Versión 2.0, octubre 1 de 2003

METODOLOGÍA PARA LA ADMINISTRACIÓN DE PROYECTOS DE DESARROLLO DE SOFTWARE

DATOS DEL DOCUMENTO			
Caracteristica	Documento borrador		
Ubicación	Archivo documentos poryecto cordata	Distribución:	
Ubicación digital			
Medio	Microsoft Word 2002 (10.0)	Paginas: 8	Imp.
Circulación	Restringida.		
Cítese como:	Naranjo C, L. J. Arias, R. Lastra. 2003. Metodologías para la Administración de Proyectos de desarrollo De Software – INVEMAR. Santa Marta D.T.C.H., Colombia, 28 pp.		
Esta obra está protegida por las normas de derechos de autor. Se permite la reproducción parcial o total citando apropiadamente la fuente.			

INTRODUCCION

El uso adecuado de una metodología de administración de proyectos de software proporciona un orden a las actividades del grupo, especifica qué es lo que se debe construir, permite dirigir y planear las tareas de los que desarrollan y del equipo, proporciona criterios para hacer seguimiento y medir productos y actividades. Adicionalmente simplifica el mantenimiento de la aplicación, el control de la calidad del producto, y la reutilización de componentes de software.

En consideración a las requerimientos del Instituto en materia de tecnologías de la información y a la necesidad de racionalizar y optimizar el uso de los recursos en este documento se plantea un proceso común para el desarrollo de aplicaciones, junto con las normas básicas de documentación que deben aplicarse durante la ejecución de las actividades relacionadas con la administración de los proyectos de desarrollo de software, en su ciclo completo. Las metodologías y normas expuestas se aplicaran de preferencia sobre otras normas similares en todas las etapas del ciclo de desarrollo de los proyectos de software. Siempre que sea necesario el documento mismo se someterá a evaluación y actualización.

1 LOS REQUERIMIENTOS DEL INVEMAR Y LOS OBJETIVOS DE LA METODOLOGIA DE ADMINISTRACION DE PROYECTOS

Las necesidades de INVEMAR en el área del desarrollo de sistemas con apoyo de tecnologías de la informática actualmente se centran en dos grandes proyectos que se integran horizontalmente entre si y verticalmente con proyectos de interés a escala nacional, regional y local, ellos son el Sistema de Información Ambiental Marino y el Sistema de Información para la Gestión Marina Ambiental. El primero ofrece como productos herramientas para el análisis de información ambiental y el apoyo a la toma de decisiones y el segundo herramientas orientadas a la planeación y a la optimización de la gestión.

Adicionalmente el Instituto trabaja en un ambiente de software abierto, con múltiples aplicaciones de uso general unas y especializadas otras que ayudan a organizar y analizar los datos y la información acopiada durante el desarrollo de los proyectos y actividades de investigación.

La complejidad y diversidad de las metas fijadas para cada uno de los proyectos de desarrollo de software, a lo que es necesario añadir la especialidad de las temáticas abordadas hace necesario implementar una metodología para su administración a fin de garantizar que los productos cumplan con los requerimientos y reúnan las funcionalidades requeridas. Los objetivos principales de una metodología de administración de proyectos son:

- Crear soluciones que se acerquen mejor a los requerimientos de la organización y de los usuarios.
- Optimizar el tiempo requerido en el ciclo de desarrollo del proyecto
- Aminorar el costo tecnológico
- Organizar eficazmente personas y recursos para el éxito del proyecto, estableciendo roles, responsabilidades y planificando las actividades.
- Mejorar el éxito de las cosas planeadas.
- Mejorar la respuesta a cosas no planeadas.
- Crear soluciones confiables y escalables.
- Administrar el riesgo inherente a todos los proyectos de desarrollo

2 METODOLOGIAS PARA LA ADMINISTRACION DE PROYECTOS DE DESARROLLO DE SOFTWARE

En general una metodología para la administración de proyectos de software busca asignar roles a las personas involucradas en el desarrollo, proporcionar un plan para la puesta en marcha y uso del sistema, establecer los mecanismos de comunicación necesarios, determinar de manera simple cual es la relación entre los componentes, evaluar el impacto de ajustes o cambios al sistema y en general proveer la información requerida para hacer las tareas de mantenimiento del sistema.

De las metodologías que tienen un reconocimiento más amplio, aplicables a la administración de proyectos de desarrollo de software en el INVEMAR se identificaron las siguientes

- Proceso Unificado Racional (RUP)
- Ciclo de Desarrollo de Sistemas en Cascada (SDLC) de la que existen múltiples variantes
- Metodologías de programación extremas (XP)

Proyectos de gran alcance son mejor implementados con las metodologías RUP y SDLC, proyectos en los que se requiere respuestas rápidas con la metodología XP. Ninguna de las alternativas se adapta exactamente a los requerimientos de una organización en particular por lo que la propuesta es tomar una de ellas como eje central y combinar algunos de los elementos que ofrecen las otras de manera sistemática y organizada.

3 PROPUESTA METODOLOGICA PARA INVEMAR

Se propone como estrategia para administrar los proyectos de desarrollo de software en su ciclo de vida completo, considerando la estructura organizacional actual y las políticas operativas del INVEMAR, una combinación de la metodología SDLC en cascada aplicada por componentes y la aplicación de las técnicas de documentación basadas en diagramas características del Lenguaje de Modelado Unificado (UML) propio del Proceso Unificado (RUP). Ambas ampliamente probadas, iterativas, parametrizables y de uso global.

Por consiguiente se consideran necesarias ejecutar cada una de las siguientes etapas durante el proceso de administración de los proyectos de software:

1. Diseño del modelo conceptual, formulación de la visión del proyecto e identificación de los componentes principales.

2. Diseño de la arquitectura para cada componente
   2.1. Definición de los requerimientos y diseño detallado
   2.2. Construcción
   2.3. Estabilización del producto

3. Integración de los componentes
   3.1. Diseño detallado
   3.2. Construcción
   3.3. Estabilización del producto

4. Estabilización global del sistema

5. Funcionamiento y mantenimiento

3.1 Diseño del modelo conceptual, formulación de la visión del proyecto e identificación de los componentes principales

En esta fase se obtiene la visión de lo que realmente se desea construir, objetivos y limitaciones.

¿Qué problema estamos tratando de resolver?
¿Quiénes son los usuarios del sistema?
¿Quiénes son los que necesitan el sistema?
¿Cuáles son las características del producto?
¿Cuáles son los requerimientos funcionales?
¿Cuáles son las limitaciones de diseño?
¿Cuáles son las limitaciones técnicas?
¿Cuáles son los requerimientos no funcionales?

¿Cuál es la justificación del proyecto?
¿Cuáles son los factores críticos a considerar para el éxito del proyecto?
¿Cuál es el impacto esperado del proyecto?

El producto esperado para esta etapa es un documento conceptual general del proyecto o **plan de desarrollo del proyecto**, en el que se especifican sus componentes y los requerimientos de funcionalidad más generales. Como diagramas de documentación apropiados se consideran válidos los de **arquitectura del sistema y los diagramas STRADIS** de flujos de información y procesos a nivel global.

A cada componente se le asigna un grado de complejidad, calificable de uno a diez, el cual esta determinado por:

- El alcance funcional
- Interacción con otros sistemas
- Número y tipo de usuarios
- Número y atributos de los datos
- Número de procesos
- Volumen de transacciones
- Complejidad de los algoritmos requeridos
- Localización geográfica de usuarios
- Relaciones entre los datos
- Tecnología requerida

Con los datos anteriores se establecen prioridades y un plan general para el desarrollo del proyecto, a su vez para cada componente y para la etapa de integración de los mismos es necesario preparar el cronograma de desarrollo y estipular los recursos necesarios.

3.1.1 Definición de los requerimientos y diseño detallado para cada componente

El diseño del sistema implica a todas las actividades de análisis que partiendo de los requerimientos crean un modelo del sistema propuesto a un nivel de abstracción que no incluye los aspectos físicos.

Comprende tres pasos: el diseño conceptual, el diseño lógico y el diseño físico. El **diseño conceptual** se considera como un análisis de actividades que se representan mediante los diagramas de casos de uso. Responde a las preguntas quién, qué, cuándo, dónde y por qué de la solución y consiste de las siguientes tareas:

- Identificar los usuarios y sus roles
- Obtener datos de los usuarios
- Evaluar la información
- Documentar los escenarios de uso
- Validar con los usuarios
- Validar contra la arquitectura de la organización

Los requerimientos de usuario, formulados de manera verificable, se resumen en documentos y modelos mediante **diagramas de casos de uso** que describan en forma no ambigua el sistema que se va a construir. Opcionalmente es conveniente recoger los requerimientos del usuario en forma de historias de usuario.

Si de la lista de requisitos se concluyera la necesidad de implementar el componente para su ejecución en ambiente Web, es necesario considerar la forma como se presenta la información, vínculos, gráficas e imágenes, técnica de navegación entre páginas, esquema de seguridad, módulos de software incluyendo aquellos que por su naturaleza no apliquen técnicas de desarrollo orientadas a objetos, marcos, formularios, hojas de estilo en cascada. El resultado de este proceso analítico se representará en diagramas UML del tipo modelo de navegación de los cuales como mínimo se incluirá el **mapa navegacional**, aunque son deseables los diagramas de contexto de navegación y vínculo de navegación.

El diseño lógico traduce los escenarios de uso creados en el diseño conceptual en un conjunto de **objetos y sus servicios**, organizando y detallando la solución las reglas y políticas específicas del problema a resolver.

El producto final del diseño lógico se plasma en el **modelo conceptual y lógico de datos y el modelo de comportamiento**. El modelo conceptual representa los conceptos más significativos en el dominio del problema utilizando clases de objetos, asociación entre clases de objetos y atributos de las clases describe la parte estática del problema. El modelo de comportamiento, define la parte dinámica, es decir, cual debe ser el comportamiento en cada situación y la forma de proceder. **Los diagramas de secuencia y de estados** son parte de este modelo.

El diseño lógico comprende las siguientes tareas:

- Identificar y definir los objetos de negocio y sus servicios
- Definir las interfases
- Identificar las dependencias entre objetos
- Validar contra los casos de uso
- Comparar con la arquitectura de la empresa
- Revisar y refinar tanto como sea necesario

Para definir los objetos de negocios y sus servicios se puede usar la técnica de análisis nombre-verbo de los escenarios de uso. También se puede emplear la técnica sujeto-verbo-objeto directo. En estas técnicas los sujetos y el objeto directo son los candidatos a objetos de negocio y los verbos activos son los candidatos a servicios.

Los objetos deben verificarse y probarse de tal manera que se asegure que los módulos operen como unidades completas de trabajo. Las tareas de verificación incluyen:

- Una verificación independiente
- Pre y post condiciones
- Lógica y funcionalidad individual
- Una verificación dependiente
- Verificación de dependencias
- Que operen como una unidad específica de trabajo

Un servicio es una unidad con capacidad de cómputo. Un servicio debe satisfacer lo siguiente:

- Ser seguro, lo que equivale a un uso correcto y con autorización
- Ser válido, qué tareas o reglas se pueden aplicar
- Manejar excepciones, informando al cliente
- Contar con un catálogo de servicios que constituye un repositorio de servicios

Con el fin de que la aplicación resulte flexible ante los cambios de requerimientos y/o de tecnología, en el diseño lógico se agrupan los servicios identificados en uno de tres niveles posibles: De usuario o cliente, de negocio o aplicaciones y de datos.

El servicio de usuario lo forman las personas, aplicaciones, otros servicios o la combinación de éstos que operan del lado del cliente. Generalmente involucra una interfase gráfica (GUI) aunque puede no serlo (mensajes o funciones), maneja todos los aspectos de la interacción con la aplicación. El objetivo central es minimizar el esfuerzo de conocimiento requerido para interpretar la información. Un servicio de usuario incluye un contenido (qué se necesita comunicar al usuario) y una forma (cómo se comunica el contenido) cuando es necesaria la comunicación.

Los servicios de negocio o aplicaciones convierten datos recibidos de los servicios de datos y de usuario en información (datos + regla de negocio) y pueden usar otros servicios de negocio para completar su tarea.

Las tareas de los servicios de negocio son:

- Dar formato a los datos
- Obtener y mover datos desde y hasta los servicios de datos
- Transformar los datos en información
- Validar los datos inmediatamente en el contexto o en forma diferida una vez terminada la transacción.

Los servicios de datos son los servicios de bajo nivel que apoyan los servicios de negocio e incluyen:

- Declaración del esquema y su evolución (estructuras de datos, tipos, acceso indexado, SQL, APIs)
- Respaldo y recuperación (recuperación de datos si un evento falla)
- Búsqueda y lectura (búsquedas, compilación, optimización y ejecución de solicitudes, formación de un conjunto de resultados)
- Inserción, actualización y borrado (procesar modificaciones consistentemente transaccional). Una transacción es atómica (ocurre o no), consistente (preserva integridad), aislada (otras transacciones ocurren antes o después) y durable (una vez completada, sobrevive).
- Bloqueo (maneja el acceso concurrente a los datos)
- Validación de datos (verifica la integridad del dominio, triggers y gateways para verificar el estado de los datos antes de aceptarlos, manejo de errores)
- Seguridad
- Administración de la conexión (mecanismos básicos para establecer una sesión de los servicios de datos). Establecer una conexión involucra: una identificación, la colocación y provisión de datos, tiempo de sesión, el tipo de interacción (conversacional, transaccional, multiusuario, monousuario).
- Distribución de datos (Distribuye información, a múltiples unidades de recuperación, bases de datos heterogéneas, según la topologías de la red).

Por seguridad se entiende como la protección de los sistemas de información contra el acceso desautorizado o la modificación de datos en almacenamiento, procesamiento o tránsito, incluyendo las medidas necesarias para detectar, documentar, y contrarrestar tales amenazas.

En particular, la seguridad del código y de los procesos de software debe ser considerada no solamente durante la fase del diseño y desarrollo sino garantizada durante la operación y el mantenimiento.

Durante el diseño lógico del sistema es necesario evaluar las técnicas a implementar para garantizar el acceso seguro a los objetos, operaciones, los permisos de usuario, de grupos

(roles) y servicios. Estas son dependientes tanto de la plataforma tecnológica como de la naturaleza de la aplicación (XML Web services, .NET Remoting, JSP, ASP, etc).

Las interfaces tienen las siguientes partes:

- Nombre
- Precondiciones, lo que debe estar presente antes de ejecutarse
- Postcondiciones, estado final
- Capacidad o funcionalidad (SQL, pseudocódigo, función matemática)
- Dependencias

Establecidas las dependencias entre objetos se definen eventos, sucesos o condiciones que inicien la ejecución de tareas coordinadamente. Para ello se debe considerar lo siguiente:

- Identificar los eventos disparadores (triggers)
- Determinar cualquier dependencia (existencial o funcional)
- Determinar cualquier problema de consistencia o secuencia
- Identificar cualquier regulación de tiempo crítica
- Considerar algún problema organizacional (transacciones)
- Identificar y auditar los requerimientos de control
- Determinar lugares y dependencias a través de la ubicación
- Determinar cuando el servicio que controla la transacción es dependiente de los servicios contenidos en otros objetos de negocio

La validación del modelo lógico debe verificar que éste sea: Completo, debe representar todos los escenarios de uso; correcto, el comportamiento lógico debe corresponder con el comportamiento conceptual; claro, los objetos de negocio y servicios no deben ser ambiguos.

El diseño físico traduce el diseño lógico en una solución implementable y costo-efectiva o económica. El diseño físico está íntimamente ligado a una alternativa tecnológica. En esta fase se refinan los esquemas conceptuales creados durante el diseño conceptual, eliminando las estructuras de datos que no se pueden implementar de manera directa sobre el modelo seleccionado para la base de datos: Orientado a objetos o relacional. Una vez hecho esto, se obtiene un primer esquema lógico que se valida mediante la normalización y frente a las transacciones que el sistema debe llevar a cabo, tal y como se refleja en las especificaciones de requisitos de usuario. El esquema lógico ya validado se puede utilizar como base para el desarrollo de prototipos. Una vez finalizada esta fase, se dispone de un esquema lógico para cada vista de usuario que es correcto, comprensible y sin ambigüedad

El componente es la unidad de construcción elemental del diseño físico. Las características de un componente son:

- Se define según cómo interactúa con otros
- Encapsula sus funciones y sus datos
- Es reusable a través de las aplicaciones

- Puede verse como una caja negra
- Puede contener otros componentes

En el diseño físico se debe cuidar el nivel de granularidad (el tamaño de un componente depende de su funcionalidad, es decir, de tamaño tal que pueda proveer de una funcionalidad compleja pero de control genérico), la agregación y la contención (un componente se puede reusar utilizando técnicas de agregación y contención, sin duplicar código).

El diseño físico define:

- Los componentes
- Las interfases o conectores de los componentes
- El diseño para distribución – debe minimizarse la cantidad de datos que pasan entre los componentes y éstos deben enviarse de manera segura por la red. Comprende especificar a distribución de los componentes en la red y la de los repositorios físicos de datos.
- El diseño para multitarea – debe diseñarse en términos de la administración concurrente de dos o más tareas distintas por una computadora y el multithreading o múltiples hilos de un mismo proceso
- El diseño para uso concurrente – el desempeño de un componente remoto depende de si está corriendo mientras recibe una solicitud.
- La tolerancia a fallas y la recuperación de errores
- La depuración – crear procedimientos para detectar y corregir errores.
- La validación de parámetros - a la entrada antes de continuar con cualquier proceso.
- La protección de los recursos críticos –manejar excepciones para evitar la falla o terminación sin cerrar archivos, liberar objetos sincronizados o memoria.
- La protección de datos importantes – contar con una excepción a la mitad de la actuación en las bases de datos.
- La protección integral de las transacciones de negocios – los errores deben regresarse al componente que llama.

De las tareas anteriores la más importante es la distribución de los datos que pueden ser centralizados, una partición, un extracto o una réplica.

Los datos centralizados equivalen a una base de datos maestra ubicada en un lugar central. No hay copias de los datos.

Una partición de datos es una segmentación de la base de datos maestra. Es útil cuando los datos se pueden fragmentar fácilmente y actualizarse en un sitio local con cambios frecuentes. No hay sobreposición entre particiones. En una partición horizontal cada hilera existe en una sola base de datos. En una partición vertical cada columna es contenida en una y solo una base de datos.

Un extracto de datos es una copia de toda o una porción de la base de datos maestra. No se permite la actualización. Se usa un timestamp o etiqueta de tiempo para indicar qué tan viejos son los datos.

Una réplica de datos es un fragmento de la bases de datos maestra que se puede actualizar. En una réplica de datos el sitio de actualización cambia a un sitio local. No se permiten actualizaciones en la base de datos réplica y en la base de datos maestra a la vez, por lo que debe de haber un proceso de sincronización entre ambas.

La tendencia actual en la arquitectura cliente/servidor es crear el back-end como un servidor robusto multitareas y multithreading y el front-end como un cliente muy liviano que no acapare al servidor comunicándose entre sí en una plataforma Internet con protocolos estándar en redes heterogéneas. El esquema de implementación mas frecuente es el de cuatro niveles, con un primer nivel para el cliente, un segundo nivel para la herramienta integradora (ASP, JSP), un tercer nivel para el servidor de aplicaciones con sus componentes y un cuarto nivel para las bases de datos.

3.1.2 Construcción

En esta etapa el producto se desarrolla, es decir, se crea el código correspondiente al resultado de la fase de diseño, siguiendo los patrones y la arquitectura escogida y se prueba. Para hacerlo se programan una o varias entregas de versiones preliminares del producto a que buscan garantizar su calidad y medir hasta que punto la solución en desarrollo satisface las necesidades del usuario. Antes de finalizar esta etapa se debe estar seguro de que el producto cumple con los requerimientos de usuario final y con los objetivos propuestos al comienzo del proyecto.

Coordinar el trabajo del equipo de desarrolladores del proyecto es aquí una actividad esencial.

Los documentos que genera esta fase del proyecto son:

- El documento técnico del software: Diagramas de clases y de procesos
- El modelo físico de la base de datos
- El diccionario de datos.
- El documento de usuario final, el cual debe incluir una guía de instalación y la descripción de la interfaz gráfica del mismo.

3.1.3 Estabilización del producto

La fase de estabilización comienza con las pruebas beta del producto y termina cuando el mismo es aceptado completamente por todos los interesados. Las pruebas dentro de esta fase tienen como objetivo medir hasta que punto el sistema responde a las necesidades reales de los usuarios y la confiabilidad de la aplicación desarrollada.

3.2 Integración de los componentes

3.2.1 Diseño

En esta etapa se atiende a las necesidades de integración de los componentes creados entre si y de estos con los sistemas ya existentes de acuerdo a los requerimientos de los usuarios y las expectativas acerca de la funcionalidad del producto. La integración implica definir los requerimientos para los sistemas existentes, identificar las arquitecturas presentes, formular un plan de integración para contenidos y datos, precisando si se requiere migrar datos, la naturaleza de estos, su volumen, la viabilidad de la migración y la estrategia para hacerla.

Sea cual sea el diseño de integración propuesto es necesario reducir al mínimo su impacto sobre la portabilidad, la seguridad, evolución y posibilidad de reutilización de los componentes.

3.2.2 Construcción

Desarrollar el plan de integración conectando cada uno de los componentes de acuerdo a los procedimientos diseñados y verificando la interoperabilidad.

3.2.3 Pruebas

Aplicar los test para las pruebas de aceptación a fin de verificar si el sistema cumple con las especificaciones propuestas. Parte integral del plan de pruebas reside en verificar que se ejecuten las correcciones necesarias a tiempo, comprobando que estas no afectan el funcionamiento de procedimientos que antes de la corrección funcionaban adecuadamente.

3.2.4 Estabilización global del sistema

Verificar el correcto funcionamiento del sistema como un todo y la eficaz integración de sus componentes. Diseñar y aplicar el **plan de pruebas** de aceptación de acuerdo con los usuarios. Ejecutar los procedimientos necesarios para evaluar la calidad del sistema y mejorar los aspectos que lo requieran.

3.3 Funcionamiento y mantenimiento

La primera tarea de esta fase consiste en la distribución e instalación del software en el entorno donde se va a desarrollar. Posteriormente, es necesario realizar una serie de pruebas para comprobar que no han surgido problemas, en tal caso será recomendable realizar una depuración, así mismo es necesario:

- Brindar asistencia técnica, orientar y capacitar a los usuarios, a partir del plan de desarrollo informático, con el propósito de garantizar el desempeño eficaz en sus funciones.
- Ejecutar el plan de distribución de la aplicación
- Mantener actualizada el área tecnológica, mediante la investigación con el propósito de promover el uso de tecnología adecuada para el desarrollo informático.
- Corregir y depurar los errores del sistema que no hayan sido detectados o que aparezcan como resultado de la evolución en las tecnologías del ambiente de trabajo de la aplicación
- Monitorear el desempeño del sistema
- Ejecutar las tareas de backup y de mantenimiento de los componentes de la aplicación de acuerdo a un programa establecido
- Adicionar funciones de acuerdo a requerimientos de los usuarios

4 DESARROLLO DE SOFTWARE DE AMBITO LIMITADO

Cuando sea necesario desarrollar pequeñas y medianas aplicaciones que den respuesta a necesidades concretas y limitadas de la organización, es necesario documentar los siguientes aspectos:

- Definir el problema que se quiere resolver
- Justificar el por qué debe resolverse el problema con una aplicación de software, así como la aplicación en sí.
- Definir cuales y cuantos son los posibles usuarios del sistema
- Especificar sobre que equipos va a funcionar la aplicación (plataforma, sistema operativo, procesador, memoria)
- El nivel de importancia de los datos
- La naturaleza de los datos que recoge y almacena el sistema
- Evaluar, cuando existan, los formularios para capturar esos datos, sean en papel o digitales.
- Describir detalladamente los requerimientos adicionales de la aplicación (consultas, reportes, formulas matemáticas para el caculo de valores no almacenados, etc.)

5 MODELO DE EQUIPO DE TRABAJO

Para el desarrollo de un producto de software los equipos deben ser multidisciplinarios, todos sus miembros deben compartir las responsabilidades, una visión clara y común del proyecto y altos estándares de calidad. Dentro del modelo de equipos de trabajo no existen líderes únicos sino roles estos son:

5.1 Administrador del producto

Su objetivo principal es entregar un producto que satisfaga las necesidades del usuario. Tiene como funciones generar el documento global que describe el proyecto, transmitir la visión del proyecto hacia el interior del grupo de trabajo y hacia la organización, especificar los recursos requeridos para la realización de las tareas, especificar el alcance del proyecto, describir como será entregada la información del producto a los usuarios.

5.2 Administrador de programa

Son sus responsabilidades, entregar el producto correcto de acuerdo a las especificaciones acordadas con los usuarios en el tiempo correcto, administrar el alcance del producto y sus especificaciones, administrar los roles y garantizar la integridad del equipo de trabajo.

5.3 Desarrollador

Es el encargado de construir el producto, como tal participa en su diseño, estima el tiempo y esfuerzo requeridos para desarrollar el producto, sirve al equipo como consultor en tecnología, y da soporte durante el proceso de instalación del producto.

5.4 Auditor o probador

Su propósito es el de desarrollar y aplicar una estrategia de pruebas, que permita detectar y corregir errores y falencias del producto.

5.5 Educador

Su función principal es la de diseñar e implementar los sistemas de soporte y ayuda al usuario.

5.6 Administrador logístico

Soportar el producto durante las pruebas y garantiza el correcto funcionamiento de la infraestructura física del proyecto.

Si los recursos o el tamaño del proyecto no permiten asignar una persona o grupo de personas para cada rol, se hace necesario combinar en una sola persona más de un rol teniendo en cuenta que los roles asignados a un individuo tengan incompatibles intrínsecas.

Compatibilidades permitidas en roles compartidos

	Administrador del proyecto	Administrador del programa	Desarrollador	Auditor	Educador	Administrador logístico
Administrador del proyecto		**N**	**N**	**P**	**P**	**NP**
Administrador del programa	**N**		**N**	**NP**	**NP**	**P**
Desarrollador	**N**	**N**		**N**	**N**	**N**
Auditor	**P**	**NP**	**N**		**P**	**P**
Educador	**P**	**NP**	**N**	**P**		**NP**
Administrador logístico	**NP**	**N**	**N**	**P**	**NP**	

N = No recomendada; **P** = Posible; **NP** = No Posible por incompatibilidades profesionales, de experiencia y/o conocimiento.

6 DOCUMENTOS A GENERAR DURANTE LA EJECUCION DE UN PROYECTO DE DESARROLLO DE SOFTWARE

El propósito de la documentación es registrar la información producida por un proceso o actividad, aclarando su necesidad, los requisitos previos a su realización y el destino y significado de la información generada.

6.1 DEL ESTILO DE LOS DOCUMENTOS

Los manuales y los documentos que se generen seguirán las normas contenidas en el **Manual de Estilo de INVEMAR,** especialmente en lo pertinente al tamaño y familia de la fuente y la enumeración y disposición de las partes del mismo. El lenguaje de redacción tendrá en cuenta siempre al público objetivo.

Los encabezados del documento y la página de presentación contendrán el nombre de la aplicación y la versión.

6.2 DESCRIPCION DE LOS DOCUMENTOS

6.2.1 Plan de desarrollo del proyecto

Este documento contiene:

- Descripción de los objetivos del proyecto
- Descripción de los principales componentes
- Prototipo de las aplicaciones
- Plan de configuración de la aplicación incluyendo sus relaciones con aplicaciones ya existentes o planeadas para el corto plazo y definidas
- Plan de administración de nuevos recursos
- Plan de aseguramiento de la calidad
- Plan de prueba y evaluación
- Cronograma de actividades

6.2.2 Requerimientos del usuario

Indicar por cada requerimiento objetivo, perfil del usuario, la forma como se evaluará la satisfacción del usuario con la solución propuesta, las tareas que deben realizarse y la asignación de recursos y responsabilidades para ejecutarlas.

Los requerimientos llevan asociadas necesidades de software, maquinas, seguridad, rendimiento, operacionales y de instalación que se deben considerar como un todo.

6.2.3 Evaluación de necesidades del producto

Describe la razón por la cual el producto es requerido, las características que debe tener, los requerimientos que debe satisfacer. Analiza las limitaciones de las soluciones propuestas teniendo en cuenta costos, carga operacional, el soporte técnico requerido, beneficios

6.2.4 Arquitectura del sistema

Describe las interrelaciones entre los componentes del sistema y de estos con el software. Debe especificar capacidad de memoria, requerimientos de las interfaces de hardware, requerimientos de las interfaces para los usuarios, características de seguridad, parámetros del sistema, componentes reusables.

6.2.5 Descripción detallada de procesos

Especifica por cada proceso objetivo, tareas y actividades a ser ejecutadas, dependencias de otros procesos, tiempo esperado para la ejecución de la tarea, producto, interfaces, identificación de roles y responsabilidades.

6.2.6 Planificación y descripción de los procesos de transición

Indicando procedimientos para evaluar, organizar y migrar la información existente, los formatos existentes, los procesos y formatos transitorios y la evaluación y validación de los datos en la nueva aplicación, los recursos necesarios para realizar la actividad y los responsables.

6.2.7 Plan de auditoria

Indica los criterios que se aplicaran para evaluar la calidad del producto, el tiempo requerido para preparar las pruebas de depuración y el plan para su ejecución asignando las personas responsables de ejecutarlo. El plan debe incluir el registro detallado de las falencias encontradas en el producto y el seguimiento necesario para verificar que los cambios necesarios se hagan.

6.2.8 Lista de distribución de la aplicación

Especificar el medio que se utilizara (CD-ROM, correo etc.) y los recursos requeridos.

6.2.9 Instrucciones de instalación

De la aplicación o de sus actualizaciones, se indicara el orden en que debe ejecutarse la tarea, el número de la versión, lista de los componentes, necesidades de backup previos y procedimientos de recuperación.

6.2.10 Evaluación de la calidad

Establece los criterios necesarios para medir el desempeño y confiabilidad del sistema, por ejemplo tiempo de procesamiento por transacción, manejo de inconsistencias.

6.2.11 Manual de usuario

Este manual tiene como objetivo orientar al usuario sobre la manera optima de utilizar la aplicación, guiarlo sobre los procedimientos que debe seguir para instalarla y resolver las dudas que sobre los procedimientos se le presenten al tiempo de operar el sistema.

El manual contendrá como mínimo las siguientes partes:

- Introducción: Presentación del sistema, su origen, propósito y proyecciones.
- Objetivos del software
- Personas a quienes va dirigido
- Requisitos de instalación tanto de software como de hardware y procedimiento a seguir para instalar
- Diagrama de componentes y de funcionamiento
- Descripción general de organización de los diferentes usuarios del sistema
- Descripción de la interfaz gráfica completa, preferiblemente siguiendo la secuencia de funcionamiento de los módulos del sistema.
- Descripción de los procesos que se ejecuten por lotes y de la manera como se administran
- Los procedimientos a seguir para la extracción o inserción de datos a partir de archivos planos, de formatos diferentes o de otras bases de datos.
- Las reglas de validación de los datos
- Los mensajes de error su significado y los procedimientos a seguir cuando se presenten
- Los procedimientos a seguir para acceder a soporte técnico
- Glosario de términos técnicos del software y de la aplicación

6.2.12 Requisitos especiales de instalación u operación

- Software requerido para instalar la aplicación
- Pasos para configurar la instalación
- Procedimientos de seguridad y administración de las diferentes cuentas de usuarios
- Descripción de las herramientas administrativas disponibles y de sus modalidades
- Descripción de los procedimientos que se ejecutan en lotes incluyendo intervalo de tiempo o condición que obliga a la ejecución de la rutina, copia del código SQL.
- Instrucciones especiales en lo referente a recursos de máquina requeridos y a los procedimientos de backup

6.2.13 Registro de la instalación del software y de las actualizaciones

Se llevará un registro de los usuarios a los que se les ha instalado la aplicación anotando por cada caso: Nombre de la entidad, nombre de la dependencia y nombre de la maquina, versión instalada y fecha, persona que recibió el software, persona que instaló, especificar si se trata de una instalación completa o de una actualización.

6.2.14 Manual del sistema

El objetivo del manual del sistema es presentar una visión general del sistema mostrando como la aplicación esta estructurada y de como se interrelacionan sus partes. El manual del sistema mostrara los diagramas de clases, los diagramas de secuencia, los diagramas entidad relación, el diagrama de componentes, el diagrama de instalación, los diagramas de estado y el diccionario de la base de datos.

Precise por componente que datos entran y que datos salen, si existe algún procedimiento especial que requiera explicarse en detalle use pseudo-código.

Si un procedimiento requiere de algún tipo de librerías particulares indique como se llaman la versión utilizada, su objetivo y la carpeta en donde el sistema las busca.

Si el sistema utiliza variables de entorno particulares indique cuales son y el rango de valores permitidos.

Explique los mensajes de error y otras condiciones de excepción que se presenten en el sistema y el modo en que deben ser manejados.

Indicará el nombre de los formularios que contienen el código fuente, el lenguaje en que se codificaron y su objetivo. Si para su correcto funcionamiento tienen algún requisito especial indicarlo.

Si la aplicación emplea plantillas para separar la lógica del negocio de las interfaces explicar la sub-división de la interface y las plantillas, cuales plantillas se aplican en que procedimientos, las variables locales y globales, las librerías y paquetes usados para conectar plantillas y contenidos.

Si se han definido interfaces con otros sistemas explique los procesos de extracción o inserción de información, el formato de los datos, los requerimientos de configuración y de back-up.

Indique cuales reportes, archivos de supervisión (logs) o productos genera el sistema, las carpetas donde se almacenan y el tipo de archivo. Si alguno de los archivos es requerido para otro proceso señale a cual.

Explique la seguridad del sistema los niveles de usuario definidos y las técnicas aplicadas para asegurar los datos.

Documentar el código fuente en todos los casos, colocar comentarios claros sobre el propósito de una función en su encabezado, aplicar las convenciones fijadas para nombrar objetos, variables y funciones. Aplicar técnicas de indentación para las escritura del código uniformes.

Si es necesario anexe ejemplo de archivos logs y de reportes y de la manera como estos deben ser interpretados, de formularios utilizados para recoger datos, o hacer seguimientos de procedimientos.

6.2.15 Registro de reportes de problemas

Indicando el nombre del usuario, componente que falla, versión de la aplicación, la descripción del problema, la información de soporte del problema si existe (reporte, archivo o tarea que falla), la severidad del problema, la fecha del reporte, la fecha tentativa esperada para dar solución al problema, el estado en que se encuentra el problema a medida que se avanza en su solución (abierto, en depuración, mas información requerida, cerrado)

6.2.16 Requerimientos de cambios

Cada vez que se soliciten algún cambio se registrará la petición indicando necesidad, impacto en el sistema, persona que lo solicita, procedimiento para implementarlo y recursos requeridos.

6.2.17 Registro de actualización de documentos

Consignar fecha, propósito, persona responsable y mecanismo o medio por el cual se distribuye la nueva versión del documento.

6.2.18 Normas

Todo documento normativo debe indicar claramente su objetivo, el destinatario, los controles aplicables para verificar que la norma se implemente, y el procedimiento para el manejo de las excepciones

7 DOCUMENTOS CONSULTADOS

Object Management Group (OMG). Unified Modeling Language Specification, Version 1.3, June 1999.

Agustín Villena M, Taller de Metodologías Ágiles de Desarrollo de Software, Estrategias de XP.

Equipo Informática, Mercado de Energía Mayorista, Estrategia Tecnológica MEM 2002.

Ministerio de Hacienda, Costa Rica. Información General de Rational unified process (RUP).

Héctor Valencia Ramírez, Un Modelo del Proceso de Desarrollo de Software, Universidad EAFIT, Escuela de Ingeniería, departamento de Informática y Sistemas, Medellín 2002

David Card, A Practical Framework for Software Measurement and Analysis, Published in Systems Development Management, 34-1016, 2000

Francisco Rueda F. Metodologias de Diseño de Sistemas Multinivel. Universidad de los Andes. 2002.

Paul Hodgetts and Denise Phillips, eXtreme Adoption eXperiences of a B2B Start Up, 2001.

Ministerio de Administraciones Públicas, Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información, España 2002.

Tabla resumen de los documentos a generar durante la ejecución de un proyecto de desarrollo de software.
(**R**= Requerido, **O**= Opcional).

ETAPA DEL PROYECTO	DOCUMENTO	RESPONSABLE	
Diseño del modelo conceptual, formulación de la visión del proyecto e identificación de los componentes principales	Plan de desarrollo del proyecto	Administrador del proyecto	**R**
	Requerimientos del usuario		**R**
	Evaluación de necesidades del producto		**R**
	Arquitectura del sistema		**R**
	Diagrama STRADIS		**R**
			O
Diseño de la arquitectura para cada componente			
Definición de los requerimientos y diseño detallado	Requerimientos del usuario	Administrador del programa	**R**
	Diagramas de casos de uso		**R**
	Descripción detallada de procesos		**R**
	Modelo conceptual de datos		**R**
	Modelo lógico de datos		**R**
	Modelo de comportamiento		**R**
	Mapa navegacional		**O**
	Planificación y descripción de los procesos de transición		**O**
	Identificación de interfaces de usuarios gráficas, no gráficas e impresas		**R**

ETAPA DEL PROYECTO	DOCUMENTO	RESPONSABLE	
	Plan de auditoria		**R**
Construcción	Modelo físico de la base de datos	Desarrollador /Educador	**R**
	Diccionario de la base de datos		**R**
	Manual del sistema		**R**
	Manual del usuario		**R**
	Instrucciones de instalación		**O**
Estabilización del producto	Evaluación de la calidad		**R**
	Lista de distribución de la aplicación		**O**
Integración de los componentes			
Diseño detallado	Diagramas de casos de uso	Administrador del programa	**R**
	Diagrama de implementación		**R**
	Diagrama de componentes		**R**
Construcción	Manual del sistema	Desarrollador /Educador	**R**
	Manual del usuario		**R**
	Instrucciones de instalación		**O**
Estabilización del producto	Registro de la instalación del software y de las actualizaciones		**O**
	Evaluación de la calidad	Auditor	**R**
	Lista de distribución de la aplicación		**O**
Estabilización global del sistema	Requisitos especiales de instalación u operación	Desarrollador /Educador	**O**
	Evaluación de la calidad	Auditor	**R**
	Evaluación del rendimiento	Auditor	**R**

ETAPA DEL PROYECTO	DOCUMENTO	RESPONSABLE	
Funcionamiento y mantenimiento	Registro de la instalación del software y de las actualizaciones	Administrador logístico	**R**
	Registro de reportes de problemas		**R**
	Requerimientos de cambios		**R**
	Registro de actualización de documentos		**R**