cada vez que se conecta a la base de datos. Además, se pueden asignar permisos independientes de acceso, modificación inserción y eliminación a por tabla a cada usuario. Lenguaje SQL compatible con SQL 92.- La implementación de SQL en Firebird es una de las más completas, incluso mejor que algunos de sus competidores de código abierto, como Postgres y MySQL, lo que asegura que se pueden realizar complejas consultas anidadas, y utilizar funciones de conversión como CAST o extracción de las partes de las fechas, con EXTRACT.

Funciones definidas por el usuario (UDF).- Cuando se requiere de funciones no integradas al SQL, o relacionadas con matemáticas, manejo avanzado de fechas, etc., Firebird permite la creación y uso de funciones externas, que al ser registradas pueden utilizarse en combinación con SQL. Esto asegura la extensibilidad de la base de datos. Además, existen algunas bibliotecas de funciones definidas por el usuario (UDF) las cuales pueden ser utilizadas sin costo alguno.

CARACTERISTICAS	FIREBIRD	INTERBASE	MYSQL	POSTGRESQL	ORACLE	MS SQL
Versión	1.5	7.1	4.1	7.4	10g	**2000**
Licencia	IDPL OPEN SOURCE	COMERCIAL	GPL - OPEN SOURCE / COMERCIAL	BSD - OpenSource	COMERCIAL	**COMERCIAL**
Hardware	Pentium	Pentium	Pentium	Pentium	Pentium 166	**PII / K6**
Plataforma	Linux/Win32/ Solaris/ OsX /FreeBSD	Linux/Win32/ Solaris	Linux/Win32/	Unix / Win32*	Linux/WinNT/ Solaris/AIX	**WinNT**
Lenguaje	SQL-92 Entry*	SQL92 Entry*	SQL92 Intermedio	SQL92 Full*	SQL99	**SQL99**
procedimientos almacenados	SI	SI	NO	SI	SI	**SI**
Triggers	SI	SI	NO	SI	SI	**SI**
Views	SI	SI	NO	SI	SI	**SI**
Transacciones	ACID	ACID	SI*	ACID	ACID	**ACID**
SMP/64bits	**SI***	**SI**	**NO**	**SI**	**SI**	**SI**

Tabla 2-7 Comparación de bases de datos Open Source[24]

[24] [24] FirebirdQL : Il DataBase opne e affidabile. Alessandro Carichini alexsoft@riminilug.it

CAPÍTULO III

RECURSOS METODOLOGICOS Y HERRAMIENTAS UTILIZADAS

3.1 HERRAMIENTAS OPEN SOURCE

3.1.1 COMPILADORES Y ENTORNOS DE DESARROLLO

El entorno de desarrollo utilizado es NetBeans versión 4.0 el cual es incluido como extra al descargar el paquete para JDK 1.5

NetBeans incluye además la herramienta de gestión de proyectos ANT el cual se encarga de la definición de CLASSPATH y la generación de archivos de clase ejecutables (bytecodes) para java.

3.1.2 BASE DE DATOS

La Base de datos utilizada en este proyecto es la versión 1.5.2 de Firebird. La cual implementa las características de un RDBMS y compatibilidad con el estándar SQL 92.

3.1.3 SOFTWARE AUXILIARES

- FlameRobin. Es un administrador de base de datos Firebird, se puede obtener gratuitamente y cuenta con una licencia libre llamada Initial Developer's Public License la cual permite la modificación y distribución de código fuente, actualmente la versión es tipo Alfa es decir continua en desarrollo. Es de fácil uso y de poco gasto de memoria y espacio en disco que puede ser distribuida como auxiliar a la base de datos,

- JayBird 1.5.4 es el driver JDBC utilizado para conectar la aplicación con la base de datos Firebird. Es de libre distribución y es compatible con el estándar de JDBC 2.0, y que incluye característica aun no compatible de la base de datos Firebird pero que se contemplan como mejoras en versiones futuras. Cabe mencionar que este driver funciona con las versiones 6.0 y 7.0 de la base de datos Internase que es comercial.
- JasperReports. Es una librería para generar reportes desde aplicaciones java utilizando XML, es distribuida bajo la licencia LGPL, la cual permite ser incluida como parte de este proyecto, esta librería permite ejecutar los reportes hechos en la herramienta auxiliar iReport.
- iReport-Designer for JasperReports es una herramienta de generación de reportes hecha 100% en java, su autor la distribuye como GNU GPL.

3.2 HERRAMIENTAS DE DISTRIBUCION GRATUITA

Algunas herramientas utilizadas en este proyecto no son de Open Source pero mantienen la característica de ser libres de cargo por el uso y la distribución, algunas son creadas por un grupo de programadores y otros por personas altruistas y usuarios. En este apartado mencionamos algunas de ellas.

- JDK de Sun. Para este proyecto se ha utilizado el compilador de java bajo la licencia de Sun, este es incluido en el paquete JDK 1.5, el cual corresponde a la última versión estable que se puede obtener de forma gratuita en Internet.
- IBEasy+. Es la herramienta para la administración de la base de datos utilizada en este proyecto. Desarrollada por Marc Grange ofrece las facilidades para administrar la base de datos Firebird. La versión utilizada es IBEasy 1.3.

3.3 ELEMENTOS APLICADOS DE XP Y CRYSTAL CLEAR

El primer paso para aplicar las metodologías es la conformación del equipo y definición de roles con lo que se establecen los siguientes productos y practicas.[25]

- Revisión y Cronograma de Versiones (CC y XP)
- Estado del proyecto (CC)
- Estructura del Equipo (CC)
- Diseño del Sistema (CC)
- Historias de Usuario (XP)
- Diseño de pantallas (CC)
- Diseño de bosquejos y notas(CC)
- Convenciones entre los programadores (XP)
- Casos de Prueba(CC y XP)
- Manual de Usuario (CC)
- Programación en Pares(XP)

[25] CC son practicas de Crystal Clear

XP se refiere a practicas de Extreme Programming

CAPÍTULO IV

PROCESO DE DESARROLLO

4.1 LOS CICLOS DE DESARROLLO

Para la descripción del proceso seguido para crear software, se han adoptado muchos esquemas lineales desde los años setenta hasta el año 2000, pero aunque estos son procesos llamados iterativos e incrementales, los esquemas representan la dependencia entre distintas actividades, por ejemplo Análisis de requerimientos, diseño, codificación, prueba, entrega. Esto creo un poco de confusión al percibirse la idea que son una serie de pasos ejecutados consecutivamente, esto se resuelve al describir los ciclos en los que las personas interactúan y generan nuevos productos y no la dependencia entre las actividades, ya que estas se repiten e incrementan en cada iteración.[26]

Según Cockburn los ciclos utilizados por Crystal Clear son los siguientes.

- Ciclo de Proyecto(Project Cycle)
- Ciclo de Entrega(Delivery Cycle)
- Ciclo de Iteración(Iteration Cycle)
- Ciclo de Semana y Dia
- Ciclo de Integración(Integration Cycle)
- Episodio de Desarrollo.

[26] Crystal Clear A Human-Powered Methodology for Small Teams. Alistair Cockburn. Capitulo 4 .

En el proceso de desarrollo llevado a cabo durante este proyecto se utilizaron los diversos ciclos antes mencionados, ya que estos proponen un tiempo y actividades a realizar por los miembros del equipo.

4.2 CICLO DE PROYECTO PARA SIPOP

4.2.1 Conformación del Equipo

El equipo esta formado por cuatro personas que ejecutan los diversos roles propuestos por la metodología Crystal Clear. Cada uno participa elaborando diferentes productos según se describe a continuación.

1. Roberto Calles
2. Rafael Paniagua
3. Gilberto Abarca,Carlos Mateus, Ana Ruth, Nery Bonilla, Evelyn de Sermeño
4. Manuel Orellana,

EQUIPO (TEAM)	PERSONA	ROL 1	SUB ROL 1	SUB ROL 2 C
	1	Coordinador (Coordinator)	Diseñador-Programador (Designer-Programmer)	Examinador (Tester) Escritor (Writer)
	2	Diseñador Lider(Lead Designer)	Diseñador-Programador (Designer-Programmer)	TESTER WRITER
	3 A	BUSINES EXPERT	Usuario Experto (User Expert)	
	4 B	Patrocinador Ejecutivo(Executiv eSponsor)	Diseñador Lider(Lead Designer) Usuario Experto (User Expert)	

Donde

A: es la persona que esta rotando en el equipo según el modulo a desarrollar, para la primera entrega, son el bodeguero y el encargado de compras del departamento. Los productos elaborados por ellos se obtienen en base a la información que proporcionan. En un proyecto no académico serian ellos quienes los elaboren, en este proyecto los productos esperados son realizados por los miembros 1 y 2.

B: Corresponde al asesor de este proyecto el cual asume diferentes roles durante el desarrollo; en un proyecto no académico esta persona es la que patrocina el proyecto y conoce las reglas de negocio.

C. Los roles descritos en esta columna se activan por demanda en el cronograma del plan de versión o entregas.

4.2.2 Requerimientos del Proyecto

La técnica utilizada para la obtención de requerimientos es llamada historias de usuario. Las presentadas en este documento corresponden a la primera versión que se utilizo para elaborar el plan del proyecto. Cada historia se clasifico según las áreas del departamento involucradas en el sistema, y que expresaban un conjunto de funcionalidades comunes. Luego de este análisis se detallan los requerimientos para desarrollar la primera versión del sistema, cada uno de ellos se obtuvo a través de una o mas historias de usuario. Al final de este apartado se encuentra el detalle de dichos requerimientos clasificados como: propuestos por los usuarios, y recomendados por el equipo de desarrollo.

Historias Relacionadas Al Modulo De Bodega

Usuario:

Evelin de Sermeño - Directora del Departamento de O y P. de la UDB

Historia de Usuario 1. Tenemos la necesidad de implantar un sistema informático que facilite el control del uso de materiales, herramientas, equipo y aparatos que están en bodega, de manera que lo que realmente se utilice sea reflejado fielmente en el sistema

Historia de Usuario 2. .Que esta información se encuentre protegida sin que pueda ser modificada ni consultada por personas no autorizadas de manera que solo el encargado de bodega tenga acceso a los registros de existencia de materiales, salidas y entradas a bodega de estos y todo lo relacionado a bodega, que ninguna otra persona tenga acceso a manipular esta información aunque el sistema esté en red o compartido a otros usuarios.

Historia de Usuario 3. Es importante que el sistema pueda generar respaldos o copias de todos los movimientos de materiales efectuados y así poder recuperarlos y consultarlos posteriormente si es necesario.

Historia de Usuario 4. El sistema debe de manejar un control de existencias mínimas y máximas que ayude a decidir si se compra o no un material específico. Seria bueno si el sistema pudiera mostrar alguna alerta o mensajes que informe sobre las existencias mínimas y máximos.

Usuario:

Gilberto Abarca - Técnico - Docente de 1er año.

Historia de Usuario 5. La conversión de unidades de medida en la presentación de los materiales es necesaria por que algunos materiales el proveedor los vende con una unidad de medida y en la bodega la salida de este material se registra con otra unidad de medida.

Historia de Usuario 6. Me ayudaría mucho si el sistema pudiera calcularme la cantidad total de material que se utilizó para una práctica de laboratorio específica de un ciclo y año determinado considerando el número de alumnos y así obtener el costo del material que servirán como estimados para los siguientes años.

Usuario:

Nery Bonilla - Encargado de bodega.

Historia de Usuario 7. El sistema debería proporcionar un control sobre la cantidad total de materiales utilizados para fabricar un producto específico para una persona o institución específica.

Historia de Usuario 8. Se necesita un sistema que registre la salida del material en bodega tanto en cantidades enteras como fraccionarias según las presentaciones o unidades de medida.

Historias Relacionadas Al Modulo De Expedientes

Usuario

Ing. Heinz Trebbin - Director del proyecto regional UDB-GTZ.

Historia de Usuario 9. Se necesita un sistema que me permita obtener reportes sobre estadísticas de los usuarios (pacientes), reportes que incluyan datos sobre el origen de la patología y la incidencia de esta dentro del grupo familiar, así como la situación socio-económica del usuario, nivel de educación, situación laboral y aspectos generales como dirección del domicilio, teléfono y otros.

Usuario

Srta. Ana Ruth - Asistente Administrativa.

Historia de Usuario 10. Lo que necesito es un sistema que me de apoyo con las búsquedas de los expedientes de los pacientes, para saber quien lo patrocina, saber quien esta atendiendo al paciente (Técnico encargado de fabricar la prótesis), cuantas visitas ha realizado, si ya ha cancelado el pago correspondientes a los servicios proporcionados por el departamento..

Historias Relacionadas Al Modulo De Producción

Usuario:

Evelin de Sermeño - Directora del Departamento de O y P. de la UDB

Historia de Usuario 11. Creo que es indispensable un módulo de control de calidad el cual permita ver registros de las requisiciones de materiales aprobadas y autorizadas y así poder controlar si el numero de requisiciones para la fabricación de un determinado producto ha sido excedido y si se necesita otra requisición para esta misma producción el sistema deberá exigir un nivel de autorización superior.

Usuario

Carlos Zelaya - Técnico-Docente 3er Año.

Historia de Usuario 12. Me gustaría que el sistema me proporcionara datos estadísticos o registros de los alumnos de 3er año considerando la cantidad y tipo de productos que han elaborado así como también un registro de los aspectos a evaluar para proporcionar una calificación a sus trabajos elaborados y tomados como prácticas. Esto me seria muy útil al momento de asignar los trabajos de las órtesis o prótesis que las personas solicitan ya que dependiendo del record del alumno: experiencia y habilidad para fabricar un determinado producto así se seleccionará el más apto para fabricar

Usuario

Gilberto Abarca - Técnico - Docente de 1er año.

Historia de Usuario 13. Si se tuviera un control de la cantidad total de material que se utiliza específicamente para las reparaciones o fabricaciones exigidas por la garantía del producto que se le ofrece a los usuarios seria favorable.

Historias Relacionadas Al Aspecto Decorativo, Grafico Y Visual

Usuario:

Ing. Heinz Trebbin - Director del proyecto regional UDB-GTZ.

Historia de Usuario 14. Es importante que el sistema incluya un atractivo visual en su diseño gráfico de tal manera que las diferentes interfaces sean del agrado a los usuarios.

Usuario

Víctor Henríquez- Asesor de computación.

Historia de Usuario 15. El sistema deberá ser fácil de usar, que no tenga procesos o mecanismos complejos, con interfaces intuitivas que puedan ser manejadas por personas con conocimiento básico de la informática.

Historias Relacionadas Al Modulo De Bodega De Herramientas

Usuario:

Nery Bonilla- Encargado de bodega.

Historia de Usuario 16. Seria muy útil que el sistema pudiera registrar los prestamos de las herramientas y equipo realizado por los alumnos, controlara la devolución de estos y generara reportes de personas con herramientas prestadas así como el vencimiento de fecha de préstamo con el fin de facilitar la creación de solvencias de los alumnos con la bodega.

4.2.2.1 Clasificación de Requerimientos

Historia de Usuario.	**Requerimientos propuesto por los usuarios.**
Historia 1	Sistema informático para el control de materiales, herramientas y equipo en bodega
Historia 2	Control de usuarios para el sistema
Historia 3	Generar Respaldados para las operaciones realizadas en la Base de datos.

Historia 4	Control de existencias máximas y mínimas para materiales en bodega
Historia 5	Facilidad para salidas de materiales en diferentes unidades
Historia 6	Cálculo de estimación del material utilizado.
Historia 9	Generar estadísticos de los clientes atendidos por el departamento
Historia 11	Control sobre la cantidad de materiales utilizados para fabricar un producto determinado.
Historia 15	El sistema debe ser fácil de usar, orientado a personas con conocimientos básicos de informática.

Historia 14	Posee una interfaz con atractivo visual, diseño del agrado de los usuarios.

Tabla 4-1

Requerimientos propuesto por el equipo de desarrollo.
• El sistema requiere de infraestructura de red local y un servidor de base de datos que registre las operaciones en el sistema. • Agilizar la entrada de datos en el sistema, orientando la interfaz a un diseño intuitivo y de acceso rápido al digitar. • Rediseñar y definir procesos para el uso efectivo del sistema en la actividades del departamento: ingreso de datos de clientes, técnicos y generación de órdenes de producción y requisiciones. • Clasificar los productos elaborados (Catalogo de productos). • Clasificar los tipos de requisiciones de materiales. • Clasificar las entradas y salidas en bodega de materiales. • Identificar por medio de consultas y reportes la información de: o ¿Qué producto se elabora? o ¿Para quién se elabora dicho producto (cliente)? o ¿Quién esta elaborando (técnico)? o ¿Qué materiales son utilizados (requisiciones)? o ¿Cuándo se entregará?

Tabla 4-2

4.2.3 Descripción del proyecto.

Este proyecto de software esta enfocado en crear un sistema para facilitar el seguimiento de las actividades de producción en el departamento de Órtesis y prótesis. Consta de varios módulos los cuales corresponde a las áreas de trabajo del departamento que se relacionan con la producción: Registro de Clientes, Bodega de Materiales, Préstamo de equipo y herramientas, Asignación de pacientes, Generación de órdenes o pedido de producción.

Durante esta primera versión solo se contempla el modulo para Bodega, Prestamos, Registro de Clientes, pero dejando las bases para continuar con el desarrollo evolutivo del sistema, basado en las metodologías ágiles.

4.2.4 Plan Inicial del Proyecto.

La planificación del proyecto se baso en la técnica blitz planning y el uso de fichas técnicas obtenidas de las historias de usuario.

La estrategia utilizada es Early Victory propuesta en Crystal Clear; la estrategia dice desarrollar lo fácil primero y los complejo después, en este caso se opto por desarrollar el modulo de bodega que era prioritario y las funcionalidades de consulta de inventario que eran menos complejas.

4.2.4.1 Fichas del Blitz Planning

Estas corresponden a una o más historias de usuarios y ayudan a determinar que hacer en el plan de versión. Cada ficha técnica tiene un tiempo estimado por los desarrolladores y que se acuerda según la solicitud de los clientes, los usuarios generalmente ponen un plazo y en caso de no ser viable técnicamente se debe

crear dos historias de usuario y nuevas fichas para ser evaluadas, esto permite tener una estimación del tiempo necesario para la iteración.

Los días contabilizados se refiere a un día típico de 8 horas de trabajo donde no existe mas preocupación que elaborar la tarea asignada, es decir sin interrupciones o distracciones que desvíen del objetivo planteado, en este sentido el día propuesto es un tiempo ideal.(VER ANEXO B)

4.2.4.2 Mapa del Proyecto

Este es un diagrama de dependencia que muestra la estructura del problema y la secuencia para atacarlo. Como puede verse no incluye fechas. Este mapa responde a las preguntas: En que orden entregaremos cada cosa? ¿Y que construiremos después? ¿Cuales son las dependencias entre versiones?

Ver figura 4-1

Los módulos origen son los que permiten alimentar el sistema con información además cada uno incluye sub-módulos con alguna funcionalidad particular.

Para el diagrama siguiente los nodos representan:

Infraestructura. Se refiere al soporte de hardware y la base de datos necesarios para iniciar el desarrollo, algunos de los módulos no requieren estar conectados inicialmente a la base de datos, pero otros como el registro de artículos en bodega si.

Ingreso de Personas. Este modulo es el que ingresa los datos generales de cualquier persona al sistema, ya sea como cliente, técnico o usuario del sistema SIPOP.

Definición de Perfiles. Es parte del modulo de Administración del sistema que determina el nivel de jerarquía en las operaciones del sistema, asignando a las personas registradas sus roles como paciente, técnico y usuario del sistema con los respectivos permisos.

Bodega de Materiales. Registra las operaciones de ingreso y salida de materiales para la producción de Órtesis y prótesis. Aquí se genera mucha de la información para módulos siguientes.

Activo Fijo. Lleva el control de depreciación de los activos del departamento incluyendo maquinaria y equipo.

Prestamos. Aquí se registran las operaciones de préstamo y devolución de equipo para la elaboración de las Órtesis y prótesis. Es un nuevo registro automatizado ya que el programa de bodega original no lo incluye.

Expedientes. Una persona registrada con el perfil de paciente, crea un expediente que sirve como insumo para otros módulos.

Ordenes de Producción. Mantiene el registro y control de los pedidos de Órtesis y prótesis ha ser elaborados, para ello necesita reconocer tres datos importantes, paciente , técnico y requisición de materiales.

Reportes. Son generados solamente si los datos existen previamente, por lo cual dependen de los otros módulos de información en mayor o menor grado.

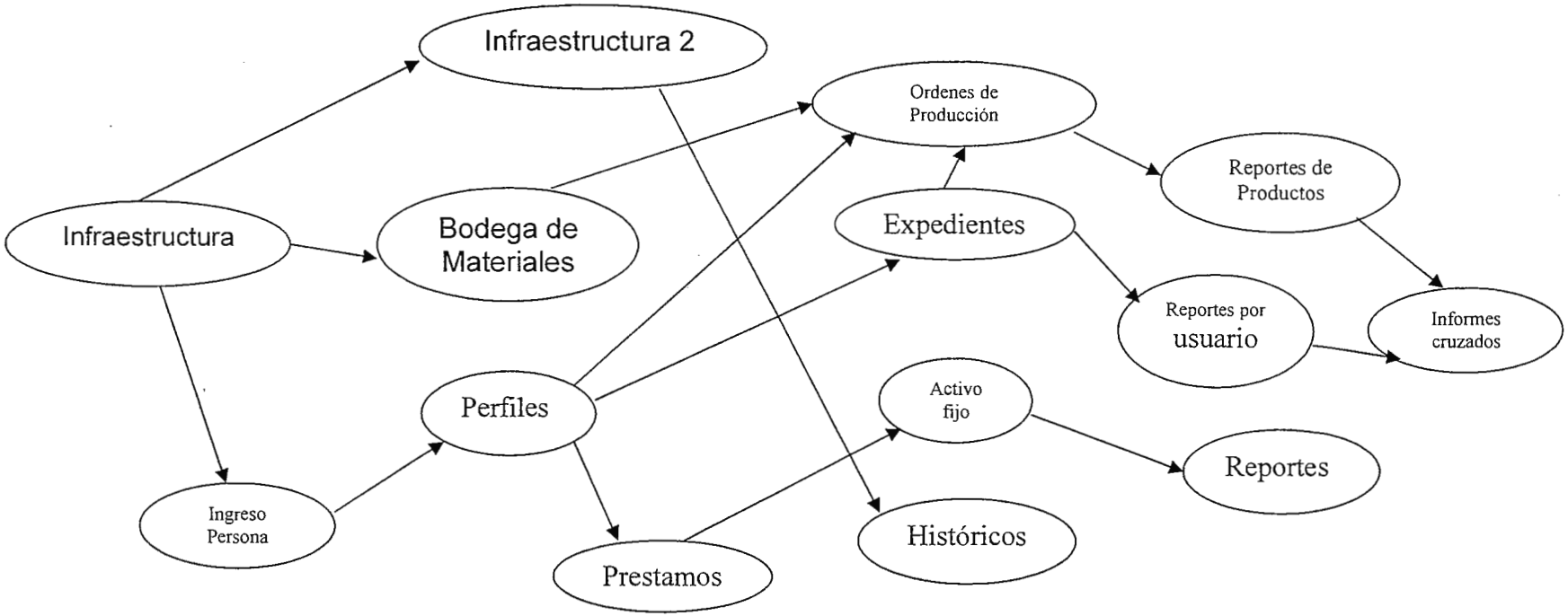

Infraestructura 2
Ordenes de Producción
Reportes de Productos
Infraestructura
Bodega de Materiales
Expedientes
Reportes por usuario
Informes cruzados
Activo fijo
Perfiles
Reportes
Ingreso Persona
Históricos
Prestamos

Figura. 4-1

4.2.4.3 Arquitectura del Sistema

La arquitectura del sistema se describe por medio de un diagrama que es seleccionado por el equipo de desarrollo, expresa la tecnología que se utiliza, como puede observarse describe la mayoría de interfaces y rutas de iteración.

Este modelo incluye una vista de los programas o módulos de software que se instalaran como clientes para acceder a la base de datos común del departamento de Órtesis y prótesis. Note que los módulos no pueden ejecutar información adicional a la que están permitidos, así como las conexiones siempre son vía la red local , asegurando que existe una sola base de datos integrada de la información del proceso de producción.

El modelo representa la arquitectura cliente /servidor ejecutada sobre terminales con sistema operativo Windows.

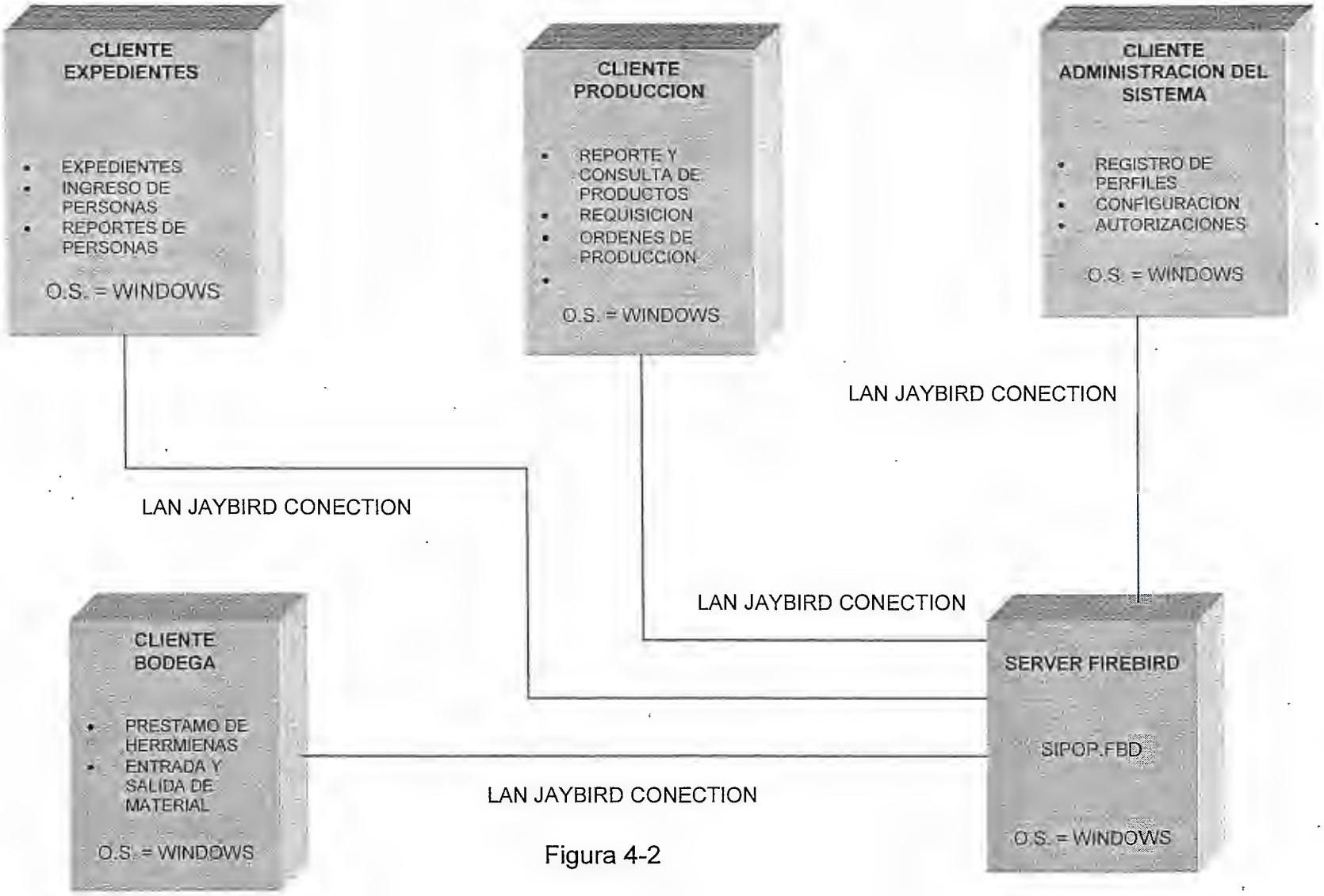

CLIENTE EXPEDIENTES
EXPEDIENTES
INGRESO DE PERSONAS
REPORTES DE PERSONAS
O.S. = WINDOWS
CLIENTE PRODUCCION
REPORTE Y CONSULTA DE PRODUCTOS
REQUISICION
ORDENES DE PRODUCCION
O.S. = WINDOWS
CLIENTE ADMINISTRACION DEL SISTEMA
REGISTRO DE PERFILES
CONFIGURACION
AUTORIZACIONES
O.S. = WINDOWS
LAN JAYBIRD CONECTION
LAN JAYBIRD CONECTION
LAN JAYBIRD CONECTION
CLIENTE BODEGA
PRESTAMO DE HERRMIENAS
ENTRADA Y SALIDA DE MATERIAL
O.S. = WINDOWS
LAN JAYBIRD CONECTION
SERVER FIREBIRD
SIPOP.FBD
O.S. = WINDOWS

Figura 4-2

4.2.4.4 Common Domain Model

Típicamente es un dibujo o esquema de base de datos o diagrama de clases, se utiliza un modelo de entidad relación que describe la base de datos creada para este proyecto. Figuras. 4-3 A, B, C, D

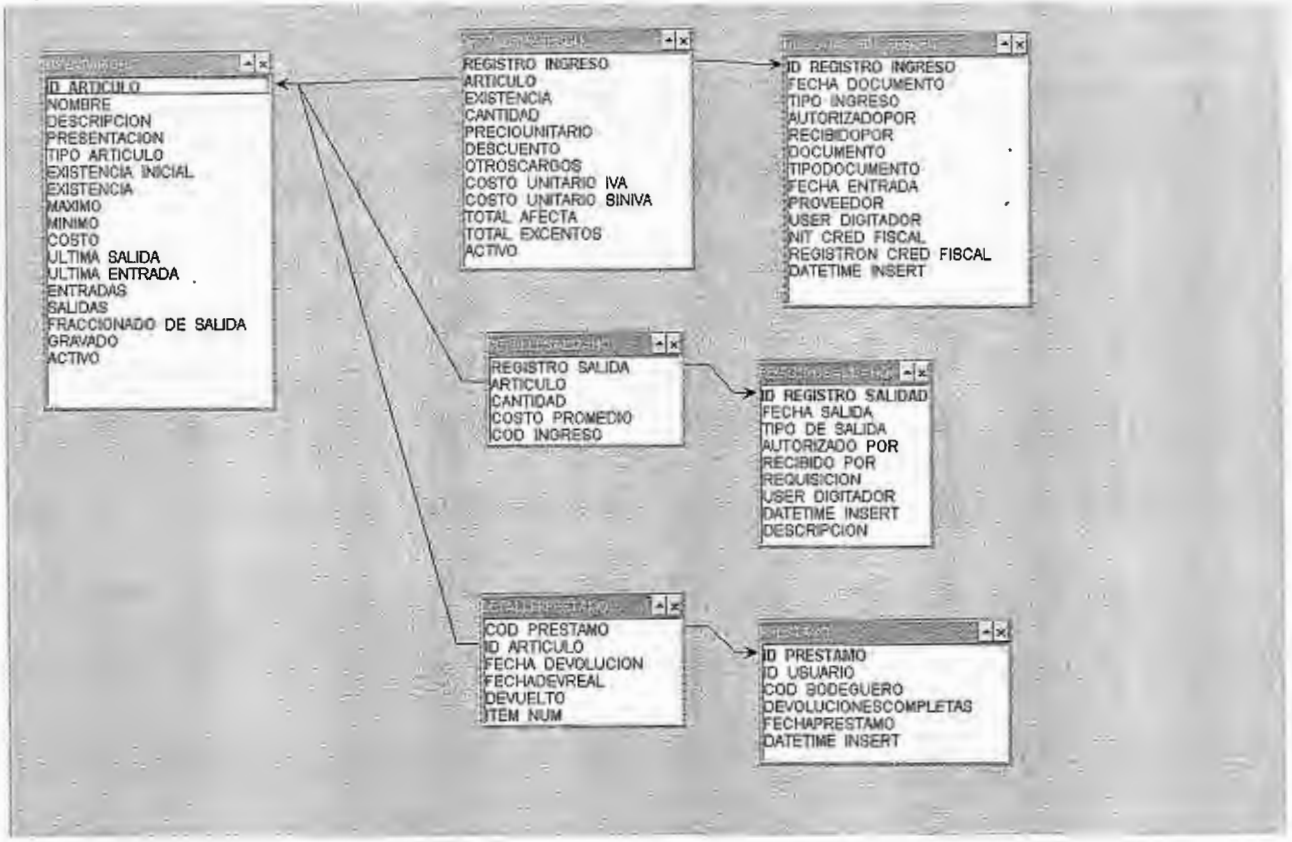


Figura. 4-3 A

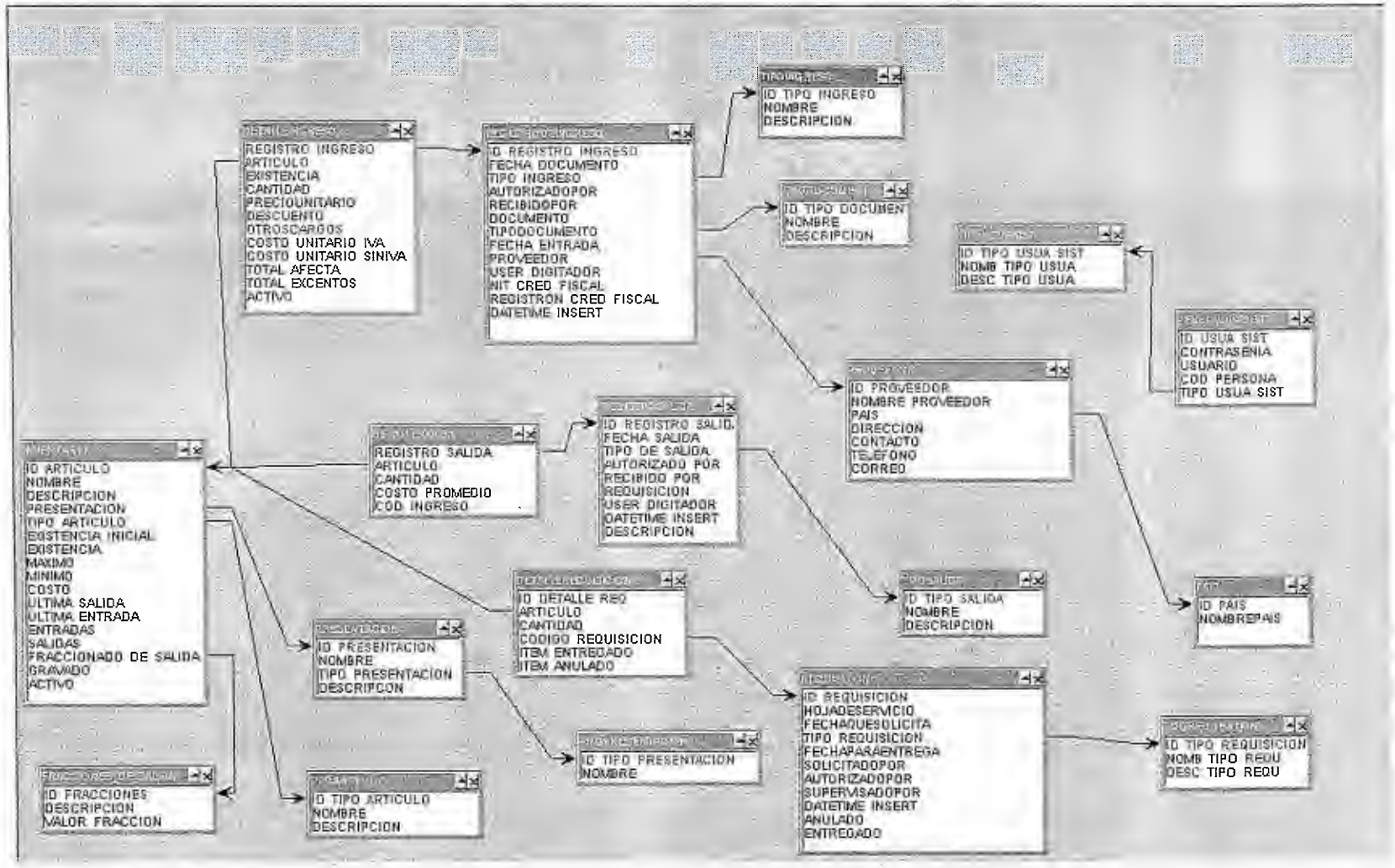

ID TIPO INGRESO
NOMBRE
DESCRIPCION
REGISTRO INGRESO
ARTICULO
EXISTENCIA
CANTIDAD
PRECIOUNITARIO
DESCUENTO
OTROSCARGOS
COSTO UNITARIO IVA
COSTO UNITARIO SINIVA
TOTAL AFECTA
TOTAL EXCENTOS
ACTIVO
ID REGISTRO INGRESO
FECHA DOCUMENTO
TIPO INGRESO
AUTORIZADOPOR
RECIBIDOPOR
DOCUMENTO
TIPODOCUMENTO
FECHA ENTRADA
PROVEEDOR
USER DIGITADOR
NIT CRED FISCAL
REGISTRON CRED FISCAL
DATETIME INSERT
ID TIPO DOCUMEN
NOMBRE
DESCRIPCION
ID TIPO USUA SIST
NOMB TIPO USUA
DESC TIPO USUA
ID USUA SIST
CONTRASENIA
USUARIO
COD PERSONA
TIPO USUA SIST
ID PROVEEDOR
NOMBRE PROVEEDOR
PAIS
DIRECCION
CONTACTO
TELEFONO
CORREO
ID REGISTRO SALID
FECHA SALIDA
TIPO DE SALIDA
AUTORIZADO POR
RECIBIDO POR
REQUISICION
USER DIGITADOR
DATETIME INSERT
DESCRIPCION
REGISTRO SALIDA
ARTICULO
CANTIDAD
COSTO PROMEDIO
COD INGRESO
ID ARTICULO
NOMBRE
DESCRIPCION
PRESENTACION
TIPO ARTICULO
EXISTENCIA INICIAL
EXISTENCIA
MAXIMO
MINIMO
COSTO
ULTIMA SALIDA
ULTIMA ENTRADA
ENTRADAS
SALIDAS
FRACCIONADO DE SALIDA
GRAVADO
ACTIVO
ID TIPO SALIDA
NOMBRE
DESCRIPCION
ID PAIS
NOMBREPAIS
ID DETALLE REQ
ARTICULO
CANTIDAD
CODIGO REQUISICION
ITEM ENTREGADO
ITEM ANULADO
ID PRESENTACION
NOMBRE
TIPO PRESENTACION
DESCRIPCON
ID REQUISICION
HOJADESERVICIO
FECHAQUESOLICITA
TIPO REQUISICION
FECHAPARAENTREGA
SOLICITADOPOR
AUTORIZADOPOR
SUPERVISADOPOR
DATETIME INSERT
ANULADO
ENTREGADO
ID TIPO REQUISICION
NOMB TIPO REQU
DESC TIPO REQU
ID TIPO PRESENTACION
NOMBRE
ID FRACCIONES
DESCRIPCION
VALOR FRACCION
ID TIPO ARTICULO
NOMBRE
DESCRIPCION

Figura. 4-3 B

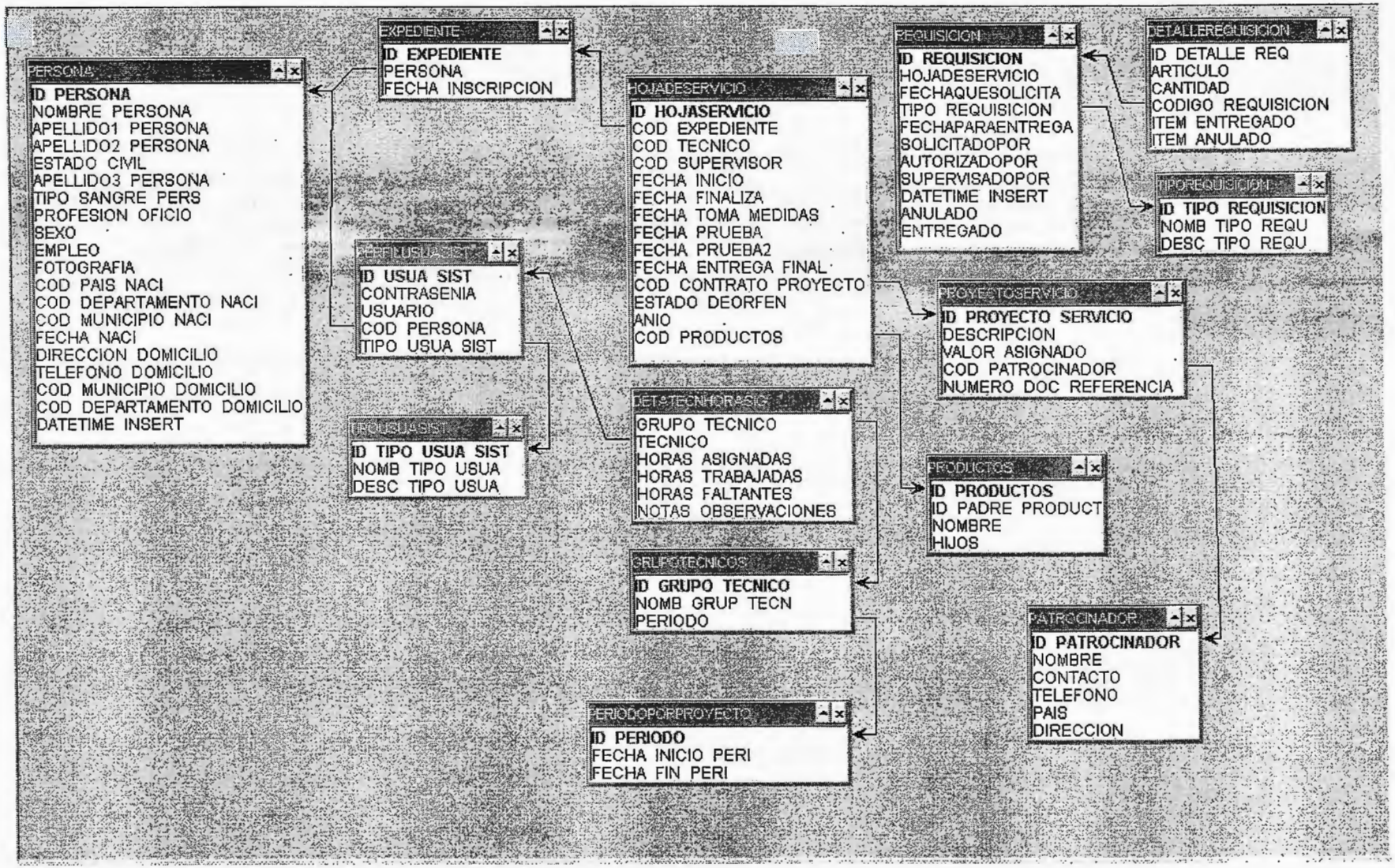

PERSONA
ID PERSONA
NOMBRE PERSONA
APELLIDO1 PERSONA
APELLIDO2 PERSONA
ESTADO CIVIL
APELLIDO3 PERSONA
TIPO SANGRE PERS
PROFESION OFICIO
SEXO
EMPLEO
FOTOGRAFIA
COD PAIS NACI
COD DEPARTAMENTO NACI
COD MUNICIPIO NACI
FECHA NACI
DIRECCION DOMICILIO
TELEFONO DOMICILIO
COD MUNICIPIO DOMICILIO
COD DEPARTAMENTO DOMICILIO
DATETIME INSERT
EXPEDIENTE
ID EXPEDIENTE
PERSONA
FECHA INSCRIPCION
PERFILUSUASIST
ID USUA SIST
CONTRASENIA
USUARIO
COD PERSONA
TIPO USUA SIST
TIPOUSUASIST
ID TIPO USUA SIST
NOMB TIPO USUA
DESC TIPO USUA
HOJADESERVICIO
ID HOJASERVICIO
COD EXPEDIENTE
COD TECNICO
COD SUPERVISOR
FECHA INICIO
FECHA FINALIZA
FECHA TOMA MEDIDAS
FECHA PRUEBA
FECHA PRUEBA2
FECHA ENTREGA FINAL
COD CONTRATO PROYECTO
ESTADO DEORFEN
ANIO
COD PRODUCTOS
DETATECNHORASIG
GRUPO TECNICO
TECNICO
HORAS ASIGNADAS
HORAS TRABAJADAS
HORAS FALTANTES
NOTAS OBSERVACIONES
GRUPOTECNICOS
ID GRUPO TECNICO
NOMB GRUP TECN
PERIODO
PERIODOPORPROYECTO
ID PERIODO
FECHA INICIO PERI
FECHA FIN PERI
REQUISICION
ID REQUISICION
HOJADESERVICIO
FECHAQUESOLICITA
TIPO REQUISICION
FECHAPARAENTREGA
SOLICITADOPOR
AUTORIZADOPOR
SUPERVISADOPOR
DATETIME INSERT
ANULADO
ENTREGADO
DETALLEREQUISICION
ID DETALLE REQ
ARTICULO
CANTIDAD
CODIGO REQUISICION
ITEM ENTREGADO
ITEM ANULADO
TIPOREQUISICION
ID TIPO REQUISICION
NOMB TIPO REQU
DESC TIPO REQU
PROYECTOSERVICIO
ID PROYECTO SERVICIO
DESCRIPCION
VALOR ASIGNADO
COD PATROCINADOR
NUMERO DOC REFERENCIA
PRODUCTOS
ID PRODUCTOS
ID PADRE PRODUCT
NOMBRE
HIJOS
PATROCINADOR
ID PATROCINADOR
NOMBRE
CONTACTO
TELEFONO
PAIS
DIRECCION

Figura. 4-3 C

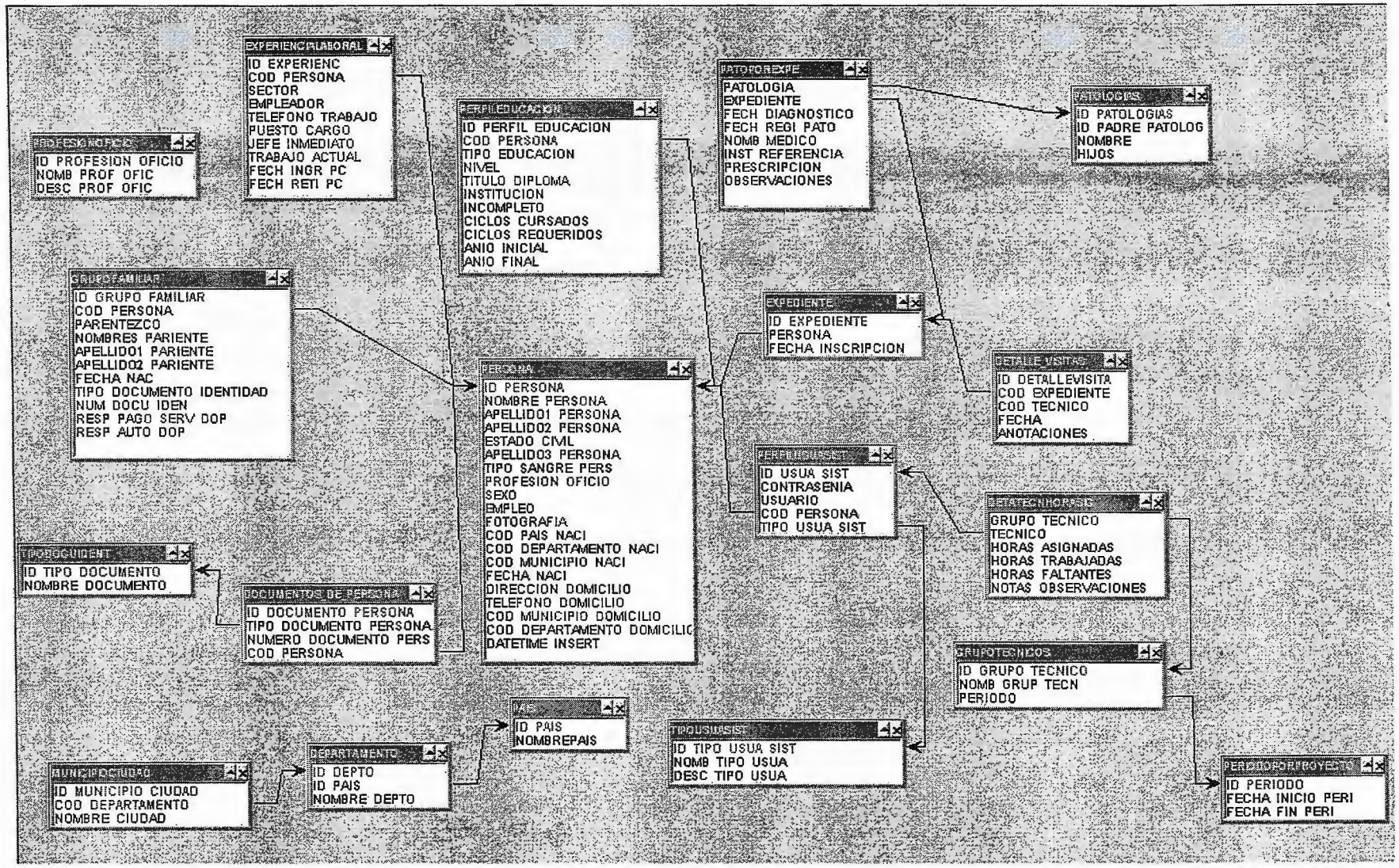

EXPERIENCIALABORAL
ID EXPERIENC
COD PERSONA
SECTOR
EMPLEADOR
TELEFONO TRABAJO
PUESTO CARGO
JEFE INMEDIATO
TRABAJO ACTUAL
FECH INGR PC
FECH RETI PC
PERFILEDUCACION
ID PERFIL EDUCACION
COD PERSONA
TIPO EDUCACION
NIVEL
TITULO DIPLOMA
INSTITUCION
INCOMPLETO
CICLOS CURSADOS
CICLOS REQUERIDOS
ANIO INICIAL
ANIO FINAL
PATOPORPERSONA
PATOLOGIA
EXPEDIENTE
FECH DIAGNOSTICO
FECH REGI PATO
NOMB MEDICO
INST REFERENCIA
PRESCRIPCION
OBSERVACIONES
PATOLOGIAS
ID PATOLOGIAS
ID PADRE PATOLOG
NOMBRE
HIJOS
PROFESIONOFICIO
ID PROFESION OFICIO
NOMB PROF OFIC
DESC PROF OFIC
GRUPOFAMILIAR
ID GRUPO FAMILIAR
COD PERSONA
PARENTEZCO
NOMBRES PARIENTE
APELLIDO1 PARIENTE
APELLIDO2 PARIENTE
FECHA NAC
TIPO DOCUMENTO IDENTIDAD
NUM DOCU IDEN
RESP PAGO SERV DOP
RESP AUTO DOP
EXPEDIENTE
ID EXPEDIENTE
PERSONA
FECHA INSCRIPCION
DETALLEVISITAS
ID DETALLEVISITA
COD EXPEDIENTE
COD TECNICO
FECHA
ANOTACIONES
PERSONA
ID PERSONA
NOMBRE PERSONA
APELLIDO1 PERSONA
APELLIDO2 PERSONA
ESTADO CIVIL
APELLIDO3 PERSONA
TIPO SANGRE PERS
PROFESION OFICIO
SEXO
EMPLEO
FOTOGRAFIA
COD PAIS NACI
COD DEPARTAMENTO NACI
COD MUNICIPIO NACI
FECHA NACI
DIRECCION DOMICILIO
TELEFONO DOMICILIO
COD MUNICIPIO DOMICILIO
COD DEPARTAMENTO DOMICILIO
DATETIME INSERT
PERFILUSUASIST
ID USUA SIST
CONTRASENIA
USUARIO
COD PERSONA
TIPO USUA SIST
DETATECNHORASIG
GRUPO TECNICO
TECNICO
HORAS ASIGNADAS
HORAS TRABAJADAS
HORAS FALTANTES
NOTAS OBSERVACIONES
TIPODOCUIDENT
ID TIPO DOCUMENTO
NOMBRE DOCUMENTO
DOCUMENTOS DE PERSONA
ID DOCUMENTO PERSONA
TIPO DOCUMENTO PERSONA
NUMERO DOCUMENTO PERS
COD PERSONA
GRUPOTECNICOS
ID GRUPO TECNICO
NOMB GRUP TECN
PERIODO
PAIS
ID PAIS
NOMBREPAIS
TIPOUSUASIST
ID TIPO USUA SIST
NOMB TIPO USUA
DESC TIPO USUA
PERIODOPORPROYECTO
ID PERIODO
FECHA INICIO PERI
FECHA FIN PERI
DEPARTAMENTO
ID DEPTO
ID PAIS
NOMBRE DEPTO
MUNICIPIOCIUDAD
ID MUNICIPIO CIUDAD
COD DEPARTAMENTO
NOMBRE CIUDAD

Figura. 4-3 D

4.3 CICLO DE ENTREGA PROYECTO PARA SIPOP

4.3.1 Plan de Primera entrega de SIPOP

Para la primera entrega (Delivery) la planificación esta muy cercana a la planificación del proyecto y puede consistir en adicionar las fechas a las actividades del mapa de proyecto, sin embargo este ciclo puede incluir uno o mas ciclos de iteración. Además del plan de versión puede servir como estatus del proyecto, ya que muestra el avance o el atraso en el desarrollo. Para SIPOP el cronograma de la primera entrega o versión se, muestra en la tabla 4-3 (Consulte el anexo para contenido de Ficha técnica).

Fechas	Actividades/
08/01/2005 hasta 15/01/2005	Planificación de Versión e iteraciones Bodega-Prestamos
17/01/2005 18/01/2005	Ficha 6,Ficha 3
18/01/2005 hasta 26/01/2005	Ficha 1 Ficha 2,
26/01/2005 hasta 30/01/2005	Ficha 15, Ficha 16
31/01/2005 1/02/2005	Ficha 18
02/02/2005	Ficha 19
03/02/2005 hasta 09/02/2005	Pruebas de Aceptación ENTREGA(DELIVERY)
10/02/2005 11/02/2005	Ficha 4,
15/02/2005	Ficha 7,Ficha 8,Ficha5
12/02/2005 19/02/2005	Ficha 10 ,Ficha 12
20/02/2005	Pruebas de Aceptación ENTREGA(DELIVERY)

Tabla 4-3

4.3.2 Re-calibración del plan de versión

Para las entregas siguientes se necesita obtener un acuerdo entre el equipo de desarrollo y los clientes sobre cuales historias de usuario que se llevaran a cabo en la siguiente versión, dado los resultados obtenidos en la primera entrega el equipo podría encontrar información importante que modifique la planificación previa.

4.3.3 Ritual de conclusión

Esta es una sesión con el propósito de hacer una reflexión utilizando la técnica de reflexión workshop, en la metodología se recomienda retroalimentar el trabajo realizado con los usuarios, el equipo de desarrollo. Aquí el equipo hace las preguntas a si mismo: ¿Qué nos agrada y queremos mantener?, ¿Que nos gustaría hacer diferente?

Para evaluar el sistema el equipo observa a los usuarios utilizando el producto. Esta es la manera en que se puede mejorar en el proceso de desarrollo evaluando en cada momento, la manera en que el equipo realiza su trabajo y los resultados obtenidos frente a los usuarios.

4.4 CICLO DE ITERACION

4.4.1 Planeación de la Iteración

Una vez hecha la planificación del ciclo de Entrega, este determina cuantas iteraciones posibles tendrá, en el caso de la primera iteración permite a los diseñadores-programadores ponerse en sintonía y aplicar algunas prácticas para mejorar su trabajo.

4.4.2 Actividades diarias de Integración

Se refiere a que el código desarrollado por los miembros del equipo se prueba en conjunto y se genera los ejecutables, de tal manera que se detecten errores de cualquier tipo, entre mas frecuente sea la integración mas rápido se corrigen los errores.

En el caso de SIPOP se utilizo la técnica de XP programación en pares que también permite mantener control de errores, las integraciones de código se realizaron diariamente.

Aunque Crystal sugiere los pruebas automatizados no son obligatorios, en SIPOP las pruebas fueron diseñadas en el IDE y realizadas manualmente por los programadores.

4.4.3 Ritual de conclusión

El equipo toma un tiempo para revisar su forma de trabajo, aspectos como su relación con los patrocinadores y usuarios, la comunicación, la forma de obtener requerimientos, las convenciones de codificación. Este es un excelente medio para mejorar los hábitos de trabajo, poner en común las dificultades y experiencias. En nuestro caso particular, SIPOP ha tenido dos iteraciones que han concluido en reflexiones sobre como mejorar nuestras formas de trabajo, eso es un beneficio adicional de la metodología.

4.5 EPISODIO DE DESARROLLO

4.5.1 Programación en pares y programación lado a lado

Para el desarrollo del proyecto SIPOP se han aplicado dos técnicas: programación en pares que propone XP y lado a lado que propone Crystal Clear. Estas técnicas han sido de mucha utilidad debido a que SIPOP ejecuta una arquitectura Cliente-Servidor la cual exige hacer pruebas en simultaneo es decir que lo se actualiza y funciona en el Servidor debe probarse en el cliente para asegurar su funcionalidad en equipos remotos.

Para el caso específico del proyecto SIPOP los siguientes ejemplos ilustran mejor las técnicas mencionadas.

Programación lado a lado.

- Conexión a la base de datos: En un inicio los primeros esfuerzos fueron enfocados en lograr la conectividad de la base datos Firebird con la aplicación desarrollada en el lenguaje JAVA para luego desplegar la información de la base de datos en controles java específicamente en componentes SWING que son utilizados para construir interfaces del tipo formulario (JFrame), implicando esta situación dos tareas diferentes pero altamente relacionadas se utilizó la técnica de programación lado a lado (Side by Side) para que de los dos desarrolladores que conforman el equipo del proyecto uno investigara y hiciera pruebas de conectividad del lado del servidor mientras que el otro hiciera pruebas de despliegue de datos con la conectividad conseguida utilizando componentes SWING del lenguaje JAVA.

- Validación de entrada y consulta de datos: Uno de los puntos de interés para los desarrolladores de la aplicación SIPOP ha sido la manera en que se le permite al usuario el ingreso de los datos, para que estos sean consistentes con las restricciones en el diseño de la base de datos en Firebird, esto se traduce en pasar los datos capturados en JAVA a un formato adecuado que sea aceptado por los tipos de campos creados en Firebird o en algunos casos lograr conversiones directas de los datos para transformarlos a tipos aceptados por la base de dato. Un caso concreto de esta situación es la validación de entrada de fechas en la que por un lado se establece la seguridad de que la fecha sea coherente con el calendario y además que permita al usuario ingresarla en el formato simple (dd-mm-yyyy) para que luego este formato sea convertido a formato de texto aceptado por el SQL de Firebird (yyyy-mm-dd) y luego hacer la conversión del tipo de dato Date que maneja la librería de JAVA java.util.Date con el tipo de dato Date que maneja el SQL de Firebird. Ahora bien la manera en que esto se implementa en el desarrollo del SIPOP es que mientras que uno de los dos desarrolladores está haciendo pruebas de código JAVA para ingresar datos validos de fecha e insertarlos a registros en la base de datos Firebird el otro simultáneamente esta haciendo el proceso inverso, desplegando los datos a un formulario JAVA (JFrame) consultando los datos fecha ingresados por su compañero programador en la base de datos Firebird haciendo las conversiones respectivas.

Programación en pares

- Definición de Controles. Dado el problema de personalizar y adaptar los controles básicos de Java que el IDE proporciona, era necesario crear nuestros

propios eventos y determinar condiciones sobre un grupo de controles en el formulario; La programación en pares permitió que la experiencia de cada uno se complementara para resolver el problema, del aporte para diseñar el evento en el formulario y luego el algoritmo para validar los controles han permitido crear un código que puede ser reutilizado y que ha sido probado por ambos programadores. Un ejemplo de esto es la consulta de materiales en Bodega que puede hacerse desde varios JCheckBox los cuales permiten seleccionar diversos criterios de búsqueda y crear a partir de la selección una consulta SELECT de SQL que es enviada a la clase que implementa la interfase con la base de datos.

4.5.2 Convenciones entre programadores

Como parte de los episodios es definir las normas para realizar el código y tener un modelo común del sistema a desarrollar. Esto se hace mas importante cuando se trabaje en programación en pares ya que cada programador posee un estilo propio para realiza su trabajo. Crystal es flexible en cuanto que permite que cada diseñador-programador mantenga su estilo trabajo pero recomienda poner un modelo común para obtener mejores resultados.

Las convenciones entre programadores para este proyecto incluyen:

Estilo de codificación Java: Los componentes Swing han mantenido el nombre estándar del IDE que los genera, pero además se le agrega un nombre descriptivo manteniendo la convención de identificadores propuesta por Sun. Ejemplo: jFrameBodega, jTextFieldCódigo, jTable1Detalles.

Se elaboran Diseños de la interfaz de usuario (UI) antes de elaborar código sobre ellas. Esto es que para cada formulario de Java generado por el IDE los programadores han definido su comportamiento y apariencia.

Los nombres de las tablas creadas en la base de datos y los campos han sido creados en base a un acuerdo, el cual puede leerse en el anexo D.

El uso de una misma estructura de proyecto, en este caso se utiliza la estructura de NetBeans 4.0 (Ver Figura 4-4) que se basa en la herramienta ANT la cual separa los códigos fuentes y los ejecutables en carpetas distintas, archivos de librería y paquetes jar de java, así como una carpeta para pruebas (test).

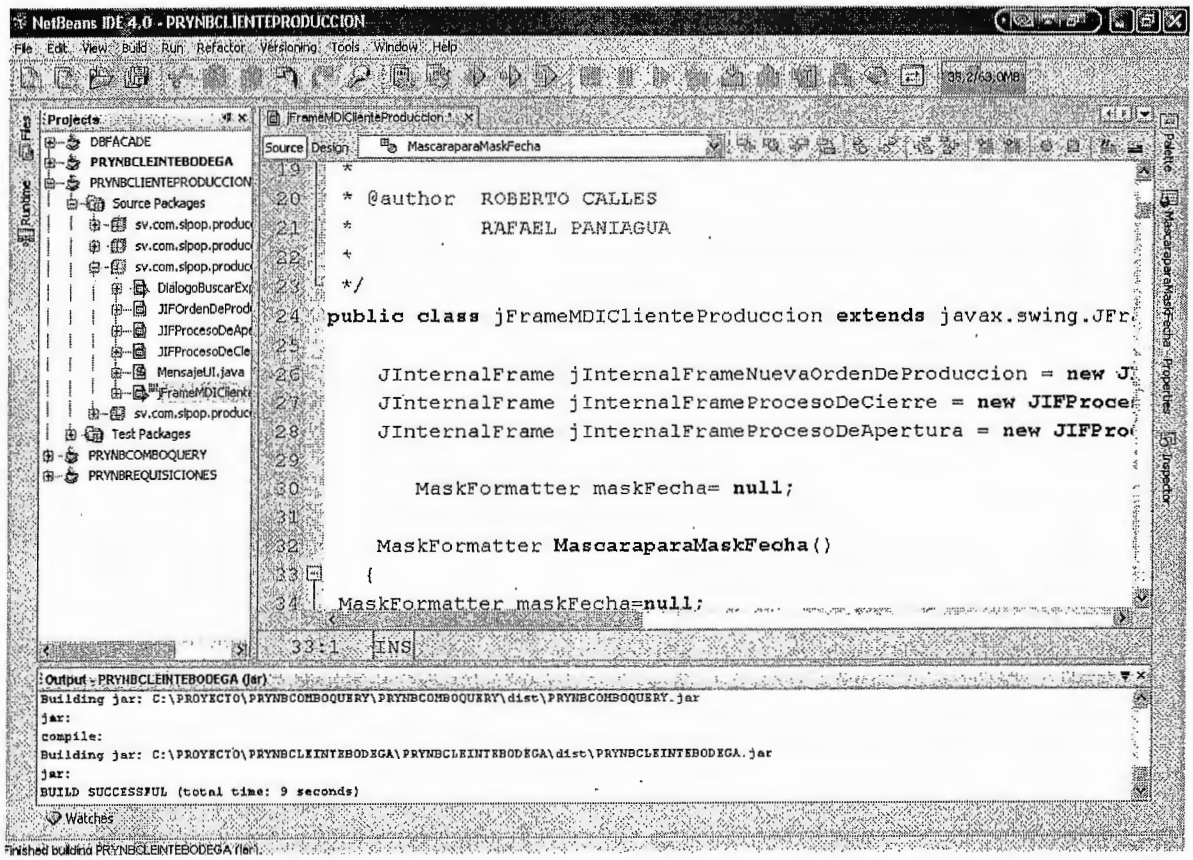


Figura 4-4

Para lograr diseñar una interfaz de usuario (UI) de fácil manejo y comprensión y a la vez obtener visualmente una apariencia moderna y decorada que permita interactuar de manera intuitiva con el sistema (SIPOP), Se ha utilizado la paleta de componentes de interfaz de usuario SWING (Ver figura 4-5) que es una de las herramienta que proporciona el IDE (NETBEANS) y es la que nos facilita la construcción de Formularos y otras clases visuales java, conocidos como contenedores, este nombre se de debe a que dentro de estos se pueden alojar o contener otros componentes visuales tales como componentes para entrada y despliegue de texto , combos, listados, tablas o grids, casillas verificadoras, botones, botones de opción, estructuras de árbol, cuadros de diálogo , en fin una gama diversa para ser utilizados tanto para introducir datos como para retornarlos.

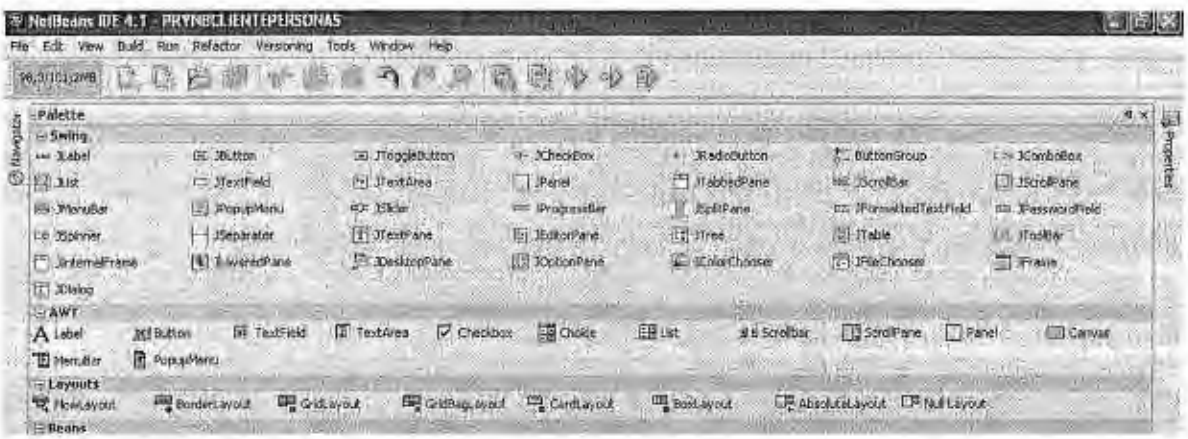


Figura 4-5

La figura 4-6 muestra una interfaz en tiempo de diseño, construida desde el IDE NetBeans, está interfaz es utilizada en el modulo cliente de Bodega para ingresar

artículos. El lenguaje java y la POO permiten personalizar el comportamiento original de estos componentes SWING, así como también su aspecto gráfico.

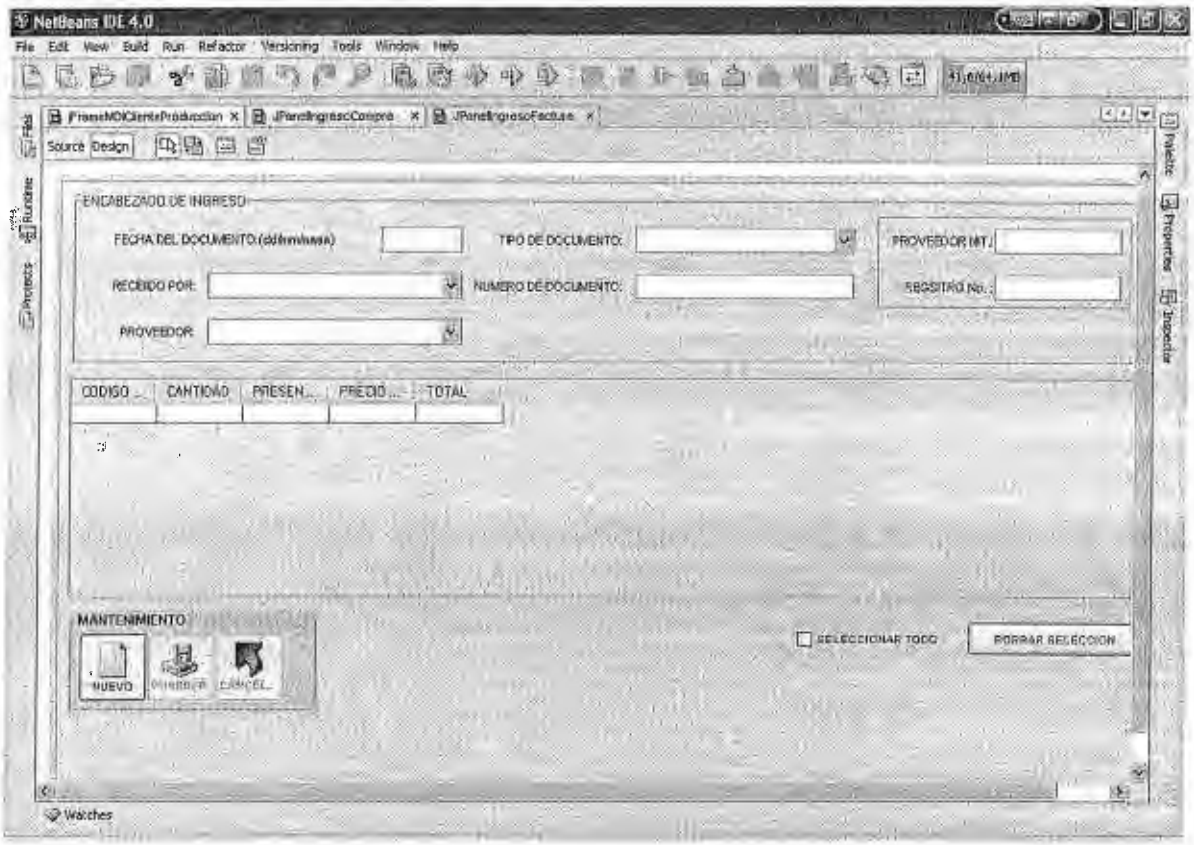


Figura 4-6

4.6 RESULTADOS DE LA PRIMERA ENTREGA DE SIPOP

A continuación se muestran algunos de los formularios elaborados para la aplicación SIPOP correspondientes a las aplicaciones clientes instaladas en la primera versión.

Las interfaces del SIPOP son tipo MDI, las cuales facilitan la consulta de los datos logrando tener diversas ventanas abiertas por medio de los componentes swing Jframe, JDesktopPane y JInternalFrame.

4.6.1 INTERFAZ CLIENTE BODEGA

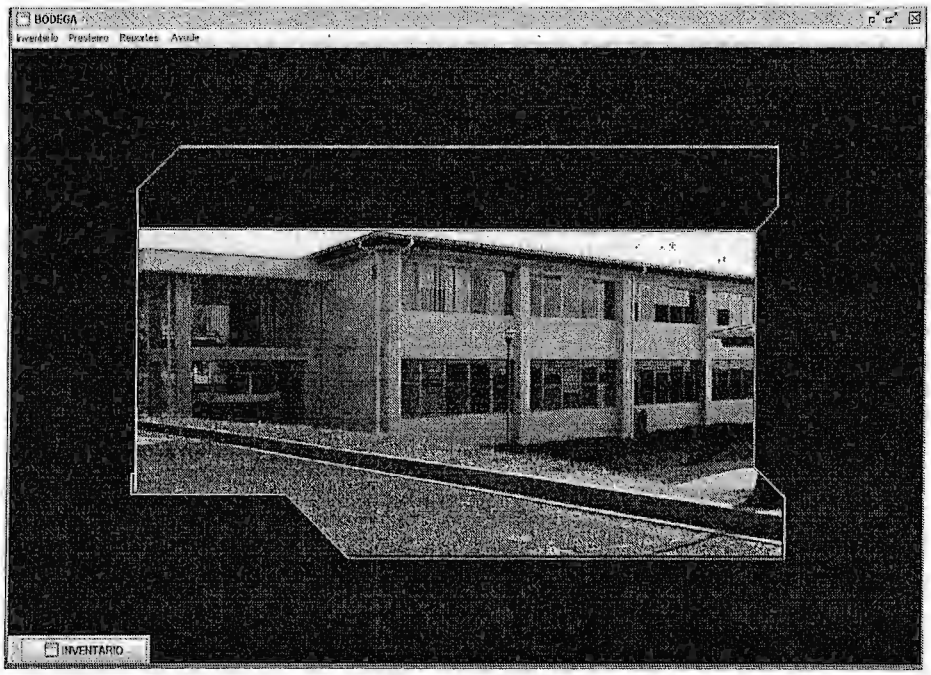


Figura. 4-7

Esta es una muestra de las aplicaciones MDI utilizadas para las aplicaciones clientes SIPOP.

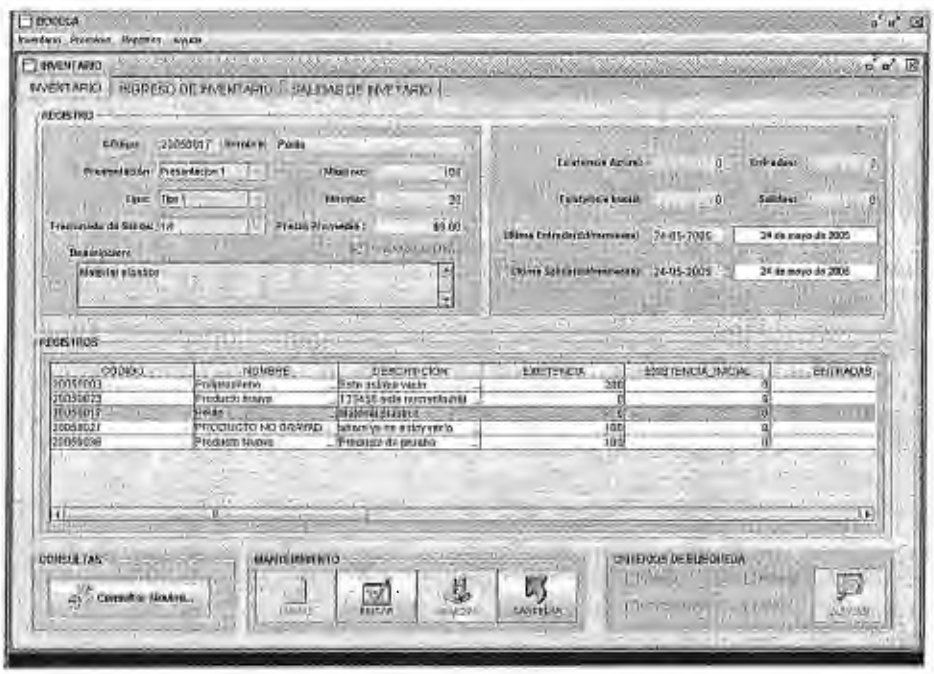


Figura.4-8

Formulario de ingreso principal para artículos en bodega de materiales

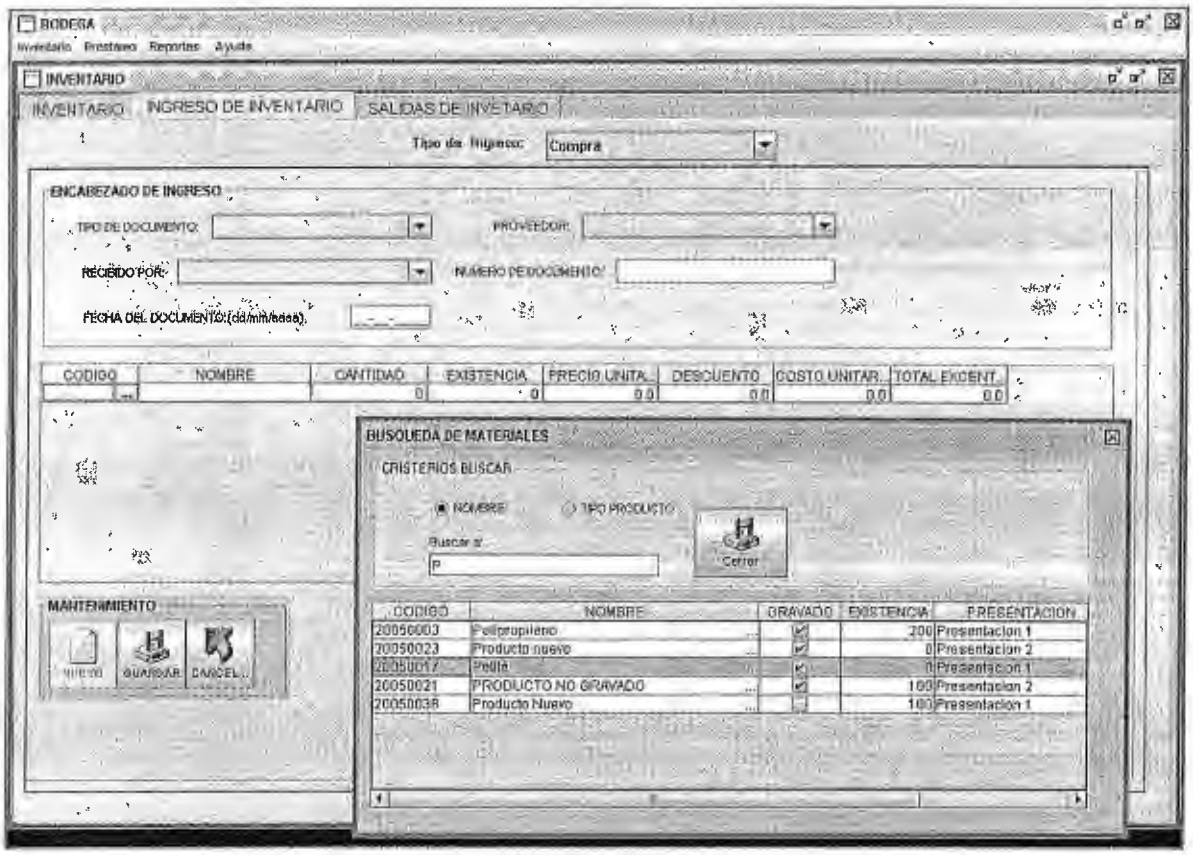


Figura. 4-9

En la figura 4-9 tenemos una captura de pantalla del Ingreso en Bodega. En la tabla de detalles de ingreso de artículos la entrada de datos se da través de una búsqueda de artículos registrados previamente, evitando el ingreso equivocado de datos. Las búsquedas de los artículos pueden ser realizadas por el nombre o por el tipo de este. Esta técnica de búsquedas es utilizada en la mayoría de consultas para registros existentes como pueden ser registros de personas, perfiles, expedientes, requisiciones, ordenes de producción.

Una de las ventajas que brindan los componentes SWING es la facultad de incorporarles iconos e imágenes como puede observarse en la mayoría de los botones, esto da como resultado interfaces más comprensibles y de fácil manejo.

La figura 4-9 además muestra cómo los datos se logran registrar de forma tabular para agilizar los procesos en el área de bodega en el Departamento de Órtesis y Prótesis de la Universidad Don Bosco

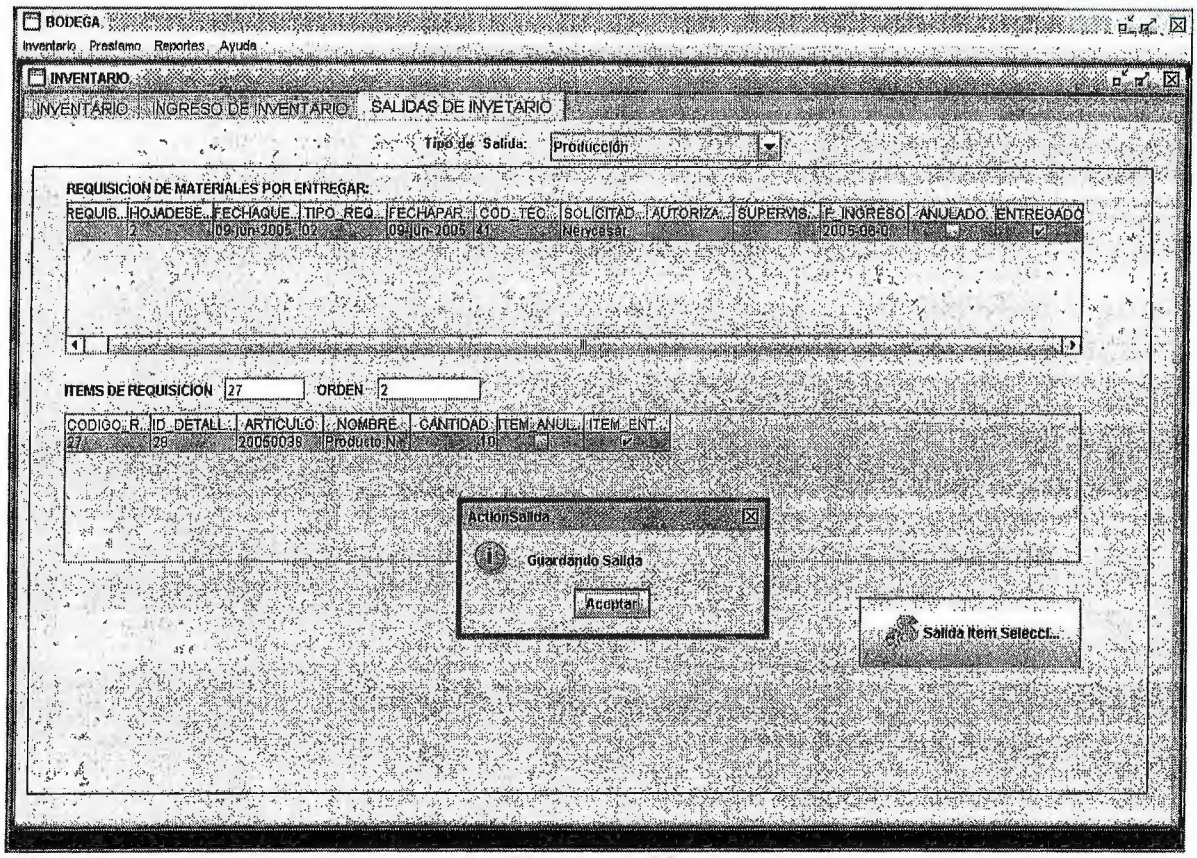


Figura. 4-10

Salida de producción. Los materiales fueron solicitados previamente por medio de una requisición y se dieron de baja en el inventario a través del tipo de salida producción (Figura 4-10). Los tipos de salida se establecen mediante la selección en el componente Combo el cual muestra una lista generada por una consulta SQL de los tipos registrados en la base de datos.

Una de las ventajas que posee este formulario es que el usuario solo debe de consultar las requisiciones registradas previamente y aceptarlas o rechazarlas por

medio de un clic sobre la columna correspondiente de la tabla o grid. En caso de ser aceptadas las cantidades de material solicitado se descuentan automáticamente del inventario del producto seleccionado

Otro modulo integrado en bodega es el mostrado en las figuras: Figura 4-11 y Figura 4-12 que corresponden al préstamo de equipo y herramientas.

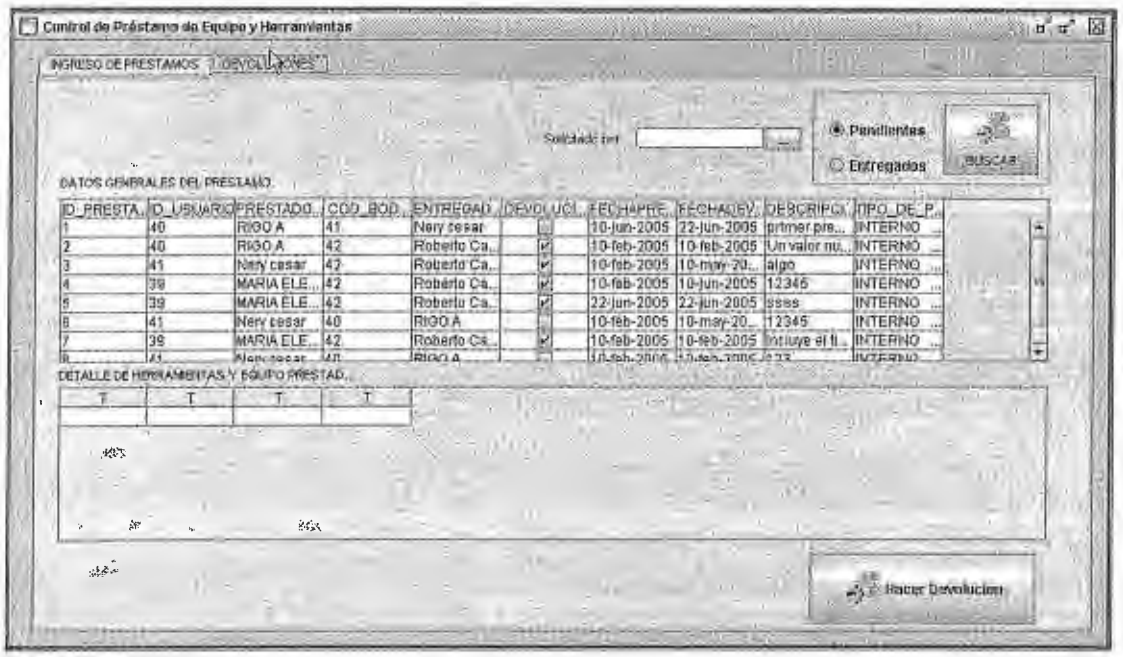


Figura 4-11

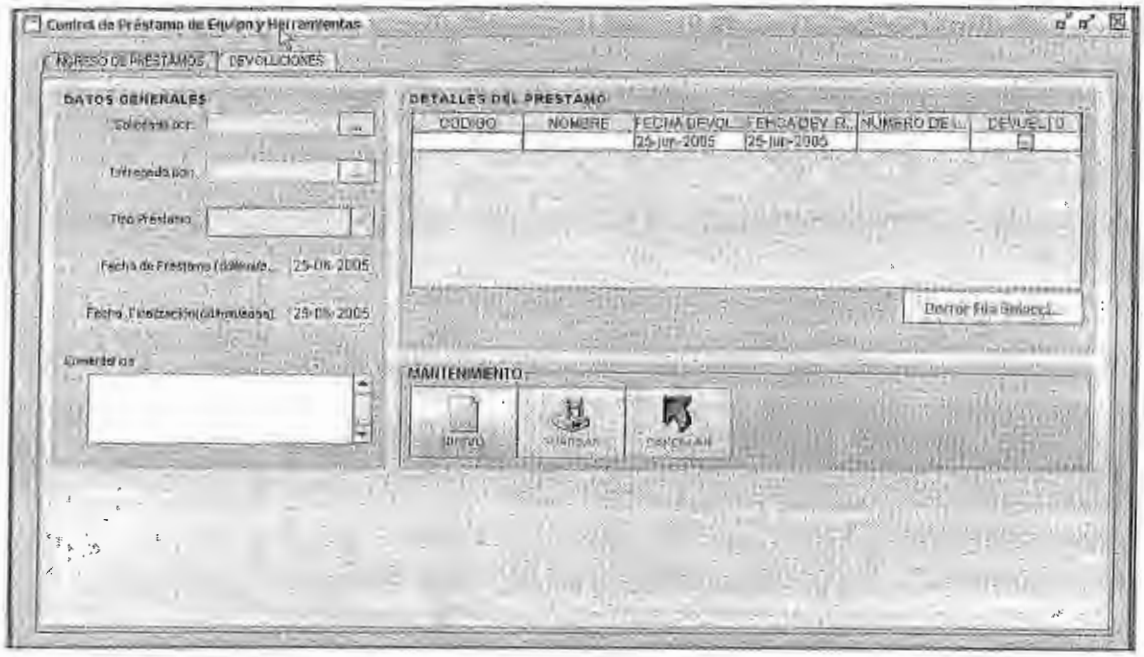

Control de Préstamo de Equipo y Herramientas
INGRESO DE PRESTAMOS
DEVOLUCIONES
DATOS GENERALES
Solicitado por:
Entregado por:
Tipo Prestamo:
Fecha de Prestamo (Ddmmaa...
25-06-2005
Fecha Terminación(Ddmmaaaa)
25-06-2005
Comentarios
DETALLES DEL PRESTAMO
CODIGO
NOMBRE
FECHA DEVOL...
25-jun-2005
FECHA DEV R...
25-jun-2005
NUMERO DE L...
DEVUELTO
Borrar Fila Selecci...
MANTENIMIENTO
NUEVO
GUARDAR
CANCELAR

Figura 4-12

4.6.2 INTERFAZ CLIENTE EXPEDIENTES

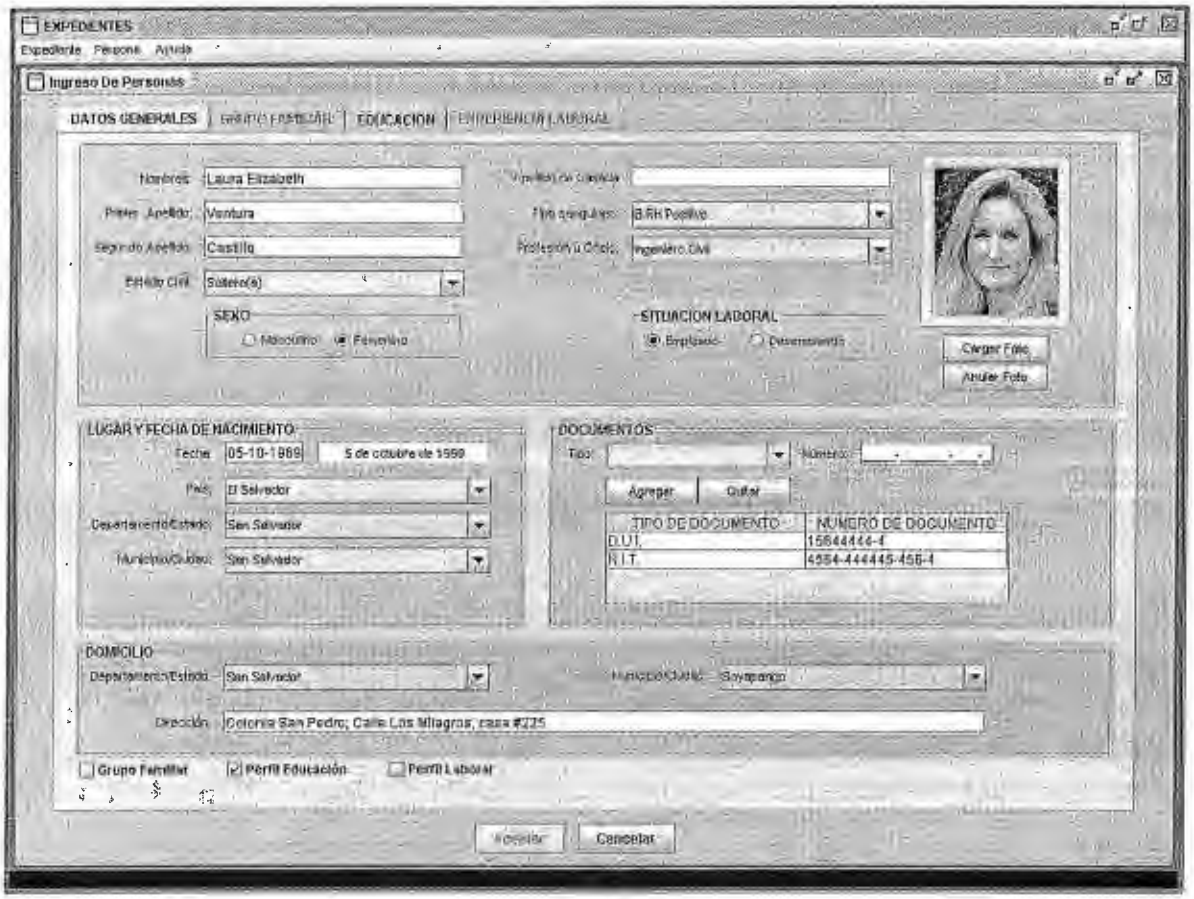


Figura. 4-13

Registro general de personas. Es un formulario para el registro de datos de todos los involucrados en actividades del departamento. Además este formulario proporciona la facultad de registra un perfil educativo y laboral correspondiente a la persona que se le esta danto ingreso de igual forma permite crear registros del grupo familiar perteneciente a esta persona (Figura 4-13).

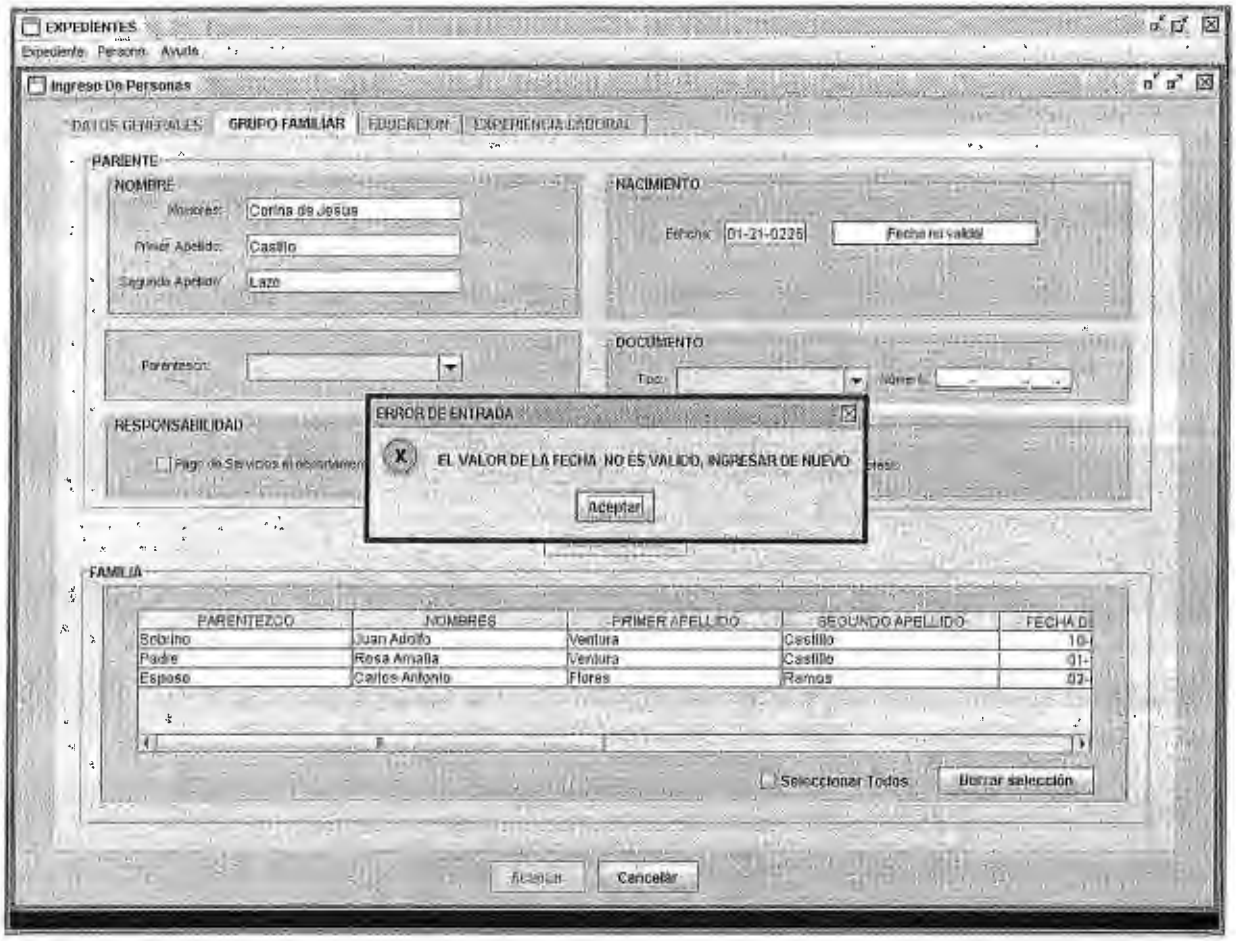


Figura. 4-14

Validación de Datos fecha a través de mensajes generados. Cuando un tipo de dato fecha es incorrecto el sistema reporta un mensaje de error el cual alerta al usurario para corregirlo en el momento, bloqueando cualquier operación hasta que el error es corregido.

En la mayoría de interfaces en el SIPOP se ha hecho uso de de cuadros de diálogos los cuales muestran mensajes correspondientes a las acciones efectuadas por el usuario, comunicando que acciones se deben tomar, consultando si se desean guardar cambios efectuados en los registros o terminar la operación sin efectuar cambios, como el mostrado en la Figura 4-14.

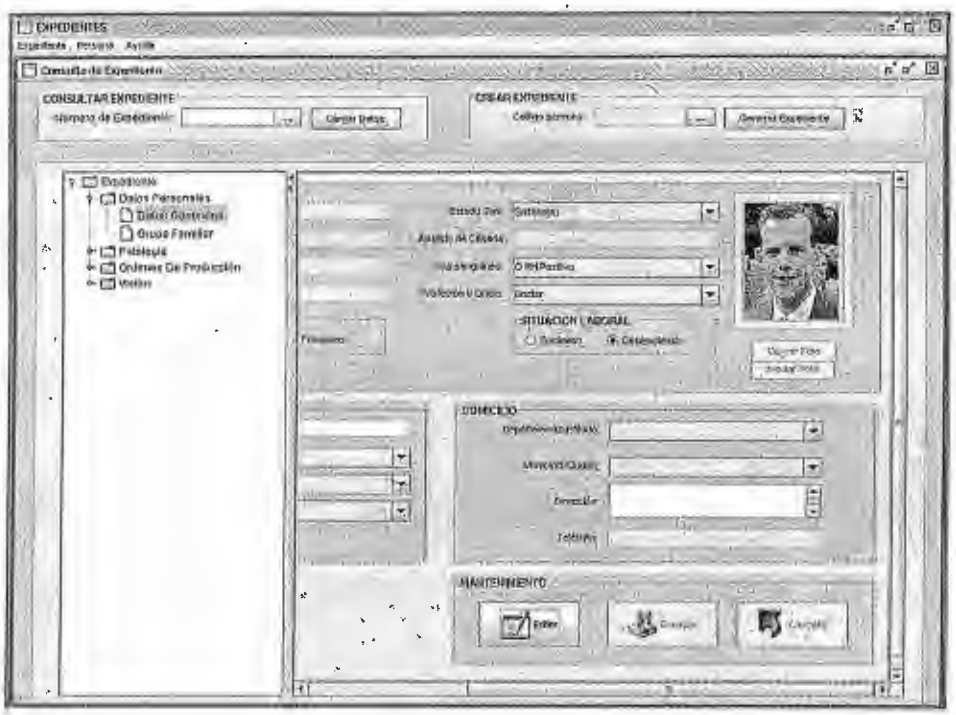


Figura. 4-15

Creación, Modificación y consulta de Expedientes

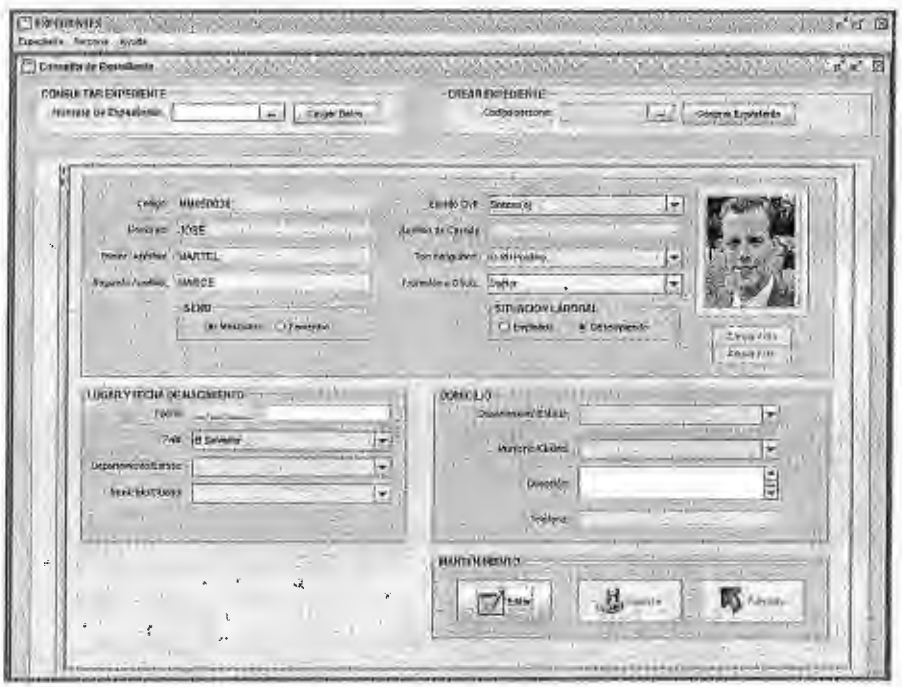


Figura. 4-16

Uso del contenedor SplitPane que amplia el espacio para la edición en el panel de datos. Este componente habilita un área para colocar información detallada según la selección hecha en el componente de árbol del panel derecho. El efecto logrado es un mejor uso del espacio y orden en el proceso de datos para el expediente. Esta interfase se puede observar en las figuras: Figura 4-15 y Figura 4-16

4.6.3 INTERFAZ CLIENTE PRODUCCION

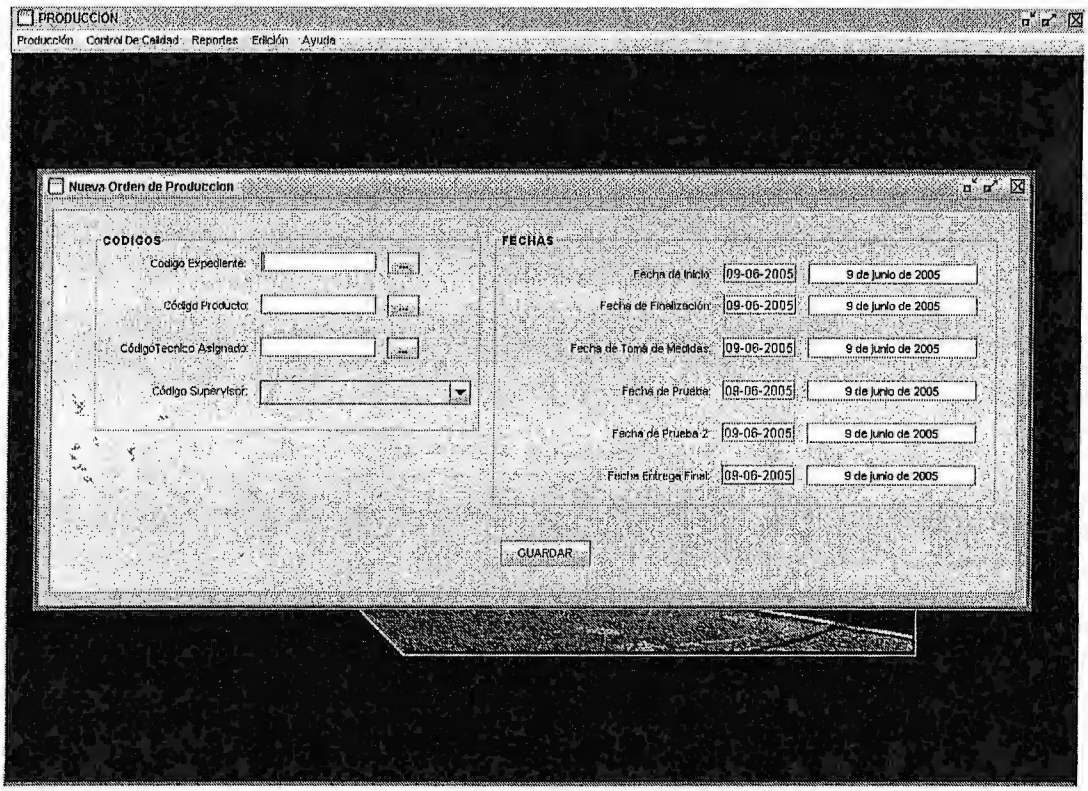


Figura. 4-17

Generación de Orden de producción para un expediente. La Orden solo puede existir si un expediente es creado previamente y asignado para la creación de un producto especifico. Este formulario se auxilia de formularios de búsqueda como en ejemplos anteriores, el propósito es evitar ingreso de datos no validos al sistema (Figura 4-17).

4.6.4 INTERFAZ CLIENTE REQUISICION

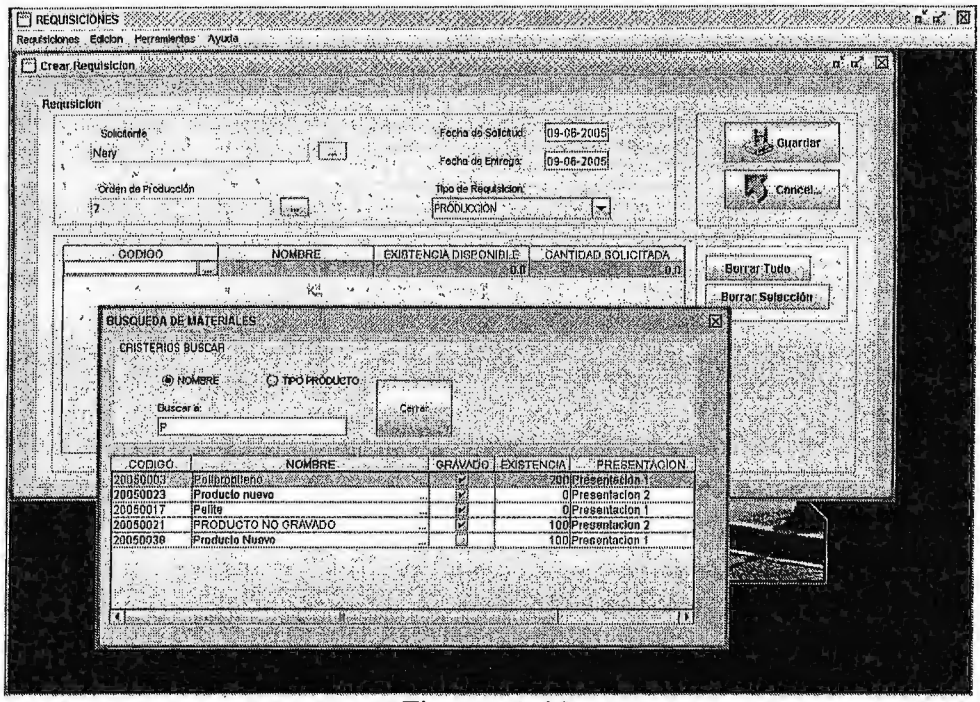


Figura. 4-18

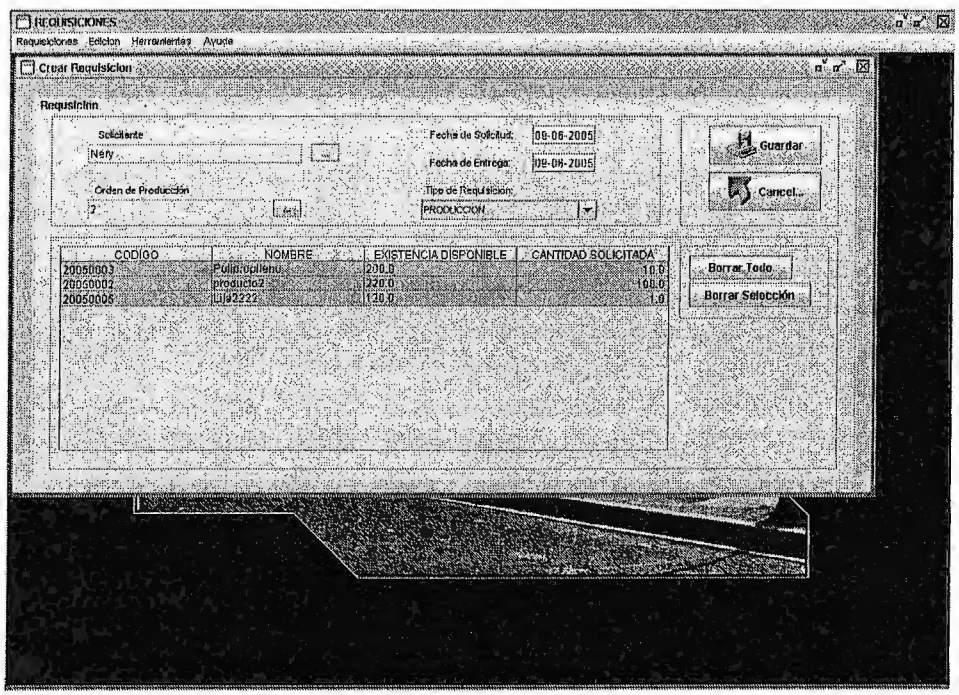


Figura. 4-19

Requisición de materiales para una Orden de producción. Como puede observar en las figuras: Figura 4-18 y Figura 4-19 la requisición de materiales posee un cliente propio.

4.6.5 INTERFAZ CLIENTE ADMINISTRACION

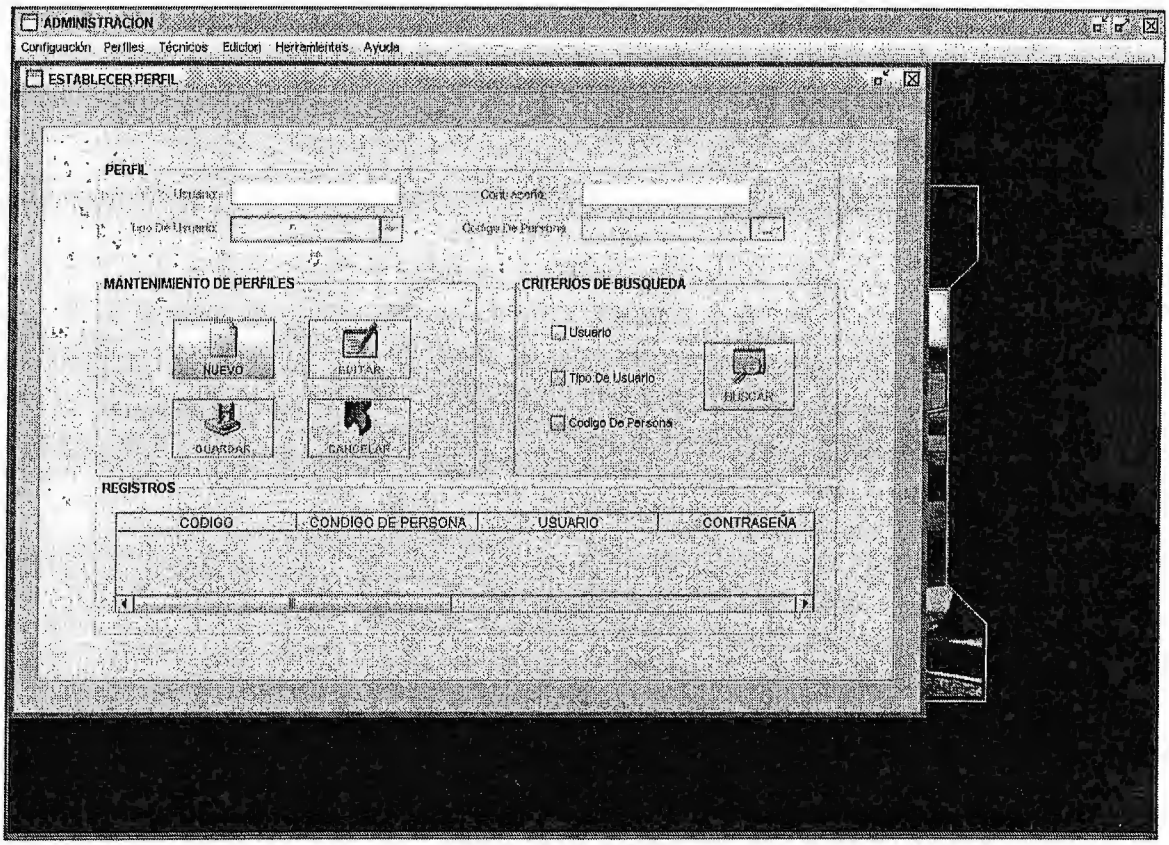


Figura. 4-20

Creación de perfiles de usuario para el SIPOP.

Los perfiles de usuario son los diferentes usuarios del sistema, así una persona registrada en el sistema puede realizar operaciones en el mismo, siempre que cuente con un perfil de usuario. Para cada perfil existente se tendrá asignado un tipo de usuario que le permite el acceso a uno o más módulos clientes, con el objetivo de dar soporte a la seguridad del sistema (Figura 4-20). Algunos de los

tipos de usuarios contemplados para el SIPOP son: Master, Paciente, Técnico, Encargado de Bodega, Asistente Administrativo.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- La selección de una metodología de desarrollo y/o de algunas de sus prácticas, dependerá del tipo de proyecto y el equipo que las utilice. Cada equipo selecciona la metodología y/o prácticas que se ajusten a su necesidad.
- La metodología Crystal Clear proporciona flexibilidad para utilizar prácticas de desarrollo del proceso de XP (Extreme Programming), esto hace que un equipo pequeño pueda adoptarla en sus proyectos y mejorar sus actividades de planificación y desarrollo.
- Las técnicas de programación en pares y lado a lado (Side by Side) no son excluyentes, así la programación lado a lado incrementa el numero de asignaciones completadas que pueden realizarse y la programación por pares es requerida en asignaciones complejas, donde la experiencia de los programadores ayuda a obtener mejores resultados al compartir tareas.
- La utilización de la base de datos Firebird así como herramientas auxiliares del tipo Open Source y de libre distribución, han demostrado ser efectivas sin agregar costos adicionales por su uso,

- El diseño del sistema SIPOP permite ampliaciones y modificaciones, como característica de las metodologías ágiles utilizadas, este diseño permitirá desarrollos futuros en las áreas aun no contempladas en este proyecto.

5.2 RECOMENDACIONES

- Cada proyecto y equipo de desarrollo posee características particulares, la selección de una metodología debe responder a estas características por ejemplo, tipo de proyecto, numero de miembros del equipo, grado de seguridad, etc, se recomienda tener claro esto antes de seleccionar una metodología ágil.
- Para equipos pequeños, es necesario que se mantenga una organización del trabajo, de acuerdo a las metodologías ágiles, esta se da por comunicación entre los programadores y los clientes, no olvidar que la planificación debe ser flexible.
- Se recomienda leer la documentación de los productos a utilizar, consultar a otros usuarios y participar en foros sobre todo si son herramientas Open Source, la mayoría posee un buen número de recursos para aprendizaje.
- Es recomendable conocer y entender el tipo de licencia de las herramientas a utilizar, en cuanto a los permisos y restricciones para el desarrollo y distribución, de los productos en si mismos o de los creados a partir de ellos
- El tópico de metodologías ágiles y mejores practicas para el desarrollo de software, debería ser incluido en el contenido de asignaturas de ingeniería de software y similares, ya que son una nueva manera de enfocar el proceso de desarrollo de software.

BIBLIOGRAFIA

- Roger S. Pressman Ingeniería de Software. Un enfoque práctico. Quinta Edición McGraw Hill 2002.
- Robert Cecil Martin. UML for java programmers Prentice Hall Inc. 2002.
- James Newkirk/Robert C Martin. La Programación extrema en la práctica. Addison Wesley 2003.
- Alistair Cockburn .Crystal Clear A Human Powered Methodology for Small Teams. Addison-Wesley 2004
- Martin Fowler. UML Distilled Third edition. Addison-Wesley 2004
- Kent Beck. Extreme Programming Explained. Addison-Wesley 2004
- Agustin Froufe. Java 2 Manual de usuario y tutorial. Alfaomega Ra-ma 2003
- Helen Borrie. The Firebird Book , A referente for Database Developers. Apress 2004
- Herbert Schildt, Java 2 Fifth Edition. The Complete Reference. MacGraw Hill 2002
- Cay Horstmann, Core Java(TM) 2, Volumen I y Volumen II. Quinta Edición. Prentice Hall 2001

ANEXOS

ANEXO A REFERENCIAS A SITIOS WEB

BASES DE DATOS

http://firebird.sourceforge.net/
http://www.firebird.com.mx
http://www.mysql.com/
http://www.postgresql.org/
http://www.sqlmanager.net
http://www.flamerobin.org/
http://firebird.sourceforge.net/index.php?op=files&id=jaybird

HERRAMIENTAS DE DESARROLLO DE SOFTWARE

www.**netbeans**.org
www.java.sun.com

METODOLOGIAS ÁGILES

http://www.agile-spain.com/
http://www.agilealliance.com
http://www.extremeprogramming.org/
http://alistair.cockburn.us/

ANEXO B FICHAS TÉCNICAS DE LAS HISTORIAS DE USUARIO

Ficha Técnica N°:1.

Tiempo estimado: 3 días

Búsqueda de artículos en el inventario a través de diversos criterios (nombre, código, tipo, presentación).

Ficha Técnica N°: 2.

Tiempo estimado: 3 días

Pantalla de presentación de datos (mínimos, máximos, existencia, código) del artículo seleccionado

Ficha Técnica N°: 3.

Tiempo estimado: 3 días

Despliegue del inventario total de artículos en bodega:

- Validación de entrada de datos:
- Fechas válidas.
- Textos en mayúsculas.
- Valores numéricos (Enteros y decimales).

Ficha Técnica N°: 4.

Tiempo estimado: 5 días

Registro de ingresos de artículos a bodega.

Ficha Técnica N°: 5.

Tiempo estimado:5 días

Ingresos de datos de los artículos en forma tabular

Ficha Técnica N°: 6.

Tiempo estimado:1 día

Cálculo automatizado de los campos monetarios PRECIO TOTAL Y COSTO TOTAL para los registros de entrada y salida de material respectivamente.

Ficha Técnica N°: 7.

Tiempo estimado: 1 día

Interfaz con Alertas para el control de existencias máximas y mínimas de los artículos.

Ficha Técnica N°: 8.

Tiempo estimado: 3 días

Validación de ingreso de códigos de artículos existentes.

Ficha Técnica N°: 9.

Tiempo estimado:3 día

Despliegue de mensajes de error o advertencias sobre las diferentes operaciones del sistema.

Ficha Técnica N°: 10.

Tiempo estimado:5 días

Registro de las salidas de artículos y materiales en bodega.

Ficha Técnica N°: 11.

Tiempo estimado:5 días

Controlar salida de materiales de manera que el primer material en entrar es el primer material en salir.

Ficha Técnica N°: 12.

Tiempo estimado:1 día

Uniformidad en el tipo de unidad de medida (presentación) tanto para el registro de entradas como el de salidas para un material especificó.

Ficha Técnica N°: 13.

Tiempo estimado: 1 día

Verificación del número de solicitud de materiales por persona (control de calidad).

Ficha Técnica N°: 14.

Tiempo estimado: 2 días

El sistema establece la unidad mínima de salida por producto.

Ficha Técnica N°: 15.

Tiempo estimado: 1 día

El sistema permite seleccionar la clasificación de los materiales entrantes en bodega.

Ficha Técnica N°: 16.

Tiempo estimado: 3 día

El sistema permite configurar los tipos de materiales, unidades de entrada, salida y presentación.

Ficha Técnica N°: 17:

Tiempo estimado: 3 día

El sistema de generar históricos de los movimientos y transacciones en bodega como de los inventarios de esta.

Ficha Técnica N°: 18:

Tiempo estimado:5 días

Configuración de archivos de propiedades y administración del sistema

Ficha Técnica N°: 19

Tiempo estimado: 3 días

El sistema debe implementar herramientas para el mantenimiento de los registros generados en la base de datos.

Ficha Técnica N°: 20:

Tiempo estimado: 1 día

Lograr conexión de la aplicación con la base de datos.

Ficha Técnica N°. 21:

Tiempo estimado:5 días

Control de transacciones, disparadores y procedimientos almacenados desde el sistema.

ANEXO C LICENCIAS DE SOFTWARE

GNU GPL

GNU LGPL

SUN

APACHE

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions
of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions
either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of
this License, you may choose any version ever published by the Free Software
Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author
to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and
of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY
FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED
OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS
TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING
WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES,
INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING
OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED
TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY

YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER
PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
>
> You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

> Gnomovision version 69, Copyright (C) year name of author
> Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'.
> This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision' (which makes passes at compilers) written by James Hacker.
>
> <signature of Ty Coon>, 1 April 1989
> Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the
library. If this is what you want to do, use the GNU Library General Public License instead of this License.

GNU LESSER GENERAL PUBLIC LICENSE
Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know

that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes
a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a

"work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

GNU LESSER GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2,

instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on

which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any
particular circumstance, the balance of the section is intended to apply,
and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add
an explicit geographical distribution limitation excluding those countries,
so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a

license version number, you may choose any version ever published by
the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the library's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.
>
> This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.
>
> You should have received a copy of the GNU Lesser General Public License along with this library; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

> Yoyodyne, Inc., hereby disclaims all copyright interest in the library `Frob' (a library for tweaking knobs) written by James Random Hacker.
>
> <signature of Ty Coon>, 1 April 1990
> Ty Coon, President of Vice

That's all there is to it!

Sun Microsystems, Inc. Binary Code License Agreement

for the JAVA 2 PLATFORM STANDARD EDITION DEVELOPMENT KIT 5.0

SUN MICROSYSTEMS, INC. ("SUN") IS WILLING TO LICENSE THE SOFTWARE IDENTIFIED BELOW TO YOU ONLY UPON THE CONDITION THAT YOU ACCEPT ALL OF THE TERMS CONTAINED IN THIS BINARY CODE LICENSE AGREEMENT AND SUPPLEMENTAL LICENSE TERMS (COLLECTIVELY "AGREEMENT"). PLEASE READ THE AGREEMENT CAREFULLY. BY DOWNLOADING OR INSTALLING THIS SOFTWARE, YOU ACCEPT THE TERMS OF THE AGREEMENT. INDICATE ACCEPTANCE BY SELECTING THE "ACCEPT" BUTTON AT THE BOTTOM OF THE AGREEMENT. IF YOU ARE NOT WILLING TO BE BOUND BY ALL THE TERMS, SELECT THE "DECLINE" BUTTON AT THE BOTTOM OF THE AGREEMENT AND THE DOWNLOAD OR INSTALL PROCESS WILL NOT CONTINUE.

1. DEFINITIONS. "Software" means the identified above in binary form, any other machine readable materials (including, but not limited to, libraries, source files, header files, and data files), any updates or error corrections provided by Sun, and any user manuals, programming guides and other documentation provided to you by Sun under this Agreement. "Programs" mean Java applets and applications intended to run on the Java 2 Platform Standard Edition (J2SE platform) platform on Java-enabled general purpose desktop computers and servers.

2. LICENSE TO USE. Subject to the terms and conditions of this Agreement, including, but not limited to the Java Technology Restrictions of the Supplemental License Terms, Sun grants you a non-exclusive, non-transferable, limited license without license fees to reproduce and use internally Software complete and unmodified for the sole purpose of running Programs. Additional licenses for developers and/or publishers are granted in the Supplemental License Terms.

3. RESTRICTIONS. Software is confidential and copyrighted. Title to Software and all associated intellectual property rights is retained by Sun and/or its licensors. Unless enforcement is prohibited by applicable law, you may not modify, decompile, or reverse engineer Software. You acknowledge that Licensed Software is not designed or intended for use in the design, construction, operation or maintenance of any nuclear facility. Sun Microsystems, Inc. disclaims any express or implied warranty of fitness for such uses. No right, title or interest in or to any trademark, service mark, logo or trade name of Sun or its licensors is granted under this Agreement. Additional restrictions for developers and/or publishers licenses are set forth in the Supplemental License Terms.

4. LIMITED WARRANTY. Sun warrants to you that for a period of ninety (90) days from the date of purchase, as evidenced by a copy of the receipt, the media on which Software is furnished (if any) will be free of defects in materials and workmanship under normal use. Except for the foregoing, Software is provided "AS IS". Your exclusive remedy and Sun's entire liability under this limited warranty will be at Sun's option to replace Software media or refund the fee paid for Software. Any implied warranties on the Software are limited to 90 days. Some states do not allow limitations on duration of an implied warranty, so the above may not apply to you. This limited warranty gives you specific legal rights. You may have others, which vary from state to state.

5. DISCLAIMER OF WARRANTY. UNLESS SPECIFIED IN THIS AGREEMENT, ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT THESE DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

6. LIMITATION OF LIABILITY. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL SUN OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR SPECIAL, INDIRECT, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF OR RELATED TO THE

USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. In no event will Sun's liability to you, whether in contract, tort (including negligence), or otherwise, exceed the amount paid by you for Software under this Agreement. The foregoing limitations will apply even if the above stated warranty fails of its essential purpose. Some states do not allow the exclusion of incidental or consequential damages, so some of the terms above may not be applicable to you.

7. TERMINATION. This Agreement is effective until terminated. You may terminate this Agreement at any time by destroying all copies of Software. This Agreement will terminate immediately without notice from Sun if you fail to comply with any provision of this Agreement. Either party may terminate this Agreement immediately should any Software become, or in either party's opinion be likely to become, the subject of a claim of infringement of any intellectual property right. Upon Termination, you must destroy all copies of Software.

8. EXPORT REGULATIONS. All Software and technical data delivered under this Agreement are subject to US export control laws and may be subject to export or import regulations in other countries. You agree to comply strictly with all such laws and regulations and acknowledge that you have the responsibility to obtain such licenses to export, re-export, or import as may be required after delivery to you.

9. TRADEMARKS AND LOGOS. You acknowledge and agree as between you and Sun that Sun owns the SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET trademarks and all SUN, SOLARIS, JAVA, JINI, FORTE, and iPLANET-related trademarks, service marks, logos and other brand designations ("Sun Marks"), and you agree to comply with the Sun Trademark and Logo Usage Requirements currently located at http://www.sun.com/policies/trademarks. Any use you make of the Sun Marks inures to Sun's benefit.

10. U.S. GOVERNMENT RESTRICTED RIGHTS. If Software is being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), then the Government's rights in Software and accompanying documentation will be only as set forth in this Agreement; this is in accordance with 48 CFR 227.7201 through 227.7202-4 (for Department of Defense (DOD) acquisitions) and with 48 CFR 2.101 and 12.212 (for non-DOD acquisitions).

11. GOVERNING LAW. Any action related to this Agreement will be governed by California law and controlling U.S. federal law. No choice of law rules of any jurisdiction will apply.

12. SEVERABILITY. If any provision of this Agreement is held to be unenforceable, this Agreement will remain in effect with the provision omitted, unless omission would frustrate the intent of the parties, in which case this Agreement will immediately terminate.

13. INTEGRATION. This Agreement is the entire agreement between you and Sun relating to its subject matter. It supersedes all prior or contemporaneous oral or written communications, proposals, representations and warranties and prevails over any conflicting or additional terms of any quote, order, acknowledgment, or other communication between the parties relating to its subject matter during the term of this Agreement. No modification of this Agreement will be binding, unless in writing and signed by an authorized representative of each party.

SUPPLEMENTAL LICENSE TERMS

These Supplemental License Terms add to or modify the terms of the Binary Code License Agreement. Capitalized terms not defined in these Supplemental Terms shall have the same meanings ascribed to them in the Binary Code License Agreement . These Supplemental Terms shall supersede any inconsistent or conflicting terms in the Binary Code License Agreement, or in any license contained within the Software.

A. Software Internal Use and Development License Grant. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software "README" file, including, but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce internally and use internally the Software complete and unmodified for the purpose of designing, developing, and testing your Programs.

B. License to Distribute Software. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software README file, including, but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce and distribute the Software, provided that (i) you distribute the Software complete and unmodified and only bundled as part of, and for the sole purpose of running, your Programs, (ii) the Programs add significant and primary functionality to the Software, (iii) you do not distribute additional software intended to replace any component(s) of the Software, (iv) you do not remove or alter any proprietary legends or notices contained in the Software, (v) you only distribute the Software subject to a license agreement that protects Sun's interests consistent with the terms contained in this Agreement, and (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

C. License to Distribute Redistributables. Subject to the terms and conditions of this Agreement and restrictions and exceptions set forth in the Software README file, including but not limited to the Java Technology Restrictions of these Supplemental Terms, Sun grants you a non-exclusive, non-transferable, limited license without fees to reproduce and distribute those files specifically identified as redistributable in the Software "README" file ("Redistributables") provided that: (i) you distribute the Redistributables complete and unmodified, and only bundled as part of Programs, (ii) the Programs add significant and primary functionality to the Redistributables, (iii) you do not distribute additional software intended to supersede any component(s) of the Redistributables (unless otherwise specified in the applicable README file), (iv) you do not remove or alter any proprietary legends or notices contained in or on the Redistributables, (v) you only distribute the Redistributables pursuant to a license agreement that protects Sun's interests consistent with the terms contained in the Agreement, (vi) you agree to defend and indemnify Sun and its licensors from and against any damages, costs, liabilities, settlement amounts and/or expenses (including attorneys' fees) incurred in connection with any claim, lawsuit or action by any third party that arises or results from the use or distribution of any and all Programs and/or Software.

D. Java Technology Restrictions. You may not create, modify, or change the behavior of, or authorize your licensees to create, modify, or change the behavior of, classes, interfaces, or subpackages that are in any way identified as "java", "javax", "sun" or similar convention as specified by Sun in any naming convention designation.

E. Distribution by Publishers. This section pertains to your distribution of the Software with your printed book or magazine (as those terms are commonly used in the industry) relating to Java technology ("Publication"). Subject to and conditioned upon your compliance with the restrictions and obligations contained in the Agreement, in addition to the license granted in Paragraph 1 above, Sun hereby grants to you a non-exclusive, nontransferable limited right to reproduce complete and unmodified copies of the Software on electronic media (the "Media") for the sole purpose of inclusion and distribution with your Publication(s), subject to the following terms: (i) You may not distribute the Software on a stand-alone basis; it must be distributed with your Publication(s); (ii) You are responsible for downloading the Software from the applicable Sun web site; (iii) You must refer to the Software as JavaTM 2 Platform Standard Edition Development Kit 5.0; (iv) The Software must be reproduced in its entirety and without any modification whatsoever (including, without limitation, the Binary Code License and Supplemental License Terms accompanying the Software and proprietary rights notices contained in the Software); (v) The Media label shall include the following information: Copyright 2004, Sun Microsystems, Inc. All rights reserved. Use is subject to license terms. Sun, Sun Microsystems, the Sun logo, Solaris, Java, the

Java Coffee Cup logo, J2SE , and all trademarks and logos based on Java are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. This information must be placed on the Media label in such a manner as to only apply to the Sun Software; (vi) You must clearly identify the Software as Sun's product on the Media holder or Media label, and you may not state or imply that Sun is responsible for any third-party software contained on the Media; (vii) You may not include any third party software on the Media which is intended to be a replacement or substitute for the Software; (viii) You shall indemnify Sun for all damages arising from your failure to comply with the requirements of this Agreement. In addition, you shall defend, at your expense, any and all claims brought against Sun by third parties, and shall pay all damages awarded by a court of competent jurisdiction, or such settlement amount negotiated by you, arising out of or in connection with your use, reproduction or distribution of the Software and/or the Publication. Your obligation to provide indemnification under this section shall arise provided that Sun: (i) provides you prompt notice of the claim; (ii) gives you sole control of the defense and settlement of the claim; (iii) provides you, at your expense, with all available information, assistance and authority to defend; and (iv) has not compromised or settled such claim without your prior written consent; and (ix) You shall provide Sun with a written notice for each Publication; such notice shall include the following information: (1) title of Publication, (2) author(s), (3) date of Publication, and (4) ISBN or ISSN numbers. Such notice shall be sent to Sun Microsystems, Inc., 4150 Network Circle, M/S USCA12-110, Santa Clara, California 95054, U.S.A , Attention: Contracts Administration.

F. Source Code. Software may contain source code that, unless expressly licensed for other purposes, is provided solely for reference purposes pursuant to the terms of this Agreement. Source code may not be redistributed unless expressly provided for in this Agreement.

G. Third Party Code. Additional copyright notices and license terms applicable to portions of the Software are set forth in the THIRDPARTYLICENSEREADME.txt file. In addition to any terms and conditions of any third party opensource/freeware license identified in the THIRDPARTYLICENSEREADME.txt file, the disclaimer of warranty and limitation of liability provisions in paragraphs 5 and 6 of the Binary Code License Agreement shall apply to all Software in this distribution.

For inquiries please contact: Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A.
(LFI#141623/Form ID#011801)

ANEXO D CONVENCIONES

TABLAS

NOMBRES

1- USAR LETRAS MAYUSCULAS
2- SIN SUBRRAYADO
3- SI SON VARIAS PALABRAS: TOMAR LAS 4 PRIMERAS LETRAS DE CADA PALABRA.
4- LAS TABLAS HISTORICAS LLEVARAN EL MISMO NOMBRE DE LAS TABLAS ORIGENES PERO CON _H AL FINAL.

CAMPOS

NOMBRES

1- USAR LETRAS MAYUSCULAS.
2- SEPARAR CON SUBRRAYADO LAS DIFERENTES PALABRAS.
3- SI SON MAS DE 2 PALABRAS: TOMAR LAS 4 PRIMERAS LETRAS DE CADA PALABRA.
4- LLEVARAN EL MISMO NOMBRE DE LAS CAMPOS DE LAS TABLAS ORIGENES PERO CON _H AL FINAL.
5- LAS LLAVES PRIMARIAS USARÁN ID_ AL INICIO DEL NOMBRE DEL CAMPO.
6- CUANDO SEAN LLAVES FORANEAS SE USARA EL NOMBRE DE LA LLAVE SIN USAR ID_
7- LOS ID SERAN DE TIPO CHAR.
8- A PARTIR DE LA 4a PALABRA SE USARAN LAS LETRAS DE INICIO DE CADA PALABRA RESTANTE.

ANEXO E

CONSULTA A PROGRAMADORES SOBRE EL USO DE HERRAMIENTAS OPEN SOURCE Y METODOLOGIAS ÁGILES PARA EL DESARROLLO DE SOFTWARE

INVESTIGACIÓN SOBRE METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

Estimado programador(a), desarrollador(a) de software:

Gracias por tu colaboración en este proyecto. Estamos recolectando información sobre la situación actual del proceso de desarrollo de software, y se pretende utilizar para fines estrictamente académicos y de investigación.

Indicaciones:

Para versión Impresa.

- *Conteste brevemente a las preguntas formuladas.*
- *Marque con una* **X** *las Casillas de Verificación (☐) para contestar* **Si** *o* **No** *y también para seleccionar algunas opciones relacionadas a la experiencia en sus proyectos.*
- *Subrayar las opciones mostradas en las* **Tablas** *para hacer referencia a su selección.*

Para versión Documento de Word.

- *Para marcar una casilla de verificación (☐) puede dar doble clic a esta y en la ventana* ***Opciones de Campo...*** *seleccionar la opción* ***Activada****, dar clic en el botón* ***Aceptar*** *y la casilla se mostrará de esta manera: ☒ indicando que está seleccionada.*
- *Modifique este documento escribiendo sus respuestas o seleccionando las opciones deseadas dando alguna referencia de su elección (Cambiando el color de la letra, usando negrita, subrayado, etc.)*

1. ¿Cuánto tiempo de experiencia en desarrollo de Software posee?

   _____Semana(s) _____Mes(es) _____Año(s)

2. ¿Hace cuanto tiempo realizó su último proyecto de software?

   ☐ Actualmente

   _____Semana(s) _____Mes(es) _____Año(s)

3. ¿Cree usted que se pueden mejorar los procesos utilizados para desarrollar el software en sus proyectos?, explique.

   _____
   _____
   _____

4. ¿De acuerdo a su experiencia cual es la importancia que el cliente le da a la documentación del sistema (Manuales)?, explique.

   _____
   _____
   _____

5. ¿Utiliza o ha utilizado Metodologías para el desarrollo de Software?

   ☐ No. ☐ Si.

Si su respuesta en la pregunta **5** es **No** entonces continuar con la pregunta **6,** si su respuesta es **Si** entonces continuar con la pregunta **7.**

6. En las tablas 1 y 2 seleccionar las opciones que describan las características para 1 o 2 proyectos que usted ha realizado. Escribir en las celdas vacías para completar los datos.

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Horas promedio al día dedicadas al proyecto:					
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 1

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Horas promedio al día dedicadas al proyecto:					
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 2

Luego de utilizar y completar las tablas continuar con la pregunta **9**.

7. Utilizar las tablas 3 y 4 para escribir el nombre de las metodologías que ha utilizado y los pasos o procesos más importantes de cada una.

NOMBRE DE LA METODOLOGIA:		
No.	Nombres de los Procesos (Pasos)	Descripción
1		
2		
3		
4		
5		

Tabla 3

NOMBRE DE LA METODOLOGIA:		
No.	Nombres de los Procesos (Pasos)	Descripción
1		
2		
3		
4		
5		

Tabla 4

8. En las tablas 5 y 6 seleccionar las opciones que describen las características para 1 o 2 proyectos que usted ha realizado. Escribir en las celdas en blanco para completar la información.

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Metodología:		**Horas promedio al día dedicadas al proyecto:**			
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 5

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Metodología:		**Horas promedio al día dedicadas al proyecto:**			
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 6

9. ¿Cuales de las siguientes herramientas utiliza para el análisis de sus proyectos de desarrollo de software?

☐ Casos de Uso ☐ Escenarios ☐ UML ☐ Ninguna

☐ Otras: _____

10. ¿Ha escuchado sobre Procesos De Desarrollo diferentes a los que Ud. utiliza?

☐ No.

☐ Si. Mencione cuales: _____

11. ¿Ha escuchado sobre RUP (Rational Unified Process)?

☐ No. ☐ Si.

12. ¿Estaría dispuesto a utilizar alguna metodología diferente a la utilizada actualmente en el desarrollo de sus proyectos de software?

☐ No.

☐ Si. Mencione cuales utilizaría (En caso de conocer sobre algunas):

_____

_____

13. ¿Ha escuchado sobre Metodologías Ágiles para desarrollo de software (SRCRUM, Extreme Programming (XP), Cristal Clear, etc.)?

☐ No.

☐ Si. Mencione Cuales: _____

14. ¿Utiliza la Metodología Extreme Programming (XP) en sus proyectos de Software?

☐ No.

☐ Si. Mencione los elementos de XP que usted ha utilizado:

_____

_____

_____

_____

_____

_____

15. Si ha utilizado alguna Metodología Ágil en sus proyectos de software, en las tablas 7 y 8 seleccionar las opciones que describen las características para 1 ó 2 de estos proyectos. Escribir en las celdas en blanco para completar la información.

NOTA: Si los Proyectos ya fueron mencionados en las tablas 5 y 6 de la pregunta **8** entonces Omitir esta pregunta.

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Metodología Ágil:		**Horas promedio al día dedicadas al proyecto:**			
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 7

NOMBRE DEL PROYECTO:					
Ítems	**Opciones**				
Tipo De Software:	B2B (Business-to-Business)	ERP (Enterprise Resource Management)	CRM (Customer Relationship Management)		Otros:
Ambiente De Desarrollo:	Intranet	Cliente/Servidor	Internet	Telefonía	Otros:
Herramientas De Desarrollo:	Visual Studio	Power Builder	FoxPro	Java	Otros:
Sistemas Operativos:	Windows(98,Me, NT, 2000, XP)	Unix	Linux	Macintosh OS	Otros:
Metodología Ágil:		**Horas promedio al día dedicadas al proyecto:**			
Numero de integrantes en equipo de trabajo:		**Duración:**			

Tabla 8

16. ¿Ha escuchado sobre herramientas de desarrollo OPEN SOURCE?

☐ No. ☐ Si.

17. ¿Tiene experiencia en el uso de herramientas de desarrollo OPEN SOURCE?

☐ No. ☐ Si.

18. ¿Ha desarrollado productos comerciales utilizando herramientas de desarrollo OPEN SOURCE?

☐ No. ☐ Si.

19. Si la respuesta en la pregunta **18** es **Si** entonces utilice las tabla 9 y 10 para mencionar algunas ventajas y desventajas que le ha brindado el uso de al menos una herramienta OPEN SOURCE en sus proyectos de software.

NOMBRE DE HERRAMIENTA OPEN SOURCE:	
VENTAJAS	DESVENTAJAS
1.	1.
2.	2.
3.	3.
4.	4.
5.	5.

Tabla 9

NOMBRE DE HERRAMIENTA OPEN SOURCE:	
VENTAJAS	DESVENTAJAS
1.	1.
2.	2.
3.	3.
4.	4.
5.	5.

Tabla 10

NOTAS:

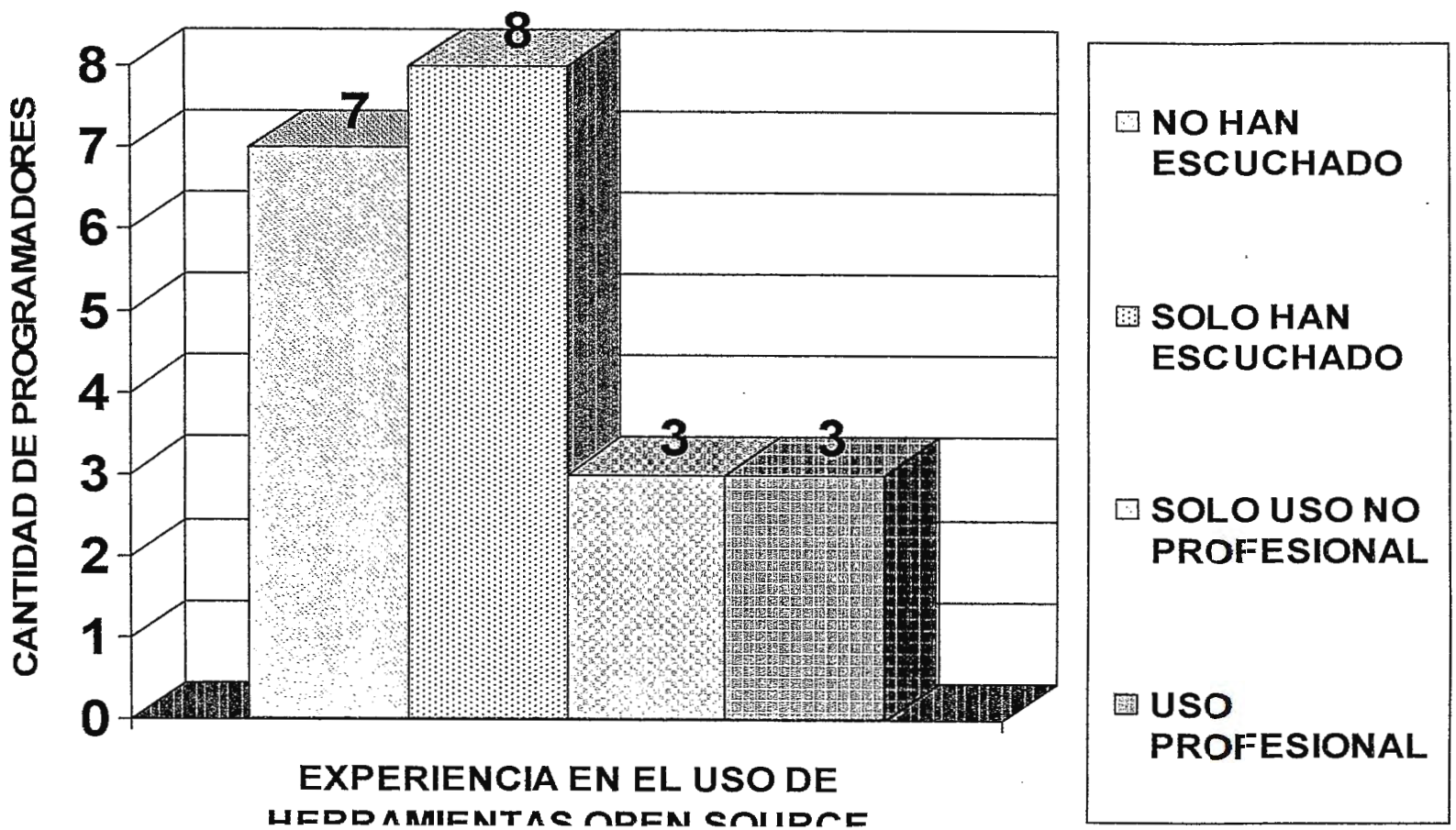
- Si usted ha llenado esta encuesta escribiendo sus respuestas modificando el documento en Microsoft Word entonces al finalizar guarde los cambios y reenvíe el documento (INVESTIGACION DESARROLLO DE SOFTWARE.doc) a la dirección de correo electrónico origen (investigaciondesoftware@hotmail.com).

- Siéntase libre de hacer sugerencias al instrumento (Encuesta) y agregar información.

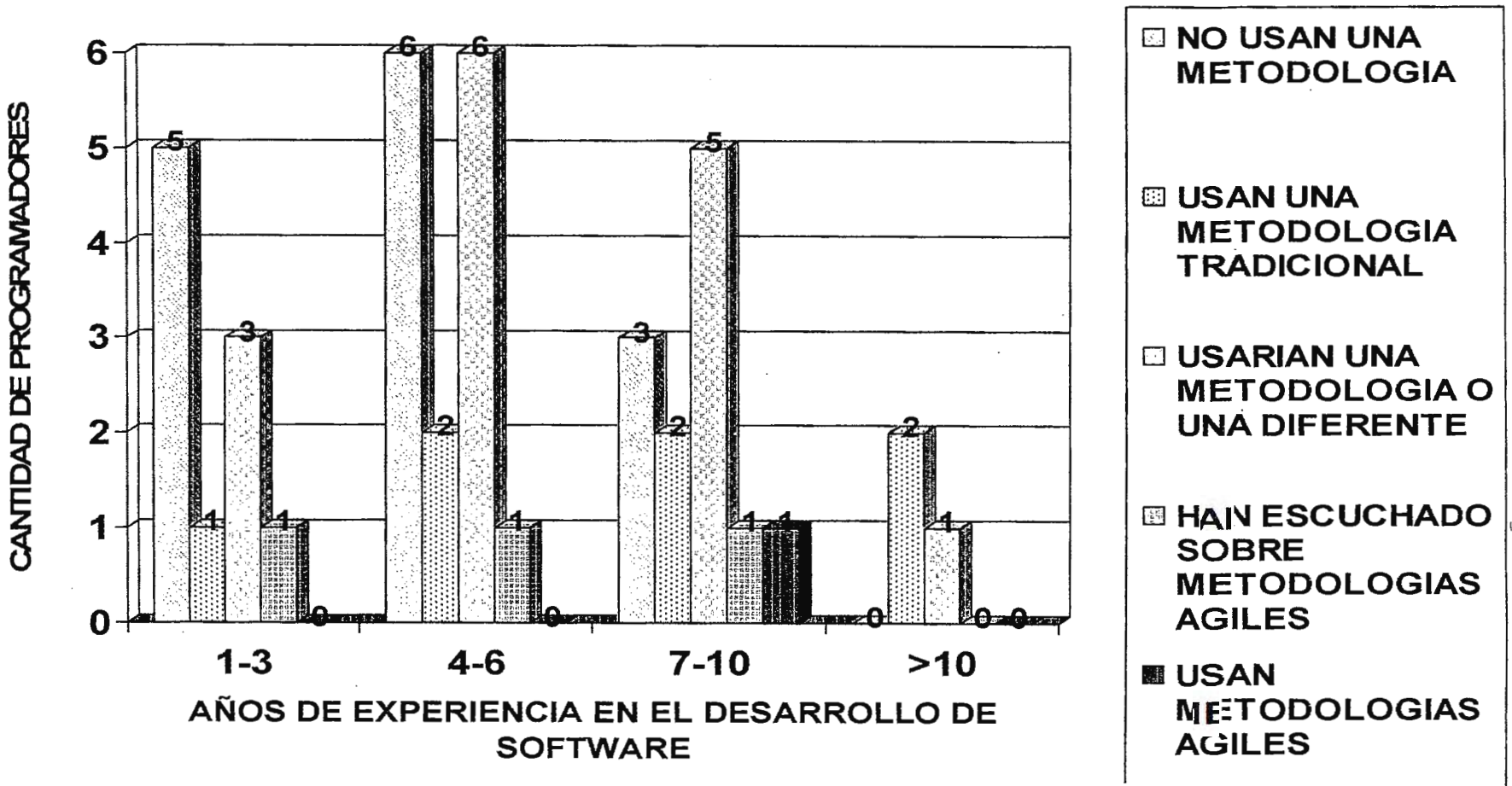
Sugerencias:

_____
_____
_____
_____
_____
_____
_____
_____
_____

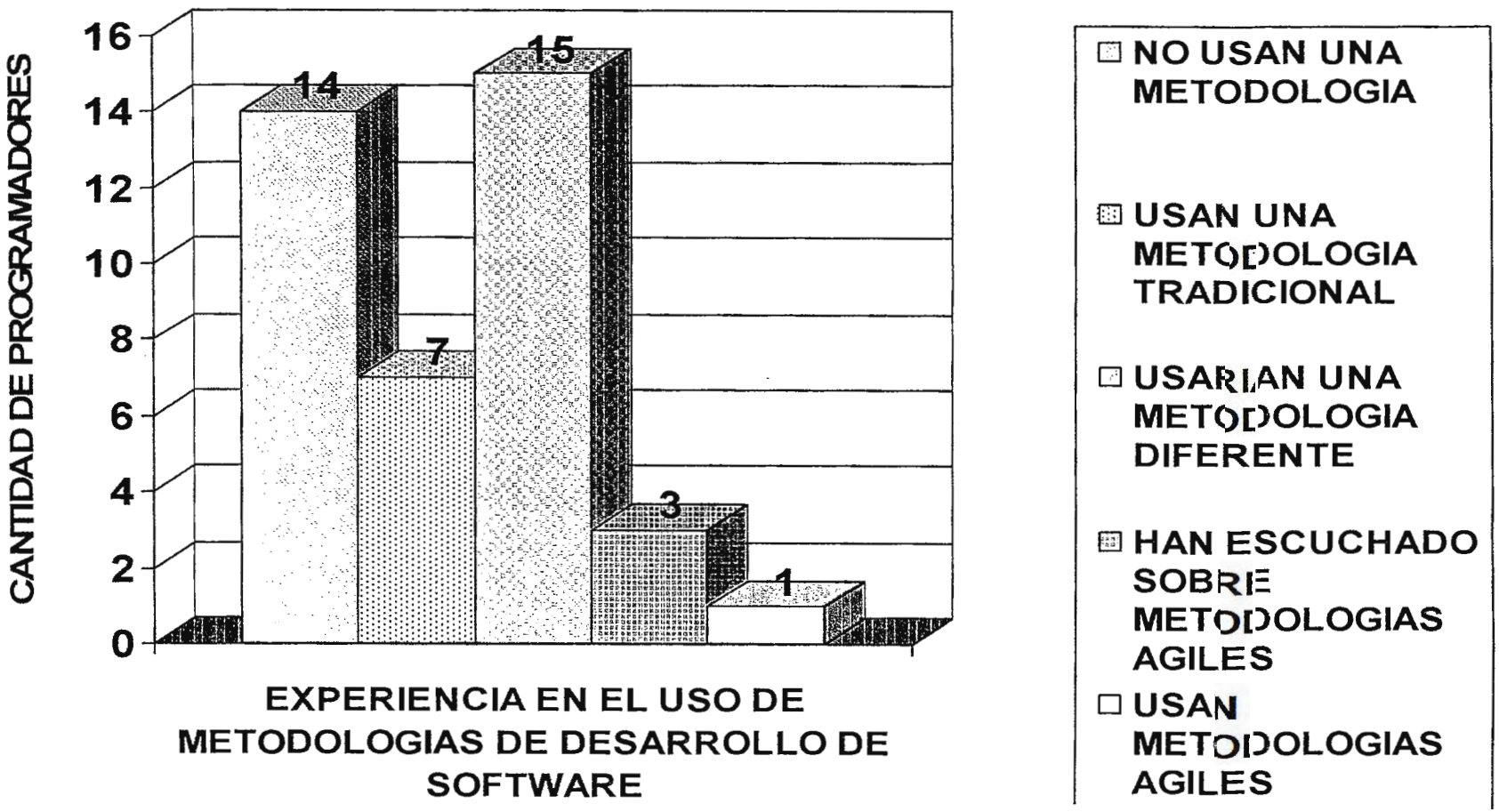
GRAFICA COMPARATIVA SOBRE EL USO DE HERRAMIENTAS OPEN SOURCE SEGÚN CONSULTA REALIZADA A 2I PROGRAMADORES EN EL MES DE ABRIL DE 2004



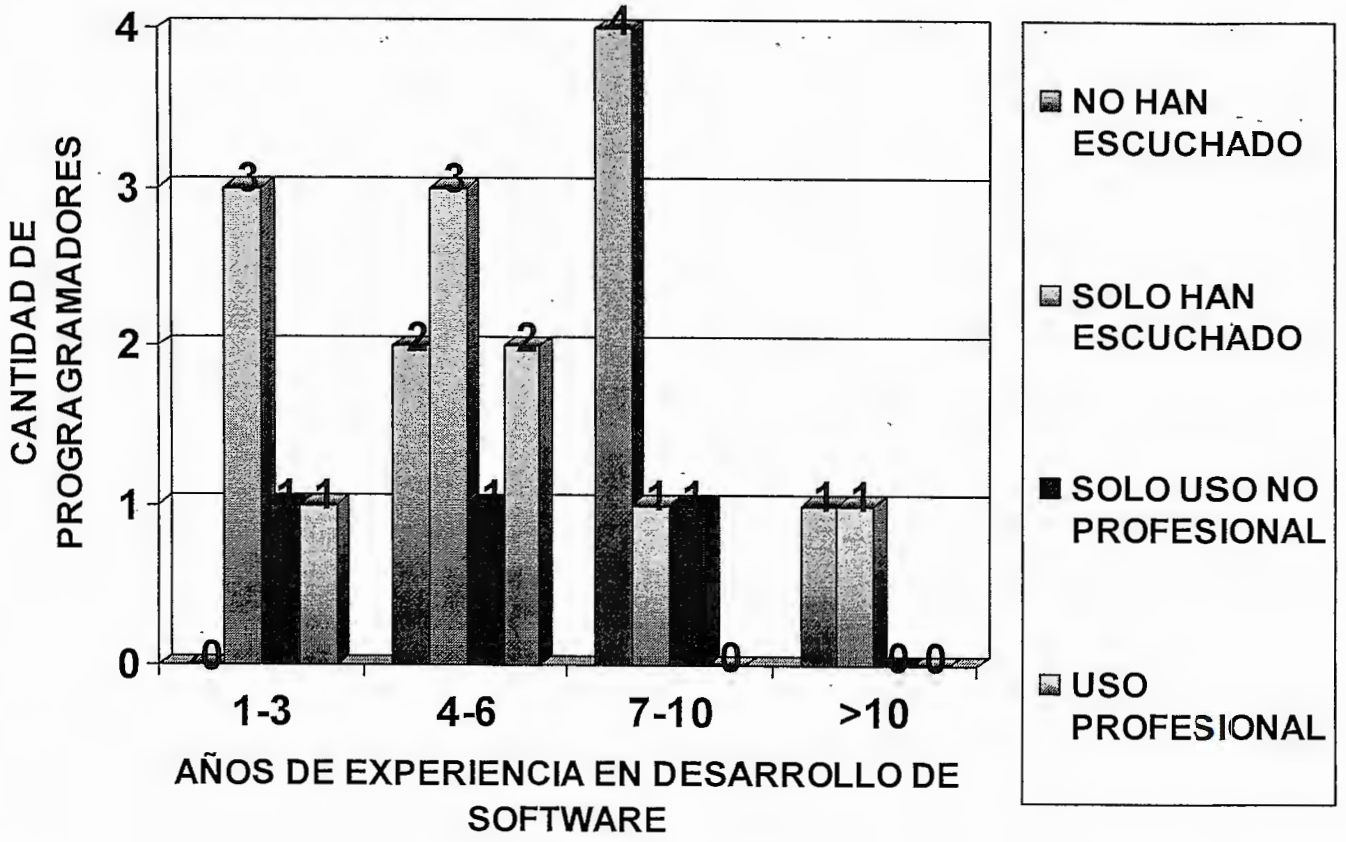
GRAFICA COMPARATIVA SOBRE EL USO DE METODOLOGIAS DE DESARROLLO DE SOFTWARE SEGÚN CONSULTA REALIZADA A 2I PROGRAMADORES EN EL MES DE ABRIL DE 2004



GRAFICA COMPARATIVA SOBRE EL USO DE METODOLOGIAS DE DESARROLLO DE SOFTWARE SEGÚN CONSULTA REALIZADA A 2I PROGRAMADORES EN EL MES DE ABRIL DE 2004



GRAFICA COMPARATIVA SOBRE EL USO DE HERRAMIENTAS OPEN SOURCE SEGÚN ENCUESTA REALIZADA A 2I PROGRAMADORES EN EL MES DE ABRIL DE 2004



ANEXO F

GLOSARIO TECNICO

Glosario Tecnico

DOP	Departamento de Ortesis y protesis
API	Application Programming Interface. Interface de Programacion de Aplicaciones
Artefacto	Producto o resultado final elaborado por un miembro del equipo que desempeña un rol asignado
AWT	Abstract Windows Toolkit. Paquete java para el manejo de ventanas en interaces de usuario
CASE	Acrónimo inglés de **C**omputer **A**ided **S**oftware **E**ngineering, que viene a significar Ingeniería de Software Asistida por Computadora.
Cristal Clear	Metodologia Agil creada por Allistair Cockburn, diseñada para equipos de trabajo pequeños, enfatiza la comunicación y centralidad en las personas.
Extreme Programming	La Programación Extrema (del inglés ExtremeProgramming) es uno de los llamados procesos o metodologías ágiles (ProcesoAgil) de desarrollo de software. Consiste en un conjunto de prácticas (ver LasPracticas) que a lo largo de los años han demostrado ser las mejores prácticas de desarrollo de software, llevadas al extremo, fundamentadas en un conjunto de valores (LosValores).
Firebird	Es un proyecto Open Source del Gestor de Base de datos Interbase.En 1997 InterBase se convirtió en una subsidiaria de Borland International. En el año 2000, la versión 6 de InterBase es liberada bajo una licencia (IPL) basada en la licencia MPL, quedando su desarrollo bajo el esquema de código abierto (Open Source).
IDE	Un **ambiente integrado de desarrollo** (*Integrated Development Environment* o IDE) es un programa de computadora que consiste de un editor de texto, un compilador, un intérprete, herramientas de automatización y un depurador. Aunque existen varios ambientes de desarrollo multilingüísticos, por lo general son dedicados a un lenguaje de programación específico, como el IDE de Visual Basic. En ocasiones son incluídos también un sistema de control de versión y herramientas para la construcción de interfaces gráficas.
Ingenieria del Sofrtware	Es el conjunto de métodos, técnicas y herramientas que controlan el proceso integral del desarrollo de software y suministra las bases para construir software de calidad de forma eficiente en los plazos adecuados
MDI	Interface de documento multiple.
Metodologias Ágiles	Nombre dado al conjunto de metodologias que enfatizan los principios del manifiesto agil
MySQL	MySQL es una de las bases de datos más populares desarrolladas bajo la filosofía de código abierto. La desarrolla y mantiene la empresa MySql AB pero puede utilizarse gratuitamente y su código fuente está disponible.
NetBeans	IDE Open Source para el desarrollo de aplicaciones Java
Open Source	Código abierto (open source en inglés) es el término por el que se conoce al software distribuido y desarrollado en una determinada forma. Este término empezó a utilizarse en 1998 por algunos usuarios de la comunidad del software libre, tratando de usarlo como reemplazo al ambiguo nombre original del software libre (free software).El significado obvio del término "código abierto" es "se puede mirar el código fuente", lo cual es un criterio más débil y flexible que el del software libre; un programa de código abierto puede ser software libre, pero también puede serlo un programa semi-libre o incluso uno completamente propietario.

Open Source	Nombre dado al software de codigo abierto, es decir aquel en que su codigo fuente puede ser modificado, son de libre distribucion y sin costo por el uso del mismo.
Practicas	Lo que hacen los programadores, se refiere a las actividades que los programadores realizan para crear en programaciòn extrema. Algunas ya existian antes que XP, pero siempre estan en vigencia.
Roles	Es el conjunto de actividades que asume un miembro del equipo de desarrollo y define el trabajo realizado dentro del proceso de desarrollo. Cada rol tiene asociado uno o mas artefactos
SIPOP	Sistema de Produccion para el departamento de Òrtesis y Prótesis
SWING	API de programacion java que permite desarrollo de interfaces de usuario, extiende las capacidades de AWT
UI	User Inteface. Interface por medio de la cual el usuario interactua con la aplicación.
UML	Lenguaje Unificado de Modelado, utilizado para el modelaje de sistemas en especial de software.Es un estandar bajo la tutela de OMG
Valores	Los valores que asumen los programadores para realizar su trabajo: comunicaciòn, simplicidad, retroalimentaciòn y coraje.
XP	Extreme Programming. Programación Extrema. Proceso agil creado por Kent Beck.
POO	Programación Orientada a Objetos
Java	Lenguaje de programación orientado a objetos.
Órtesis	Dispositivo utilizado para proteger, soportar o mejorar la función de segmentos del cuerpo que se mueven.
Prótesis	elemento artificial, fabricado para reemplazar las superficies de articulaciones que se han dañado y no pueden ser recuperadas nuevamente.
Ordén De Producció	Entidad del SIPOP utilizada para registrar el tipo de producto a fabricar, nombre de la persona a quien se le fabricará, nombre del técnico quien la fabricará fechas del inicio y fin de la fabricacion asi como la fecha de entrega y los estados de la etapa de fabrcacion (EN PROCESO O FINALIZADO)
Requisición	Documento para registrar materiales para la elaboración de un producto, registrado en el módulo cliente Requisiciones