Contenido

TEMARIO **2**

INTRODUCCIÓN **4**

INGENIERÍA DEL SOFTWARE **5**

EL INICIO **6**

GESTIÓN DE PROYECTOS **10**

INGENIERÍA DE SISTEMAS **20**

ANÁLISIS DE REQUERIMIENTOS **22**

DISEÑO DE LA SOLUCIÓN **30**

CONSTRUCCIÓN **47**

PRUEBAS **50**

LIBERACIÓN A PRODUCCIÓN **56**

BIBLIOGRAFÍA **57**

TEMARIO

LABORATORIO DE SISTEMAS DE INFORMACIÓN

Plan: 98 Clave: 1840 Semestre: 8º.
Créditos: 8
Licenciatura: informática
Área: desarrollo de sistemas
Horas por clase: 2
Clases por semana: 2

OBJETIVO GENERAL

Al finalizar el curso el alumno habrá adquirido experiencia práctica en el desarrollo de sistemas de información, mediante el desarrollo de un sistema y la aplicación de los conocimientos y las habilidades adquiridas durante sus estudios de licenciatura.

PLAN DE TRABAJO PROPUESTO

1. ANTECEDENTES

Uno de los objetivos de la carrera de Licenciado en Informática es el ejercicio práctico de los conocimientos adquiridos durante toda la carrera, es así que a punto de terminar, se creara una asignatura que cumpliera cabalmente con este propósito. **"Laboratorio de Sistemas de Información"** reúne a través de la vinculación con casos reales, la aplicación de todos sus conocimientos en la resolución e implantación de Sistemas de Información que lleven a las personas y a las organizaciones soluciones basadas en el manejo de la información.

Entre los propósitos principales de colaboración en la UNAM, esta asignatura se ha convertido en una proveedora directa de la satisfacción de necesidades internas dentro de la FCA así como de la propia UNAM, sin restringir la creatividad y accesibilidad a que los alumnos promuevan sus propios proyectos, siempre y cuando éstos tengan una razón de beneficio social para alguna organización pública o proyecto universitario, en bien de la comunidad universitaria.

El desarrollo de proyectos empresariales se da también como una opción, siempre y cuando estén acordes a los programas de vinculación empresarial; ya sea del propio programa emprendedores de la FCA o de la UNAM.

2. RESTRICCIONES

Considerando las anteriores premisas debe establecerse que los alumnos en estos semestres ya están laborando en organizaciones dedicadas al desarrollo e implementación de aplicaciones y actividades relacionadas con el manejo de información, es preciso observar como una restricción para presentar y aprobar esta asignatura, la presentación de algún proyecto relacionado con su actividad laboral que la vincule directamente a usar información ya conocida o elaborada dentro de la organización; el proyecto deberá ser inédito y de beneficio social.

El alumno deberá adecuarse a una metodología de trabajo.

El alumno deberá desarrollar en los estándares definidos por el titular de la asignatura o de los tutores.

El uso de un lenguaje de programación diferente al acordado o permitido, y que no se tengan las herramientas necesarias para ejecutarlo o modificarlo, se considera puntos menos.

3. METODOLOGÍA

El titular de la asignatura propone los proyectos a realizar por los alumnos.

El titular asigna y designa tutores especializados para cada uno de los proyectos, a fin de que éstos puedan orientar a los alumnos en el desarrollo; el tutor ayudará en la gestión necesaria para el desarrollo del proyecto.

Los alumnos forman equipos para la elaboración de los proyectos.

El número de equipos y de integrantes por equipo será de acuerdo a las características del proyecto, incluso considerando que pueden existir proyectos que se elaboren por una sola persona o bien por varios equipos, según sea la magnitud.

El número de integrantes lo definirá el titular de la asignatura.

La forma de organización es responsabilidad del equipo.

Los alumnos deberán presentar un plan de trabajo para la realización del proyecto.

El tutor deberá llevar un registro periódico de las actividades y avances de acuerdo al plan de trabajo que presentó el equipo al inicio del proyecto.

El proyecto deberá cumplir con los siguientes puntos:

Documentación
Análisis de procedimientos administrativos
Análisis conceptual
Diseño
Codificación
Pruebas de validación
El sistema funcionando y a satisfacción de las necesidades de la organización
Manual técnico
Manual de usuario

4. PUNTOS A CONSIDERAR (por el titular de la asignatura)

Dadas las características de la asignatura, el proyecto entregado completo, equivale a la aprobación de la asignatura; proyecto no terminado, se considera asignatura no aprobada.

Considerando los posibles problemas que se pueden presentar, el titular será, quien a consideración del tutor, formule prórrogas de tiempo para la entrega del proyecto o bien para su entrega parcial.

INTRODUCCIÓN

Durante el curso de Análisis y Diseño de Sistemas de Información se revisó a detalle la teoría y conceptos básicos de cada una de las fases que conforman el ciclo de vida de un sistema de información, el propósito de este documento es servir de guía para desarrollar, mediante el enfoque de ingeniería de sistemas, un proyecto, aplicando la teoría previamente estudiada (métodos, herramientas y técnicas para el análisis y diseño de sistemas de información) para obtener un sistema en papel, la programación del mismo se deja a un lado pues el alumno ya debe conocer algún lenguaje de programación para realizar esta actividad.

Sirvan estos apuntes para dar al alumno una perspectiva práctica de su futura vida profesional en el área de Análisis y Diseño de aplicaciones de software.

Atentamente,

L.I. Ma. Isabel Méndez Morales

México, D.F., enero de 2006.

INGENIERÍA DEL SOFTWARE[1]

La meta de todo desarrollador de aplicaciones de cómputo es crear productos que hagan bien las cosas, para esto se requiere apegar a una disciplina, a un enfoque de ingeniería.

La ingeniería del software es una disciplina o área de la Informática o Ciencias de la Computación, que integra procesos, métodos y herramientas para desarrollar y mantener software de calidad. Hoy día es cada vez más frecuente la consideración de la ingeniería, y el ingeniero de/l software comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir.

La ingeniería del software trata con áreas muy diversas de la informática y de las ciencias de la computación, tales como construcción de compiladores, sistemas operativos o desarrollos en Intranet/Internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas.

Los temas más sobresalientes de la ingeniería del software son:

Inspección de software crítico.
Software de tecnologías de procesos de negocios.
Arquitecturas de software distribuido.
Introducción a UML.
Control técnico de proyectos de software.
Marcos de trabajo (frameworks) de empresas orientadas a objetos.
Estrategias de ingeniería inversa para migración de software.
Ingeniería de objetos.
Modelado y análisis de arquitectura de software.
Objetos distribuidos.
Sistemas Cliente/Servidor.
Reingeniería.
CASE.
Análisis y diseño orientados a objetos.
...

La ingeniería del software la realizan personas creativas, con conocimiento, que deberían trabajar dentro de un proceso del software definido y avanzado apropiado para los productos que se construyen y para la demanda del mercado.

[1] El concepto de ingeniería de software surgió en unas reuniones de trabajo organizadas por la Organización del Tratado del Atlántico Norte (OTAN) en 1968 y 1969 para estudiar lo que entonces se describía como “la crisis del software”. Había demasiados proyectos de desarrollo de soporte lógico que experimentaban fallos, los cuales se atribuían al rápido aumento en la escala y complejidad del software en cuestión. Se reconoció que era necesario un planteamiento más sistemático en el desarrollo de software, que debía basarse en principios de ingeniería ya establecidos. Biblioteca de Consulta Microsoft ® Encarta ® 2005. © 1993-2004 Microsoft Corporation.

EL INICIO

Para que una organización decida automatizar alguna tarea rutinaria debe existir una poderosa razón que le lleve a iniciar un estudio de factibilidad dependiendo de su necesidad:

1. Automatizar un proceso manual que genera errores frecuentes en los datos de entrada y por lo tanto entrega reportes incorrectos para la toma de decisiones.

2. Mejorar un proceso automático ya existente pero que puede incrementar la competitividad de la organización.

3. Reducción de costos a nivel organización al detectar procesos y/o tareas duplicadas lo que merma considerablemente la competitividad de la misma en el mercado.

Es casi imposible que los sistemas de información resulten eficaces si son desarrollados en forma independiente de los objetivos, valores y metas de la organización, para la que fueron diseñados. Por esto las solicitudes de proyectos se deben preparar y evaluar de acuerdo a este principio.

ASPECTOS A CONSIDERAR EN EL DESARROLLO DE APLICACIONES

1. **UNA METODOLOGÍA.** Cuando se trabaja para construir un producto o un sistema, es importante seguir una serie de pasos predecibles, un proceso. Este proceso es muy importante porque proporciona estabilidad, control y organización a una actividad que puede, si no se controla, volverse caótica. Desde el punto de vista técnico, un proceso de software es un marco de trabajo de las tareas que se requieren para construir software de alta calidad[2].

2. **APOYO DEL USUARIO SOLICITANTE.** Los ingenieros de software y sus gestores adaptan el proceso a sus necesidades y entonces lo siguen, contando siempre con el apoyo del usuario solicitante.

Desde el punto de vista de un ingeniero de software, los productos obtenidos son programas, documentos y datos que se producen como consecuencia de las actividades de ingeniería del software definidas por el proceso.

MECANISMOS DE EVALUACIÓN DEL PROCESO

Para estar seguro de que el ingeniero de software ha realizado su trabajo de manera correcta existen mecanismos de evaluación del proceso del software que permiten a las organizaciones determinar la madurez de su proceso del software. Pero la calidad, oportunidad y viabilidad a largo plazo del producto que está construyendo son los mejores indicadores de la eficiencia del proceso utilizado.

[2] En los últimos años se ha hecho mucho énfasis en la "madurez del proceso", el *Software Engineering Institute* (SEI) ha desarrollado un método completo que se basa en un conjunto de funciones de ingeniería de software que deberían estar presentes conforme organizaciones alcanzan diferentes niveles de madurez del proceso. Para determinar el estado actual de madurez, el SEI utiliza un cuestionario de evaluación y un esquema de cinco grados: inicial, repetible, definido, gestionado y optimización. (Para mayor referencia consultar **http://www.sei.cmu.edu**).

El trabajo que se asocia a la ingeniería del software se puede dividir en tres fases distintas, con independencia del área de aplicación, tamaño o complejidad del proyecto, definición, desarrollo y mantenimiento.

1. **FASE DE DEFINICIÓN.** Se centra sobre el ***qué***. Durante esta fase, el que desarrolla el software intenta identificar qué información ha de ser procesada, que función y rendimiento se desea, qué comportamiento del sistema, qué interfaces han de ser establecidas, que restricciones de diseño existen, y qué criterios de validación se necesitan para definir un sistema correcto. En pocas palabras, han de identificarse los requisitos clave del sistema y del software.

   TAREAS PRINCIPALES

   - Ingeniería de sistemas o de información
   - Planificación del proyecto de software
   - Análisis de los requisitos

2. **FASE DE DESARROLLO.** Se centra sobre el ***cómo***. Durante esta fase un ingeniero del software intenta definir cómo han de diseñarse las estructuras de datos, cómo ha de implementarse la función de dentro de una arquitectura de software, cómo han de implementarse los detalles procedimentales, cómo han de caracterizarse interfaces, cómo ha de traducirse el diseño en un lenguaje de programación y cómo ha de realizarse la prueba.

   TAREAS PRINCIPALES

   - Diseño del software
   - Generación de código
   - Prueba del software

3. **FASE DE MANTENIMIENTO**[3]. Se centra sobre el ***cambio*** que va asociado a la corrección de errores, a las adaptaciones requeridas a medida que evoluciona el entorno del software y a cambios debidos a las mejoras producidas por los requisitos cambiantes del cliente o su entorno. En esta fase se pueden dar cuatro tipos de cambios:

   - **Correctivo**. Cambia el software para corregir los defectos.
   - **Adaptativo**. Produce modificación en el software para acomodarlo a los cambios de su entorno externo.
   - **Perfectivo**. Lleva al software más allá de sus requisitos funcionales originales.
   - **Preventivo** (reingeniería del software). Hace cambios en programas de cómputo a fin de que se puedan corregir, adaptar y mejorar más fácilmente.

Además de estas actividades de mantenimiento, los usuarios de software requieren un mantenimiento continuo.

Estas fases y los pasos relacionados anteriormente comentados se complementan con un número de *actividades adicionales,* aplicadas a lo largo de todo el proceso del software, como las que a continuación se mencionan:

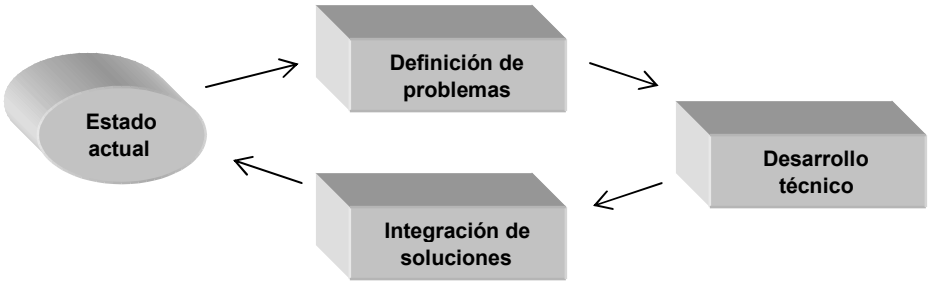
[3] El término mantenimiento, en este contexto, es mucho más que una simple corrección de errores.

- Seguimiento y control del proyecto de software
- Revisiones técnicas formales
- Garantía de calidad del software
- Gestión de configuración del software
- Preparación y producción de documentos
- Gestión de reutilización
- Mediciones
- Gestión de riesgos

MODELOS DE PROCESO DEL SOFTWARE

Para resolver los problemas un ingeniero del software o un equipo de ingenieros debe incorporar una estrategia de desarrollo que acompañe al proceso, métodos y capas de herramientas, esta estrategia, llamada modelo de proceso o paradigma de ingeniería del software.

Todo el desarrollo del software se puede caracterizar como bucle de resolución de problemas, como lo indica la siguiente figura:



Fases de un ciclo de resolución de problemas.

DEFINICIÓN DE PROBLEMAS. Identifica el problema específico a resolverse.

DESARROLLO TÉCNICO. Resuelve el problema a través de la aplicación de alguna tecnología.

INTEGRACIÓN DE SOLUCIONES. Ofrece los resultados (documentos, programas, datos, nueva función comercial, nuevo producto, etc.) a los que solicitan la solución en primer lugar.

Existen diferentes modelos de proceso para la ingeniería del software:

- **Modelo lineal secuencial** (ciclo de vida clásico o modelo en cascada).Sugiere un enfoque sistemático, secuencia, para el desarrollo del software que comienza en un nivel de sistemas y progresa con el análisis, diseño, codificación, pruebas y mantenimiento.

- **Modelo de construcción de prototipos.** Cuando el cliente tiene una necesidad legítima, pero está desorientado sobre los detalles, el primer paso es desarrollar un prototipo.

- **Modelo DRA** (Desarrollo Rápido de Aplicaciones). Es una adaptación a “alta velocidad” del modelo lineal secuencial en el que se logra el desarrollo rápido utilizando una construcción basada en componentes.

- **Modelos evolutivos de proceso del software:** modelo incremental, modelo en espiral, modelo espiral WINWIN, modelo de desarrollo concurrente.
- **Desarrollo basado en componentes.**
- **Modelo de métodos formales.**
- **Técnicas de cuarta generación (T4G).**

La elección del modelo para el desarrollo de aplicaciones depende de las características propias de cada organización y del tiempo asignado para esta actividad. Cabe mencionar que algunas empresas diseñan sus propios modelos de ingeniería a partir de los ya existentes.

GESTIÓN DE PROYECTOS

La gestión de proyectos implica la planificación, supervisión y control del personal, del proceso y de los eventos que ocurren mientras evoluciona el software desde la fase preliminar a la implementación operacional. La gestión de un proyecto es una actividad intensamente humana

La construcción de software es una empresa compleja, particularmente si participa mucha gente, trabajando durante un periodo de tiempo relativamente largo. Esta es la razón por la cual los proyectos de software necesitan ser gestionados.

GESTIÓN DE PERSONAL

El personal debe estar organizado para desarrollar el trabajo con efectividad. La comunicación con el cliente debe ocurrir para que se comprendan el alcance del producto y los requisitos.

Participantes en el proceso de software:

- **Gestores superiores.** Definen los aspectos de negocios que a menudo tienen una significativa influencia en el proyecto.
- **Gestores técnicos del proyecto.** Deben planificar, motivar, organizar y controlar a los profesionales que realizan el trabajo de software.
- **Profesionales.** Proporcionan las capacidades técnicas necesarias para la ingeniería del una aplicación.
- **Clientes.** Especifican los requisitos para la ingeniería del software y otros elementos que tienen menor influencia en el resultado.
- **Usuarios finales.** Interaccionan con el software una vez que se ha entregado para la producción.

Para ser eficaz, el equipo del proyecto debe organizarse de manera que maximice las habilidades y capacidades de cada persona. Este es el trabajo del jefe del equipo.

GESTIÓN DEL PROCESO

Debe seleccionarse el proceso adecuado para el personal y el producto[4]. Un proceso de software proporciona la estructura desde la que se puede establecer un detallado plan para el desarrollo del software.

El gestor del proyecto debe decidir qué modelo de proceso es el más adecuado para los clientes que han solicitado el producto y la gente que realizará el trabajo, las características del producto en sí y el entorno del proyecto en el que trabaja el equipo de software.

[4] Las fases genéricas que caracterizan el proceso de software (definición, desarrollo y soporte) son aplicables a todo software.

Cuando se selecciona un modelo de proceso, el equipo define entonces un *plan de proyecto preliminar* basado en un conjunto de actividades estructurales.

Una vez establecido el plan preliminar, empieza la descomposición del proceso: se debe crear un plan completo reflejando las tareas requeridas a las personas para cubrir las actividades estructurales.

GESTIÓN DEL PRODUCTO

Antes de poder planificar un proyecto, se deberían establecer los objetivos y el ámbito del producto, se deberían considerar soluciones alternativas e identificar las dificultades técnicas y de gestión (entiéndase como "producto" cualquier software que será construido a petición de otros). Sin esta información, es imposible definir estimaciones razonables del costo, una valoración efectiva del riesgo, una subdivisión realista de las tareas del proyecto o una planificación del proyecto asequible que proporcione una indicación fiable del progreso. El desarrollador de software y el cliente deben reunirse para definir los objetivos del producto y su ámbito. Una vez que se han entendido los objetivos y el ámbito del producto, se consideran soluciones alternativas.

- **Ámbito del software**. El ámbito de un proyecto de software debe ser unívoco y entendible a niveles de gestión y técnico. Se define respondiendo a las siguientes cuestiones:

- **Contexto**. ¿Cómo encaja el software a construir en un sistema, producto o contexto de negocios mayor y qué limitaciones se imponen como resultado del contexto?

- **Objetivos de información.** ¿Qué objetos de datos visibles al cliente se obtienen del software? ¿Qué objetos de datos son requeridos de entrada?

- **Función y rendimiento.** ¿Qué función realiza el software para transformar la información de entrada en una salida? ¿Hay características de rendimiento especiales que abordar?

- **Descomposición del problema**. Para desarrollar un plan de proyecto razonable, se tiene que descomponer funcionalmente el problema a resolver, en otras palabras, un problema complejo se parte en problemas más pequeños que resultan más manejables (estrategia que se aplica al inicio de la planificación del proyecto). Se aplica en dos áreas principales: la funcionalidad que debe entregarse y el proceso que se empleará para entregarlo.

GESTIÓN DEL PROYECTO[5]

El proyecto debe planificarse estimando el esfuerzo y el tiempo para cumplir las tareas; definiendo los productos del trabajo, estableciendo puntos de control de calidad y estableciendo mecanismos para controlar y supervisar el trabajo definido en la planificación.

Un gestor de proyectos hace lo correcto cuando estimula al personal para trabajar juntos como un equipo efectivo, centrando su atención en las necesidades del cliente y en la calidad del producto.

[5] Para gestionar un proyecto de software con éxito, debemos comprender qué puede ir mal (para evitar esos problemas) y cómo hacerlo bien.

La actividad de gestión del proyecto comprende medición y métricas, estimación, análisis de riesgos, planificación del programa, seguimiento y control.

El proceso del software y las métricas del producto son una medida cuantitativa que permite a la gente del software tener una visión profunda de la eficacia del proceso del software y de los proyectos que dirigen utilizando el proceso como un marco de trabajo.

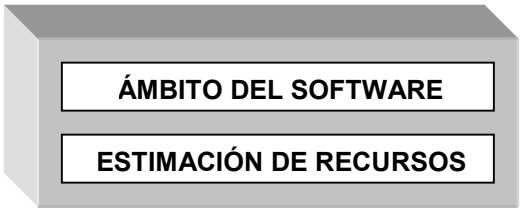
PLANIFICACIÓN DE PROYECTOS

Antes de que el proyecto comience, el gestor y el equipo de software deben realizar una estimación del trabajo a realizar, de los recursos necesarios y del tiempo que transcurrirá desde el comienzo hasta el final de su realización.

El objetivo de la planificación del proyecto de software es proporcionar un marco de trabajo que permita al gestor hacer estimaciones razonables de recursos, costo y planificación temporal.

La planificación implica la estimación (intento por determinar cuánto dinero, esfuerzo, recursos y tiempo supondrá construir un sistema).

Esta actividad es realizada por los gestores del software utilizando la información solicitada a los clientes y a los ingenieros de software y los datos de métricas de software obtenidos de proyectos anteriores.



Actividades de la planificación de proyectos de software

ÁMBITO DEL SOFTWARE

La primera actividad de la planificación del proyecto de software es determinar el ámbito del software, en el cual se describe el control y los datos a procesar, la función, el rendimiento, las restricciones, las interfaces y la fiabilidad.

Obtención de la información necesaria para el ámbito. La técnica utilizada con más frecuencia para acercar al cliente y al desarrollador, y para hacer que comience el proceso de comunicación es establecer una reunión o una entrevista preliminar[6]. El analista puede comenzar haciendo *preguntas de contexto libre*, como las siguientes:

¿Quién está detrás de la solicitud de este trabajo?
¿Quién utilizará la solución?
¿Cuál será el beneficio económico de una buena solución?
¿Hay otro camino para la solución?

[6] En realidad, la sesión P&R sólo se debería utilizar para el primer encuentro, reemplazándose posteriormente por un tipo de reunión que combine elementos de resolución de problemas, negociación y especificación.

Las siguientes preguntas permiten que el analista comprenda mejor el problema y que el cliente exprese sus percepciones sobre una solución:

¿Cómo caracterizaría [el cliente] un resultado "correcto" que se generaría con una solución satisfactoria?
¿Con qué problema(s) se afrontará esta solución?
¿Puede mostrarme (o describirme) el entorno en el que se utilizará la solución?
¿Hay aspectos o limitaciones especiales de rendimiento que afecten a la forma en que se aborde la solución?

La última serie de preguntas se centra en la efectividad de la reunión:

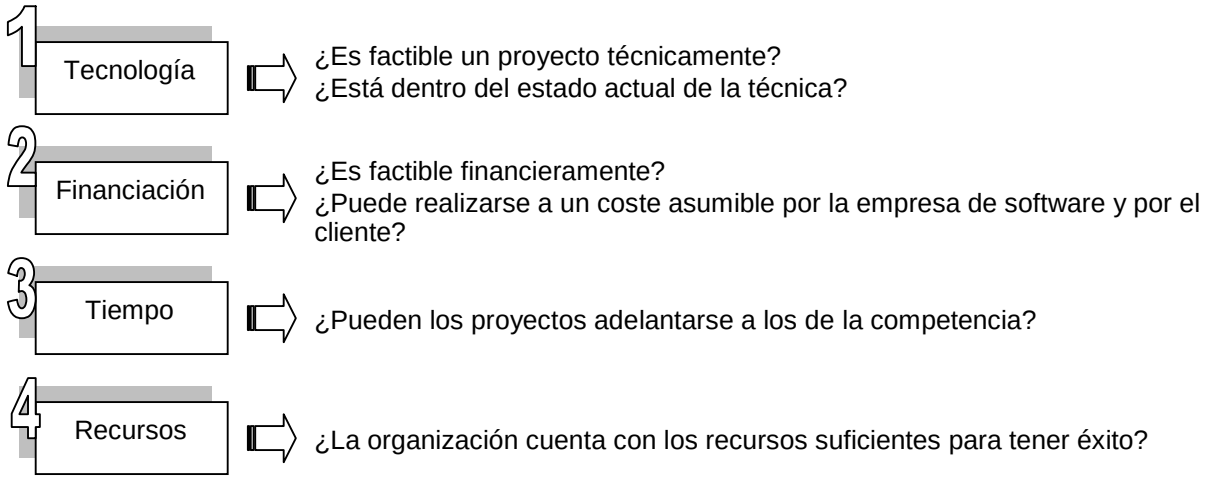
¿Es usted la persona apropiada para responder a estas preguntas? ¿Son "oficiales" sus respuestas?
¿Son relevantes mis preguntas para su problema?
¿Estoy realizando muchas preguntas?
¿Hay alguien más que pueda proporcionar información adicional?
¿Hay algo más que debiera preguntarle?

Estas preguntas (y otras) ayudarán a "romper el hielo" y a iniciar la comunicación esencial para establecer el ámbito del proyecto.

VIABILIDAD

Una vez que se ha identificado el ámbito (con la ayuda del cliente), es razonable preguntarse: "¿podemos construir el software de acuerdo a este ámbito? ¿Es factible el proyecto?

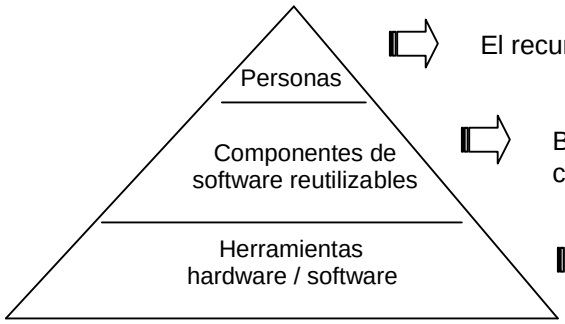
La factibilidad del software tiene cuatro sólidas dimensiones:



Dimensiones de la factibilidad del software.

ESTIMACIÓN DE RECURSOS

La segunda tarea de la planificación del desarrollo de software es la estimación de los recursos requeridos para acometer el esfuerzo de desarrollo de software.



Para realizar estimaciones seguras de costes y esfuerzos se tienen varias opciones:

1. Dejar la estimación para más adelante. Esta opción, aunque atractiva, no es práctica. Las estimaciones de costes han de ser proporcionadas *a priori.*

2. Basar las estimaciones en proyectos similares ya terminados. Esta opción pude funcionar razonablemente bien si el proyecto actual es bastante similar a los esfuerzos pasados.

3. Utilizar "técnicas de descomposición". Mediante la descomposición del proyecto en sus funciones principales y en las tareas de ingeniería del software correspondientes, la estimación del coste y del esfuerzo puede realizarse de una forma escalonada idónea.

4. Utilizar uno o más modelos empíricos de estimación: modelo COCOMO (*Constructive Cost Model*), COCOMO II, ecuación del software.

LA DECISIÓN DE DESARROLLAR-COMPRAR.

Algunas veces es más rentable adquirir el software de computadora que desarrollarlo:

- El software se puede comprar (con licencia) ya desarrollado
- Se pueden adquirir componentes de software y modificarse e integrarse para cumplir las necesidades específicas.
- El software puede ser construido de forma personalizada por una empresa externa para cumplir las especificaciones del comprador.

En el análisis final, la decisión de desarrollar comprar se basa en las condiciones siguientes:

1. ¿La fecha de entrega del producto de software será anterior que la del software desarrollado internamente?
2. ¿Será el coste de adquisición junto con el de personalización menor que el coste de desarrollo interno del software?
3. ¿Será el coste del soporte externo menor que el coste del soporte interno?

SUBCONTRATACIÓN (OUTSOURCING)

En algún momento toda compañía que desarrolla software de computadora se hace la siguiente pregunta: ¿existe alguna forma por la que podamos conseguir a bajo precio el software y los sistemas que necesitamos? La respuesta a esta pregunta no es simple, y las discusiones sobre esta cuestión dan como respuesta una simple palabra: subcontratación (*outsourcing*) en donde las actividades de ingeniería del software se contratan con un tercero, quien hace el trabajo a bajo coste asegurando una alta calidad. El trabajo de software llevado dentro de la compañía se reduce a una actividad de gestión de contratos.

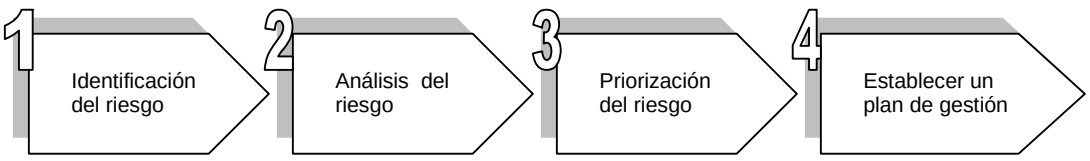
ANÁLISIS Y GESTIÓN DEL RIESGO

El análisis y la gestión del riesgo son una serie de pasos que ayudan al equipo del software a comprender y a gestionar la incertidumbre[7].

PREMISAS:

1. El riesgo afecta a los futuros acontecimientos. ¿Cómo enfrentar el riesgo en un proyecto de software?
2. El riesgo implica cambio, que puede venir dado por cambios de opinión, de acciones, de lugares, … ¿Cómo afectarán los cambios en los requisitos del cliente, en las tecnologías de desarrollo, en las computadoras a las que va dirigido el proyecto y a todas las entidades relacionadas con él, al cumplimiento de la planificación temporal y al éxito en general?
3. El riesgo implica elección, y la incertidumbre que entraña la elección. ¿Qué métodos y herramientas se deberán emplear, cuánta gente deberá estar implicada, cuánta importancia hay que darle a la calidad.
4. El riesgo es una de las pocas cosas inevitables en la vida. ¿Cómo minimizar sus efectos en el proyecto de software?

PASOS:



Pasos en el análisis y gestión del riesgo.

1. **IDENTIFICACIÓN DEL RIESGO.** Es un intento sistemático para especificar las amenazas al plan del proyecto (estimaciones, planificación temporal, carga de recursos, etc.). Identificando los riesgos conocidos y predecibles, el gestor del proyecto da un paso adelante para evitarlos cuando sea posible y controlarlos cuando sea necesario. La mejor estrategia para el control

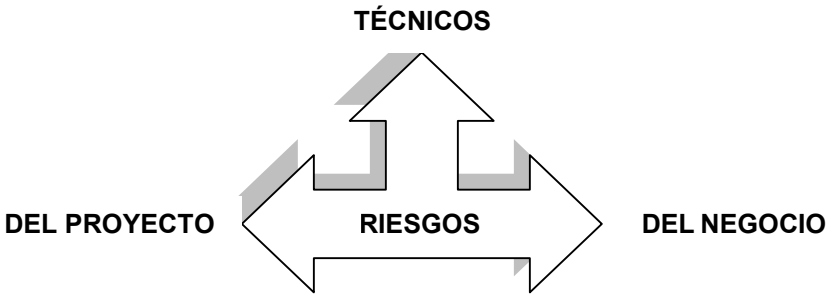
[7] El acontecimiento que caracteriza al riesgo, puede o no ocurrir.

del riesgo es ser proactivo. Un método para identificar riesgos es crear una *lista de comprobación de elementos de riesgo*, la cual se puede utilizar para identificar riesgos y se centra en un subconjunto de riesgos conocidos y predecibles en las siguientes subcategorías genéricas:

Tamaño del producto a construir o modificar.
Impacto en el negocio.
Características del cliente.
Definición del proceso y seguimiento.
Entorno de desarrollo (herramientas a emplear).
Tecnología a construir.
Tamaño y experiencia de la plantilla de personal.

Las respuestas a estas preguntas permiten al planificador del proyecto estimar el impacto del riesgo.

2. **ANÁLISIS DEL RIESGO**. A continuación, cada riesgo es analizado para determinar la probabilidad de que pueda ocurrir y el daño que puede causar si ocurre. Cuando se analizan los riesgos es importante cuantificar el nivel de incertidumbre y el grado de pérdidas[8] asociado con cada riesgo. Para hacerlo, se consideran diferentes categorías de riesgos:



Tipos de riesgos probables mientras se construye el software.

- **RIESGOS DEL PROYECTO.** Si los riesgos del proyecto se hacen realidad, es probable que la planificación temporal del proyecto se retrase y que los costos aumenten.
- **RIESGOS TÉCNICOS.** Amenazan la calidad y la planificación temporal del software que hay que producir. Ocurren porque el problema es más difícil de resolver que lo que se pensaba. Si se convierten en realidad, la implementación puede llegar a ser difícil o imposible.
- **RIESGOS DEL NEGOCIO.** Amenazan la viabilidad del software a construir. A menudo ponen en peligro el proyecto o el producto.

3. **PRIORIZACIÓN DEL RIESGO.** Siguiendo una estrategia proactiva se identifican los riesgos potenciales, se evalúa su probabilidad y su impacto y se establece una prioridad según su importancia.

[8] Si el riesgo se convierte en una realidad, ocurrirán consecuencias no deseadas o pérdidas.

4. **PLAN DE GESTIÓN DEL RIESGO.** El primer objetivo es evitar el riesgo, pero como no se pueden evitar todos los riesgos. El equipo trabaja para desarrollar un plan de contingencia que le permita responder de una manera eficaz y controlada.

PLANIFICACIÓN TEMPORAL Y SEGUIMIENTO DEL PROYECTO

Las actividades de estimación y análisis de riesgos y las técnicas de planificación temporal se implementan a menudo bajo la restricción de una fecha límite definida.

La planificación temporal de un proyecto de software es una actividad que distribuye el esfuerzo estimado a lo largo de la duración prevista del proyecto, asignando el esfuerzo a las tareas específicas de la ingeniería del software.

PRINCIPIOS BÁSICOS DE LA PLANIFICACIÓN TEMPORAL

Cada uno de los siguientes principios se aplican a medida que evoluciona la planificación temporal del proyecto:

1. **COMPARTIMENTACIÓN.** El proyecto debe dividirse en un número de actividades y tareas manejables.

2. **INTERDEPENDENCIA.** Se deben determinar las interdependencias de cada actividad o tarea compartimentada. Algunas tareas deben ocurrir en una secuencia determinada; otras pueden darse en paralelo. Algunas actividades no pueden comenzar hasta que el resultado de otras no esté disponible. Otras actividades pueden ocurrir independientemente.

3. **ASIGNACIÓN DE TIEMPO.** A cada tarea que se vaya a programar se le debe asignar cierto número de unidades de trabajo. Además, a cada tarea se le debe asignar una fecha de inicio y otra de finalización.

4. **VALIDACIÓN DE ESFUERZO.** Todos los proyectos tienen un número definido de miembros de la plantilla.

5. **RESPONSABILIDADES DEFINIDAS.** Cada tarea que se programe debe asignarse a un miembro del equipo específico.

6. **RESULTADOS DEFINIDOS.** Cada tarea programada debería tener un resultado definido.

7. **HITOS DEFINIDOS.** Todas las tareas o grupos de tareas deberían asociarse con un hito del proyecto.

HERRAMIENTAS DE PLANIFICACIÓN TEMPORAL DE PROYECTOS

PERT

CPM

Tanto PERT (técnica de evaluación y revisión de programa) como CPM (método del camino crítico) proporcionan herramientas cuantitativas que permiten al planificador del software:

1. Determinar el camino crítico, la cadena de tareas que determina la duración del proyecto.

2. Establecer las dimensiones de tiempo “más probables” para las tareas

individuales aplicando modelos estadísticos.

3. Calcular las limitaciones de tiempo que definen una “ventana” de tiempo de una tarea determinada.

Ambas técnicas son dirigidas por la información ya desarrollada en actividades anteriores de la planificación del proyecto:

- Estimaciones de esfuerzo.
- Una descomposición de la función del producto.
- La selección del modelo de proceso adecuado y de conjunto de tareas.
- La descomposición de tareas.

GRÁFICOS GANTT

Un gráfico de tiempo permite conocer qué tareas serán conducidas en un punto determinado en el tiempo. El esfuerzo, duración y fecha de inicio son las entradas de cada tarea y se asignan las tareas a individuos específicos. Se puede desarrollar un gráfico de tiempo para todo el proyecto. Alternativamente se pueden desarrollar diferentes gráficos para cada función del proyecto o para cada individuo que trabaje en el proyecto.

Una vez que se ha introducido la información necesaria para generar el gráfico de tiempo, la mayoría de las herramientas de la planificación temporal de proyectos de software producen *tablas de proyecto* (listado tabular de todas las tareas del proyecto, sus fechas previstas y reales de inicio y finalización e información relativa al tema). Empleando las tablas junto con los gráficos de tiempo, le permiten al gestor del proyecto hacer un seguimiento del progreso.

SEGUIMIENTO DE LA PLANIFICACIÓN TEMPORAL

El seguimiento se puede hacer de diferentes maneras:

1. Realizando reuniones periódicas del estado del proyecto en las que todos los miembros del equipo informan del progreso y de los problemas.
2. Evaluando los resultados de todas las revisiones realizadas a lo largo del proceso de ingeniería del software.
3. Determinando si se han conseguido los hitos formales del proyecto en la fecha programada.
4. Comparando la fecha real de inicio con las previstas para cada tarea del proyecto listada en la tabla del proyecto.
5. Reuniéndose informalmente con los profesionales del software para obtener sus valoraciones subjetivas del progreso hasta la fecha y los problemas que se avecinan.
6. Utilizando el análisis del valor ganado[9] para evaluar el progreso cuantitativamente.

[9] El sistema de valor ganado proporciona una escala de valor común para cada tarea, independientemente del tipo de trabajo que esté siendo llevado a cabo. Se estiman entonces el total de horas para realizar el proyecto completo y a cada tarea se le da un valor ganado basado en su porcentaje estimado respecto al total.

PLAN DEL PROYECTO

Es un documento relativamente breve dirigido a una audiencia diversa y debe:

1. Comunicar el ámbito y recursos a los gestores del software, personal técnico y al cliente.
2. Definir los riesgos y sugerir técnicas de aversión al riesgo.
3. Definir los costes y planificación temporal para revisión de la gestión.
4. Proporcionar un enfoque general del desarrollo del software para todo el personal relacionado con el proyecto.
5. Describir cómo se garantizará la calidad y se gestionan los cambios.

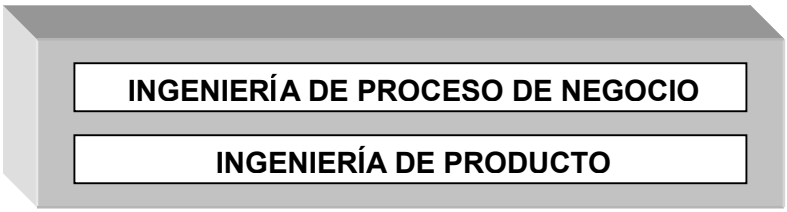
El plan del proyecto del software no es un documento estático ya que el equipo del proyecto lo consulta repetidamente, actualizando riesgos, estimaciones, planificaciones e información relacionada, a la vez que el proyecto avanza y es más conocido.

INGENIERÍA DE SISTEMAS

La ingeniería de sistemas de computadora es un proceso de modelado. El ingeniero de sistemas crea modelos que:

- Definan los procesos que satisfagan las necesidades de la visión en consideración.
- Representen el comportamiento de los procesos y los supuestos en los que se basa el comportamiento.
- Definan explícitamente las entradas exógenas[10] y endógenas de información al modelo.
- Representen todas las uniones (incluyendo las salidas) que permitan al ingeniero entender mejor la visión.

La ingeniería de sistemas abarca a la ingeniería de proceso de negocio y a la ingeniería de producto[11].



Componentes de la Ingeniería de Sistemas

INGENIERÍA DE PROCESO DE NEGOCIO (IPN)

El objetivo de la IPN es definir arquitecturas que permitan a las empresas emplear la información eficazmente.

Se deben analizar y diseñar tres arquitecturas diferentes dentro del contexto de objetivos y metas de negocio: de datos, de aplicaciones e infraestructura tecnológica.

ARQUITECTURA DE DATOS

Proporciona una estructura para las necesidades de información de un negocio o de una función de negocio. Los ladrillos de la arquitectura son los objetos de datos[12] que emplea la empresa.

Una vez definido el conjunto de datos, se identifican sus relaciones. Una relación indica como los objetos están conectados. Los objetos de datos fluyen entre las funciones de negocio, están

[10] Las entradas exógenas unen un elemento de una visión dada con otros elementos al mismo o a otros niveles; las entradas endógenas unen componentes individuales de un elemento en una visión particular.

[11] Capítulo 10, ingeniería del software, Pressman.

[12] Un objeto de datos contiene un conjunto de atributos que definen aspectos, cualidades, características o descriptor de los datos que han sido descritos.

organizados dentro de una base de datos y se transforman para proveer información que sirva a las necesidades del negocio.

Objeto: Cliente Atributos: Nombre Nombre de la compañía Dirección comercial Información de contacto Producto(s) de interés Compra(s) anteriores

Definición del objeto Cliente.

ARQUITECTURA DE APLICACIÓN

Comprende aquellos elementos de un sistema que transforman objetos dentro de la arquitectura de datos por algún propósito del negocio. Es el sistema de programas (software) que realiza esta transformación.

INFRAESTRUCTURA TECNOLÓGICA

Proporciona el fundamento de las arquitecturas de datos y de aplicaciones. La infraestructura comprende el hardware y el software empleados para dar soporte a las aplicaciones y datos.

INGENIERÍA DE PRODUCTO

La meta de la ingeniería de producto es traducir el deseo de un cliente, de un conjunto de capacidades definidas, a un producto operativo. Para conseguir esta meta, la ingeniería de producto debe crear una arquitectura y una infraestructura. La arquitectura comprende cuatro componentes de sistemas distintos: software, hardware, datos (bases de datos) y personas. Se establece una infraestructura de soporte e incluye la tecnología requerida para unir los componentes y la información que emplea para dar soporte a los componentes.

ANÁLISIS DE REQUERIMIENTOS

El análisis de los requisitos es una tarea de ingeniería del software que cubre el hueco entre la definición del software a nivel sistema y el diseño del software. Su objetivo es traducir las necesidades del usuario en una solución de software a la medida debidamente especificada funcionalmente.

Permite al ingeniero de sistemas especificar las características operacionales del software (función, datos y rendimientos), indica la interfaz del software con otros elementos del sistema y establece las restricciones que debe cumplir el software.

El análisis de requisitos del software puede dividirse en cinco áreas de esfuerzo:

1	**Reconocimiento del problema**	¿Cuál es el problema real?
2	**Evaluación del problema y síntesis de la solución**	¿Qué datos produce y consume el sistema, qué funciones debe realizar el sistema, qué interfaces se definen y qué restricciones son aplicables?
3	**Modelado**	Creación de modelos de función y comportamiento para entender mejor la entidad que se va a construir.
4	**Especificación de requisitos**	Un proceso de representación.
5	**Revisión de la especificación**	Un intento por asegurarse de que la especificación sea completa, consistente y correcta.

1. **Reconocimiento del problema.** Es importante entender el software en el contexto de un sistema y revisar el ámbito del software que se empleó para generar las estimaciones de la planificación. A continuación, se debe establecer la comunicación para el análisis de manera que nos garantice un correcto reconocimiento del problema. El objetivo del analista es el reconocimiento de los elementos básicos del problema tal y como los percibe el cliente/usuario.

2. **Evaluación del problema y síntesis de la solución.** El analista debe definir todos los objetos de datos observables externamente, evaluar el flujo y contenido de la información, definir y elaborar todas las funciones del software, entender el comportamiento del software en el contexto de acontecimientos que afectan al sistema, establecer las características de la interfaz del sistema y descubrir restricciones adicionales del diseño. Una vez que se han identificado los problemas, el analista determina qué información va a producir el nuevo sistema y qué información se le proporcionará al sistema. Una vez evaluados los problemas actuales y la información deseada (entradas y salidas), el analista empieza a sintetizar una o más soluciones. Para empezar, se definen en detalle los datos, las funciones de tratamiento y el comportamiento del sistema. El proceso de evaluación y síntesis continúa hasta que el analista y el cliente se sienten seguros de que se puede especificar adecuadamente el software para posteriores fases de desarrollo.

3. **Modelado.** El modelado es el fundamento para el diseño del software y base para la creación de una especificación del software.

a. **Modelos funcionales.** Empieza con un sencillo modelo a nivel de contexto, después de una serie de iteraciones, se consiguen más y más detalles funcionales, hasta que se consigue representar una minuciosa definición de toda la funcionalidad del sistema.

b. **Modelos de comportamiento.** Crea una representación de los estados del software y los sucesos que causan que cambie de estado.

4. **Especificación de requisitos.** Completa descripción de la información, descripción detallada de la función y el comportamiento, indicación de los requisitos del rendimiento y restricciones del diseño, criterios de validación apropiados y otros datos pertinentes a los requisitos.

5. **Revisión de la especificación.** Es esencial para asegurarse que el cliente y el desarrollador tienen el mismo concepto del sistema. Profundiza en el detalle, examinando no solo las descripciones superficiales, sino la vía en que los requisitos son expresados. Una vez que se ha completado la revisión, se firma la especificación de requisitos de software por el cliente y desarrollador, convirtiéndose en un "contrato" para el desarrollo del software. Durante esta fase se pueden recomendar cambios a la especificación[13] y puede ser extremadamente difícil valorar el impacto global de un cambio, para esto, las herramientas CASE pueden ayudar a resolver estos problemas.

[13] Las peticiones de cambios en los requisitos una vez que se ha finalizado la especificación no se eliminarán, pero el cliente debe saber que cada cambio a posteriori significa una ampliación del ámbito del software, y por tanto pueden aumentar los costos y prolongar los plazos de la planificación temporal del proyecto.

Anexos:

1. **Propuesta de solución**
2. **Especificación funcional**
3. **Cotización preliminar de propuesta de solución**

Anexo 1. Propuesta de solución

1. Datos de identificación del requerimiento

Folio del requerimiento:	<Número de folio>	**Nombre del proyecto:**	<Nombre completo del proyecto>
Fecha de formalización del requerimiento:	<dd-mmm-aaaa>	**Fecha de inicio de la propuesta de solución:**	<dd-mmm-aaaa>
Descripción del requerimiento: <Descripción detallada del requerimiento>			
Razón fundamental del proyecto: <Explicar la razón fundamental para crear una función nueva o modificar una existente, incluyendo justificaciones de recursos y/o esfuerzos (recursos x tiempo)>			
Objetivos del requerimiento: <Explicar el resultado esperado del requerimiento>			
Datos generales del líder del proyecto:			

Nombre:	
Puesto:	**Extensión telefónica:**
Área de adscripción:	**Dirección general a la que pertenece:**

Participantes en la solución del requerimiento: <Especificar los datos generales de estos participantes>

2. Alcance del proyecto

Alcance del proyecto: *<Detallar el alcance del proyecto desde el punto de vista del negocio>*

3. Áreas de negocio involucradas

<Indicar las áreas de negocio involucradas o impactadas en la solución>

4. Alcance del requerimiento

<Especificar el alcance real del requerimiento, así como las consideraciones y supuestos>

5. Restricciones

<Indicar las reglas a considerar en el diseño de la solución>

6. Requerimientos funcionales genéricos

<Indicar cada función que debe tener la solución para resolver un problema del negocio>

- **Tecnología**
- **Recursos humanos**
- **Recursos materiales**
- **Mercadotecnia**
- **Finanzas …**

7. Situación actual

Esquema conceptual <Elaborar el esquema conceptual de la situación actual, indicando los elementos relativos a flujos de entrada, de salida y los elementos que intercambian información>
Narrativa del esquema conceptual <Describir textualmente la forma en que operan los elementos indicados en el diagrama del esquema conceptual>
Diccionario de términos <Definir los conceptos propios del negocio utilizados en la propuesta>

8. Situación propuesta

Esquema conceptual <Elaborar el esquema conceptual de la situación actual, indicando los elementos relativos a flujos de entrada, de salida y los elementos que intercambian información>
Narrativa del esquema conceptual <Describir textualmente la forma en que operan los elementos indicados en el diagrama del esquema conceptual>
Diccionario de términos <Definir los conceptos propios del negocio utilizados en la propuesta>

9. Análisis de riesgo – beneficio

	Retos	***Beneficios***
Hacer el proyecto	<Indicar las metas que deben ser alcanzadas en el proyecto>	<Cuantificar de manera genérica, los ingresos o costos que serán ahorrados por el proyecto> <Indicar los beneficios al concluir el proyecto>
	Riesgos	***Costos***
No hacer el proyecto	<Indicar los efectos por seguir trabajando como actualmente se hace>	<¿Cuál sería la cantidad perdida de ingresos?>

10. Criterios de aceptación de la solución

<Indicar los criterios que serán usados para validar el correcto cumplimiento del requerimiento>

11. Aprobación

<incluir el visto bueno del líder o coordinador del proyecto>

Anexo 2. Especificación funcional

1. Datos de identificación del requerimiento

Folio del requerimiento:	<Número de folio>	**Fecha inicio de la especificación:**	<dd-mmm-aaaa>
Nombre del requerimiento:	<Nombre completo del requerimiento>		
Descripción del requerimiento: <Descripción detallada del requerimiento>			

2. Detalles funcionales del requerimiento

Requerimiento funcional genérico: <*Folio del requerimiento funcional indicado en la propuesta de solución*>
Descripción detallada del requerimiento: <Describir paso por paso, en qué consiste el requerimiento>
Validaciones y/o reglas de negocio: <operaciones o manipulación sobre los datos>
Comportamiento: <Descripción de las acciones que serán ejecutadas (guardar, validar, salir, etc.) en base a la activación de teclas, movimientos del cursor, etc.>
Datos mínimos necesarios: <Especificación de los datos mínimos necesarios para realizar alguna acción de consulta, validación, ejecución, etc>
Resultados esperados: <Especificación del resultado a obtener de acuerdo a la acción ejecutada y en caso de error por parte del usuario se debe informar en forma adecuada (ventana, barra de status, etc.)>
Interfaces con otros módulos: <Especificación de la forma en que compartirá datos con otros módulos (cadenas de caracteres, archivos, etc) indicando la forma en que se compone esta cadena de caracteres o bien el nombre del archivo con su descripción de campos>
Dependencias (funciones relacionadas): <Se informan las dependencias entre módulos>
Observaciones: <Cualquier comentario que clarifique la función que esta siendo detallada>

3. Anexos

<incluir los anexos necesarios>

4. Aprobación

<incluir el visto bueno del líder o coordinador del proyecto>

Área:	<Nombre oficial del área de adscripción>
Nombre:	<Nombre completo>
Firma:	<Firma registrada en el catálogo de firmas>

Anexo 3. Cotización preliminar de propuesta de solución

1. Datos de identificación del requerimiento

Folio del requerimiento:	<Número de folio>	**Fecha de elaboración de la propuesta:**	<dd-mmm-aaaa>
Nombre del proyecto:	<Nombre completo del proyecto>		
Descripción del requerimiento: <Descripción detallada del requerimiento>			

Complejidad técnica:	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1 = Baja complejidad 10 = Alta complejidad										

2. Resumen de costos

Inversión	**$**
Gasto	**$**
Total costo	**$**
Gasto recurrente	**$**
Gasto fijo de personal	**$**
Total gasto corriente	**$**

3. Estimación de esfuerzo

<Indicar el esfuerzo en horas/hombre>

Entregable	Esfuerzo (hrs/hombre)

4. Estimación del costo del proyecto

<Desglose de los costos del proyecto de cada concepto del resumen arriba especificado>

Inversión		
Concepto	**Descripción**	**Monto**
		$
		$
Subtotal:		**$**

Gasto		
Concepto	**Descripción**	**Monto**
		$
		$
Subtotal:		**$**

Costo del proyecto (inversión + gasto)	**$**

Gastos recurrentes		
Concepto	**Descripción**	**Monto**
		$
		$
Subtotal:		**$**

Gastos fijos de personal		
Concepto	**Descripción**	**Monto**
		$
		$
Subtotal:		**$**

Total gasto corriente (gastos recurrentes + gastos fijos de personal)	**$**

5. Aprobación de los participantes en el proyecto

<incluir el visto bueno de cada participante en la evaluación del proyecto>

Área:	<Nombre oficial del área de adscripción>
Nombre:	<Nombre completo>
Firma:	<Firma registrada en el catálogo de firmas>

Área:	<Nombre oficial del área de adscripción>
Nombre:	<Nombre completo>
Firma:	<Firma registrada en el catálogo de firmas>

Área:	<Nombre oficial del área de adscripción>
Nombre:	<Nombre completo>
Firma:	<Firma registrada en el catálogo de firmas>

DISEÑO DE LA SOLUCIÓN

Una vez que se analizan y especifican los requisitos del software, el diseño del software es la primera de las tres actividades técnicas (diseño, generación de código y pruebas) que se requieren para construir y verificar el software.

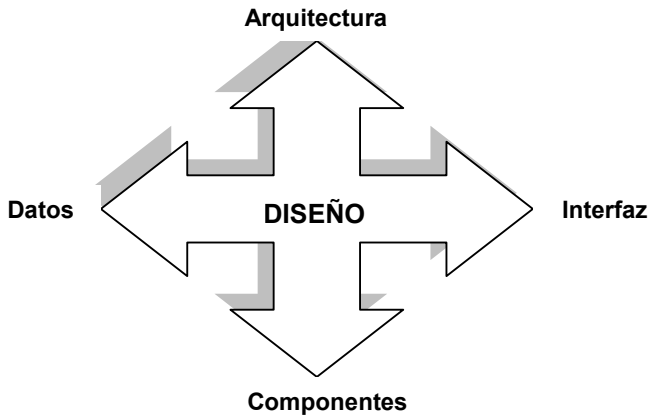
En esta fase se tiene el objetivo de producir el diseño detallado del sistema a desarrollar que satisfaga los requerimientos planteados por el usuario solicitante y que sirva de guía al equipo de desarrollo para la construcción del producto.

Los requisitos del software, manifestados por los modelos de datos funcionales y de comportamiento, alimentan la tarea del diseño.

La importancia del diseño del software se puede describir con una sola palabra: *calidad*. El diseño es el lugar en donde se fomentará la calidad en la ingeniería del software.

El diseño es la única forma de convertir exactamente los requisitos de un cliente en un producto o sistema de software finalizado.

En el contexto de la ingeniería del software, el diseño se centra en cuatro áreas importantes de interés: la estructura de datos, la arquitectura del sistema, la representación de la interfaz y los detalles a nivel componentes.



Modelos de diseño requeridos para la especificación completa

DISEÑO DE DATOS

Transforma el modelo del dominio de información que se crea durante el análisis en las estructuras de datos que se necesitarán para implementar el software. Los objetos de datos y las relaciones definidas en el diagrama relación entidad y el contenido de datos detallado que se representa en el diccionario de datos proporcionan la base de la actividad del diseño de datos.

DISEÑO ARQUITECTÓNICO

Define la relación entre los elementos estructurales principales del software, los patrones de diseño que se puedan utilizar para lograr los requisitos que se han definido para el sistema, y las restricciones que afectan a la manera en que se puedan aplicar los patrones de diseño arquitectónicos.

DISEÑO DE LA INTERFAZ

Describe la manera de comunicarse el software dentro de sí mismo, con sistemas que interoperan dentro de él y con las personas que lo utilizan. Una interfaz implica un flujo de información (por ejemplo, datos y/o control) y un tipo específico de comportamiento.

DISEÑO A NIVEL DE COMPONENTES

Transforma los elementos estructurales de la arquitectura del software en una descripción procedimental de los componentes del software.

DIRECTRICES DE DISEÑO:

1. Un diseño deberá presentar una estructura arquitectónica que: a) se haya creado mediante patrones de diseño reconocibles; b) que esté formada por componentes que exhiban características de buen diseño y c) que se puedan implementar de manera evolutiva, facilitando así la implementación y la comprobación.

2. Un diseño deberá ser modular, es decir, el software deberá dividirse lógicamente en elementos que realicen funciones y subfunciones específicas.

3. Un diseño deberá contener distintas representaciones de datos, arquitectura, interfaces y componentes (módulos).

4. Un diseño deberá conducir a estructuras de datos adecuadas para los objetos que se van a implementar y que procedan de patrones de datos reconocibles.

5. Un diseño deberá conducir a componentes que representen características funcionales independientes.

6. Un diseño deberá conducir a interfaces que reduzcan la complejidad de las conexiones entre los módulos y con el entorno externo.

7. Un diseño deberá derivarse mediante un método repetitivo y controlado por la información obtenida durante el análisis de los requisitos del software.

El diseño de software es tanto un proceso como un modelo. El *proceso* de diseño es una secuencia de pasos que hacen posible que el diseñador describa todos los aspectos del software que se va a construir. El *modelo* de diseño es el equivalente a los planos de un arquitecto para una casa.

La modularidad (tanto en el programa como en los datos) y el concepto de abstracción permiten que el diseñador simplifique y reutilice los componentes del software.

El refinamiento proporciona un mecanismo para representar sucesivas capas de datos funcionales.

El programa y la estructura de datos contribuyen a tener una visión global de la arquitectura del software, mientras que el procedimiento proporciona el detalle necesario para la implementación de los algoritmos.

DOCUMENTACIÓN DEL DISEÑO

La especificación del diseño aborda diferentes aspectos del modelo de diseño y se completa a medida que el diseñador refina su propia representación del software.

En primer lugar, se describe el ámbito global del esfuerzo realizado en el diseño. La mayor parte de la información que se presenta aquí se deriva de la especificación del sistema y del modelo de análisis (especificación de los requisitos del software).

A continuación, se especifica el diseño de datos. Se definen también las estructuras de las bases de datos, cualquier estructura externa de archivos, estructuras internas de datos y una referencia cruzada que conecta objetos de datos con archivos específicos.

El diseño arquitectónico indica cómo se ha derivado la arquitectura del programa del modelo de análisis. Para representar la jerarquía del módulo se utilizan gráficos de estructuras.

Se representa el diseño de interfases internas y externas de programas y se describe un diseño detallado de la interfaz hombre-máquina.

Los componentes se describen inicialmente con una narrativa de procesamiento en cualquier idioma, explicando la función procedimental de un componente (módulo). Posteriormente, se utiliza una herramienta de diseño procedimental para convertir esa estructura en una descripción estructural.

La especificación del diseño contiene una referencia cruzada de requisitos. El propósito de esta referencia (normalmente representada con una matriz simple) es:

1. Establecer que todos los requisitos se satisfagan mediante el diseño del software, y
2. Indicar cuales son los componentes críticos para la implementación de requisitos específicos.

Una vez que se han establecido las interfaces y la estructura de programa podremos desarrollar las líneas generales para comprobar los módulos individuales y la integración de todo el paquete.

Las restricciones de diseño, tales como limitaciones físicas de memoria o la necesidad de una interfaz externa especializada, podrán dictar requisitos especiales para ensamblar o empaquetar el software.

La última sección de la especificación del diseño contiene datos complementarios. También se presentan descripciones de algoritmos, procedimientos alternativos, datos tabulares, extractos de otros documentos y otro tipo de información relevante, todos mediante notas especiales o apéndices separados.

Anexos:

1. **Diseño del prototipo**
2. **Alternativas de diseño**
3. **Diseño detallado**
4. **Arquitectura del sistema**
5. **Diseño de interfaces**
6. **Elementos impactados**
7. **Modelo de datos**
8. **Matriz de pruebas**
9. **Configuración**
10. **Esquema de contingencia**

Anexo 1. Diseño del prototipo

1. Datos del proyecto

Folio del proyecto:	<Número de folio>	**Fecha inicio del diseño:**	<dd-mmm-aaaa>
Nombre del proyecto:	<Nombre completo del proyecto>		
Descripción del proyecto: <Descripción detallada del proyecto>			

2. Objetivo del prototipo

<Describir claramente el objetivo por el cual se esta solicitan el prototipo>

3. Plataformas involucradas

<Informar las plataformas involucradas en el funcionamiento del prototipo, incluyendo los requisitos de software>

No.	Plataforma	Versión	Requisitos de software

4. Tecnología propuesta

<Describir la tecnología a usar en la programación del prototipo>

Lenguaje de programación:
Criterio:

Base de datos
Criterio:

Otros productos:
Criterio:

5. Resultados esperados

<Describir claramente los resultados que se esperan obtener en el prototipo>

6. Resultados obtenidos

<Describir claramente los resultados que se obtuvieron en el prototipo>

7. Observaciones

<Observaciones adicionales al prototipo>

8. Aprobación del prototipo

<Indicar si el prototipo fue o no aprobado>

9. Descripción detallada de componentes

Nombre del componente de reuso	Ubicación del componente	Componentes que lo utilizan	Aplicativos que lo usan actualmente

Descripción de cada componente:

Componente 1 <Nombre> <Describir el propósito y la funcionalidad del componente: procesamiento, validaciones, reglas de negocio >

Parámetros e interfaces de componentes:

<Describir las interfaces que implementará el componente (parámetros, retornos y tipos de error o excepciones)>

Manejo de errores:

<Explicar la forma en que se realizará el manejo de errores dentro del componente y la forma en que se retroalimentará a la entidad que mande llamar el componente cuando se presente un error>

Id del error	Mensaje	Acciones

10. Infraestructura

Requisitos de hardware:

<Especificar los elementos de hardware que constituyen la infraestructura del diseño de la solución: servidores, redes, dispositivos especiales, etc. >

No.	Elemento	Plataforma	Descripción

Requisitos de software:

<Especificar los elementos de software que constituyen la infraestructura del diseño de la solución: manejadores de bases de datos, software de comunicaciones, lenguajes de programación, librerías, browsers, etc.>

Software requerido	Versión

11. Anexos

<Documentos necesarios para completar la especificación detallada de la solución>

Diseño de interfaces:

Modelo de datos:

Anexo 2. Alternativas de diseño

1. Alternativas de solución

Descripción de cada alternativa *<Explicación de las diferentes alternativas de solución y los criterios considerados para la elección (costo, beneficio, confiabilidad, seguridad, esfuerzo, tiempo, etc.)>*

No.	Descripción de la alternativa

Alternativa seleccionada *<Número de alternativa seleccionada>*

2. Criterios de selección para la alternativa elegida

<Justificación de la selección>

Anexo 3. Diseño detallado

1. Descripción detallada de componentes

<Componentes reutilizados >

Nombre del componente de reuso	Ubicación del componente	Componentes que lo utilizan	Aplicativos que lo usan actualmente

Descripción de cada componente:

Componente 1 <Nombre> <Describir el propósito y la funcionalidad del componente: procesamiento, validaciones, reglas de negocio>

Parámetros e interfaces de componentes:

<Describir las interfaces que implementará el componente (parámetros, retornos y tipos de error o excepciones)>

Mapeo de campos:

Campo de elemento origen	Campo de elemento destino

Manejo de errores:

<Explicar la forma en que se realizará el manejo de errores dentro del componente y la forma en que se retroalimentará a la entidad que mande llamar el componente cuando se presente un error>

Id del error	Mensaje	Acciones

2. Infraestructura

Requisitos de hardware:

<Especificar los elementos de hardware que constituyen la infraestructura del diseño de la solución: servidores, redes, dispositivos especiales, etc. >

No.	Elemento	Plataforma	Descripción

Requisitos de software:

<Especificar los elementos de software que constituyen la infraestructura del diseño de la solución: manejadores de bases de datos, software de comunicaciones, lenguajes de programación, librerías, browsers, etc.>

Software requerido	Versión

3. Anexos

<Documentos involucrados en el diseño detallado de la solución>

1. Arquitectura del sistema
2. Diseño de interfaces
3. Elementos impactados
4. Modelo de datos
5. Matriz de pruebas
6. Configuración
7. Esquema de contingencia
8. Otros

Anexo 4. Diseño de la arquitectura del sistema

1. Estructura general de componentes e interfaces

Diagrama de componentes *<Insertar diagrama>*

Descripción del componente *<Descripción del componente >*

Nombre del componente	Tipo	Plataforma	Descripción de la funcionalidad

2. Reuso de componentes y relaciones con otros aplicativos

Componente	Aplicativos que lo usan actualmente

Diagrama de relaciones con otros aplicativos *<Insertar diagrama>*

3. Modelo de datos

Diagrama entidad – relación *<Insertar diagrama>*

Tabla de relaciones

Entidad	Entidad relacionada	Tipo de relación

Anexo 5. Diseño de interfaces

1. Formato de archivos

Definición de extracción de datos o registros *<Descripción de las características de los campos o registros que forman al archivo de interfaz>*

Nombre del archivo: Descripción del archivo: Aplicación que lo genera o consume: Periodicidad: Tipo de Interfaz (E / S): Medio de Transporte (FTP, etc.): Ruta donde se deposita:

Layout de registros <*Especificar cada campo y delimitador>*

ID	Campo	Tipo	Longitud	De	A

2. Pantallas

<Describir cada pantalla>

Título	<Título de la pantalla>
Propósito	<Propósito de la pantalla>
Reglas de negocio	<Reglas de negocio de la pantalla>
ID de pantalla	<Id de la pantalla>

Imagen de la pantalla: *<Incluir la imagen de la pantalla y una descripción del contenido de la misma>*

Controles de la pantalla:

Grupo o sección	Nombre	Tipo	Long	Valores		Propiedades				
				Posibles	Inicial	V	R	H	E	T

V .-Visible (S/N)
R.- Requerido (S/N)
H.- Habilitado (S/N)
E.- Editable (S/N)
T.- Orden (Tab Order)

Comportamiento de la pantalla:

Nombre del control	Evento	Acciones
Button 1	Clic	Guarda datos
Button 2	Clic	Guarda datos y llama pantalla de salida

Descripción de las teclas de función:

Tecla	Letrero	Acción	Consideraciones
<Identificar la tecla de función (F1, F4, F7, etc.)>	*<Indicar texto representativo de la función>*	*<Indicar la acción asociada a la tecla>*	*<Reglas de Negocio>*

Mapeo de campos:

Datos de entrada de la pantalla: *<Descripción del mapeo de los datos que proporciona el usuario a través de la capa de presentación y que son pasados al elemento de la siguiente capa>*

Campo de la Pantalla	Mapea a (parámetro de otro elemento)

Datos de salida de la pantalla: *<Descripción del mapeo de los datos que recibe la interfaz de usuario provenientes de otro elemento>*

Parámetro recibido	Mapea a (campo de la pantalla)

3. *Reportes*

Título	*<Título del reporte>*
Propósito	*<Propósito del reporte>*
Dispositivo de Impresión	*<Dispositivo de impresión del reporte>*
Id. Reporte	*<Id del reporte>*

Imagen del reporte: *<Incluir la imagen del reporte y una descripción del contenido del mismo>*

Mapeo de campos:

Campo del reporte	Parámetro recibido

Anexo 6. Elementos impactados

1. Listado de elementos y sistemas impactados

Identificación de elementos. *<Listar los elementos (programas, tablas, archivos, etc.) del sistema que se vean afectados por algún requerimiento. Listar también, si fuera el caso, otros sistemas que se vean afectados>*

Requerimiento funcional	Elemento impactado / Aplicativo	Descripción del impacto

2. Observaciones

Anexo 7. Modelo de datos

1. Diseño de tablas

Diseño de tablas (tablas, tipos de datos, constraints, triggers) <*Incluir el diagrama de relación de la base de datos*>

Especificar el nombre de la tabla que se está diseñando y para cada columna:

- Tipo de dato
- Longitud y precisión
- Si permite o no valores nulos
- Valor por default (si aplica)
- Rangos de valores permitidos, fórmula o expresión regular que permita validarlo
- Descripción de la columna

Tabla <Nombre>

Nombre de columna	PK (S/N)	Tipo de dato	Longitud / precisión	Rango de valores permitidos	Descripción de columna

Volumen esperado de crecimiento por tabla <Registros y tamaño esperado>

2. Procedimientos almacenados

Procedimiento almacenado <*Nombre y descripción de la funcionalidad*>

Parámetros de E/S <*Descripción de parámetros*>

Procesamiento <*Descripción del procesamiento indicando, en orden, las principales tareas que ejecutará y las validaciones correspondientes*>

3. Índices

Índice <*Nombre*>

Índice sobre la tabla <*Nombre de la tabla*>

Tipo de índice <*unique constraint / index / binary / etc.*>

¿Puede almacenar valores repetidos? <*Si / No*>

¿Requiere estar ordenado físicamente? <*Si / No*>

Nombre de columna	Orden

4. Opciones de mantenimiento

Opciones de mantenimiento: jobs, depuración

Nombre	Periodicidad	Forma de ejecución (manual / automática)	Descripción de la funcionalidad

Anexo 8. Matriz de pruebas

1. Matriz de pruebas

Nombre del proyecto *<Nombre del proyecto al que corresponden las pruebas>*

No. De caso de prueba	Descripción del caso	Resultado esperado	Status de la prueba

Postcondiciones *<condiciones en las que se encontrará el sistema después de realizar las actividades descritas >*

2. Bitácora de errores

Error	Tipo de error	Fecha de prueba	Fecha de reporte	Fecha de status	Observaciones

Tipo de error: Programación, datos, comunicaciones, conexión a base de datos, usuario, etc.

Anexo 9. Configuración

<Especificar todos los elementos de configuración que requiere la aplicación para su correcto funcionamiento>

1. ***Sistema operativo*** *<Configuración del sistema operativo: procesos batch, horario, idioma, versión, etc.>*

2. ***Plataforma*** *<Plataforma en que correrá la aplicación: Windows, Unix, etc.>*

3. ***Software de arquitectura*** *<Software que sirve de apoyo al aplicativo: IIS, MQ Series, compiladores, etc.>*

4. ***Registry*** *<Para el caso de sistemas operativos Microsoft cuando se utilicen llaves de registro para la operación>*

5. ***Archivos de configuración*** *<Especificar las características a configurar y sus valores >*

6. ***Variables de ambiente*** *<Descripción a nivel sistema y nivel usuario>*

7. ***Cargas iniciales*** *<En caso de que se requiera tener información de tablas o memoria >*

8. ***Tablas / catálogos*** *<Cambios a tablas/catálogos en las aplicaciones liberadas >*

9. ***Accesos*** *<Especificar la relación entre las entidades a las que se brindará acceso y los elementos de software>*

10. ***Otros*** *<Algún otro aspecto de configuración no especificado previamente>*

Anexo 10. Esquema de contingencia

1. Casos especiales

<Casos en que la ejecución del sistema se ve interrumpida por causas ajenas al mismo (fallas de hardware, fallas de energía, error de comunicaciones, etc.) >

Estado	Precondiciones	Actividades	Postcondiciones

Estado *<Nombre>*
Precondiciones *<Explicar la forma en que se llegó a ese estado>*
Actividades *<Pasos que se tienen que ejecutar para salir de ese estado >*
Postcondiciones *<condiciones en las que se encontrará el sistema después de realizar las actividades descritas >*

2. Descripción de la contingencia

<Describir por cada contingencia>

Descripción del caso *<Nombre del caso, descripción y causas que lo provocaron >*

Acciones a ejecutar *<Acciones que se deben ejecutar para solucionar la contingencia >*

CONSTRUCCIÓN

El propósito de esta fase es realizar la codificación (construcción, modificación e instalación) de software que resuelva lo planteado en la especificación funcional y diseño detallado cumpliendo con los estándares y políticas vigentes. Para iniciar este proceso se debe contar con la aprobación del proyecto.

En caso de cancelación del proyecto en esta fase de deberá documentar la causa por la cual se ha decidido no desarrollar el sistema e informar al usuario solicitante.

Es responsabilidad del analista construir la solución de software con las siguientes consideraciones:

1. Mínimo tiempo invertido en la codificación del producto.
2. Máxima velocidad de ejecución del proceso.
3. Optimización del uso de recursos.
4. Documentación interna apropiada con el fin de facilitar el mantenimiento posterior a la liberación del producto.
5. Claridad del algoritmo.
6. Organización visual del código.

Para la validación del desarrollo se deben realizar pruebas de todos los elementos (nuevos y modificados) con el fin de verificar el cumplimiento del documento de la especificación funcional y la funcionalidad misma.

Anexos:

1. **Explicación detallada de la construcción**

Anexo 1. Explicación detallada de la construcción

1. Datos del problema / requerimiento

<Nombre del proyecto>

<Descripción detallada>

2. Explicación detallada de la construcción

<Responsable de la codificación> <Equipo de trabajo> <Extensión>

<Causa>

<Solución>

<Actividades realizadas>

<Tiempo invertido en el desarrollo de la solución>

3. Observaciones

<Observaciones>

PRUEBAS

Una vez generado el código fuente, el software debe ser probado para descubrir (y corregir) el máximo de errores posible antes de la instalación de la aplicación en el ambiente productivo.

El objetivo de esta fase es generar una serie de casos de prueba que tengan una alta probabilidad de encontrar errores.

Las pruebas de software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación.

Las técnicas de pruebas del software facilitan una guía sistemática para diseñar pruebas que:

1. Comprueben la lógica interna de los componentes de software desarrollados

2. Verifiquen los dominios de entrada y salida del programa para describir errores en la funcionalidad, el comportamiento y rendimiento.

El software debe probarse desde dos perspectivas diferentes:

1. La lógica interna del programa se comprueba utilizando técnicas de diseño de casos de prueba de “caja blanca”.

2. Los requisitos del software se comprueban utilizando técnicas de diseño de casos de prueba de “caja negra”.

En ambos casos, se intenta encontrar el mayor número de errores con la menor cantidad de esfuerzo y tiempo.

Objetivos de las pruebas[14]:

1. La prueba es el proceso de ejecución de un programa con la intención de descubrir un error.

2. Un buen caso de prueba es aquel que tiene una alta probabilidad de mostrar un error no descubierto hasta entonces.

3. Una prueba tiene éxito si descubre un error no detectado hasta entonces.

PRUEBA DE CAJA BLANCA

La prueba de caja blanca, denominada a veces *prueba de caja de cris*tal es un método de diseño de casos de prueba que usa la estructura de control del diseño procedimental para obtener los casos de prueba que:

1. Garanticen que se ejercita por lo menos una vez todos los caminos independientes de cada módulo

[14] La prueba no puede asegurar la ausencia de defectos; sólo puede demostrar que existen defectos en el software.

2. Ejerciten todas las decisiones lógicas en sus vertientes verdadera y falsa

3. Ejecuten todos los bucles en sus límites y con sus límites operacionales

4. Ejerciten las estructuras internas de datos para asegurar su validez.

La prueba del camino básico y la prueba de la estructura de control son dos técnicas de prueba de caja blanca[15].

PRUEBA DE CAJA NEGRA

Las pruebas de caja negra, también denominada prueba de comportamiento, se centran en los requisitos funcionales del software, es decir, permiten al ingeniero del software obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa.

La prueba de caja negra intenta encontrar errores de las siguientes categorías:

1. Funciones incorrectas o ausentes

2. Errores de interfaz

3. Errores en estructuras de datos o en accesos a bases de datos externas

4. Errores de rendimiento

5. Errores de inicialización y de terminación.

[15] Capítulo 17, ingeniería del software, Pressman.

Anexos:

1. **Matriz de pruebas**
2. **Plan de pruebas**
3. **Evidencias de pruebas**

Anexo 1. Matriz de pruebas

1. Matriz de pruebas

Nombre del sistema *<Nombre del sistema al que corresponden las pruebas>*

No. De caso	Descripción del caso	Resultado esperado	Datos de entrada	Datos de salida	Status de la prueba	Observaciones

Status:

1. Probado con errores críticos
2. Probado con errores no críticos
3. No probado
4. Probado OK
5. Probado parcialmente
6. Otro

2. Bitácora de errores

<Descripción de errores detectados>

Error	Tipo de error	Fecha de prueba	Fecha de reporte	Fecha de corrección	Status	Observaciones

Tipo de error: Ambiente, producción, desarrollo, datos, especificación funcional.

Status:

1. En espera
2. Terminación OK

3. Bitácora de mejoras

<Descripción de mejoras detectadas>

Descripción de la mejora	Reportado a:	Hora reporte	Fecha reporte	Aplicación mejora	Observaciones

Anexo 2. Plan de pruebas

1. Alcance del plan de pruebas

<Especificar el alcance que tendrá este plan >

2. Riesgos en la ejecución de las pruebas

<Especificar el riesgo que se tendrá en la ejecución de las pruebas >

3. Planeación de las pruebas

<Informar la fecha y tiempo en horas que llevará cada etapa de pruebas>

Etapa	Fecha realización	Status	Horas invertidas
Pruebas integrales			
Pruebas unitarias			
Análisis de resultados			
Liberación			
Seguimiento en producción			
Total de horas invertidas:			

Anexo 3. Evidencias de pruebas

1. Proyecto

<Especificar el nombre del proyecto sobre el cual se presentan las evidencias>

2. Detalle de las evidencias

<Especificar a detalle las evidencias obtenidas de las pruebas ejecutadas>

No. De caso de prueba	Fecha en que se realizaron las pruebas	Analista de pruebas	Extensión del analista	Observaciones

<Imagen de la evidencia >

LIBERACIÓN A PRODUCCIÓN

Una vez probado y corregido el sistema en el ambiente de pre-producción por el área correspondiente y el usuario se tiene un producto final validado y aprobado que cumple con los requisitos originalmente expuestos por el usuario solicitante. Ahora es el momento de generar la solicitud para el pase a producción de la nueva aplicación:

Solicitud de pase a producción

1. Datos generales del cambio

<Datos del solicitante>

<Fecha y hora de instalación>

<Tipo de cambio>

2. Detalles del cambio

<Descripción del cambio>

<Razón del cambio>

<Recursos requeridos para el cambio>

<Recursos afectados por el cambio>

<Usuarios afectados por el cambio>

<Servicios afectados por el cambio>

3. Documentación incluida

<Relación de elementos>

<Plan de pruebas >

<Plan de instalación>

<Plan de trabajo>

<Manual de instalación>

<Procedimiento de operación>

<Plan de retorno en caso de falla>

La liberación de la nueva aplicación requiere una capacitación al usuario final del uso del sistema (operación), la cual debe ser planeada previamente a la puesta en producción. Además de esta capacitación operativa también se entrega un manual de operación del sistema, manual técnico y un plan de soporte en sitio o en línea (de acuerdo a lo estipulado en el contrato firmado).

BIBLIOGRAFÍA

Ingeniería del software, un enfoque práctico. Roger S. Pressman. 5a. edición. Mc Graw Hill.

Análisis y diseño de sistemas de información. James A. Senn. 2ª. Edición. Mc Graw Hill.