UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

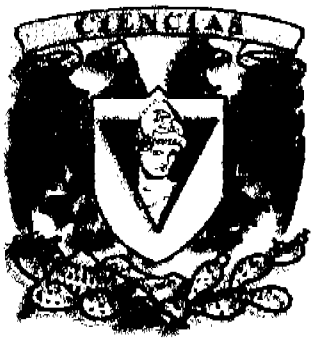
PRÁCTICAS PARA LA INGENIERÍA DE SOFTWARE

P R O Y E C T O

QUE PARA OBTENER EL TÍTULO DE
LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

P R E S E N T A :

GUADALUPE AGUILAR MORALES



FACULTAD DE CIENCIAS
UNAM

DIRECTORA DEL PROYECTO:

M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ

2005

Autorizo a la Dirección General de Bibliotecas de la UNAM a difundir en formato electrónico e impreso el contenido de mi trabajo recepcional.

NOMBRE: Guadalupe Aguilar Morales

FECHA: 10 - Enero - 2005

FIRMA:

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

ACT. MAURICIO AGUILAR GONZÁLEZ
Jefe de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo escrito:

Prácticas para la Ingeniería de Software

realizado por Aguilar Morales Guadalupe

con número de cuenta 09522561-8 **, quien cubrió los créditos de la carrera de:**

Lic. en Ciencias de la Computación

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario M.en C.María Guadalupe Elena Ibargüengoitia González

Propietario Dra. Hanna Oktaba

Propietario Mat. Ana Luisa Solís González Cosío

Suplente Dra. Amparo López Gaona

Suplente M. en C. Gustavo Arellano Sandoval

Consejo Departamental de Matemáticas

Dr. Francisco Hernández Quiroz

CONSEJO DEPARTAMENTAL DE MATEMÁTICAS

Con cariño para mis padres,
por que en todo momento he
contado con su apoyo,
comprensión y aliento para
seguir adelante.
Por que a su lado he pasado
momentos muy felices.

A mi novio por apoyarme en
todo momento, por ser mi apoyo
y darme ánimos para seguir
adelante.

AGRADECIMIENTOS

Agradezco a Dios la oportunidad de permitirme terminar mi carrera universitaria, por darme la paciencia y las ganas de seguir adelante. Y sobre todo por darme a los mejores padres del mundo que me han dado todo su amor y cariño.

A mis padres(Margarita y Justo): Gracias por sus palabras y consejos siempre de aliento, por hacerme ver mis debilidades y errores a tiempo. Y por estar en los malos y buenos momentos. Por su comprensión. Por su confianza. Por creer en mí. Ustedes son mis más estrictos sinodales; me obligan a estudiar y dar siempre un mayor esfuerzo. Muchas gracias. ¡Que Dios los Bendiga!

Mamita: Nadie me hace poner los pies sobre la tierra mejor que usted. Le agradezco por esa sutil combinación de dureza y amor. Es usted mi mamá más mala del mundo.

Papito: Gracias por su apoyo incondicional, su paciencia, su amor, nadie enseña con el ejemplo mejor que usted.

Abuelita y Familiares: Gracias por todo su cariño hacia mí y por todas sus palabras llenas de humildad y por el gran apoyo recibido.

Agradezco a la M. Ma. Guadalupe Elena Ibargüengoitia González por su gran calidez humana, su afinada dirección y por sus palabras de aliento que me motivaron a terminar este trabajo.

Agradezco también los valiosos comentarios y sugerencias de mis sinodales: Doctora Hanna Oktaba, Doctora Amparo López Gaona, Matemática Ana Luisa Solís González Cosío y M. en C. Gustavo Arellano Sandoval.

ÍNDICE

CAPÍTULO X.CONCLUSIONES

BIBLIOGRAFÍA

APÉNDICES

GLOSARIO

INTRODUCCIÓN

El objetivo de este proyecto es facilitar y servir como auxiliar en el aprendizaje de la ingeniería de software en la aplicación de prácticas como método de enseñanza didáctica. En este proyecto se plantea la realización de prácticas en el área de la ingeniería de software para apoyar el aprendizaje en el desarrollo de software.

En este trabajo se puede apreciar cómo se aplican los principios de la ingeniería de software para la construcción de un software que realizan los alumnos. El curso de Ingeniería de software (IS) en la Facultad de Ciencias se basa en el libro "Team Software Process (TSPi)" [Humphrey, 2000].

Durante el curso, el grupo se divide en equipos de 5 personas que desarrollarán un producto de software en 2 ciclos. Para el desarrollo se siguen las fases propuestas por TSPi, cada fase tiene definidas prácticas para apoyar a los alumnos a obtener los productos de esa fase.

Para completar todas las fases se generaron 2 proyectos de titulación, cada proyecto desarrolló 11 prácticas. La repartición es la siguiente:

Principios de TSPI.....**Aguilar Morales Guadalupe**
Lanzamiento.....Avalos Monroy Daniela
Estrategia.....Avalos Monroy Daniela
Planeación .....**Aguilar Morales Guadalupe**
Requerimientos.....Avalos Monroy Daniela
Diseño.....**Aguilar Morales Guadalupe**
Implementación.....**Aguilar Morales Guadalupe** y Avalos Monroy Daniela
Pruebas.....**Aguilar Morales Guadalupe**
Post Mortem.....**Aguilar Morales Guadalupe** y Avalos Monroy Daniela

Este trabajo consta de nueve capítulos donde se subraya la importancia de las prácticas como un método de enseñanza en la ingeniería de software. En el capítulo I, II y III veremos la importancia de las técnicas de enseñanza, cuál es el significado de una práctica como método de enseñanza y una breve introducción al mundo de la ingeniería de software. A partir del capítulo IV se van desarrollando diversas prácticas dependiendo de cada fase en el desarrollo de software, incluye una actividad práctica. La finalidad de estas actividades es diversa: facilitar la compresión de los conceptos, saber organizarse en cada fase para el desarrollo de software, reforzar el aprendizaje de algunos conceptos y familiarizarse con los diversos procedimientos que se tienen que desarrollar.

Agradezco a Microsoft por la donación de Microsoft Project y Visio, para la organización del proyecto de forma fácil y rápida, dándole la oportunidad a los alumnos de utilizar estas herramientas para la realización de sus proyectos.

CAPÍTULO I. TÉCNICAS DE ENSEÑANZA

1.1 INTRODUCCIÓN

La efectividad de la enseñanza se ha convertido en el centro de atención de muchas personas. Las técnicas de enseñanza desempeñan un papel esencial. No basta con perfeccionar libros de texto, programas y planes de estudio, sino que es necesario perfeccionar las técnicas de enseñanza, para que de esta forma se eleve la calidad de la labor de profesores y alumnos.

Se necesita aprender a resolver problemas, a analizar críticamente, a identificar conceptos técnicos, aprender a pensar, a hacer, a ser, a convivir y por último, descubrir el conocimiento profesional de una manera amena, interesante y motivadora.

Uno de los objetivos fundamentales que debe cumplir la enseñanza es llevar al alumno por caminos similares a los que caminó el investigador para llegar a sus conclusiones. En este camino no sólo debe apropiarse activamente del conocimiento, sino de la lógica de la ciencia en la solución de un problema determinado.

1.2 ¿QUÉ ES UNA TÉCNICA DE ENSEÑANZA?

La enseñanza es un conjunto de normas y disciplinas, donde cada individuo es capaz de formular su propio proyecto de vida, mejorando así la capacidad de análisis, síntesis, comparación, investigación. [Borges , 1995]

Una técnica es un conjunto de métodos de organización que se utilizan para resolver las necesidades que se plantean. Este conjunto de procedimientos tiene por objetivo llegar a un resultado determinado. Las Técnicas de enseñanza son métodos de aprendizaje, que permiten perfeccionar la educación y estimular al estudio individual. La experiencia en el

uso de estas técnicas sólo viene a través de la práctica. El objetivo es contribuir al proceso de técnica-enseñanza de tal manera que se pueda aplicar cada técnica de acuerdo con las características planteadas, así como del interés y la estructura institucional.

Las siguientes técnicas de enseñanza son muy importantes:

1.2.1 Presentación oral

Se utiliza solamente por un período corto de tiempo.

En esta técnica de enseñanza, el profesor se dirige al grupo usando notas preparadas con anticipación, con ayudas visuales o gráficas y sin dar oportunidad para que el grupo haga preguntas, sino hasta el final. Esta técnica es sugerida cuando se va a presentar información completamente nueva para el grupo, es recomendable decir antes de empezar, que se va a hablar por poco tiempo y después tendrán la oportunidad para discutir sobre el tema. Al combinar esta técnica con otras, como debates en grupos pequeños o la técnica de preguntas y respuestas permite al grupo expresar su opinión.
En esta técnica se transmite información de expertos.

1.2.2 Presentación oral Ilustrada

Esta técnica de enseñanza es parecida a la anterior, su principal función es apoyarse con dibujos, carteles, copias de artículos publicados y otros materiales. El poder utilizar este tipo de técnica da una amplia visión del tema a tratar, de tal manera que los dibujos o carteles no necesitan ser profesionales, solamente interesantes y claros.

1.2.3 Debate

En esta técnica se da la oportunidad para que cada uno exprese sus ideas, se pueden organizar en pequeños grupos y escoger diferentes temas a discutir. Al aplicar esta técnica existe la ventaja de compartir experiencias, ideas e información. En esta técnica se participa activamente y se aprende, siempre y cuando se contribuyan con ideas para iniciar el debate. En ella surgen puntos de vista diferentes para ayudar al grupo a enfrentar un problema polémico en donde la discusión tiene que enfocarse hacia un solo objetivo. El grupo debe de saber:

I. Qué tipo de temas se van a discutir.
II. Cómo formar equipos.
III. Cómo conducir la discusión.
IV. El tiempo que va a durar la discusión.
V. Cómo reportar el trabajo.

1.2.4 Técnica de preguntas y respuestas

El uso de la técnica de preguntas y respuestas es una manera rápida y efectiva para compartir el conocimiento que el grupo tiene. Se elaboran preguntas para estimular la atención y concentración del grupo en la materia. Esta técnica de preguntas y respuestas se puede lograr de diferentes formas: Una de ellas es compartir el conocimiento que se tiene, estimular la atención y concentración en el tema que se presenta. Otra forma es formular la pregunta y dirigirse específicamente a una persona determinada. El responsable soluciona las preguntas personalmente o pregunta a algún experto en la materia. Puede solicitar preguntas de un grupo determinado y a la vez dirigirlas a otros miembros del grupo para que respondan.

1.2.5 Demostraciones

En esta técnica se demuestra y se explica cómo hacer algo. Otra clase de demostración es la demostración de resultados. Ésta es una manera efectiva de enseñanza. Esta técnica permite demostrar visualmente los resultados que se pueden obtener al experimentar con objetos. Esta técnica es muy efectiva para la enseñanza.

1.2.6 Sesión de trabajo o ejercicio práctico

Esta técnica se puede utilizar junto con cualquiera de los anteriores ya que se tiene la oportunidad de "aprender mientras se va haciendo". La técnica de sesión de trabajo o ejercicio práctico es básica para probar y practicar nuevas experiencias de aprendizaje.

Es una técnica que permite la libre expresión de las ideas de los participantes, sin restricciones o limitaciones, con el propósito de obtener el mayor número de datos, opiniones y soluciones sobre algún tema.

1.2.7 La experimentación

La experimentación consiste básicamente en un conjunto de procesos utilizados para verificar las ideas que se tienen. Es una técnica exitosa para todos aquellos que realizan un estudio individual o de grupo. Aquí es donde se ve si las ideas funcionan o no, muchos de los conocimientos adquiridos proporcionan la experiencia y es una técnica que permite estar seguro de lo que se está haciendo.

Al aplicar esta técnica, lo que se trata de buscar es una solución de calidad y que pueda satisfacer las necesidades que se tengan, promoviendo a la búsqueda de soluciones distintas. Desafortunadamente no en todas las investigaciones se puede aplicar esta técnica, ya que éstas dependen del grado de conocimiento del interesado.

1.2.8 Recorridos y visitas

La técnica de recorridos y visitas se utiliza cuando alguna persona tiene algún proyecto y no se puede transportar prácticamente a cada reunión del grupo. Se puede realizar la visita a la casa del miembro, por una persona especializada y por otros miembros del grupo, para dar la oportunidad de hacer sugerencias constructivas al miembro que se visita. También se puede visitar a las personas con proyectos a largo plazo o donde se puedan ver resultados de proyectos parecidos a los del grupo.

1.2.9 Exhibiciones

Se muestra o se habla de un tema relacionado a un proyecto específico.

Esta última técnica de enseñanza es una actividad de proyecto, que sirve para que otras personas se den cuenta de lo que se está aprendiendo en los proyectos, esta técnica comparte una experiencia de aprendizaje con otras personas y facilita la comunicación interpersonal y grupal en forma ordenada. La técnica de enseñanza habla de un tema relacionado a un proyecto.

1.3 ¿QUÉ ES UNA PRÁCTICA?

La práctica consiste en el desarrollo de una determinada temática, para la que se debe aportar los conocimientos adquiridos, a fin de valorar si el estudiante aplica las metodologías inculcadas para llegar a la correcta solución del tema seleccionado. Según el problema elegido a solucionar, es el grado de definición que se alcanza con la solución buscada. Por tanto, una práctica es el desarrollo de ciertas actividades, en las que se siguen ciertos pasos, que nos enseñan a llegar a hacer algo y así poder lograr los objetivos formativos de aprendizaje, aplicando experimentación.

Las prácticas se enfocan en las diferentes técnicas de enseñanza, como por ejemplo:

- Presentación oral ilustrada: Se utilizan dibujos, carteles, copias de artículos publicados, como complemento de otras técnicas para apoyar temas expuestos.
- Demostraciones: Para demostrar como se realizó algo, paso a paso.
- Sesión de trabajo o ejercicio práctico: Practicar las nuevas experiencias.
- La experimentación: Se verifica que las ideas funcionen.
- Recorridos y visitas: Para reforzar los conocimientos e ideas, apoyándose así de algún proyecto.
- Exhibiciones: Sirve para compartir experiencias con otras personas, donde se habla de algún tema, relacionado con algún proyecto de interés.

Aspectos a considerar en una práctica:

El profesor debe tener en cuenta las características grupales e individuales de los estudiantes: conocimientos, estilo cognitivo, intereses. Se debe tener en cuenta la definición previa de los objetivos que se pretenden, lograr la preparación, selección y secuenciación del contenido, tener un diseño de actividades de alta potencialidad didáctica con metodologías de trabajo activas y muchas veces colaborativas, estas actividades son las que promoverán interacciones (entre los estudiantes y el entorno) generando la enseñanza y el aprendizaje.

El desarrollar una práctica es importante, para conocer el avance de los estudiantes, sus logros, sus dificultades y facilitar el asesoramiento, la orientación de la actividad de los estudiantes cuando convenga. Se realizará un desarrollo flexible adecuando la estrategia didáctica a las circunstancias, hechos e incidencias que se produzcan. Por último, se realiza una reflexión del proceso realizado, analizando los resultados obtenidos y los posibles cambios a realizar para mejorar la intervención educativa.

1.3.1 Metodologías para la resolución de prácticas

Para diseñar una práctica, es necesario imaginar una solución, escribirla y comprobar que funciona. Las prácticas de laboratorio permiten valorar el nivel de conocimientos y habilidades que va adquiriendo el estudiante. En las prácticas de laboratorio los alumnos realizan actividades que les permiten, valorar el nivel de asimilación de conocimientos y las habilidades que van adquiriendo.

En toda práctica se debe tener en cuenta:

- Preparación teórica del estudiante. (conocimientos indispensables.)
- El desarrollo de habilidades y hábitos. (capacidad para realizar ciertas tareas o resolver algún tipo de problemas.)
- El vínculo con otras asignaturas.
- La captura de datos y la realización del informe.

Aspectos que toma en cuenta el profesor:

El profesor diseña la práctica con todos los requerimientos necesarios, realizando todas las observaciones técnicas y de seguridad, así cómo las formas o caminos para la realización de ésta. Divide al grupo en subgrupos de estudiantes de acuerdo a los puestos de trabajo. Nombra a un subgrupo de estudiantes bajo la dirección de uno de ellos, que será rotativa. El primer subgrupo realiza la práctica, mientras el otro subgrupo observa y controla. En todo el proceso anterior los estudiantes han tenido que estar muy atentos, pues se les exige responsabilidad a ambos subgrupos, a uno por la realización y al otro porque responde ante él.

El subgrupo que controlaba pasa a realizar el experimento y el que realizó primero, pasa a ser el controlador. Es decir, rotan y permutan las funciones, de crítico a criticado y de observador a observado.

1.3.2 Diseño de las prácticas de laboratorio

Aspectos para el diseño de la práctica de laboratorio :

- **Fase.** Tema en que se trabaja o fase del proceso.
- **Título de la práctica.** Debe enumerarse y titularse la práctica acorde a la secuencia lógica y el objetivo.
- **Objetivos.** Deben abarcar el contenido de la práctica, tanto conocimientos nuevos, como los que se están consolidando, es decir, concretar las habilidades o la demostración de las leyes.
- **Introducción teórica.** Debe haber una introducción teórica, donde al alumno se le explica en que consiste la práctica, así cómo el procedimiento a seguir para su realización exitosa, hacer énfasis en algunos aspectos del contenido teórico y desarrollar habilidades en la interpretación correcta de esquemas.
- **Equipos y materiales a utilizar y sus parámetros.** Describir los equipos y materiales a utilizar. De los aspectos señalados parecería éste uno de los menos importantes, sin embargo se considera que es fundamental. En la preparación de la práctica y la selección de los equipos, materiales, dispositivos y medios necesarios. El profesor sabe cómo realizar la práctica, estudiando cuidadosamente los contenidos teóricos.
- **Desarrollo.** Aquí se debe dar una lista breve y compacta de pasos a seguir para la correcta realización de la práctica.
- **Conclusiones.** Por último, los alumnos brindan una opinión de lo que significó para ellos el desarrollo de dicha práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
- **Preguntas de control.** Se enfoca a realizar preguntas sobre el tema tratado.
- **Bibliografía.** Enlistar el material bibliográfico de apoyo, utilizado para la realización de las prácticas. Material que los alumnos puedan consultar individualmente.

Para resolver las prácticas en programación, es necesario que el alumno siga una secuencia de pasos: En primer lugar se debe leer el enunciado varias veces, asesorado por el profesor u otros alumnos si es necesario, hasta entender bien el problema que se quiere resolver. Pensar, cómo resolver el problema mediante varios pasos: cómo va a representar los conceptos clave del problema y sus relaciones. (haciendo dibujos, anotaciones o gráficos que permitan comprender el problema y cómo va a resolverlo, cómo construir el objeto, qué atributos poner en cada clase, cómo escribir la cabecera de un método, cómo escribir el cuerpo.) Escribir el código (el programa) que soluciona el problema, comentar las líneas inconclusas. Ejecutar el programa con diferentes datos y en diferentes situaciones. Si se encuentran errores, localizar la parte del programa que no funciona correctamente y corregirla. Y por último, es entregar un informe elaborado con los datos obtenidos en la práctica y profundizar teóricamente en los mismos, lo que permitirá la adquisición de hábitos de estudio independiente. [Álvarez, 1989]

CAPÍTULO II LA INGENIERÍA DE SOFTWARE Y SU PROBLEMÁTICA

2.1 INTRODUCCIÓN

2.1.1 ¿Qué es la ingeniería de software?

Es el conjunto de métodos, técnicas y herramientas que controlan el proceso del desarrollo de software. En 1980 y 1990, dos aspectos dominaron la ingeniería de software. Uno fue el crecimiento explosivo de aplicaciones, incluidas las asociadas a Internet. Otro fue el desarrollo de nuevas tecnologías (orientado a objetos). La ingeniería de software es una rama muy joven de la ingeniería y por ello se encuentra en un estado de cambio rápido. La British Computer Society, por ejemplo, la primera ocasión en que otorgó el certificado Chartered Engineer a los ingenieros de software fue al principio de la década de 1990. En 1998 por primera vez fue posible registrarse como ingeniero de software profesional en un lugar de Estados Unidos: el estado de Texas. [Braude, 2003]

Evolución

Primera era. Principios de 1960`s es donde se investigó principalmente el hardware. Al software se le veía como un añadido. El software se desarrollaba sin ninguna planificación.

Características:

- Distribución limitada.
- Orientación por lotes (tarjeta → resultado). No interactivo.
- Software a la medida.

Segunda era. Mediados de los 60`s, finales de los 70`s, en esta etapa los sistemas informáticos llegan a mucha más gente. Por tanto, hay muchas posibilidades de que el software llegue a más usuarios. El software cobra más importancia: Aparecen casas desarrolladoras de software.

Características:

- Multiusuarios.

- tiempo-real.
- bases de datos.
- producto de software.

Tercera era. Mediados de los 70`s , finales de los 80`s, aquí es donde se desarrollan todos los dispositivos de hardware actuales: Sistemas distribuidos, redes, programación estructurada, explosión de metodologías estructuradas. El hardware se estandariza, pero el software no.

- sistemas distribuidos.
- inteligencia embedida.
- bajo costo de hardware.
- impacto al consumidor.

La gente gasta más dinero en el software que en el hardware de su empresa.

Cuarta era. Desde principios de los 90`s. Todo el mundo tiene acceso al hardware más potente. Los avances están viniendo en nuevos conceptos de software.

Características:

- computadoras personales poderosas.
- tecnologías orientadas a objetos.
- sistemas expertos.
- redes neuronales artificiales.
- cómputo paralelo.
- redes de computadoras.

2.2 ACTIVIDADES PARA LA CONSTRUCCIÓN DE SOFTWARE

Las actividades básicas requeridas para la construcción del software son: [Braude, 2003]

- **Definición del proceso de desarrollo de software.** La ingeniería de software estudia el proceso de construir aplicaciones de alcance práctico, el cual incluye las herramientas, métodos, lenguajes de programación y apoyo.

- **Administración del proyecto de desarrollo.** Consiste en gestionar la producción de un producto dentro de un tiempo dado y con límites en el costo. No sólo involucra la organización técnica y las habilidades organizacionales, también incluye el arte de administrar personas.
- **Descripción del producto de software que se quiere**. En esta etapa el objetivo es describir el producto (función del producto o finalidad).
- **Diseño del producto**. En esta etapa se deben construir los modelos para la realización del producto. Su meta es preparar por completo el proyecto para la implementación.
- **Implementación del producto**. En esta etapa se comienza a trabajar en la programación. Se deben satisfacer los requerimientos de la manera en que se especifica en el diseño del producto.
- **Prueba de las partes del producto.** Las pruebas son para detectar en que parte una aplicación no es satisfactoria. La fase de pruebas consume más de la mitad del tiempo dedicado al producto.
- **Integración de las partes del producto y pruebas del producto completo.** El proceso de integración consiste en unir los elementos que sirven y revisar que el producto construido esté completo. Se verifica que todas las partes necesarias estén presentes en la construcción.
- **Mantenimiento del producto.** Son las actividades realizadas sobre la aplicación, una vez entregado el producto. En esta etapa se corrigen los defectos, se adapta al cambio del entorno y se mejora el desempeño.

La ingeniería de software incluye personas, proceso, proyecto y producto.(FIGURA 2.1)

- Una *persona* se refiere a los interesados en el proyecto: personas que ganan o pierden algo con su resultado. Éstas incluyen el cliente y el usuario final.
- Un *producto* de software es el resultado del diseño para un usuario. También se llama artefacto.
- Un *proyecto* es el conjunto de actividades para producir los artefactos requeridos. Incluye contacto con el cliente, escribir la documentación, desarrollar el diseño.
- Un *proceso* es el conjunto de actividades a realizar par producir una aplicación, está consta de varias etapas: Requerimientos(reúne las necesidades del

producto), diseño (describe la estructura interna del producto, se representa con diagramas), implementación(o la programación), pruebas(donde el producto no satisface las condiciones pedidas). [Braude, 2003]

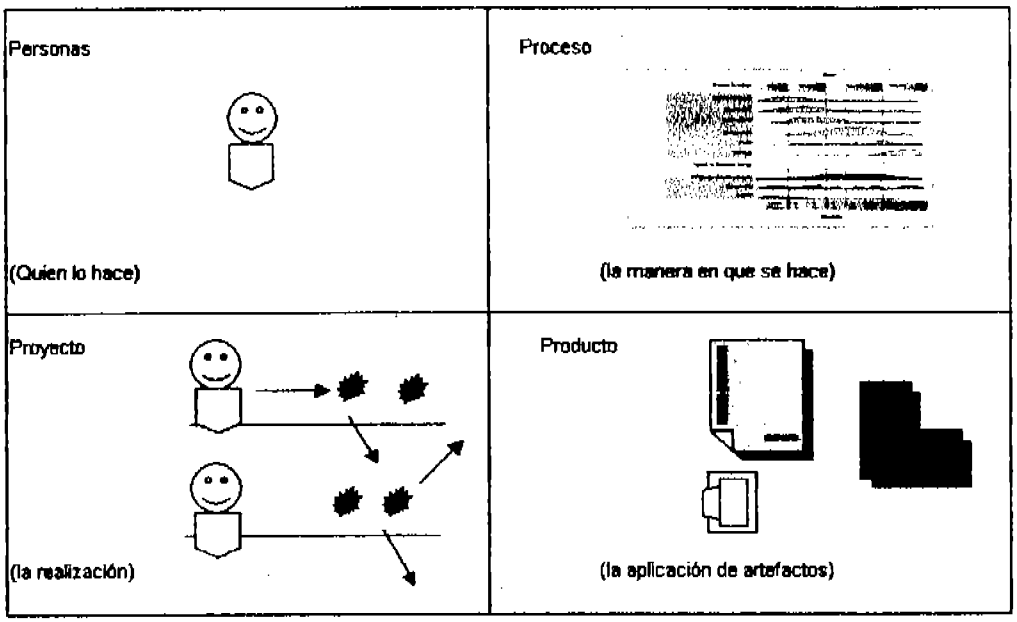


FIGURA 2.1 Las cuatro "P" de la ingeniería de software. [Braude, 2003]

2.3 LA INGENIERÍA DE SOFTWARE Y SU PROBLEMÁTICA

Software

El software es un concepto lógico, no físico y de aquí sus características difieren de las del hardware.

2.3.1 Características del software

El software se desarrolla, no se fabrica en un sentido clásico. El software no se estropea, solamente se deteriora. El software sufre cambios y esos cambios introducen nuevos defectos.

La mayoría del software se construye a medida, en lugar de ensamblar componentes existentes.

2.3.2 Aplicaciones de Software

- **Software de sistemas.** Conjunto de programas que han sido escritos para servir a otros programas. Existe una fuerte interacción con el hardware de la computadora. Tiene un soporte para multiusuarios y cuenta con una óptima gestión de recursos y procesos.

  Ejemplos: Sistemas Operativos, Emuladores, Compiladores, Editores.
- **Software de tiempo real.** Software que mide, analiza, controla sucesos del mundo real conforme ocurren. Formatea la información reciclada y la analiza, se tiene un control de salida al mundo externo.

  Ejemplos: Control de robots, Estaciones meteorológicas, Centrales aeronáuticas.
- **Software de gestión.** Procesamiento de información comercial.

  Ejemplos: Contabilidad, Control de recursos, Nóminas, Cuentas de débito.
- **Software de Ingeniería y científico.** Está caracterizado por los algoritmos de manejo números. Gran importancia de las aplicaciones CAD, Simulación.

  Ejemplos: Funciones digitales de un automóvil, tales como control de la gasolina.
- **Software empotrado.** Los programas se encuentran en la memoria de sólo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo.

  Ejemplos: televisores, sistemas de control de un coche.
- **Software de inteligencia artificial.** Hace uso de algoritmos no numéricos para resolver problemas complejos para los que son adecuados el calculo o el análisis directo.

  Ejemplos: Sistemas expertos, redes neuronales, algoritmos genéticos.

2.3.3 Problemas relacionados con el software

Uno de los problemas del software es el no satisfacer la demanda de nuevos programas, teniendo así una gran batalla para crear software altamente confiable. Existen problemas para entender y extender problemas existentes debido a un pobre diseño y a recursos inadecuados. Las aplicaciones que se tienen, necesitan reconstruirse. Muchas veces el diseño es difícil de entender.

2.4 CICLO DE VIDA DEL DESARROLLO DE SOFTWARE

El desarrollo de software va unido a un **ciclo de vida** compuesto por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto de software, hasta que el producto deja definitivamente de ser utilizado por el último de sus usuarios.

Cuando se diseña un producto de software se suele poner atención al costo de desarrollo, pero rara vez se toma en consideración el costo de funcionamiento. El costo de funcionamiento corresponde al de la operación normal, etapa durante la cual es necesario efectuar manutención al software, sea para eliminar errores, sea para mejorar su rendimiento, sea para adaptarlo a nuevas condiciones del entorno.

Se admite que la operación y manutención es la fase más costosa y crítica del ciclo de vida del software. El proceso de desarrollo de software requiere por un lado de un conjunto de conceptos, una metodología y un lenguaje propio. A este proceso también se le llama el ciclo de vida del software.

2.4.1 Ciclo de vida clásico (o en cascada)

Se llama modelo de cascada porque sus etapas se distribuyen de manera secuencial, es decir, una después de la otra.

Este modelo de desarrollo de software se ilustra en la (FIGURA 2.2) y sus fases son:

- *Análisis de requerimientos :* En esta etapa se determina, si el proyecto se puede o no realizar, determinando aproximadamente los costos. Consiste en reunir las necesidades del producto y casi siempre su salida es texto. Se debe identificar y documentar los requerimientos exactos del sistema según las necesidades de los usuarios finales.

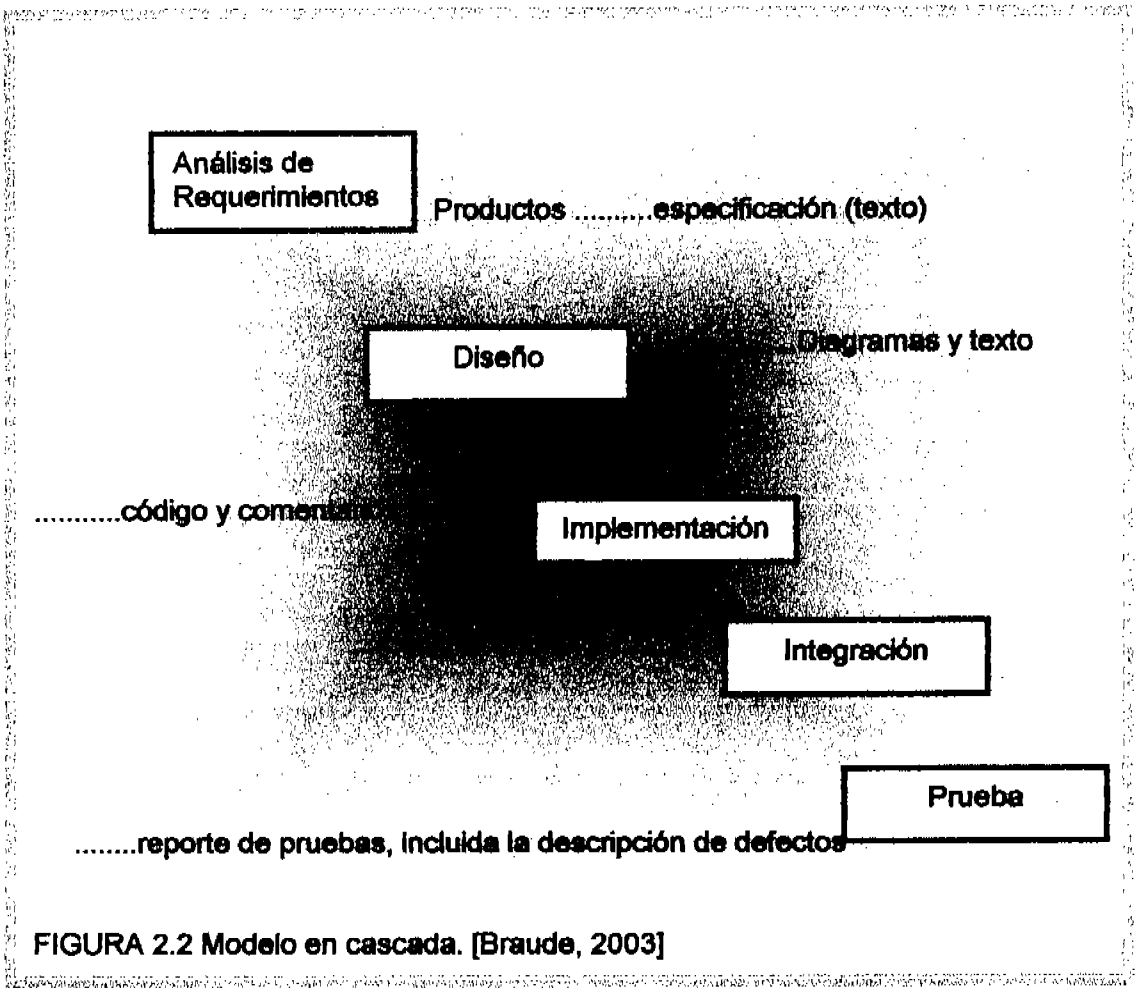


FIGURA 2.2 Modelo en cascada. [Braude, 2003]

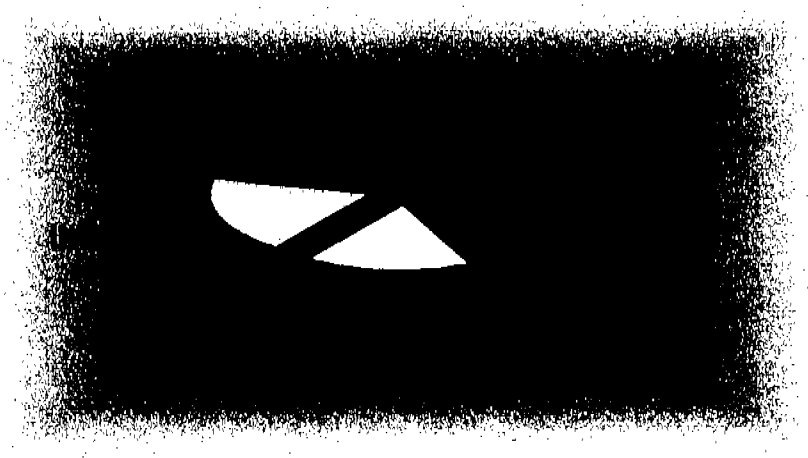
- *Diseño:* El diseño del software es realmente un proceso multipaso basado en tres atributos distintos del programa:
  1) La estructura de los datos.
  2) La arquitectura del software.
  3) El detalle procedimental.

  El proceso de diseño traduce los requisitos en una representación del software, que pueda ser establecida de tal forma que obtenga la calidad requerida antes de comenzar la codificación y estableciendo que hará cada parte, donde el diseño debe permitir implementaciones que verifiquen los requerimientos.
- *Implementación:* Significa programación. El producto de esta etapa es el código en cualquier nivel. El diseño es traducido en una forma legible para la máquina. Si el

diseño se realiza de una manera detallada, la implementación puede realizarse mecánicamente y traducirse a un lenguaje de programación.

- *Integración:* Es el proceso de ensamblar las partes para completar el producto.
- *Prueba:* Una vez generado el código, comienza la prueba del programa. La prueba se centra en la lógica interna del software, asegurando que todas las sentencias se han probado y en las funciones externas se realizarón pruebas que aseguren que la entrada definida produce los resultados que realmente se requieren. La distribución de cada fase se presenta en la siguiente (GRÁFICA 2.3).

CICLO DE VIDA CLÁSICO



GRAFICA 2.3 La distribución del esfuerzo para un ciclo de vida clásico.
[http://www.ii.uam]

2.5 PSP (Personal Software Process)

PSP : Creado por Watts Humphrey. [Humphrey, 1999] PSP se concentra en las prácticas personales de trabajo de los ingenieros, en una forma individual. El principio fundamental de PSP es servir para producir software de calidad, donde cada ingeniero debe trabajar,

para realizar un buen trabajo de calidad. PSP se diseñó para ayudar a profesionales del software para que utilicen constantemente prácticas sanas de la ingeniería de software.

2.5.1 Principios de PSP (Personal Software Process)

El diseño de PSP se basa en los principios de planeación y de calidad. El Proceso de Software Personal o Personal Software Process (PSP) es un proceso de automejora diseñado para ayudar a controlar, administrar y mejorar la forma de trabajar.
[Humphrey, 95]

- Cada ingeniero es diferente, debe planear su trabajo y basar sus planes en sus propios datos personales.
- Para mejorar constantemente su funcionamiento, los ingenieros deben utilizar procesos bien definidos y medidos.
- Para desarrollar productos de calidad, los ingenieros deben sentirse comprometidos con la calidad de sus productos.
- Cuesta menos encontrar y arreglar errores en la etapa inicial del proyecto, que encontrarlos en las etapas subsecuentes. Es más eficiente prevenir defectos que encontrarlos y arreglarlos.

2.5.2 Niveles de PSP

PSP proporciona métodos detallados para estimar, planear y mostrar a los ingenieros cómo dar seguimiento a su desempeño contra estos planes, y explicar cómo los procesos definidos pueden guiar su trabajo.

PSP tiene el propósito de desarrollar buenos hábitos de programación. PSP incorpora el uso de cuatro niveles de un proceso definido, empezando en el nivel 0 contando con sus propios requerimientos: PSP0: Mediciones básicas de tiempo y de conteo de defectos. PSP1: Mediciones y estimaciones de tamaño. PSP2: Diseño y revisiones. PSP3: Ciclos de desarrollo. (FIGURA 2.4)

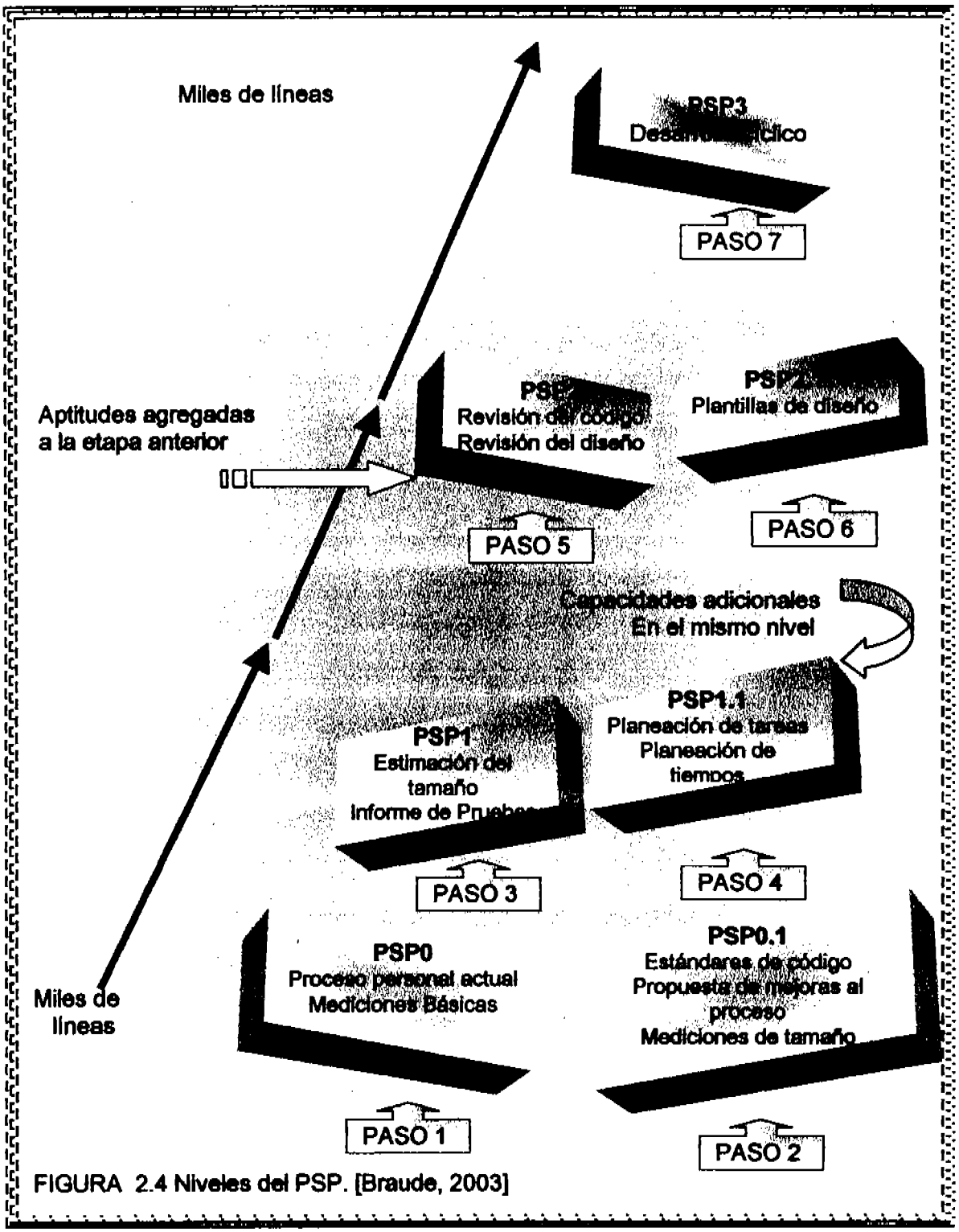


FIGURA 2.4 Niveles del PSP. [Braude, 2003]

PSP0: Proceso base PSP0 acepta las prácticas de desarrollo actuales del estudiante y requiere :

- Mantener un registro del tiempo dedicado a trabajar en un proyecto.
- Registrar los defectos encontrados.
- Registrar los tipos de defectos.

PSP1: Proceso de planeación personal El PSP1 está diseñado para ayudar al ingeniero a entender la relación entre el tamaño de los programas y el tiempo que toma desarrollarlos. Proporciona un marco de trabajo ordenado dentro del cual el individuo puede realizar estimaciones, hacer compromiso, evaluar el estado y registrar los resultados.

PSP2: Proceso de administración de la calidad personal El PSP2 está diseñado para ayudar a los ingenieros a "manejar de manera realista y objetiva" los defectos de programación. La idea es estimar tantos defectos como sea posible antes de someter el programa a una inspección formal.

PSP3: Proceso personal cíclico El PSP3 está diseñado para escalar el PSP para manejar las unidades de código grandes (en miles de líneas) dividiendo un programa grande en pequeños incrementos. PSP3 agrega la aplicación de PSP a cada incremento para producir una alta base de calidad para los incrementos sucesivos y el uso de pruebas de regresión para asegurar que las pruebas diseñadas para los incrementos anteriores todavía son buenas en los nuevos incrementos. [Braude. 2003]

2.6 TSP (Team Software Process)

2.6.1 Introducción

TSP sirve para construir y guiar equipos interdisciplinarios, construye un equipo de trabajo con calidad, se basa en ingenieros acostumbrados individualmente a trabajar con calidad, según PSP. Integra equipos independientes de alto rendimiento que planeen y registren su trabajo, estableciendo metas y siendo dueños de sus procesos y planes. Muestra a los gerentes cómo monitorear y motivar a sus equipos de trabajo y ayudarlos a alcanzar su máxima productividad.

Acelerar la mejora continua de procesos.

- Proporciona:
  - Un proceso definido para construir el grupo.
  - Un marco para el trabajo en grupo.
  - Un ambiente de gestión para soportarlo.
- Diseño para equipos de desarrollo y mantenimiento que incluye:
  - Un proceso definido para el trabajo en grupo.
  - Roles definidos para los miembros del grupo.

2.6.2 Enfoque de TSP

- Planificar el trabajo antes de comenzarlo.
- Usar un proceso definido.
- Medir y seguir el tiempo de desarrollo, tamaños y búsqueda de defectos.
- Planificar y medir la calidad del producto.
- Analizar cada tarea y utilizar los resultados para la mejora del proceso.

2.6.3 Fases de TSP

El Proceso de Software en Equipo (TSP) es un marco de trabajo definido para cursos de ingeniería de software realizándose en equipo. Cada fase es una guía de pasos a seguir, indicando cómo se debe aplicar el conocimiento de la ingeniería de software y los principios del proceso en un ambiente de equipo de trabajo, definiendo los roles para cada uno de los integrantes.

Las formas de TSP están adaptadas a la idiosincrasia de México según la experiencia (ver apéndice). Las fases son:

- *Principios de TSPi*.- Definir roles, prácticas y métodos de desarrollo, para generar beneficios necesarios al enfrentarse con problemas del proyecto y así poder obtener soluciones efectivas.
- *Lanzamiento*

Cada proyecto empieza con un lanzamiento.

- Lleva 3 o más días.
  - Es parte del proyecto.

- Dirigido por un mentor preparado en TSP.

❖ En el lanzamiento:

- Se eligen roles personales que pueden ser:
  - ✦ Administrador de Planeación.
  - ✦ Administrador de Calidad.
  - ✦ Administrador de Apoyo o Soporte.
  - ✦ Administrador de Desarrollo.
  - ✦ Líder del equipo.

➢ *Estrategia.-* Crear una estrategia para realizar el trabajo. Crear el diseño conceptual del producto. Documentar la estrategia.

➢ *Planeación.-* Se realiza el plan de desarrollo del proyecto y el plan de calidad. Aplicando el criterio de estándares de calidad para generar un producto de alta calidad.

➢ *Requerimientos.-*Se describe el proceso de requerimientos en TSPi.

➢ *Diseño.-* En TSPi, la fase de diseño se enfoca a la estructura completa del sistema, describe la estructura interna del producto y suele representarse con diagramas y texto. Su meta es preparar por completo el proyecto para su implementación.

➢ *Implementación.-* En esta fase es donde se empieza a construir el código, de acuerdo a la manera en que se diseñó.

➢ *Pruebas.-* En TSPi, el propósito de las pruebas consiste en evaluar el producto. La calidad de un producto se determina cuando se desarrolla.

➢ *Postmortem.-* Es el último paso en el proceso de TSPi, donde se revisa que el equipo de trabajo haya terminado todas las tareas y registrado todos los datos requeridos. Proporciona una manera estructurada de aprendizaje y mejora, porque se evalúa el desempeño personal y del equipo. (FIGURA 2.5)

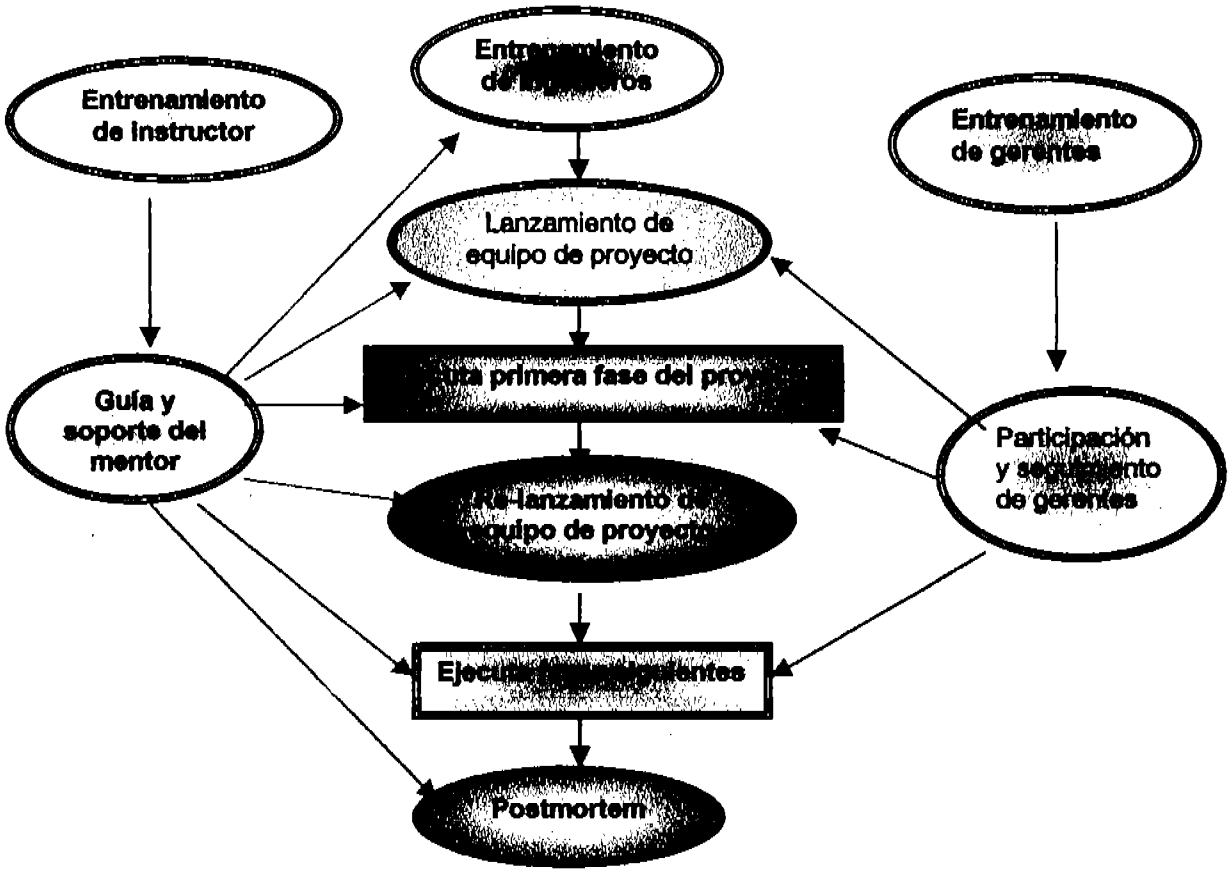


FIGURA 2.5 TSP Visión general. [Humphrey , 1999]

2.7 MATERIAL DE APOYO A LA INGENIERÍA DE SOFTWARE (IS)

2.7.1 Introducción

El material de apoyo es necesario para la enseñanza de la (IS) donde se proporciona información sobre un tema. Como material de apoyo se tienen *libros*, como objetos portables y duraderos, que han ayudado a preservar y difundir el conocimiento. El servicio de *Internet* permite intercambiar información en una colección de archivos, que incluye información en forma de textos, gráficos, y además tiene vínculos con otros archivos sobre un tema determinado.

2.7.2 Material de apoyo a IS

- Gregory W. Jones, Software Engineering, Jhon Wiler & Sons, 1990.
- Ian Sommerville, Software Engineering, Addison-Wesley.
- Lem O. E, Software Engineering with formal metrics, QED technical Publishing Group.
- Ronald J. Leach, Introduction to Software Engineering, CRCpress 2000.
- UML Resource Center.

Disponible en: http://www-306.ibm.com/software/rational/uml/

- Watts S. Humphrey, A discipline for Software Engineering, Addisson-Wesley, 2000.
- Seguridad de la ingeniería de software.

Disponible en: http://www.acm.org/crossroads/espanol/xrds7-4/onpatrol74.html

2.7.3 Material de apoyo a TSP

TSP es una guía para los grupos en cada una de las etapas de desarrollo del proyecto. Para ello también se tiene material de apoyo en Internet, cuya función consiste en proporcionar información de TSP (Team Software Process).

- Building High Performance Teams Using Team Software ProcessSM (TSP) and Personal Software ProcessSM (PSP).

Disponible en: http://www.sei.cmu.edu/tsp/

- Team Software Process Reliability Results.

Disponible en: http://www.softwaretechnews.com/stn3-4/teamspi.html

- Teaching the Team Software Process.

Disponible en: http://oce.spsu.edu/cseet2002/Anthony-Lattanze.pdf

- Watts S. Humphrey, "A Discipline for Software Engineering", Addison-Wesley, 1995.
- Watts S. Humphrey, Introduction to the Personal Software Process, Addison-Wesley, 2000.

CAPÍTULO III. LAS PRÁCTICAS EN INGENIERÍA DE SOFTWARE

3.1 INTRODUCCIÓN AL CURSO DE (IS) EN LA FACULTAD DE CIENCIAS (UNAM)

El curso de ingeniería de software en la Facultad de Ciencias se basa en el libro Team Software Process (TSPi) [Humphrey, 2000]. El proceso de TSPi está diseñado para desarrollar el producto de software en dos ciclos, el cual requiere de la participación de los estudiantes en proyectos por equipos de 5 personas que desarrollarán el producto de software, en el cual cada alumno tendrá a su cargo un rol determinado. Con TSPi, todos los integrantes del equipo tienen objetivos específicos. Los Roles son los siguientes:

- Líder del equipo: Mantener un equipo unido.
- Administración de Planeación: Generar un plan completo, detallado para el equipo.
- Administrador de desarrollo: Utilizar las destrezas y habilidades de los integrantes.
- Administrador de calidad y proceso: Realizar reportes de los datos de TSPi.
- Administrador de apoyo:Facilitar al equipo las herramientas de apoyo.

Para el desarrollo de dicho producto se siguen las fases propuestas por TSPi:

1.- Principios de TSPi	2.- LANzamiento	3.- ESTRATegia
4.- PLANeación	5.- REQuerimientos	6.- DISeño
7.- IMPlementación	8.- PRUeba	9.- Postmortem

Para la realización de cada fase, se deben de llenar determinadas formas, las cual se encuentran en el apéndice A, B. Se cuenta con un guión para el desarrollo de Proceso de Software en Equipos (Ver apéndice C). Se utilizan dos herramientas importantes para la realización de dicho proyecto: Microsoft Project y Visio.

3.2 TEMAS DE INGENIERÍA DE SOFTWARE (IS) QUE REQUIEREN PRÁCTICAS

3.2.1 Temas de importancia en IS

La importancia de proporcionar prácticas para cada una de las etapas por las que pasa un ingeniero de software, es para realizar un buen producto en el momento del desarrollo,

ya se mencionó en el apartado anterior, que pasa por varias etapas y cada etapa es de vital importancia. Por eso se deben desarrollar prácticas para todas las etapas. Para ello una parte importante de la ingeniería de software es saber cómo hacer un *documento,* en el se incluye la documentación separada del código, al igual que la documentación asociada a él. Esto sirve para entender el código .

Los temas para los que se harán las prácticas son:

- *Principios de TSPi.-* Consiste en definir prácticas y métodos de desarrollo, para generar beneficios necesarios al enfrentarse con problemas del proyecto.
- *Lanzamiento .-* Consiste en formar equipos y definir roles.
- *Estrategia.-* Crear el diseño conceptual del producto. Documentar la estrategia.
- *Planeación.-* Se realiza el plan de desarrollo del proyecto y el plan de calidad.
- *Análisis de requerimientos:* Es un proceso de conceptualización y expresión de los conceptos de forma concreta, errores en el análisis de requerimientos que son los más caros para reparar.
- *Diseño:* Es preparar por completo el proyecto para su implementación.
- *Implementación* : En esta etapa se debe tener el plan de desarrollo y la estrategia terminados; las especificaciones de requerimientos y de diseño terminadas, revisadas y actualizadas.
- *Pruebas:* El propósito de las pruebas consiste en evaluar el producto y no en modificarlo.
- *Postmortem.-* Se revisa que el equipo de trabajo haya terminado todas las tareas y registrado todos los datos requeridos. [Braude, 2003]

3.3 LISTADO DE PRÁCTICAS

En el listado de prácticas que a continuación se presenta, se seguirá TSP. En donde cada fase consta de sus respectivas prácticas. La secuencia de las prácticas están en orden, pero la presentación de cada uno varia, ya que son dos tesis complementarias. Pero trabajadas a la par para el desarrollo de software.

3.3.1 Principios de TSPI

- Práctica1.- Motivación al trabajo en equipo.
- Práctica2.- Llenado de formas básicas Semana Personal, Semana del equipo, REGT, REGD.

3.3.2 Lanzamiento

- Práctica 3.-Definir el equipo.
- Práctica 4.- Establecer objetivos.

3.3.3 Estrategia

- Práctica 5.- Definición de estándares.
- Práctica 6.- Estrategia de desarrollo.

3.3.4 Planeación

- Práctica 7.- Introducción a Microsoft Project.
- Práctica 8.- Planeación del proyecto y plan de calidad.

3.3.5 Requerimientos

- Práctica 9.- Diagramas de caso de uso con Visio.
- Práctica 10.- Detallar los casos de uso.
- Práctica 11.- Construcción del prototipo.
- Práctica 12.- Requerimientos no funcionales y plan de pruebas del sistema.

3.3.6 Diseño

- Práctica 13.- Diseño de la Arquitectura del sistema y Plan para la prueba de Integración.
- Práctica 14.- Diagramas de clases y de secuencia.

3.3.7 Implementación

- Práctica 15.- Creación y conexión a Bases de datos (BD).
- Práctica 16.- Paso del modelo de clases a código java.
- Práctica 17.- Desarrollo Web JSP´s, SERVLETS Y TOMCAT.
- Práctica 18.- Pruebas Unitarias.

3.3.8 Pruebas

- Práctica 19.- Pruebas de Integración y del sistema.
- Práctica 20.- Manuales del sistema.

3.3.9 Postmortem

- Práctica 21.- Recolección de métricas del proceso.

- Práctica 22.- Evaluación de roles en el equipo.

3.4 PLANTILLA DE PRÁCTICAS

Para realizar una buena práctica, se necesita conocer los pasos que se deben seguir. Ya se había mencionado la importancia de realizar prácticas como un método de enseñanza y para ello una práctica consta de (FIGURA 3.1) :

- **Fase.** Tema en que se trabaja o fase del proceso.
- **Título de la práctica.** Debe enumerarse y titularse la práctica acorde a la secuencia lógica y el objetivo.
- **Objetivos.** Deben abarcar el contenido de la práctica, tanto conocimientos nuevos, como los que se están consolidando, es decir, concretar las habilidades o la demostración de las leyes.
- **Introducción teórica.** Debe haber una introducción teórica, donde al alumno se le explique en que consiste la práctica, así cómo el procedimiento a seguir para su realización exitosa, y hacer énfasis en algunos aspectos del contenido teórico, así cómo desarrollar habilidades en la interpretación correcta de esquemas.
- **Equipos y materiales a utilizar y sus parámetros.** Describir los equipos y materiales a utilizar. De los aspectos señalados parecería éste uno de los menos importantes, sin embargo se considera que es fundamental. En la preparación de la práctica y la selección de los equipos, materiales, dispositivos y medios necesarios. El profesor sabe cómo realizar la práctica, estudiando cuidadosamente los contenidos teóricos.
- **Desarrollo.** En el desarrollo se da una lista breve y compacta de pasos a seguir para la correcta realización de la práctica.
- **Preguntas de control.** Se enfoca a realizar preguntas sobre el tema tratado.
- **Conclusiones.** Por último los alumnos brindan una opinión de lo que significó para ellos el desarrollo de dicha práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
- **Bibliografía.** Enlistar el material de apoyo utilizado para la realización de las prácticas. Material que los alumnos puedan consultar individualmente.

PRÁCTICA # N

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: Tema en que se trabaja o fase del proyecto.

INGENIERÍA DE SOFTWARE

TÍTULO DE LA PRÁCTICA

Debe enumerarse y titularse la práctica acorde a la secuencia lógica y el programa que se desarrolla.

OBJETIVOS :

- Debe abarcar el contenido de la práctica, que actividades se deben realizar.

INTRODUCCIÓN :

Explicar a los alumnos en que consiste la práctica y cuales son las bases fundamentales.

MATERIAL:

Especificar que material (o software) se utilizará para la realización de la práctica.

DESARROLLO:

Se da una breve y compacta lista de pasos a seguir para la correcta realización de la práctica.

PREGUNTAS DE CONTROL:. Es una parte muy importante de la práctica ya que en ella se realizan preguntas sobre el tema a tratar.

CONCLUSIONES : Los alumnos explican de manera informal los resultados obtenidos y brindan una opinión de lo que significó la realización de dicha práctica.

BIBLIOGRAFÍA : Ofrecer información acerca del autor, título, del que abarca dicho tema.

FIGURA 3.1 Plantilla para prácticas.

CAPÍTULO IV FASE PRINCIPIOS DE TSPi (Práctica #1, Práctica #2)

4.1 INTRODUCCIÓN

En este cuarto capítulo se presentan las prácticas sobre los principios de TSPi. Esta fase consta de dos prácticas para el inicio del desarrollo de software, donde se aprende la motivación al trabajo en equipo y la manera de llenar las formas que propone TSP, será de gran apoyo para llevar un control del tiempo de los miembros del equipo.

4.2 MOTIVACIÓN AL TRABAJO EN EQUIPO

El objetivo es definir una meta para el equipo y que cada uno de los integrantes identifiquen lo necesario para alcanzar esa meta y que son necesarios para lograr un acoplamiento eficaz entre ellos.

En un equipo se trabaja para lograr una meta, aceptando ciertas normas, compartiendo trabajo y formando un espíritu de equipo. Los miembros del equipo necesitan pasar tiempo juntos, tener intereses y habilidades complementarias para poder formar un buen equipo. Para ser un miembro efectivo de un equipo, cada persona necesita conocer sus habilidades e intereses. En esta práctica identificaremos nuestras habilidades e intereses.

4.3 LLENADO DE FORMAS BÁSICAS

El objetivo de esta práctica es aprender a llenar las formas TSPi , aprender a llevar un control del tiempo para cada tarea realizada, llevar un control de defectos. Evaluarse en equipo e individualmente. En esta práctica se conocerán estas formas básicas, para qué sirven, qué campos tienen y cómo se llenan.

PRÁCTICA # 1

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **Principios de TSPI.**

INGENIERÍA DE SOFTWARE

MOTIVACIÓN AL TRABAJO EN EQUIPO.

OBJETIVOS :

- Formación de un buen equipo.
- Motivar la reflexión personal sobre las habilidades de cada persona.
- Hacer una tarea en equipo.

INTRODUCCIÓN :

Equipo: Conjunto de dos o más personas que trabajan para un mismo objetivo, interaccionando entre ellas, aceptando ciertas normas y compartiendo emociones, participando de un sentimiento común llamado espíritu de equipo.

Para trabajar en equipo se deben tener los siguientes elementos:

- **Objetivo común:** Es la característica que define y da vida al equipo. Para lograr los objetivos que asigna a cada individuo un rol, y cada rol representa una faceta en las actividades a realizar en el equipo.
- **La estructura:** Es la organización del equipo para desarrollar sus actividades.
- **Los roles:** Un rol es un conjunto de responsabilidades para cada individuo.
- **Las normas** Son las reglas de funcionamiento de un equipo. Deben ser conocidas y aceptadas por el equipo.
- **Valores y creencias** Es el cuerpo de conocimientos que resumen el aprendizaje colectivo de un equipo.
- **Sistema de comunicación interno y externo** Es la forma en que los miembros, y el líder, han establecido para intercambiar mensajes entre ellos y con el exterior. La comunicación permitirá al equipo fijar y desarrollar sus normas de actividades, y consolidar su estructura, pueden también definir, diagnosticar y resolver problemas comunes.

DESARROLLO:

PARTE I.- En esta primera parte reflexionaremos sobre las características de cada persona. Contesta sinceramente las siguientes preguntas

- *Como persona:*
  - Algo que hago bien es:
  - Lo mejor de ser yo es:
  - Me gusta hacer trabajo rutinario o me gusta hacer cambios.
  - Me gusta trabajar bajo presión o no.
  - Soy analítico o intuitivo.

- ❖ Planeo excesivamente, o soy atrevido y audaz sin planeación.
- ❖ Mi gran fuerza personal es:
- ❖ Me gusta trabajar solo o en equipo.

➢ *Mi desempeño en equipo:*

- ❖ Estás integrado en algún grupo (juvenil, fútbol, coro, etc..) y que papel desempeñas.
- ❖ Al estar en un grupo nuevo. ¿Cómo te sientes bien?
- ❖ En tu opinión, que características son importantes para poder llevar una buena comunicación con los que te rodean.

PARTE II.- Mostrar la importancia de preparar el terreno, identificando sus habilidades al resolver juntos un problema.

CARRERA DE AUTOMÓVILES:

Paso1.- Dividirse en equipos. Se recomiendan 5 personas por equipo. El trabajo del equipo consistirá en analizar la información y llegar en equipo a una respuesta correcta.

Paso2 .- Hay 8 coches de marcas y colores diferentes, que están alineados para salir en carrera. Lo que se debe hacer es establecer el orden en que están dispuestos, basándose en la siguiente información : (Es importante llevar el control del tiempo.)

- ❖ El Ferrari está entre los coches rojo y ceniza.
- ❖ El coche ceniza está a la izquierda del Lotus.
- ❖ El McLaren es el segundo a la izquierda del Ferrari y el primero a la derecha del coche azul.
- ❖ El Tyrell no tiene ningún coche a su derecha y está a continuación del coche negro.
- ❖ El coche negro está entre el Tyrell y el coche amarillo.
- ❖ El Shadow no tiene ningún coche a su izquierda y está a la izquierda del coche verde.
- ❖ A la derecha del coche verde está el March.
- ❖ El Lotus es el segundo a la derecha del coche crema y el segundo a la izquierda del coche marrón.
- ❖ El Lola es el segundo a la izquierda del Iso.

Identifica el orden de los carros:

1.-Shadow azul 2.- McLaren verde 3.- March rojo

4.- Ferrari crema 5.- El Lola ceniza 6.- El Lotus amarillo

7.- El Iso negro 8.- El Tyrell marrón

PREGUNTAS DE CONTROL:
1. ¿Qué habilidades encontraste en tí que sean complementarias a las de tus compañeros?
Escribe tus respuestas aquí.
2.- En el equipo ¿Cómo se organizaron para realizar la tarea encomendada?
Escribe tus respuestas aquí.
3. ¿Qué sistema de comunicación interno siguieron?
Escribe tus respuestas aquí.
4. ¿Hubo necesidad de un líder formal?
Escribe tus respuestas aquí
5. ¿Se perdió tiempo organizando?
Escribe tus respuestas aquí

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- http://www.mercado.com.ar/mercado/mo/lazzati/l25-1097/htm/L25-1097.asp#IV

PRÁCTICA # 2

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **Llenado de formas básicas**

INGENIERÍA DE SOFTWARE

LLENADO DE FORMAS BÁSICAS.
(Semana Personal, Semana Equipo, REGT, REGD)

OBJETIVOS :

- Conocer las formas básicas de TSPi.
- Aprender a llevar el control del registro del tiempo.
- El control de registro de defectos.

INTRODUCCIÓN :

Una práctica importante que propone PSP y TSPi, es llevar un control personal y del equipo, de los tiempos que se ocupan en efectuar cada una de las tareas de desarrollo de software. Para eso, se proponen diversas formas que ayudan a registrar los tiempos, tamaños y defectos.

En cada una de las formas es indispensable:

- Registrar el nombre, fecha, nombre del equipo y del instructor.
- Registrar el nombre de la fase.
- Registrar el número de ciclo.

La forma de ***Registro de Tiempo*** **REGT,** se utiliza para registrar el tiempo dedicado a cada tarea del proyecto por cada persona. En esta forma:

- Se registra el tiempo dedicado al proyecto.
- Se registra el tiempo en minutos.
- Se debe ser lo más preciso posible.
- Si se necesita más espacio, se utiliza otra forma.
- Si se olvido registrar el inicio, pausas, tiempo de interrupción, se registra el tiempo estimado.

En la forma **REGT** se llenan los siguientes campos:

- FECHA.- Es el día en que se realiza la actividad.
- INICIO.- Se registra la hora en que se inicia una tarea de desarrollo, ejemplo 12:00
- FIN.- Se registra la hora en que terminó la tarea.
- TIEMPO DE INTERRUPCIÓN.- El tiempo en que interrumpido, por realizar otras actividades y el motivo, ejemplo 13:45 – descanso.
- TIEMPO TOTAL.- Es el tiempo total de la hora de inicio +hora de término - tiempo de interrupción. Se puede medir en minutos u horas.
- PRODUCTO O ACTIVIDAD.- Se registra el nombre de la actividad o producto que se generó en ella, ejemplo: plan de pruebas del sistema, código.

- TAMAÑO O COMENTARIOS.- Cuando un producto está terminado, se anota el tamaño que tuvo. Se pueden poner comentarios sobre la actividad que se está realizando, como su complejidad, grado de avance, etc. (FIGURA 4.1)

"Registro de Tiempo , REGT"

Nombre:	Fecha:
Equipo:	Ciclo:

Fecha	Inicio	Fin	Tiempo de interrupción	Tiempo total	Producto o actividad	Tamaño o Comentarios

FIGURA 4.1 REGT. [Humphrey, 2000]

La forma ***Semana Personal***, como su nombre lo indica es individual y resume los tiempos dedicados al proyecto de esa semana y se forma del resumen que cada persona hace, apuntando en su REGT de la semana. Consta de los siguientes campos:

- Tareas de desarrollo efectuadas.- Es la tarea que se está realizando.
- Horas dedicadas.- Cuánto tiempo se le dedicó a determinada tarea en esta semana.
- Tamaño del producto.- Si ya está terminado el producto se pone el tamaño que tuvo, puede ser en páginas, LOC, etc.
- Total.- Es la suma de horas y tamaños. (FIGURA 4.2)

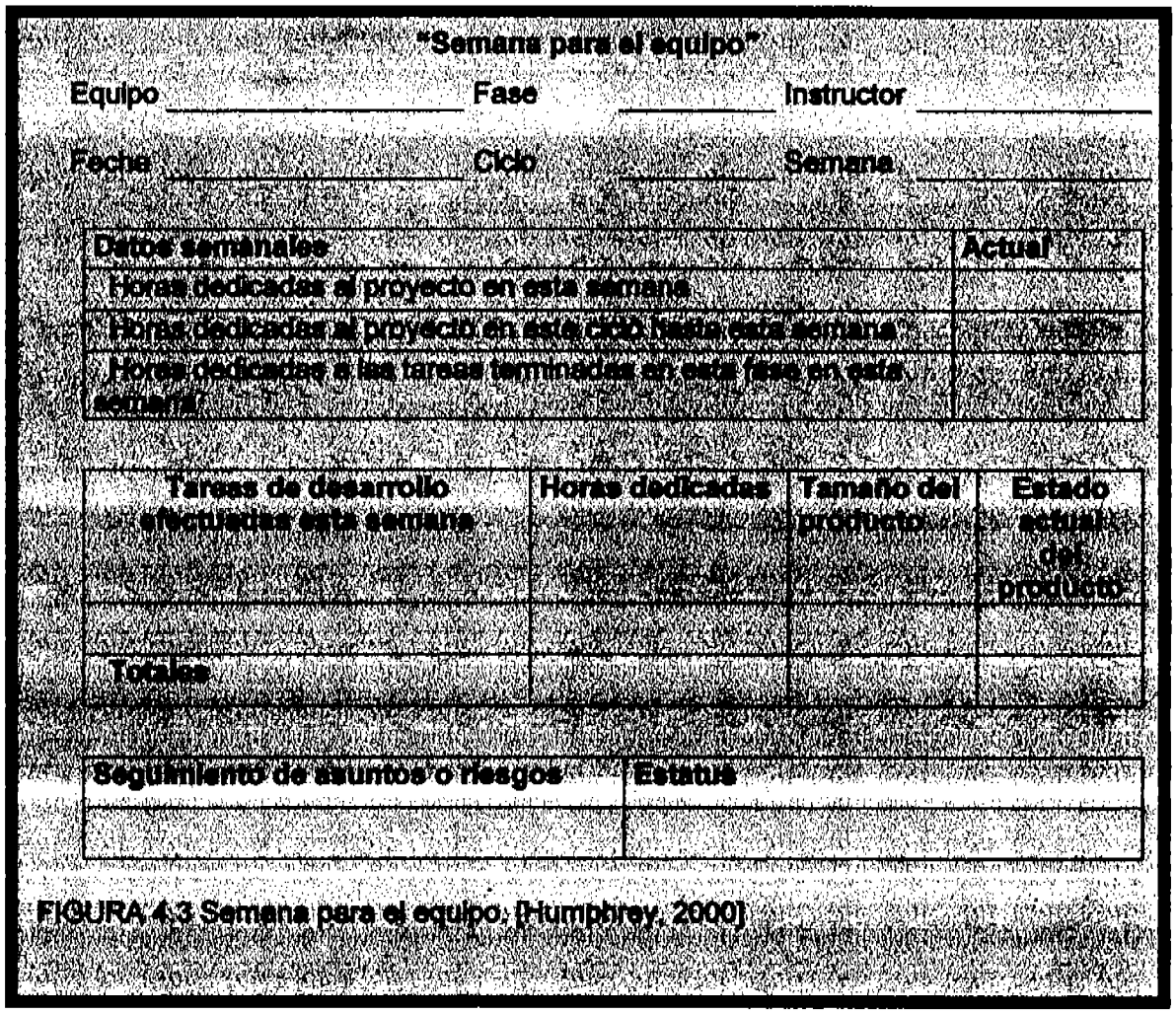
"Semana Personal"

Nombre: Equipo: Instructor:
Fecha: Ciclo: Semana #:

Tareas de desarrollo efectuadas	Horas dedicadas	Tamaño del producto	Estado de producto
Totales			

FIGURA 4.2 Semana Personal. [Humphrey, 2000]

La forma ***Semana para el equipo***, es el reporte que se prepara cada semana y que resume el trabajo de todos los integrantes del equipo. Cada integrante del equipo entrega en la forma semanal personal el trabajo realizado durante la última semana. Y en la del equipo se resume el trabajo realizado en la semana por todo el equipo. (FIGURA 4.3)



"Semana para el equipo"

Equipo _____ Fase _____ Instructor _____

Fecha _____ Ciclo _____ Semana _____

Datos semanales	Actual
Horas dedicadas al proyecto en esta semana	
Horas dedicadas al proyecto en este ciclo hasta esta semana	
Horas dedicadas a las tareas terminadas en esta fase en esta semana	

Tareas de desarrollo efectuadas esta semana	Horas dedicadas	Tamaño del producto	Estado actual del producto
Totales			

Seguimiento de asuntos o riesgos	Estatus

FIGURA 4.3 Semana para el equipo, [Humphrey, 2000]

- Primera parte

DATOS SEMANALES

- *Horas dedicadas al proyecto en esta semana.*- Cuántas horas se le dedicó al proyecto en la semana. Se suman todos los totales de los integrantes.

- *Horas dedicadas al proyecto en este ciclo hasta esta semana.*- Es la suma de las horas que lleva el equipo en este ciclo desde que se comenzó.
- *Horas dedicadas a las tareas terminadas en esta fase en esta semana.*- Cuantas horas se llevaron, de las tareas terminadas en cada fase hasta el día de hoy.

❖ Segunda parte

TAREAS DE DESARROLLO EFECTUADAS ESTA SEMANA

- *Horas dedicadas.*- cuántas horas le dedicaron a determinada tarea todos los miembros del equipo.
- *Tamaño del producto.*- Puede ser en páginas, LOC, etc.
- *Estado actual del producto.*- Si está terminado o no.
- *Total.*- Es sacar la suma de cada uno.

❖ Tercera parte

SEGUIMIENTO DE ASUNTOS O RIESGOS

- *Estatus* .- Resumir el estado de asuntos y riesgos que preocupen al equipo, como de cualquier cambio importante, acontecido en la semana.

La forma de ***Registro de Defectos*** **REGD** es útil para registrar los defectos encontrados y corregidos que se encontraron en los productos al hacer la revisión de los mismos. Los datos a poner son (FIGURA 4.4):

❖ FECHA.- Se registra la fecha en la que se encontró el defecto.
❖ NUMERO.- El número de defectos encontrados y se va enumerando sucesivamente .
❖ DESCRIPCIÓN DEL DEFECTO.- Se describe el defecto para tener información que permita recordar el error y saber porque se realizó.
❖ ESTADO DEL DEFECTO.- Indicar si el defecto está encontrado o corregido.
❖ TOTAL DE DEFECTOS.- Cuántos defectos fueron encontrados y cuántos corregidos.

"Registro de Defectos, REGD"

Nombre _____ Fecha _____

Equipo _____ Ciclo _____

Fecha	Número	Descripción del defecto	Estado del defecto

Total de defectos	Encontrados	Corregidos

FIGURA 4.4 REGD. [Humphrey, 2000]

MATERIAL :

- Las formas que se bajan del sitio del curso.

DESARROLLO:

Llenar las siguientes formas con tu equipo, del trabajo realizado en esta práctica. Las tareas son leer estas instrucciones, bajar las formas del sitio del curso y llenarlas. Cada integrante del equipo toma los tiempos en realizar estas tareas y luego se resume en lo que se tardaron todos en la forma semanal del equipo.

Para llenar la REGD, cada quien revisa las formas de otro miembro del equipo y registra en esta forma los defectos (faltas de ortografía, errores, faltas de dedo, etc.) que encuentre en las formas de su compañero.

PREGUNTAS DE CONTROL:.
1. ¿Por qué crees que se debe llevar un registro de tiempo y un registro de defectos?
Escribe tus respuestas aquí.
2. ¿Para qué se deberá resumir el tiempo que tarda el equipo en una tarea o producto?
Escribe tus respuestas aquí.
3. ¿Por qué se deberán registrar los defectos encontrados en los productos?
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Watts S. Humphrey . Introduction to the Team Software Process, 2000.

5.1 INTRODUCCIÓN

En la **fase de planeación** como su nombre lo dice, es empezar a planear todas las actividades que se van a realizar, donde los interesados participarán compartiendo sus necesidades, expectativas y así aplicar sus conocimientos, habilidades y técnicas a las actividades del proyecto. (FIGURA 5.1)

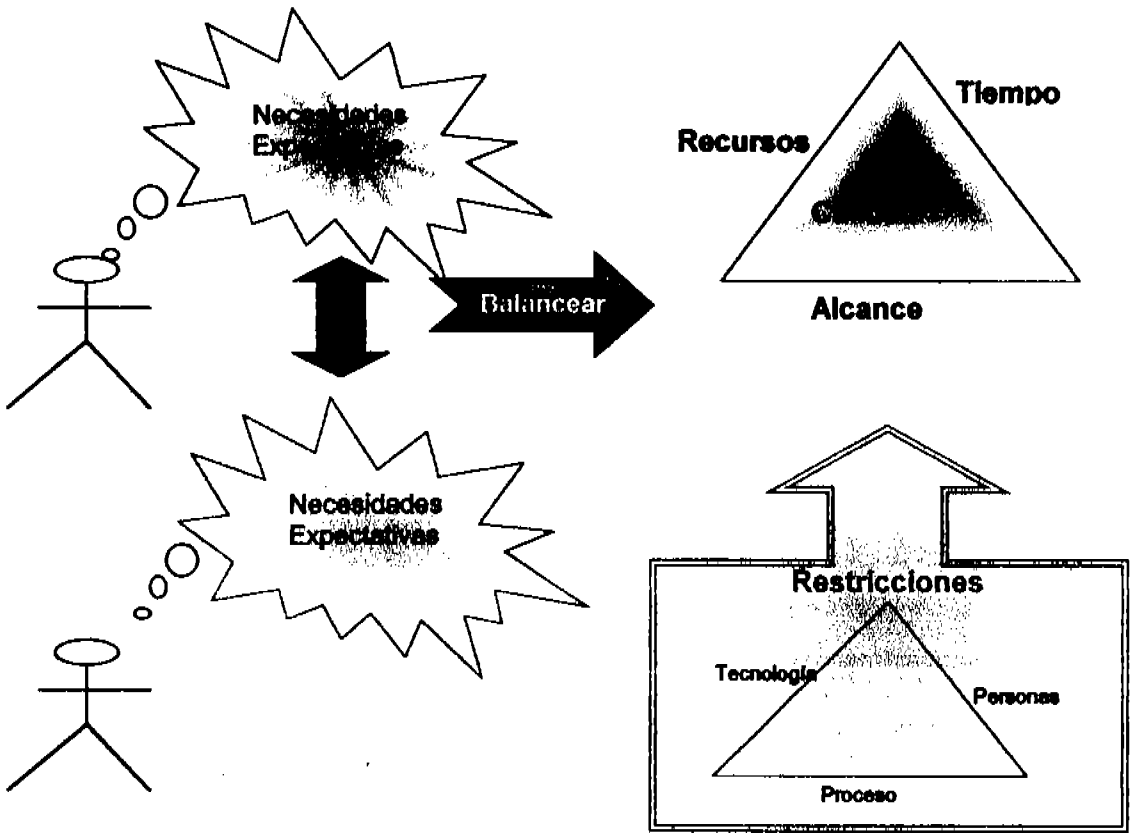


FIGURA 5.1 Planeación. [http://www.fing.edu.uy/inco/grupos/gris/]

Un plan permite mantener un seguimiento y administración sobre el trabajo necesario en el desarrollo de software. La fase de planeación se dividió en dos prácticas, en una de ellas contamos con un paquete que nos ayuda a realizar dichas actividades. **Microsoft**

Project que es de gran utilidad en la práctica 7, donde se hace una introducción a su uso. La práctica 8 es la **Planeación del Proyecto y del Plan de Calidad** donde se enfoca a cómo realizar dicho plan.

5.2 INTRODUCCIÓN A MICROSOFT PROJECT

Esta práctica es una introducción a Microsoft Project, su funcionalidad y conocer las características generales de esta herramienta, para la gestión de proyectos. Aprendiendo así el manejo básico de Microsoft Project.

Conocer cuáles son las elementos importantes de un proyecto, qué es un proyecto, cuáles son los objetivos primordiales que hay que conocer y los componentes principales. (FIGURA 5.2)

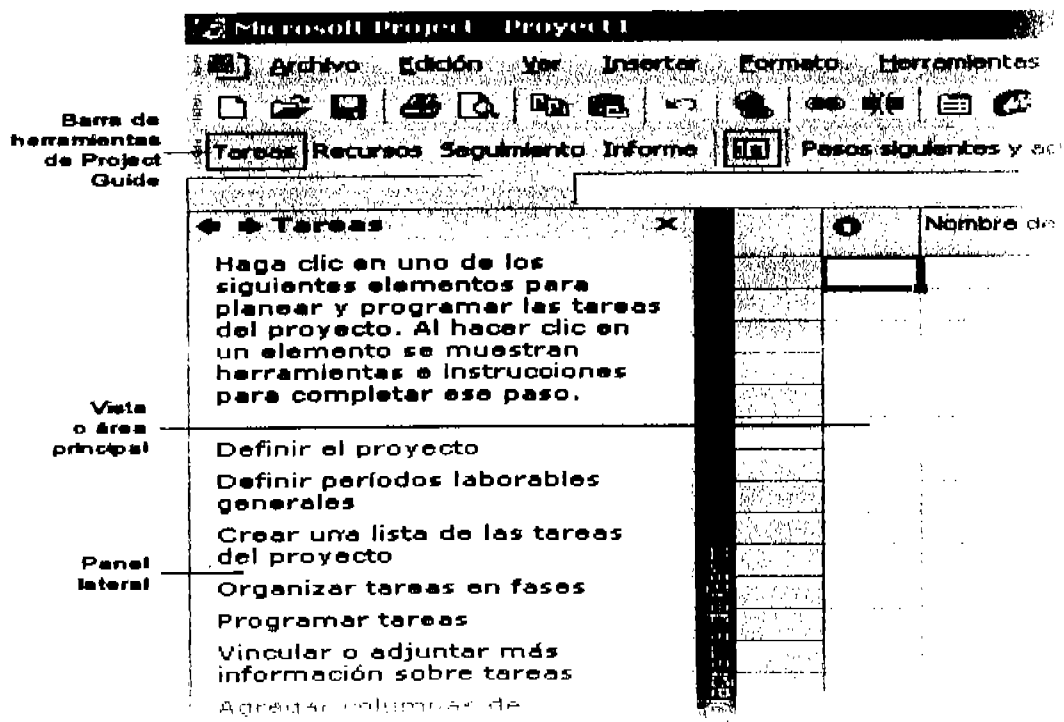


FIGURA 5.2 Vista Global de Microsoft Project.

[http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm]

5.3 PLANEACIÓN DEL PROYECTO Y DEL PLAN DE CALIDAD

En la práctica 8 se van a definir las actividades que se harán, cuándo y por quién. Para que así se pueda revisar el progreso del proyecto.

En esta actividad se detallará con mucho cuidado el plan de calidad, que servirá para satisfacer los requerimientos, así cómo también las necesidades de cada usuario.

Este proceso de planeación es el contexto mediante el cual se diseña la manera en la que se hará un trabajo. Al planear se entablan compromisos donde se establecen fechas y se detecten los recursos necesarios. Una de las causas principales que originan problemas de horario es la carga de trabajo no balanceada, sobre todo cuando se asigna mayor trabajo a algunos que a otros denominado *Un plan balanceado*, en donde los integrantes del equipo terminan las tareas planeadas en el orden apropiado y casi al mismo tiempo.

Este plan le permitirá mantener un seguimiento y administración sobre el trabajo necesario en el desarrollo de software. Donde cada integrante debe estimar 10 o menos horas a las tareas asignadas, habrá tareas como requerimientos y diseño de alto nivel que necesitarán más de 10 horas porque serán desempeñadas por los integrantes.

PRÁCTICA # 7

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **PLANeación**

INGENIERÍA DE SOFTWARE

INTRODUCCIÓN A MICROSOFT PROJECT.

OBJETIVOS :

- Conocer las características generales de una herramienta de software, para la gestión de proyectos.
- Aprender el manejo básico de Microsoft Project.

INTRODUCCIÓN :

¿Qué es un proyecto?

Es una secuencia bien definida de eventos con un principio y un final identificados, que se centra en alcanzar un objetivo claro.

1. La Administración de Proyectos es el proceso de planificar, organizar y administrar tareas y recursos para alcanzar un objetivo concreto, generalmente con delimitaciones de tiempo, recursos o costo.
2. Plan de Proyecto: Es una lista de tareas, sus fechas de comienzo y fin, y las relaciones entre ellas.

Elementos de un Proyecto:

Tarea: Son acciones que se requieren para completar un proyecto.
Indicador o Hito: Representa un evento o condición que marca la finalización de un grupo de tareas relacionadas o la finalización de una fase del proyecto.
Recurso: Es cualquier persona o elemento que sea necesario para realizar una tarea.
Producto: Resultado de efectuar una tarea.

Microsoft Project es una herramienta diseñada para organizar proyectos de forma fácil y rápida. Project puede ayudarle a planear y dar seguimiento a los tiempos dedicados a los proyectos.

Un **proyecto** tiene de **4 fases:**

1.- Definir el objetivo del proyecto

Primero debemos identificar el objetivo. El objetivo debe ser medible, debe definir un fin concreto del proyecto.

2.- Generar el plan del proyecto

Una vez que teniendo el objetivo del proyecto, se debe averiguar la mejor manera de llegar a ese punto. Para ello, tendrá que recopilar información del proyecto y construir la lista de las tareas que han de realizarse y las estimaciones del tiempo que requerirá cada una. Así como los productos a obtener en cada tarea y los recursos necesarios.

El plan del proyecto constituye la guía del proyecto. Es un modelo que indica las tareas que se van a llevar a cabo, quién las va a realizar y cuándo. Probablemente, la parte más importante del plan es la programación del proyecto, que incluye las fechas de comienzo y de fin de cada tarea, el tiempo que requerirá cada una de ellas, la duración y la fecha de fin de todo el proyecto. El plan del proyecto también puede incluir información acerca de los costos del uso de los recursos.

3.-Administrar y realizar un seguimiento del proyecto

Una vez que se inicia el proyecto, el equipo de colaboradores se encarga de ejecutar el plan. Sin embargo, tendrá que seguir de cerca su progreso, puesto que generalmente surgen problemas imprevistos. Al utilizar Microsoft Project para realizar un seguimiento del progreso del proyecto, se podrá ver en todo momento su estado e identificar y resolver lo antes posible los problemas que puedan afectar su correcto desarrollo.

4.- Cerrar el proyecto

Cada proyecto es una experiencia de la que siempre se aprende. Por muy bien que se planee el comienzo, al final del proyecto comprobará que su plan ha cambiado con respecto a la versión original. Si guarda el plan original en Microsoft Project, puede aprovechar mejor su experiencia comparando la información del proyecto original con la forma en que progresó realmente el proyecto.

Los tres **componentes** principales de Microsoft Project son los siguientes:

- **La barra de herramientas** de **Project** es una barra de comandos para el desplazamiento de nivel superior que contiene botones para las principales áreas de Project: **Tareas**, **Recursos**, **Seguimiento** e **Informe**. Estas áreas representan las fases fundamentales en la administración de un proyecto de las tareas de planeación, administración de recursos, seguimiento del proyecto durante su realización y creación de informes y análisis de los datos del proyecto.
- **Panel lateral** muestra las tareas de cada área de Project y proporciona instrucciones sobre cómo llevar a cabo las actividades. Al hacer clic en un área en la barra de herramientas de **Project**, el panel lateral muestra la lista de tareas para esta área. Por ejemplo, si un usuario hace clic en **Tareas** en la barra de herramientas predeterminada de **Project**, el panel lateral mostrará una lista de las

actividades que se deben completar como parte de la planificación de las tareas de un proyecto como, por ejemplo, elaboración de la lista de tareas, organización de éstas en fases y programación de las mismas.

- **Área de vista principal** permite ver y editar la información de los proyectos y la zona donde los usuarios realizan su trabajo. Las vistas y páginas mostradas dependen de la tarea que el usuario esté realizando en el panel lateral. Este espacio muestra las vistas de los datos del proyecto. Así mismo, se puede utilizar para mostrar las páginas Web de Microsoft Project Server. La vista o página mostrada cambia dependiendo de la actividad que el usuario esté siguiendo en Project. Por ejemplo, si el usuario está introduciendo tareas, se mostrará la opción

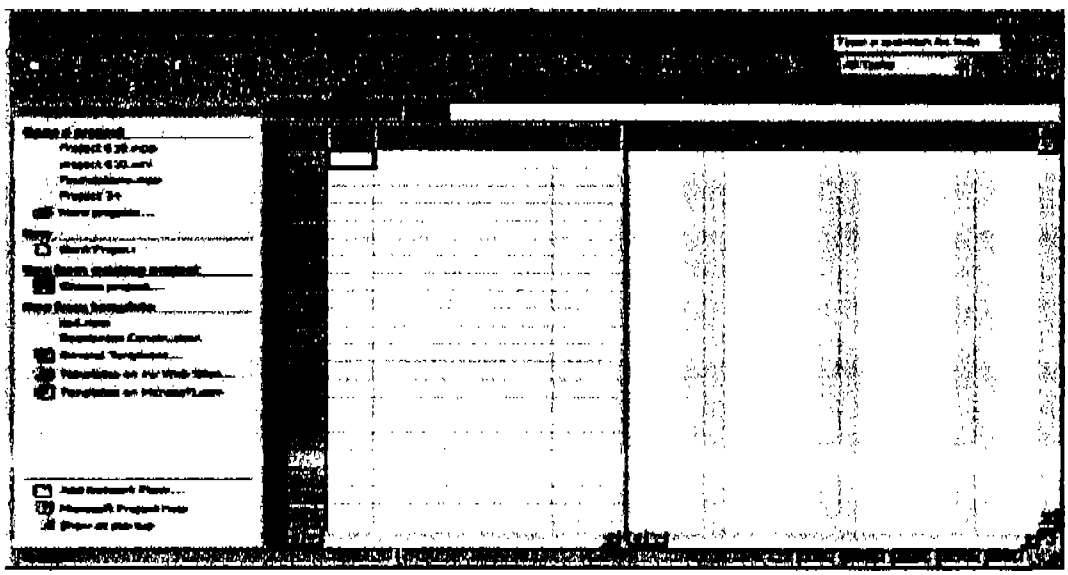


Diagrama de Gantt a la derecha. (FIGURA 5.3)

FIGURA 5.3 Diagrama de Gantt.

[http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm]

MATERIAL :

- Microsoft Project.

DESARROLLO:

PARTE I.- Crear un nuevo proyecto.

El primer paso: para crear un proyecto.

- Seleccionar en la barra de tareas de Windows el botón de:

  Inicio /Programas/MicrosoftProject (FIGURA 5.4)

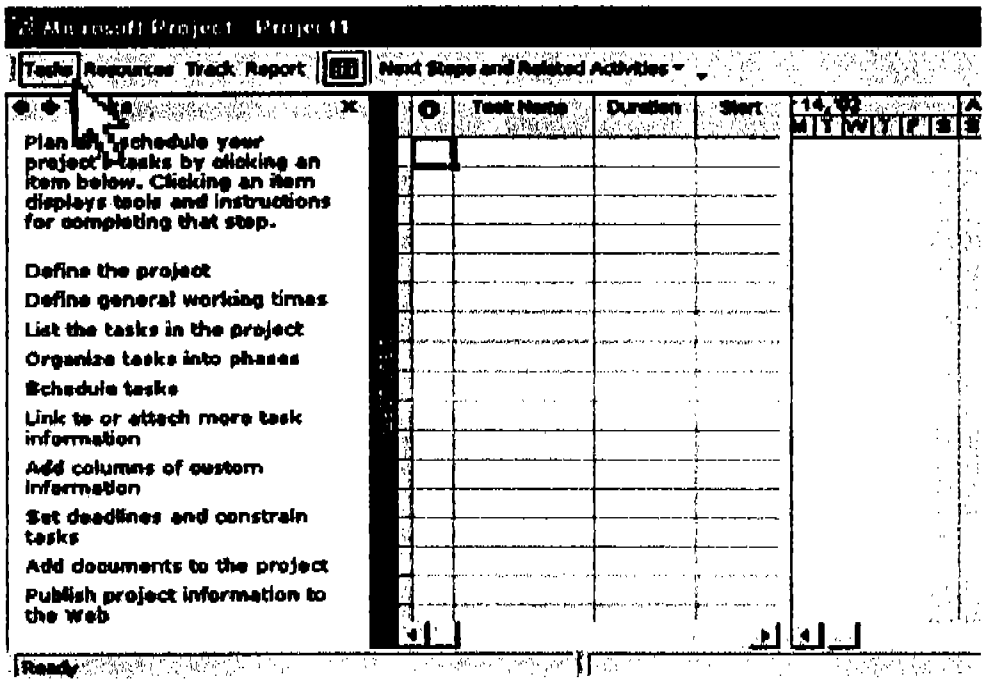


FIGURA 5.4 Microsoft Project.

[http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm]

Sugerencia Si los planes del proyecto cambian, puede modificar la información en cualquier momento haciendo clic en información del proyecto en el menú Proyecto.

El segundo paso: Información clave del proyecto.

- Escribir el título del proyecto, el nombre del administrador y las notas importantes.

Cada proyecto se compone de un conjunto único de elementos: las tareas, las personas que las realizan y el objetivo del proyecto que se espera alcanzar. Como ayuda para

recordar y comunicar detalles importantes, puede escribir información acerca del proyecto y consultarla o imprimirla cuando la necesite.

- ❖ En el menú Archivo, haga clic en Propiedades.
- ❖ En la ficha Resumen, escriba información acerca del proyecto, por ejemplo, las personas que administrarán el proyecto y que se encargarán del mantenimiento del archivo de proyecto. El indicado para hacer esto es el Administrador de Planeación (AP).

Sugerencia Si lo desea, puede imprimir esta información.

Tercer paso: Establecer la programación del trabajo.

- ➢ Puede cambiar los días y horas laborables del calendario del proyecto para reflejar la programación de trabajo de todas las personas que trabajan en el proyecto.

PARTE II.- Introducir y organizar una lista de tareas.

Considera las tareas definidas en el guión del curso, y establece el plan para estas tareas según las semanas planeadas.

Primer paso: Especificar una Tarea Normal.

- ➢ En el menú **Ver**, seleccionar Diagrama de Gantt.
- ➢ En el campo *Nombre de Tarea*, escribir el nombre de una tarea y, a continuación, presionar la tecla TAB.
- ➢ Microsoft Project introduce una duración estimada de un día para la tarea, seguida de un signo de interrogación.
- ➢ En el campo **Duración**, escribir la cantidad de tiempo que llevará cada tarea en meses ,semanas o días, sin contar los períodos no laborables. Se pueden usar las abreviaturas siguientes:

  meses =me, semanas =s, días = d.
- ➢ En el campo **Comienzo**, introduce la fecha de inicio de la tarea.
- ➢ En el campo **FIN,** el día que la tarea finaliza. Presionar la tecla ENTRAR.

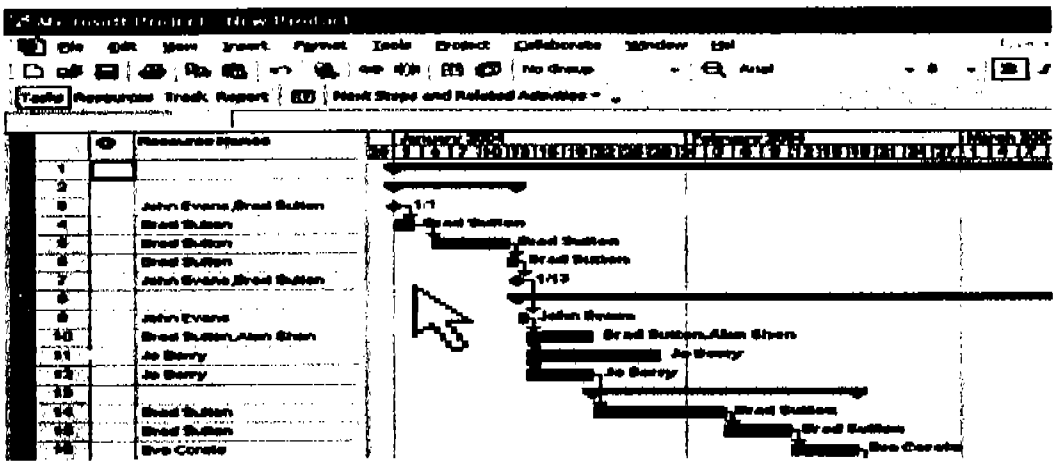


FIGURA 5.5 Especificar tarea.

[http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm]

PREGUNTAS DE CONTROL:
1.-¿Cuál es el objetivo del proyecto en el curso de IS?
Escribe aquí tus respuestas
2.-¿ Para ese proyecto qué es una tarea, un hito o indicador y un recurso?
Escribe aquí tus respuestas
3.- ¿Cuáles son las partes del plan de proyecto?
Escribe aquí tus respuestas

CONCLUSIONES: Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe aquí tus respuestas

BIBLIOGRAFÍA :

- Microsoft Press. Microsoft Project 1998 paso a paso. McGraw-Hill Interamericana, 1998.
- http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm

PRÁCTICA # 8

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **PLANeación**

INGENIERÍA DE SOFTWARE

PLANEACIÓN DEL PROYECTO Y DEL PLAN DE CALIDAD.

OBJETIVOS :

- Realizar el plan de desarrollo del proyecto.
- Definir el plan de calidad.
- Aplicar el criterio de estándares de calidad, para generar un producto de alta calidad.

INTRODUCCIÓN :

Planear es definir las actividades que se harán, cuándo y por quién. Planear permite revisar el progreso de un proyecto.

Planear sirve para:

- que cada quien sepa que debe hacer y ser más eficiente en su trabajo.
- permite considerar tareas que se podrían olvidar.
- cumplir con los compromisos.

Los planes deben ser para todo el equipo y de ellos se extraen los planes personales. Los planes personales de los miembros del equipo deben estar balanceados en las cargas de trabajo. Los planes deben ser detallados.

TSPi requiere que cada integrante estime el tiempo para las tareas asignadas, aunque habrá tareas desempeñadas por varios integrantes, cada quien estima el tiempo que le dedicará.

Calidad

Calidad de un producto de software es el grado en que éste satisface los requerimientos especificados así como las necesidades y expectativas del usuario.

Para lograr la calidad necesitamos planear las actividades de calidad como son la verificación y validación de los productos de software durante todo el proceso de desarrollo.

Verificación es la actividad para revisar la satisfacción de los requerimientos especificados.

Validación es la actividad para revisar la satisfacción de las necesidades y las expectativas del usuario.

Una de las técnicas de verificación son las Inspecciones (o Revisiones entre Colegas), se establece un plan en que cada miembro revisa los productos del equipo en una semana. Se establece desde esta fase, cuándo le tocará a cada quien hacer la revisión de los productos de esta semana.

MATERIAL :

- Microsoft Project.

DESARROLLO:

PARTE I .- Planear las actividades.

A) Hacer una lista de las actividades a realizar:

- Identificar las tareas necesarias para cada producto y repartirlas entre los miembros del equipo. Las tareas de cada participante deberán estar equilibradas.

B) Con un Diagrama de Gantt efectuar la planeación para cada semana por persona y del equipo con auxilio de Microsoft Project.

- Construir el diagrama de Gantt para definir que tareas se harán en cada semana.
- Construir los diagramas de Gantt de cada participante. Si alguno tiene mucho trabajo en una semana, reasignar las tareas. Si otro no tiene mucho trabajo, repartir de nuevo las tareas.
- Estime horas de trabajo para las tareas del equipo y de cada uno.
- Incorporar actividades de inspección por cada producto.

C) Formas Semana personal y del equipo.

- Cada semana el administrador de planeación, resume en la forma Semana del equipo, las actividades realizadas y las compara con la planeación que se hizo en el diagrama de Gantt. Si hay retrasos, les comunica al equipo para tomar las

acciones correctivas. Si están en tiempo, se continúa así. Si hay ganancia de tiempo se comunica al equipo para disponer de él según se ofrezca.

Para que el AP pueda conjuntar lo realizado cada semana, cada participante del equipo deberá llenar su forma SEMANA personal para informarle las actividades realizadas.

PARTE II.- Generar el Plan de calidad y definir las inspecciones.
En el plan de calidad se establece para cada semana, quien participará en la inspección de los productos de esa semana. Todos deben participar al menos una semana en esta tarea.

En las reuniones de inspección el Administrador de Calidad/Proceso:

1. Planear fecha y horario de la inspección y seleccionar participantes.
   Recibir producto a revisar.
   Preparar la lista de verificación.
   Seleccionar los participantes en la inspección.
   Acordar fecha y hora de la inspección.
2. Dirigir la reunión de la inspección.
   Revisar que todos hicieron revisiones individuales.
   Discutir los defectos encontrados.
   Registrar defectos en la forma REGD.
3. Realizar la revisión final del producto corregido.

Equipo de revisión:

1. Revisar de forma individual el producto apoyándose en la lista de verificación.
2. Registrar los defectos en la forma REGD.
3. Asistir a la reunión de Inspección para discutir los defectos.

Propietario del producto:

1. Discutir y analizar los defectos presentados en la reunión de Inspección.
2. Corregir defectos encontrados.
3. Entregar el producto corregido al Adm. de Calidad.
4. Hacer este plan de calidad.

PREGUNTAS DE CONTROL:
1.- ¿Qué se representa en los diagramas de Gantt?
Escribe aquí tus respuestas
2.-¿Cómo se balancean las tareas de los miembros del equipo?
Escribe aquí tus respuestas
3.-¿Qué es un defecto en un producto?
Escribe aquí tus respuestas
4.-¿Por qué es importante hacer revisiones entre colegas?
Escribe aquí tus respuestas

CONCLUSIONES: Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe aquí tus respuestas

BIBLIOGRAFÍA :

- Watts S. Humphrey. Introduction to the Team Software Process. Cap5, 2000.
- http://www.fing.edu.uy/inco/grupos/gris/

CAPÍTULO VI FASE DISEÑO (Práctica #13, Práctica #14)

6.1 INTRODUCCIÓN

Este capítulo se enfoca en las prácticas para hacer el diseño. En TSPi, la fase de diseño se enfoca a la estructura del sistema.

En equipo se puede trabajar más rápido dividiendo el producto en partes y que cada uno de los integrantes diseñen e implementen una o más de esas partes.

Un diseño arquitectónico define las partes principales de un producto, describe cómo esas partes interactúan y especifica cómo unirlas para producir el resultado final. El tener distintos diseños permite cumplir con los requerimientos, pero cada uno ofrece ventajas. (FIGURA 6.1)

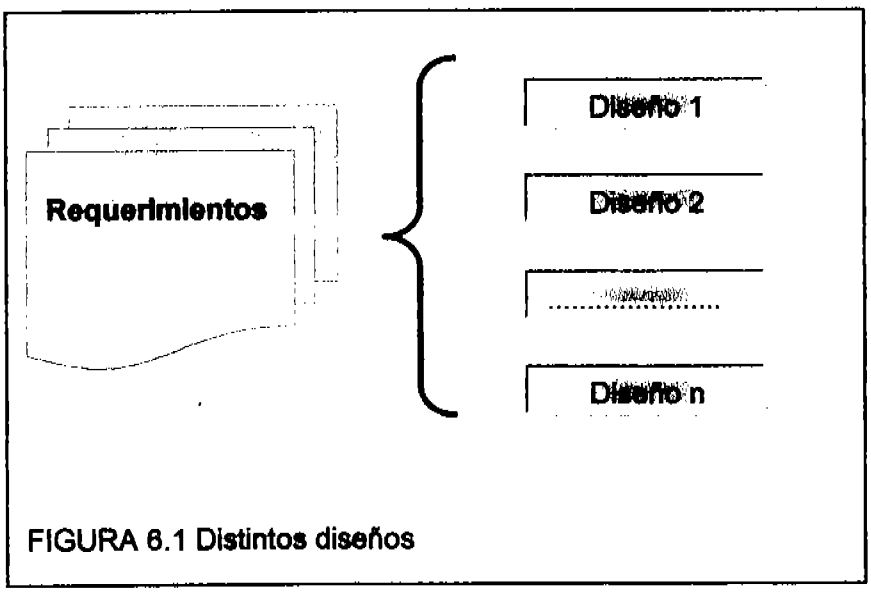


FIGURA 6.1 Distintos diseños

FIGURA 6.1 Diseño. [Humphrey, 2000]

6.2 DISEÑO DE LA ARQUITECTURA DEL SISTEMA Y PLAN PARA LA PRUEBA DE INTEGRACIÓN

Esta fase se enfoca a la estructura o arquitectura del sistema, donde se produce el documento de Especificación de Diseño de Software (**EDS**), el cual documenta el diseño de alto nivel (**DAN**). En esta fase se decide como se va a construir el producto, describiendo como interactúan las partes y especificando como unirlas para producir el producto final. Es importante producir el Plan de Pruebas de Integración donde se realizarán las especificaciones de diseño. Ya que con ellas se van revisando y verificando las interfaces que hay entre los componentes del sistema.

6.3 DIAGRAMAS DE CLASES Y DE SECUENCIA

El objetivo primordial de esta práctica, es conocer el lenguaje modelado, sabiendo diferenciar entre un diagrama de clases y un diagrama de secuencia.

El **Diagrama de Clase** es el diagrama principal de diseño y análisis para un sistema. En él, la estructura de clases del sistema que son los componentes básicos del desarrollo orientado a objetos, se especifican, con relaciones entre clases y estructuras de herencia. Durante el análisis del sistema, el diagrama se desarrolla buscando una solución ideal. Durante el diseño, se usa el mismo diagrama y se modifica para satisfacer los detalles de las implementaciones.

Un **diagrama de secuencia** muestra las interacciones entre objetos ordenadas en secuencia temporal. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. En aplicaciones grandes, además de los objetos, se muestran también los componentes de los casos de uso. El mostrar los componentes tiene sentido ya que se trata de objetos reutilizables.

PRÁCTICA # 13

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: DISeño

INGENIERÍA DE SOFTWARE

DISEÑO DE LA ARQUITECTURA DEL SISTEMA Y PLAN PARA LA PRUEBA DE INTEGRACIÓN.

OBJETIVO :

- Proporcionar un diseño de alto nivel y de alta calidad que se utilice como base para la fase de Implementación.

INTRODUCCIÓN :

Esta fase se enfoca en los principios de diseño y su proceso. En TSPi, la fase de diseño se enfoca a la estructura completa del sistema. Aquí se produce el documento Especificación de Diseño de Software (**EDS**), el cual documenta el diseño de alto nivel (**DAN**). En la fase de implementación se hace referencia al diseño detallado.

- **Principios de diseño**

El diseño es el proceso creativo mediante el cual se decide como construir un producto. El diseño de alto nivel define las partes principales del producto, describe como interactúan y especifica como unirlas para producir el producto final.

El diseño de alto nivel **DAN** tiene que producir una especificación que varios ingenieros puedan usar para diseñar cada parte independiente de los demás.

El objetivo principal es que el código fuente realice correctamente todas las funciones especificadas, que utilice adecuadamente todas las facilidades del sistema, que incorpore el reuso disponible y que siga los estándares de codificación. El producto final implementado tiene que ser un programa fuente que se compile y ejecute sin errores o problemas.

- **Arquitectura del sistema**

La "Arquitectura" es equivalente a "diseño de alto nivel". Un marco de trabajo es una colección de clases que se aplica a una familia de aplicaciones.

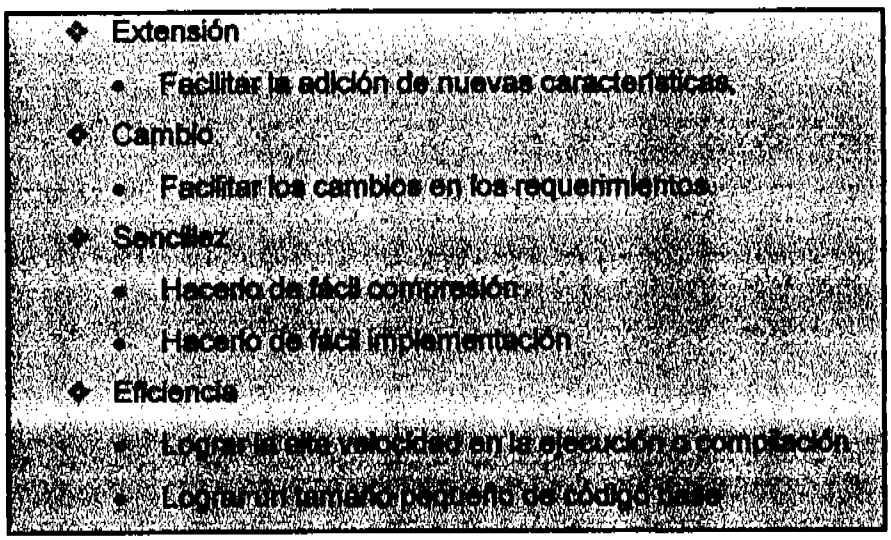
Arquitectura del sistema = diseño de alto nivel

La especificación de la arquitectura del sistema es indispensable en trabajos de desarrollo con más de una persona. Esto se debe a que las aplicaciones grandes deben diseñarse e implementarse en partes (módulos) y después ensamblarse.

Metas de selección de la arquitectura

Para un proyecto de desarrollo de software puede existir varias arquitecturas adecuadas a elegir, decidir cual es la más adecuada.

Metas de diseño más importantes:



Es necesario describir las aplicaciones desde varias perspectivas, esas perspectivas reciben el nombre de vistas. La arquitectura del sistema se expresa con un modelo de cómo debe funcionar la aplicación. Hay varios modelos. Uno muy popular es el de capas.

Una capa de una arquitectura, es una colección de artefactos de software con un objetivo específico, por ejemplo la capa de interfaz, la de la seguridad, etc. En su forma más común, una capa usa cuando mucho otra capa y es usada por otra capa. Construir aplicaciones capa por capa puede simplificar mucho el proceso. Una forma común de capas es la arquitectura cliente-servidor.

La capa cliente se apoya de la capa del servidor para obtener los servicios que requiere. En general el cliente reside en la computadora del usuario y el servidor en una computadora más grande centralizada. Con frecuencia el servidor se refiere a una base de datos. La arquitectura por capas tiene grandes beneficios para el reuso, por la

independencia entre las capas y facilita los cambios que se pueden localizar, sabiendo a que capa corresponde y no afectar a las otras.

Para describir el diseño es importante ponerse de acuerdo en el equipo, en como describirlo, para eso se debe definir varios estándares

❖ **Estándares de diseño**

Existen varios estándares de diseño como son:

- Convenciones en nombramiento

Se especifica la estructura de nombramiento. El AA debe establecer un glosario del sistema. Se deben definir los nombres de forma que muestre la jerarquía a la que pertenece el sistema, producto, componente, módulo u objeto; las convenciones usadas para nombrar los programas, archivos, variables, parámetros y los procedimientos para establecer, controlar y cambiar nombres. Esto ayuda a el equipo a saber como se llamaran los componentes que desarrollen sus compañeros.

- Formatos de interfaz

Aquí se definen los formatos y el contenido de las interfaces de los componentes, indicando como definir los parámetros para variables, código de error u otras condiciones. Cuando las interfaces están especificadas consistentemente se tendrán menos errores y se localizaran rápidamente durante las revisiones e inspecciones.

- Mensajes del sistema de error

Es conveniente establecer formatos estándares y procedimientos para los mensajes de error y del sistema. Un sistema útil debe tener consistencia y los mensajes deben ser comprendidos fácilmente.

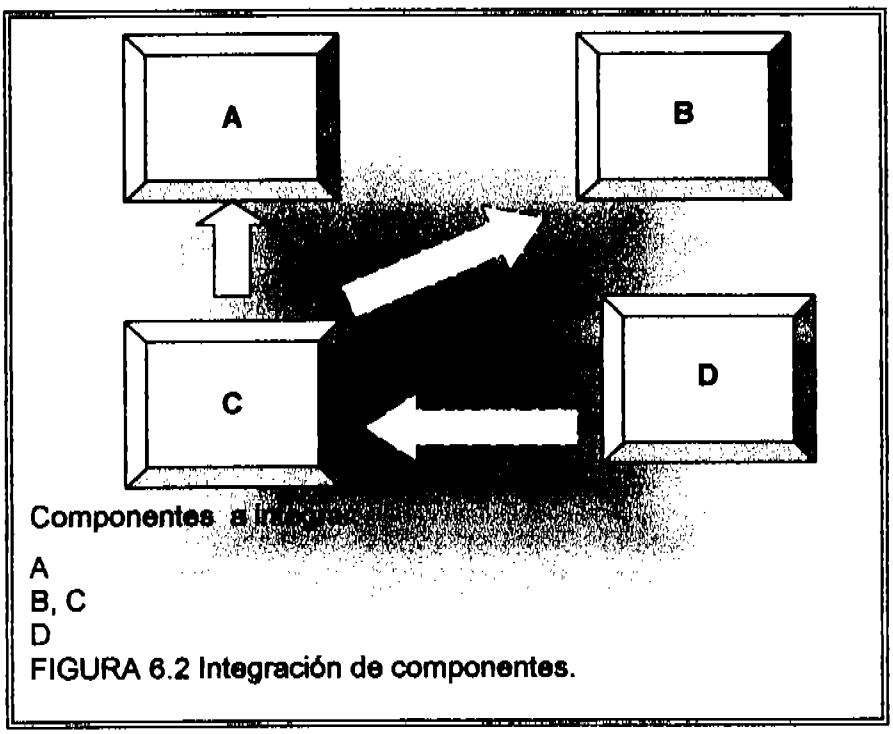
❖ **Plan de pruebas de integración (PPI)**

Una vez que se construyeron las capas, se deben juntar, a eso se le llama integrar el sistema. En el diseño se planea como se hará la integración.

En este documento al realizar las especificaciones de diseño. Con las pruebas de integración se revisan y verifican todos las interfaces entre los componentes del sistema. Para asegurar que a todas las interfaces se les han aplicado las pruebas es conveniente inspeccionar el Plan de Pruebas de Integración junto con la Especificación del Diseño de Software (EDS).

Se hace una tabla que indique que componente depende de cuáles y cómo se integrarán (FIGURA 6.2).

Ejemplo:



Componentes a integrar

A
B, C
D

FIGURA 6.2 Integración de componentes.

MATERIAL :

- Ninguno

DESARROLLO:

- ❖ Desarrolle la arquitectura del sistema.

  - Estructura del sistema, que tipo de arquitectura aplicará (proponer varias capas y dar las razones para la selección definitiva).
  - Construir el estándar de los componentes del diseño.

- Ambiente de implementación: Describir el ambiente en que se implementará la arquitectura.
- Revisar la especificación de Diseño de Software terminada. Inspeccionar y decir en donde se define que capa tendrá el sistema, quien hará cada capa, que contendrá cada capa, corregir el diseño incluyendo todos los materiales de diseño necesarios.
- Construir la tabla del plan de integración.

PREGUNTAS DE CONTROL:.
1.¿Cómo se relaciona la "arquitectura " con el "diseño"?
Escribe tus respuestas aquí.
2.¿Para qué sirven los estándares de diseño?
Escribe tus respuestas aquí.
3.¿Qué es el plan de integración?
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Watts S. Humphrey. Introduction to the Team Software Process. Cap7, 2000.
- Eric J Braude. INGENIERÍA DE SOFTWARE "Una perspectiva orientada a objetos", 2003.

PRÁCTICA # 14

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: DISeño

INGENIERÍA DE SOFTWARE

DIAGRAMAS DE CLASES Y DE SECUENCIA.

OBJETIVO :

- Proporcionar una base formal para entender el lenguaje de modelado.

INTRODUCCIÓN :

DIAGRAMA DE CLASES

Un diagrama de clases modela la estructura estática del sistema, muestra el conjunto de clases importantes que forman parte de un sistema, junto con las relaciones existentes entre ellas. Muestra de una manera estática la estructura de la información del sistema y la visibilidad que tiene cada una de las clases, dada por sus relaciones con los demás en el modelo. Los diagramas de clase son el pilar básico del modelado orientado a objetos para mostrar como el sistema puede ser construido.

Las clases tienen un nombre y describen sus propiedades o atributos y el comportamiento a través de sus métodos (operaciones) (FIGURA 6.3). Una **clase** representa un conjunto de entidades que tienen propiedades comunes. Una clase es un constructor que define la estructura y comportamiento de una colección de objetos. Un objeto es una instancia de la clase. En UML la clase está representada por un rectángulo con tres divisiones internas, son los elementos fundamentales del diagrama.

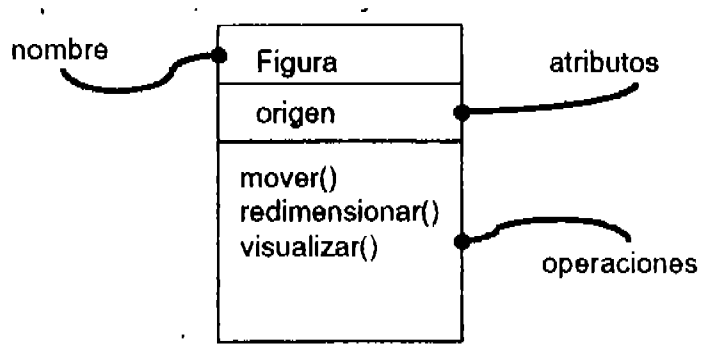


FIGURA 6.3 Diagrama de clase. [http://www-306.ibm.com/software/rational/uml/]

Atributo: Representa una propiedad de una entidad. Cada atributo de un objeto tiene un valor que pertenece a un dominio de valores determinados.

Visibilidad <nombre>: tipo = valor inicial { propiedades} Donde visibilidad puede ser # protegido o - privado. , se define su tipo y se le puede asignar un valor inicial.

Operación o método: El conjunto de operaciones describe el comportamiento de los objetos de una clase. La sintaxis de una operación en UML es:

Visibilidad nombre (lista de parámetros): tipo que retorna { propiedades}

Objeto: Es una instancia de una clase. Se caracteriza por tener una identidad o nombre único, un estado definido por el conjunto de valores para los atributos y un comportamiento representado por sus operaciones.

Asociación (rol, multiplicidad, calificador): representan las relaciones entre instancias de clase. Una asociación es una línea que une dos clases.

Rol: Es el nombre de la asociación entre clases, describe la semántica de la relación en el sentido indicado. Cada asociación tiene dos roles; un rol para cada dirección en la asociación.

Multiplicidad: Describe la cardinalidad de la relación, es decir, cuantos objetos de esa clase pueden participar en la relación.

Herencia: Es la relación de generalización / especialización, donde hay una clase que tiene las propiedades mas generales que comparten sus subclases que especializan esas propiedades. Se representa por una flecha hacia la superclase.

Agregación: Esta relación representa una asociación en donde una clase está compuesta por otras o que una clase es parte de otra. Esta relación se representa por un rombito en la clase que contiene a las otras clases. Ver el ejemplo en la (FIGURA 6.4)

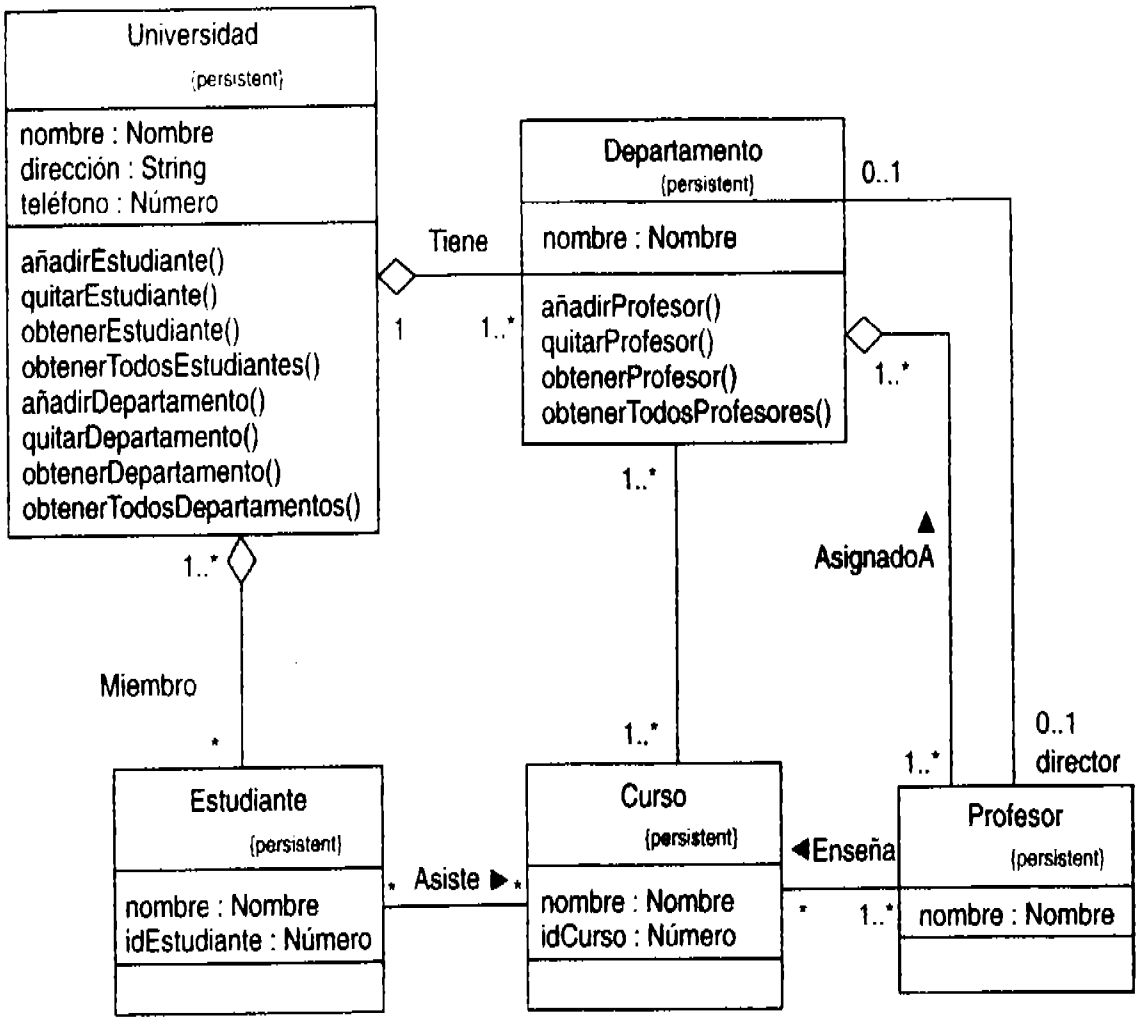


FIGURA 6.4 Ejemplo de un diagrama de clases.
[http://www-306.ibm.com/software/rational/uml/]

DIAGRAMAS DE SECUENCIA

Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en el tiempo. Muestra los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el caso de uso.

Para mostrar la interacción con el usuario o con otro sistema se introducen en los diagramas de secuencia las **clase de interfaz**. En las primeras fases de diseño el propósito de introducir estas clases, es capturar y documentar los requisitos de interfaz, no mostrar como se va a implementar dicha interfaz.

Los diagramas de secuencia, muestran la interacción entre objetos. Documentan el diseño desde el punto de vista de los casos de uso.

A la hora de documentar un diagrama de secuencia resulta importante mantener los enlaces entre los mensajes y los métodos apropiados de las clases. (FIGURA 6.5):

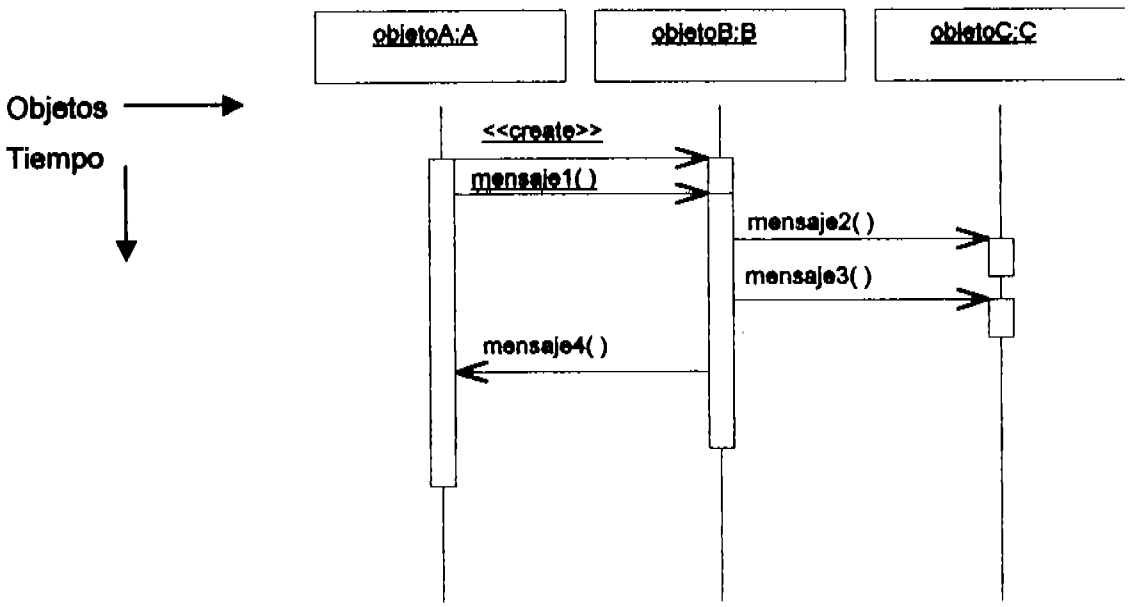


FIGURA 6.5 Construcción de un diagrama de secuencia.
[http://www-306.ibm.com/software/rational/uml/]

Los conceptos más importantes relacionados con los diagramas de secuencia son:

- **Línea de vida de un objeto** (*lifeline*): La línea de vida de un objeto representa la vida del objeto durante la interacción. En un diagrama de secuencia un objeto se representa como una línea vertical punteada con un rectángulo de encabezado y con rectángulos delgados a través de la línea principal que denotan la ejecución

de métodos (activación). El rectángulo de encabezado contiene el nombre del objeto y su clase, en un formato *nombreObjeto : nombreClase.* Cada objeto representa una columna distinta.

- **Mensaje**: El envío de mensajes entre objetos se denota mediante una línea sólida dirigida, desde el objeto que emite el mensaje hacia el objeto que lo ejecuta. El orden relativo de los objetos no tiene significado aún cuando resulta útil organizarlos de modo que se minimice la distancia de las flechas.
- **Activación**: Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación, bien sea por sí mismo o por medio de delegación a alguno de sus atributos. Se denota como un rectángulo delgado sobre la línea de vida del objeto
- Se pone una X grande en el punto en que deja de existir el objeto o en el punto en que el objeto se destruye a sí mismo.

Los diagramas de secuencia incluyen secuencias temporales, pero no incluyen las relaciones entre objetos. El objetivo de los diagramas de secuencia es mostrar el comportamiento de un caso de uso. Se construye uno para el comportamiento normal del caso de uso y uno para los comportamientos excepcionales.

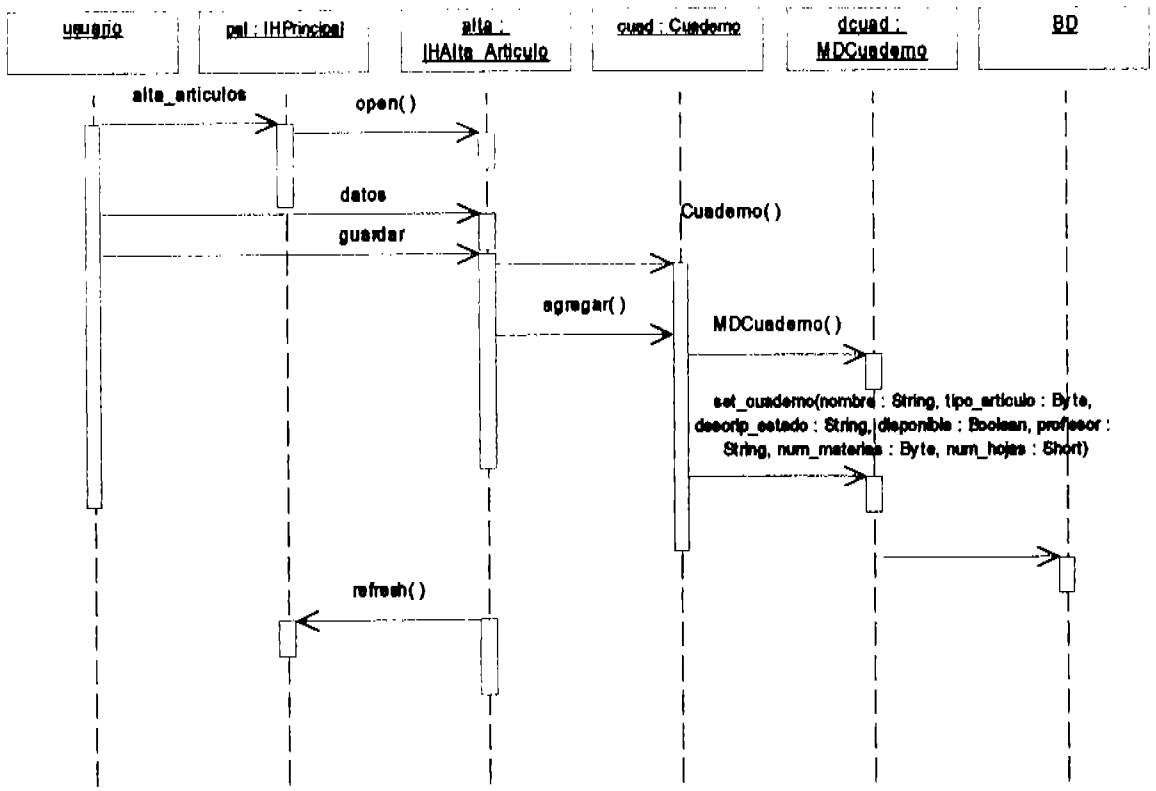


FIGURA 6.6 Ejemplo de un diagrama de secuencia.

[http://www-306.ibm.com/software/rational/uml/]

MATERIAL :

- Visio o un diagramador de UML.

DESARROLLO:

Para un sistema de inscripción a un video centro:

1.- Identificar las clases participantes (al menos 3).

2.- Construir el diagrama de clases con las relaciones , roles y multiplicidades entre las clases.

3. Desarrolla un diagrama de secuencia para sacar el caso de uso de una película del video centro.

PREGUNTAS DE CONTROL:.
1.Define qué es una clase y un objeto.
Escribe tus respuestas aquí.
2.¿Qué elementos hay en un diagrama de clase?
Escribe tus respuestas aquí.
3.¿Qué elementos hay en un diagrama de secuencia?
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- J Rumbaugh, M. Blaha, W. Premerlani, F. Eddy y W. Lorensen. Object-oriented modeling and design. Prentice- HallBraude.
- http://www-306.ibm.com/software/rational/uml/

CAPÍTULO VII FASE IMPLEMENTACIÓN (Práctica #15, Práctica #16)

7.1 INTRODUCCIÓN

El objetivo de esta práctica es iniciar con la implementación del código para el producto y se necesita tener las especificaciones de requerimientos y diseño terminadas, revisadas, actualizadas; un estándar de codificación definido y documentado, estar listos para construir el código.

Lo ideal es que los integrantes que realizarón el diseño sean quienes realicen la implementación. En la Implementación generalmente se crea una base de datos, se hace la conexión al código, utilizando para ello *postgreSQL que es un administrador de bases de datos de dominio público.*

Postgres ofrece una característica adicional al incorporar los siguientes conceptos básicos de tal forma que los usuarios puedan extender el concepto de un sistema orientado a objetos que está basado en una colección de objetos :

- Atributo.- Unidad mínima de información en una base de datos. Por ejemplo, un campo puede ser un nombre, un código de artículo, la edad de una persona.
- Entidad.- Colección de atributos, formando una unidad de información coherente. Por ejemplo, si necesitamos los datos de un empleado, tendremos un registro empleado que constará de varios atributos como su nombre, dirección.
- Clave.- Dentro de una entidad, hay atributos que permiten diferenciar de manera unívoca a dicha entidad; a esos atributos se les llaman *claves*. Por ejemplo, en el registro empleado, la clave será el campo DNI, puesto que no puede haber dos empleados distintos con el mismo DNI, mientras que si eligiésemos como clave el campo Apellidos, no podríamos distinguir a algunos empleados, puesto que dos empleados distintos pueden tener los mismos apellidos.
- Tabla .- Colección de entidades. Podemos tener, por ejemplo, una tabla formada por todos los datos sobre empleados, otra tabla con los datos de clientes, ... y

todas las tablas serán lo que forme la Base de Datos.

- Relación .- Es una asociación entre diferentes entidades.
- Objeto.- Contiene fragmentos de código que se llaman métodos y donde un objeto puede acceder a los datos de otro objeto.
- Clases.- Es un objeto que contiene los mismos tipos de valores y los mismo métodos que se agrupan.

El modelo orientado a objetos se basa en el encapsulamiento de datos donde cada objeto está asociado con:

- Un conjunto de variables que contiene los datos del objeto, las variables corresponden al modelo Entidad-Relación (E-R).
- Un conjunto de mensajes, cada mensaje puede no tener parámetros o tener uno o varios.
- Un conjunto de métodos, cada uno de los cuales es código que implementa un mensaje, el método devuelve un valor.

Otras características aportan potencia y flexibilidad adicional al manejador de bases de datos como:

- Restricciones (Domain Constraints) Verifica siempre que se introduce en la base de datos un nuevo elemento y examina las consultas para asegurar de que tengan sentido. La cláusula **check** permite restringir los dominio y por ejemplo para asegurar que un dominio de sueldo por hora sólo permita valores mayores que un valor especificado (como puede ser el sueldo mínimo) tal y como se muestra aquí:

```
create domain sueldo-por-hora numeric (7,1)
        constraint comprobación-valor-sueldo
                check (value>= 800,0)
```

- Integridad Referencial en SQL Aseguran que un valor que aparece en una relación para un conjunto de atributos dado aparezca también para un conjunto de atributos concreto en otra relación.
- Asertos (Assertions) es un predicado que expresa una condición que se desea que la base de datos satisfaga siempre. En SQL los asertos adoptan la siguiente forma:

  Create assertion < nombre-aserto> **check** <predicado>
- Disparadores (triggers) Es una orden que el sistema ejecuta de manera automática como efecto secundario de la base de datos. Se debe cumplir con especificar las condiciones en las que se va a ejecutar el trigger , especificar las acciones que se van a realizar cuando se ejecuta el disparador. A veces se denominan *reglas o reglas activas.*

Estas características coloca a Postgres en la categoría de las Bases de Datos identificadas como *objeto-relacionales* y son diferentes a las referidas como *orientadas a objetos*, que en general no son bien aprovechables para soportar lenguajes de Bases de Datos relacionales tradicionales. Postgres tiene algunas características que son propias del mundo de las bases de datos orientadas a objetos.

7.2 CREACIÓN Y CONEXIÓN A BASES DE DATOS (BD)

El objetivo es crear una base de datos dentro de Postgres, para ello se instala Postgres, bajando las fuentes de http://www.postgresql.org/, con el comando **create base-de-datos**. **createdb**, es un comando hecho para crear la base de datos, que puede estar bajo el ambiente de Windows o Linux. Para ello se utiliza el lenguaje Structered Query Language (SQL), está compuesto por comandos, operaciones que nos sirven para crear y manipular la base de datos.

La estructura básica de una expresión en SQL consiste en tres cláusulas:

- **Select** corresponde a la operación de proyección del álgebra relacional.

- **From** Corresponde al producto cartesiano de la operación del álgebra relacional, lista la relación que va ser buscado al evaluar una expresión.
- **Where** Corresponde a la selección del predicado del álgebra relacional, el cual opera en la clase from.

La estructura tiene la siguiente forma:

select A_1 , A_2,.....A_n //Cada A_n representa un atributo

from r_1,r_2,.....r_m // Cada r_m representa una relación

where P // P es el predicado

El Álgebra relacional consta de un conjunto de operaciones que toma como entrada una o dos relaciones y produce como resultado una nueva relación, siendo así un lenguaje de consulta procedimental.

7.3 PASO DEL MODELO DE CLASES A CÓDIGO JAVA

El objetivo de esta práctica, es pasar de los diagramas de clases y de secuencia a código java.

Por tanto, teniendo los modelos de las clases, se pasan a código Java, lo que reduce en un 50% los errores más comunes de programación al eliminar muchas de las características de éstos.

PRÁCTICA # 15

NOMBRE DEL ALUMNO:	
INSTRUCTOR:	
FECHA:	
FASE:	**IMPlementación**
INGENIERÍA DE SOFTWARE	

CREACIÓN Y CONEXIÓN A UNA BASE DE DATOS.

OBJETIVOS :

- Crear la base de datos a usar.
- Acceder a la BD para construir las tablas.
- Insertar datos.
- Salir.
- Acceso a BD desde JAVA.

INTRODUCCIÓN :

El Sistema Manejador de Bases de Datos Relaciónales conocida como PostgreSQL, ofrece la posibilidad de incorporar los siguientes cuatro conceptos de forma que los usuarios puedan modelar fácilmente los datos de un sistema.

- **INSTRUCCIONES SQL**

El lenguaje Structured Query Language (SQL) está compuesto por comandos, cláusulas, operadores y funciones. Estos elementos se combinan en las instrucciones para crear, actualizar y manipular las bases de datos, es un lenguaje estándar de comunicación para bases de datos.

Comandos

Existen dos tipos de comandos SQL:

- Lenguaje de Definición de datos **(DLL)** es un conjunto de tablas que permiten crear y definir nuevas bases de datos, campos e índices.
- Lenguaje de manipulación de datos **(DML)** que permiten a los usuarios generar consultas para ordenar, filtrar tablas y extraer datos de la base de datos.

Comandos DLL	
Comando	**Descripción**
CREATE	Crear nuevas tablas, campos e índices
DROP	Eliminar tablas e índices
ALTER	Modificar las tablas agregando campos o cambiando la definición de los campos.
Comandos DML	
Comando	**Descripción**
SELECT	Consultar registros de la base de datos que satisfagan un criterio determinado
INSERT	Cargar conjuntos de datos en la base de datos en una operación.
UPDATE	Modificar los valores de los campos y registros especificados

DELETE	Eliminar registros de una tabla de una base de datos

Cláusulas

Las cláusulas son condiciones de modificación utilizadas para definir los datos que desea seleccionar o manipular.

Cláusula	Descripción
FROM	Especificar la tabla de la cual se van a seleccionar los registros
WHERE	Especificar las condiciones que deben reunir los registros que se van a seleccionar
GROUP BY	Separar los registros seleccionados en grupos específicos
HAVING	Expresar la condición que debe satisfacer cada grupo
ORDER BY	Ordenar los registros seleccionados de acuerdo con un orden específico

Operadores Lógicos

Operador	Uso
AND	Es el "y" lógico. Evalúa dos condiciones y devuelve un valor de verdad sólo si ambas son ciertas.
OR	Es el "o" lógico. Evalúa dos condiciones y devuelve un valor de verdad si alguna de las dos es cierta.
NOT	Negación lógica. Devuelve el valor contrario de la expresión.

Operadores de Comparación

Operador	Uso
<	Menor que
>	Mayor que
<>	Distinto de

<=	Menor o igual que
>=	Mayor o igual que
=	Igual que
BETWEEN	Especificar un intervalo de valores.
LIKE	Comparación de un modelo
In	Especificar registros de una base de datos

Funciones de Agregado

Las funciones de agregado se usan dentro de una cláusula SELECT en grupos de registros para devolver un valor único que se aplica a un grupo de registros.

Función	Descripción
AVG	Calcular el promedio de los valores de un campo determinado
COUNT	El número de registros de la selección
SUM	Devolver la suma de todos los valores de un campo determinado
MAX	El valor más alto de un campo especificado
MIN	El valor más bajo de un campo especificado

Orden de ejecución de los comandos

Dada una sentencia SQL de selección que incluye todas las posibles cláusulas, el orden de ejecución de las mismas es el siguiente:

- Cláusula FROM
- Cláusula WHERE
- Cláusula GROUP BY
- Cláusula HAVING
- Cláusula SELECT
- Cláusula ORDER BY

Comandos

ABORT — Aborta la transacción en curso.

BEGIN — Comienza una transacción en modo encadenado.

CLOSE — Cierra un cursor.

CLUSTER — Proporciona aviso de almacenaje agrupado (clustering) al servidor.

COMMIT — Realiza la transacción actual.

COPY — Copia datos entre ficheros y tablas.

CREATE AGGREGATE — Define una nueva función de agregado.

CREATE DATABASE — Crea una nueva base de datos.

CREATE FUNCTION — Define una nueva función.

CREATE GROUP — Crea un grupo nuevo.

CREATE RULE — Define una nueva regla.

CREATE TABLE — Crea una nueva tabla.

CREATE TABLE AS — Crea una nueva tabla.

CREATE TRIGGER — Crea un nuevo disparador.

DELETE — Borra filas de una tabla.

DROP AGGREGATE — Elimina la definición de una función agregada.

DROP DATABASE — Elimina una base de datos existente.

DROP RULE — Quita una regla existente de la base de datos.

DROP SEQUENCE — Quita una secuencia existente.

DROP TABLE — Elimina tablas de una base de datos.

DROP TRIGGER — Borra la definición de un disparador.

DROP VIEW — Retira una vista definida en una base de datos.

END — Lleva a cabo la transacción actual.

INSERT — Inserta filas nuevas en una tabla.

ROLLBACK — Interrumpe la transacción en curso.

SELECT — Recupera registros desde una tabla o vista.

SELECT INTO — Crear una nueva tabla a partir de una tabla o vista ya existente.

SET — Fija parámetros de tiempo de ejecución para la sesión.

SHOW — Muestra los parámetros en tiempo de ejecución de la sesión.

TRUNCATE — Vacía una tabla.

UPDATE — Substituye valores de columnas en una tabla.

VACUUM — Limpia y analiza una base de datos Postgres.

- ***CREACIÓN DE UNA BASE DE DATOS***

  - **Obtener una cuenta de acceso a Postgres.**

PARA WINDOWS

//Permitir el acceso vía TCP/IP

$ ipc-deamon.exe&

//Inicializamos la BD.

$ adduser postgres

$ mkdir/usr/local/pgsql/data

$ chown postgres/usr/local/pgsql/data

$ su – postgres

$ usr/local/pgsql/bin/initdb –D/usr/local/pgsql/data

$ usr/local/pgsql/bin/postmaster –D/usr/local/pgsql/data

PARA LINUX

$su –postgres

$createuser name

  - **Creación de una base de datos.**

Windows/Linux

Supongamos que quiere crear una base de datos llamada *Mibase*. Puede hacerlo con el siguiente orden:

$/home/name> createdb *Mibase*

  - **Acceso a la base de datos.**

Puede que quiera ejecutar el programa psql. Puede activarlo para la base de datos *Mibase* escribiendo la orden:

$/home/name> psql *Mibase*

>>

Recibirá como respuesta el siguiente mensaje:

Welcome to the Postgres interactive sql monitor:

type \? for help on slash commands

type \q to quit

type \g or terminate with semicolon to execute query

You are currently connected to the database: *Mibase*

Mibase =>

Este indicativo le informa que el monitor de terminal se encuentra dispuesto y que puede escribir consultas SQL en el espacio de trabajo creado por el citado monitor de terminal.

- **OPERACIONES.**

Hay dos tipos de operaciones que son:

- Instrucciones.- Son un conjunto de operaciones.

>> CREATE DATABASE name

- Scripts.-

>> \i crea Tablas.sql

- **TIPOS DE DATOS.**

- Numéricos : int, date, char(n), varchar(n), text, bool.

  - **Insertar datos.**

>> \i script de datos.sql *// carga el archivo.*

>> Insert into

  - **Operaciones.**

INSERT ***//Lo que hace esta operación es insertar datos en la base de datos.***

INSERT INTO MY_TABLE VALUES (3, 'HOLA')

SELECT ***//Consulta registros de la base de datos.***

*SELECT * FROM MY_TABLE ;* ***//Selecciona toda la información que hay en la // tabla llamada MY_TABLE.***

SELECT a, b FROM table1 ORDER BY a; ***//Selecciona a, b de table1 ordenada.***

DELETE **// Elimina registros de una tabla de la base de datos.**

DELETE FROM table1 WHERE id=10;

DROP **// Elimina tablas e índices.**

DROP TABLE name ***//Elimina mi tabla.***

$dropbd nombredb **//Elimina la base de datos.**

El programa psql responde a códigos de escape que comienzan con la barra invertida, "\". Por ejemplo, puede obtener ayuda sobre la sintaxis de varias órdenes SQL de Postgres escribiendo:

nombredb=> \h

Una vez que ha terminado de introducir sus consultas en el espacio de trabajo, puede pasar el contenido de éste al servidor de Postgres escribiendo:

nombredb=> \g

Entonces el servidor procesa la consulta. Si termina su consulta con punto y coma, no es necesario que introduzca la secuencia \g. psql procesará automáticamente consultas terminadas en punto y coma. Para leer las consultas de un archivo, en lugar de introducirlas interactivamente, escribir:

nombredb=> \i nombre_fichero

Para **salir** de psql escribir:

nombredb=> \q

- **Destrucción de una base de datos.**

% dropdb *nombredb*

Esta acción elimina físicamente todos los archivos asociados con la base de datos y no puede ser invertida, por lo que sólo debe ser usada con gran precaución. Es también posible destruir una base de datos desde una sesión SQL usando:

> drop database *nombredb*

- **Copia de seguridad y restauración.**

Postgres proporciona dos comandos para realizar las copias de seguridad de su sistema: ***pg_dump*** para copias de seguridad de bases de datos individuales.

pg_dumpall para realizar copias de seguridad de toda la instalación de una sola vez.

La copia de seguridad de una base de datos puede realizarse usando la siguiente orden:
% pg_dump *nombredb* > *nombredb*.pgdump

y puede ser restaurada usando:
cat *nombredb*.pgdump | psql *nombredb*

- ***ACCESO A BD DESDE JAVA.***

Para hacer uso de Postgres desde java es necesario contar con un drive que nos permita establecer la conexión con la BD.

Driver: jdbc-XXX-XX-XX.jar

- Depende de la versión que se tenga de postgresSQL instalado.
- Para hacer uso del drive en las clases se necesita incluir java.sql.
- Para cargar el drive en la clase:

```
Class.forname("org.postgresql.Drive");   //Trabajando con el drive JDBC
```

Conexión:

- Abrir una conexión.

  ```
  Connection db = DriverManager.getConnection(url,username,password);
  jdbc:postgresql:database
  jdbc:postgresql://host /database
  jdbc:postgresql://host.port /database
  ```

- Cerrar una conexión.

  ```
  Db.close();
  ```

Operaciones:

- Objeto en el cual definimos la operación a ejecutar:

  ```
  Statment st= db.createStatement ();
  ```

- Ejecución de la consulta:
  - ResultSet rs= st.executeQuery("SELECT * FROM mytable where id= 500");
  - Int total = st.executeUpdate("DELETE FROM persona where id= 500")

- PreparedStatement st= db.prepareStatement("SELECT * FROM mytable where columnfoo = ?");

```
st.selint(1. foovalue);
ResultSet rs= st.executeQuery ();
```

Procesamiento de resultados obtenidos.

- executeUpdate:

```
int rowsDeleted = st.executeUpdate();
```

- executeQuery:

```
while(rs.next())
{
System.out.print("Column 1 returned ");
System.out.println(rs.getString(1));
}
```

Cerrar conexión.

- Cerrar conexión definida por el Statement y la conexión a la BD

```
rs.close();
st.close();
```

MATERIAL :

- PostgreSQL.

DESARROLLO:

1. Obtener una cuenta de acceso a postgres y ponerle el nombre que quieras a la base de datos.
2. Entrar al manejador de consultas.
3. Crear la tabla alumno con los siguientes campos:
   Nombre, edad, rfc, domicilio, teléfono.
4. Realizar las siguientes instrucciones SQL:
   - Insertar datos.

- Seleccionar registros de una BD.
- Actualizar registros de una BD.
- Eliminar registros de una BD.

5. Crear la clase Alumno.java y Conexión.java para poder acceder a la tabla alumno.

PREGUNTAS DE CONTROL:.
1. Define qué es PostgresSQL.
Escribe tus respuestas aquí.
2. Comandos indispensables en SQL y su función.
Escribe tus respuestas aquí.
3. Para qué se hace una copia de seguridad y restauración.
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Abraham Silberschatz, Henry F. Korth. DATABASE SYSTEM CONCEPTS, Third edition.

PRÁCTICA # 16

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **IMPlementación**

INGENIERÍA DE SOFTWARE

PASO DEL MODELO CLASES A CÓDIGO JAVA.

OBJETIVO :

- Teniendo los modelos de clases pasarlos a código java.

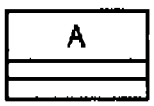
INTRODUCCIÓN :

La implementación se refiere a la programación. El propósito de la implementación es satisfacer los requerimientos de la manera en que especifica el diseño detallado. Y de lo que se trata es que teniendo los diagramas de UML pasarlos a código java.

❖ CLASES

Representa un conjunto de entidades que tienen propiedades comunes.
Una clase es un constructor que define la estructura y comportamiento de una colección de objetos. Un objeto es una instancia de la clase.

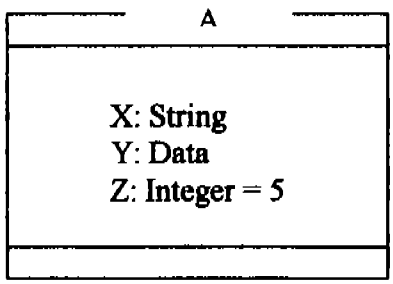
public Class A { };



❖ ATRIBUTOS

Representa una propiedad de una entidad. Cada atributo de un objeto tiene un valor que pertenece a un dominio de valores determinado.
Implementar cada atributo como una variable de instancia según su tipo y visibilidad
+ público, # protegido, - privado.

```
public Class A {
private String x;
private Data y;
private Integer z;
```



Para los métodos Set y Get de los atributos:

- **Set** de un atributo .-Sirve para poner o cambiar el valor de los atributos

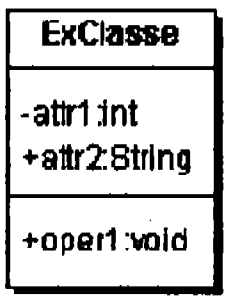
```
public Void set_x ( String x ) {
        This.x = x;
        }
```

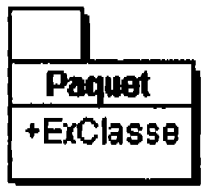
- **Get** de un atributo.- Este método sirve para obtener el valor de un atributo

```
public String get_x () {
        Return This.x;
                }
```

❖ CLASE DE UN PAQUETE

```
package Paquet;

class ExClasse {
    private Int attr1;
    private String attr2;

void oper1() {
  }
 }
```

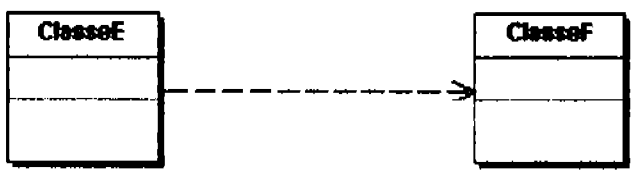




❖ DEPENDENCIAS

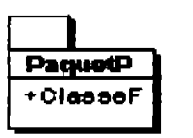
- Importar una clase de un paquete.

```
public class ClasseF {
}
import ClasseF;
public class ClasseE {
}
```



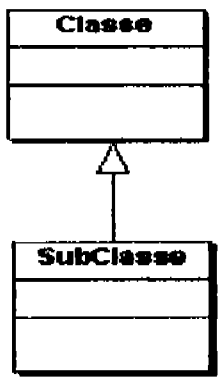
- Importar un clase de un paquete diferente.

```
package PaquetP;
public class ClasseF {
}
import PaquetP.ClasseF;
public class ClasseE {
}
```



❖ GENERALIZACIÓN

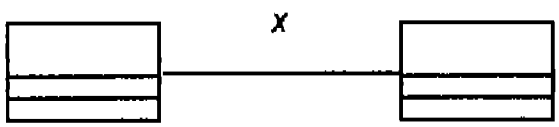
Es un proceso de abstracción en el cual un conjunto de clases existentes, que tienen atributos y métodos comunes, es referido por una clase genérica a un nivel mayor de abstracción. La relación de generalización denota una relación de herencia entre clases. Se representa dibujando un triángulo sin rellenar en el lado de la superclase. La subclase hereda todos los atributos y métodos descritos en la superclase.




```
public class Classe {
        }
public class SubClasse extends Classe {  }
```

❖ ASOCIACIÓN

Asociación (rol, multiplicidad): representan las relaciones entre instancias de clase. Una asociación es una línea que une dos o más clases y que tiene un nombre de rol que define para que es la asociación. Además puede incluir multiplicidades.



- Principales métodos de asociación

  A) Asocia_x ;(crea una asociación)
  B) Desasocia_x;(destruye una asociación)
  C) Get_x; (Sirve para hacer una consulta)

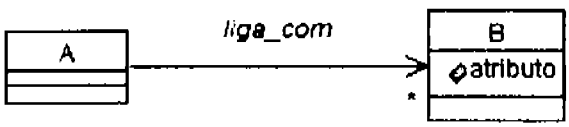
- Tipos de asociación
  - Unidirecional con multiplicidad de uno:

```
public Class A {        private B b ;
        public A ( B b ) {
  this.associa_ligaCom ( b ) ;
        };
  public Void associa_ligaCom ( B b ) {
                this.b = b ;
        };
public B get_ligaCom () {
                Return this.b;
        }
  }
```

- Unidirecional con multiplicidad 0..1:


```
public Class A {        Private B b ;
public Void associa_ligaCom ( B b ) {
        this.b = b ;
        };
public desassocia_ligaCom.( ) {
        this.b = Null
        };
public B get_ligaCom () {
        Return this.b
        }
            }
```

- Unidirecional con multiplicidad de *:



¬Ordenada

```
Import java.util.*;
public Class A {
private Set b = new Set ;
public Void associa_ligaCom ( B b ) {
        this.b.add(b) ;
    };
public desassocia_ligaCom ( B b ) {
        this.b.remove(b);
    };
public B get_ligaCom (String atributo) {
        Return this.b.get(atributo)
    }
  }
```

- Ordenada:


```
Import java.util.*;
public Class A {
private List b = new List ;
public Void associa_ligaCom ( B b , Integer posicao) {
                this.b.addAt(b, posicao) ;
        };
public desassocia_ligaCom (Integer posicao) {
this.b.removeAt(posicao);
        };
public B get_ligaCom (Integer posicao) {
        Return this.b.getAt(posicao)
        }
  }
```

- Bidireccional:


```
public Class A {
        private Set b ;

public Void associa_aux_ligaCom ( B b ) {
                this.b.add( b );
        };
public Void desassocia_aux_ligaCom ( B b ) {
                this.b.remove( b );
        };
public Void associa_ligaCom ( B b ) {
        this.associa_aux_ligaCom (b );
        b.associa_aux_ligaCom ( this ) ;
         };
public Void desassocia_ligaCom ( B b ) {
        this.desassocia_aux_ligaCom (b );
        b.desassocia_aux_ligaCom ( this ) ;
         };
public B get_ligaCom () {
        Return this.b;
        }
          }
```

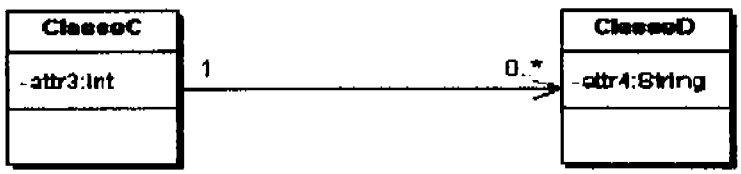
❖ ASOCIACIÓN Y MULTIPLICIDAD 1 –1 DIRIGIDA


```
public class ClasseB {
    }
public class ClasseA {
private ClasseB lnkClasseB;
    }
```

❖ ASOCIACIÓN Y MULTIPLICIDAD 1 –1 NO DIRIGIDA

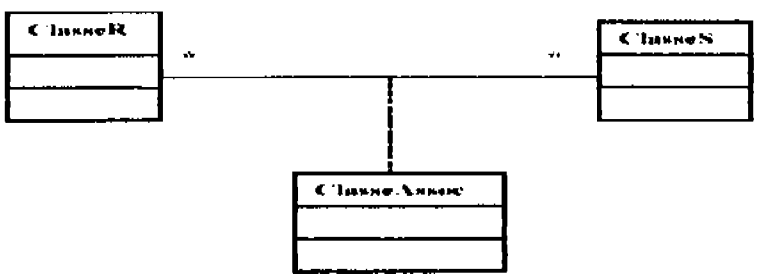

```
public class ClasseM {
private ClasseN lnkClasseN;
        }
public class ClasseN {
private ClasseM lnkClasseM    }
```

❖ ASOCIACIÓN Y MULTIPLICIDAD 1- 0.....*



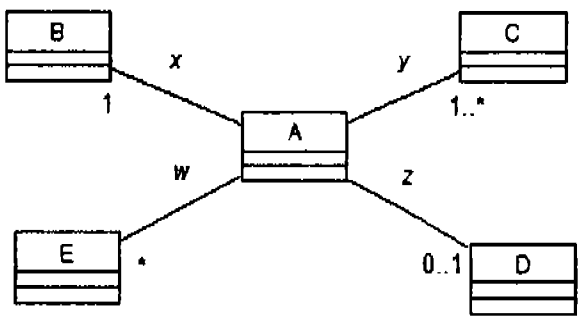

```
public class ClasseD {

private String attr4;
}
import java.util.Vector;
public class ClasseC {

private int attr3;
private Vector lnkClasseD;
}
```

❖ CLASE ASOCIACIÓN



- ❖ **CONSTRUCTOR**
- ➢ Crea una instancia
- ➢ Representa asociaciones para *
- ➢ Recibe como parámetros objetos (de 1 a 1....*)

```
public Class A {
  private B b ;
  private Set c;
  private D d;
  private Set e;
  Public A ( B b ; C c ) {
        this.c = new Set;
      this.e = new Set;
      this.b = b;
      this.c.add( c );
      }  }
```



MATERIAL :

- Diagramas en UML.
- Java.

DESARROLLO:

1. A partir de los diagramas de clases de tu sistema pasarlos a código java .

PREGUNTAS DE CONTROL:.
1. ¿Por qué es bueno tener los diagramas de clases para después hacer el código?
Escribe tus respuestas aquí.
2. ¿Cómo se denota una clase en java?
Escribe tus respuestas aquí.
3. ¿Cómo se ponen las relaciones en un sentido y multiplicidad de 0....k?

Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica además de especifica los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Grady Booch, James Rumbaugh, Ivar Jacobson. El lenguaje unificado de Modelado. Addisson-Wesley.
- J Rumbaugh, M. Blaha, W. Premerlani, F. Eddy y W. Lorensen. Object-oriented modeling and design. Prentice-HallBraude.

CAPÍTULO VIII FASE PRUEBAS(Práctica #19, Práctica #20)

8.1 INTRODUCCIÓN

El objetivo consiste en evaluar el producto, detectando los defectos antes de la fase de pruebas. Se aprenderá, qué es la calidad de un producto, el cual se determina cuando se va construyendo desde el modelado.

Las etapas de las pruebas son pruebas de integración y pruebas del sistema, se integran las partes del sistema en uno completo y listo para aplicarle pruebas. Las pruebas de integración aseguran que todas las partes se hayan incluido en el sistema y que las interfaces trabajen en conjunto. Finalmente, las pruebas del sistema, validan las funciones y desempeño del sistema completo contra los requerimientos.

La documentación también es parte esencial de cada producto de software. Es tan importante que el código del programa. En TSPi, el trabajo de documentación se incluye en la fase de pruebas. La importancia de la documentación podría ser comparada con la importancia de la existencia de una póliza de seguro; mientras todo va bien no existe la precaución de confirmar si nuestra póliza de seguros está o no vigente.

La documentación adecuada y completa, de una aplicación que se desea implantar, mantener y actualizar en forma satisfactoria, es esencial en cualquier sistema de información, sin embargo, frecuentemente es la parte a la cual se dedica el menor tiempo y se le presta menos atención.

Siempre se debe documentar un sistema como si estuviera a punto de irse a Siberia el siguiente mes, para nunca volver. Si la documentación del sistema es incompleta el diseñador continuamente estará involucrado.

8.2 PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA

El objetivo es verificar que el producto a los que se les aplican las pruebas son de alta calidad. Para ello se debe realizar la prueba de integración y la prueba de integración del sistema. En las pruebas de integración se pueden planear y llevar acabo en conjunto con las pruebas unitarias. En cambio una prueba del sistema es la culminación de las pruebas de integración. Estas pruebas se realizan mientras la aplicación se ejecuta en su entorno. Este tipo de prueba asegura que los requerimientos se cumplan, validando así cada requerimiento.

8.3 MANUALES DEL SISTEMA

En esta práctica se desarrollarán los manuales del sistema, el cual consiste en el material que explica las características técnicas y la operación de un sistema. Es esencial para proporcionar entendimiento de un sistema a quien lo vaya a usar para mantenerlo, para permitir auditoria del sistema y para enseñar a los usuarios como interactuar con el sistema y a los operandos como hacerlo funcionar. Existen varios tipos de manuales dependiendo de quien lo vaya a usar. El del *código* explica la lógica de un programa e incluye descripciones, diagramas de flujo, listados de programas y otros documentos; el del *usuario* describe en forma general la naturaleza y capacidades del sistema y cómo usarlo.

PRÁCTICA # 19

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **PRUebas**

INGENIERÍA DE SOFTWARE

PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA.

OBJETIVO :

- Guiar al equipo en la integración y pruebas de los componentes del producto del primer ciclo de desarrollo.

INTRODUCCIÓN :

En TSPi, el propósito de las pruebas consiste en evaluar el producto y no tanto en modificarlo. Los defectos debieron haberse detectado y arreglado antes de la fase de pruebas. La calidad de un producto se determina cuando se desarrolla.

La fase de pruebas tiene como objetivo verificar que los productos a los que se les aplican las pruebas son de alta calidad. Las actividades principales son:

- Utilizar las partes desarrolladas y a las que se les ha aplicado pruebas unitarias para construir el sistema.
- Utilizar las pruebas de integración para verificar que el sistema está debidamente construido, que todas las partes se encuentran desarrolladas y que funcionan en conjunto.
- Validar con las pruebas del sistema que el producto realiza los requerimientos del sistema.

La estrategia de integración verifica que todos los componentes estén presentes y que las llamadas y otras interacciones trabajen adecuadamente.

"PRUEBAS DE INTEGRACIÓN"

El software actual es complejo, se construye con partes que se desarrollan por separado. La **"Integración"** es el proceso de ensamble. Las pruebas de integración se realizan sobre un sistema parcialmente construido para verificar que el resultado de integrar software opera como estaba planeado.

Manera de planear la integración:

- Tener una arquitectura sencilla
- Usar el plan de integración desarrollado junto con la arquitectura (Práctica 13)
- Planear las pruebas
- Refinar el programa para reflejar los resultados

Los casos de uso son una fuente ideal de casos de prueba para las pruebas de integración. Es recomendable integrar por casos de uso y que se vaya incorporando los que ya están. Se organizan los casos de pruebas en paquete de clases especialmente creadas para las pruebas. Finalmente, un paquete se dedica a probar toda la aplicación.

Las pruebas de integración se van realizando mientras las construcciones avanzan. Este tipo de pruebas se aplican en periodos regulares y frecuentes, dan seguridad a los programadores para seguir trabajando de la misma forma, o indican problemas que podrían ocasionar grandes demoras en la integración.(FIGURA 8.1)

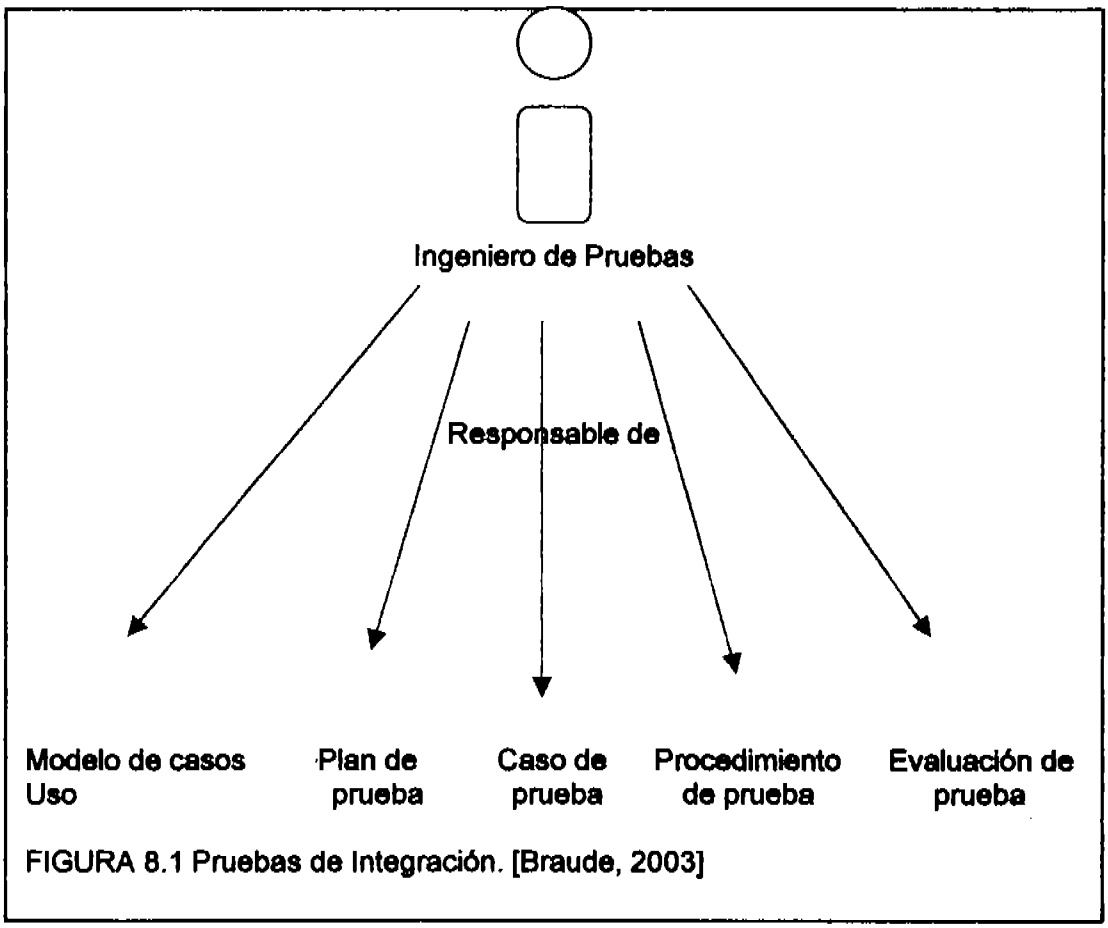


FIGURA 8.1 Pruebas de Integración. [Braude, 2003]

Documentos a usar para las pruebas de Integración

- *Modelo de casos de uso:* Conjunto de casos de uso que describen el uso típico de la aplicación y los diagramas de secuencia que los describen con detalle.
- *Casos de prueba:* Los datos de entrada para cada prueba.

- *Procedimientos de prueba:* La manera en que se deben ejecutar las pruebas, y evaluar los resultados. Estos pueden ser los procedimientos manuales o usar herramientas para pruebas.
- *Evaluación de pruebas:* Resumen, detalles y efectos de los defectos encontrados.
- *Plan de pruebas:* Plan global para realizar las pruebas; incluye el orden.
- *Componentes de las pruebas:* Código fuente para las pruebas en sí, y para el código de la aplicación que se quiere probar.
- *Defectos:* Informes con los defectos descubiertos como resultado de este proceso, clasificado por la severidad y tipo.

"PRUEBAS DEL SISTEMA"

La prueba del sistema es la culminación de las pruebas de integración. Esta prueba se realizan en su entorno. Este tipo de prueba asegura que los requerimientos se cumplan, validando así cada requerimiento.

Se realiza en toda la aplicación. Las pruebas del sistema validan los requerimientos funcionales y los requerimientos no funcionales como desempeño, velocidad de ejecución y uso de almacenamiento.

La estrategia de pruebas pretende responder cuatro preguntas:

1. ¿El sistema desempeña las funciones especificadas?
2. ¿Reúne los objetivos de calidad?
3. ¿Operará apropiadamente en condiciones normales?
4. ¿Operará apropiadamente bajo condiciones adversas?

Tipos de pruebas del sistema

- Volumen: Somete al producto a la entrada de grandes cantidades de datos
- Utilidad: Valida la aceptación de la aplicación por los usuarios, mide la reacción del usuario (por ejemplo: calificación de 0 a 10)

- ❖ Desempeño: Mide la velocidad para varias circunstancias.
- ❖ Configurabilidad: Configura los distintos elementos de hardware / software (por ejemplo: mide el tiempo de preparación)
- ❖ Compatibilidad: Con otras aplicaciones designadas por ejemplo: mide el tiempo de adaptación.
- ❖ Seguridad: Sujeta a intentos comprometedores (ejemplo: mide el tiempo promedio para entrar tirar al sistema)
- ❖ Confiabilidad / disponibilidad: Mide el tiempo de operación en periodos largos
- ❖ Uso de recursos: Mide el uso de RAM o espacio en disco.
- ❖ Aptitud de instalación: Mide el tiempo de instalación
- ❖ Recuperabilidad: modelar actividades para reactivar la aplicación. (mide el tiempo de recuperación)
- ❖ Funcionalidad: Se refiere a la facilidad o dificultad con que la aplicación se mantiene operativa (mide el tiempo de servicio)
- ❖ Carga / tensión: Sujeta a datos extremos y trafico de eventos.

MATERIAL :

- Casos de usos.
- Plan de pruebas de integración.

DESARROLLO:

Utilizando el sistema del curso elabora las pruebas de integración y del sistema:

Efectuar mediciones de las pruebas, registrando los defectos en la REGD y los tiempos en la REGT, responde las siguientes preguntas:

1. ¿Cuánto tiempo se necesitó para ejecutar la prueba?
2. ¿Cuántos defectos se encontraron?
3. ¿Requiere intervención manual o puede procesarse con otras pruebas?

- ❖ Haz un registro de pruebas indicando lo siguiente:

1. La fecha de ejecución de la prueba.

2. El nombre de quien aplica la prueba.
3. Nombre o número de pruebas.
4. El producto y configuración que se está probando.
5. La hora de inicio.
6. La hora de terminación.
7. El número de defectos encontrados, guardando los datos en REGD.
8. Los resultados de las pruebas.
9. La configuración del sistema que se está probando.
10. Herramientas o facilidades utilizadas.
11. Si se requirió la intervención de algún operador y cuánto tiempo.

PREGUNTAS DE CONTROL:.
1.¿Qué es una prueba de integración y de sistema?
Escribe tus respuestas aquí.
2. ¿Cuántos defectos se encontraron?
Escribe tus respuestas aquí.
3.Menciona el tiempo total que se llevaron las pruebas.
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica además de especifica los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Watts S. Humphrey. Introduction to the Team Software Process. Cap9, 2000.
- Eric J Braude. INGENIERÍA DE SOFTWARE "Una perspectiva orientada a objetos", 2003.

PRÁCTICA # 20

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **PRUebas**

INGENIERÍA DE SOFTWARE

MANUALES DEL SISTEMA.

OBJETIVOS :

- Conocer los tipos de manuales que existen.
- Cuántos tipos de manuales existen.

INTRODUCCIÓN :

Los manuales son parte esencial de cada producto de software, son tan importantes como el código del programa. TSPi incluye el trabajo de generación de manuales en la fase de pruebas. En grandes sistemas, debería iniciarse antes y continuar durante las pruebas. Es conveniente aplicar pruebas a manuales de los usuarios como parte de las pruebas del sistema.

Los manuales debe estar escritos para quien los va a leer. Hay distintos tipos de lectores de manuales:

- el cliente que lo encargó.
- el usuario final.
- quién instalará el sistema.
- quién lo mantendrá.
- quién lo desarrolló.
- quién administra el proceso de desarrollo.

Tipos de Manuales:

- Manual de usuario.

Escribir manuales de usuario útiles es un reto para los Ingenieros de Software.
Un manual bien diseñado debe organizarse de acuerdo a las necesidades del lector y no a la estructura del producto. En general, la primera sección debería contener las necesidades del usuario y no la estructura del producto. La primera sección explica al usuario cómo empezar a utilizar el producto, después se deberá explicar al usuario qué puede hacer con el producto.

Es necesario utilizar:

- Un glosario para definir términos que no estén en el diccionario estándar.
- Incluir una sección con mensajes de error y de recuperación de errores.
- Tener un índice de términos importantes.
- Proporcionar una tabla de contenido detallado.

Este manual debe contener información para poder usar el sistema.
Cómo se entra, qué aparece en la interfaz, qué se espera en cada opción o campo, qué responderá el sistema, cómo resolver problemas a la hora de la ejecución.
Podría contener información sobre la instalación.

- ❖ Manual de instalación.

Debe estar escrito de forma que se pueda hacer la instalación del software. Indicará la versión que se está instalando. Necesidades de hardware indicando capacidades y velocidades mínimas. Necesidades de software con versiones mínimas. Procesos para la instalación. A quién recurrir en caso de problemas.

- ❖ Manual de desarrollo.

Este manual es la documentación del objetivo y características del software, de las decisiones tomadas y la historia de toda la documentación generada. Incluye los documentos de las fases de requerimientos, diseño, implementación y pruebas.

- ❖ Manual de administración del proceso de desarrollo.

Contiene la información sobre el lanzamiento, estrategia y planeación del proceso. Incluye la evaluación postmortem para cada ciclo.

La información necesaria es:

- Contenido.
- Nombre del sistema.
- Describir el nombre del sistema a implantar en la empresa.
- Equipo encargado del sistema.
- Nombre del personal encargado del análisis y diseño del sistema.
- Resumen Administrativo.

El propósito es permitir enterarse en forma somera de la propuesta del sistema. En este punto aparece por primera vez el nombre del sistema, el cual debe ser único, éste deberá conservarse invariable en todos los documentos referentes a ese sistema.

Revisión de los manuales

Se aconseja que uno o más colegas revisen los manuales generados. Si el producto es para clientes o usuarios, aplique pruebas de usuario para asegurarse que la escritura es clara, precisa, completa y entendible. A continuación se presentan los siguientes elementos para las revisiones.

1. *Organización de los manuales.*

¿El manual está organizado de acuerdo a lo que el usuario hará o de acuerdo al contenido del producto?. Los manuales del usuario deberían estar organizados de acuerdo a sus necesidades y no a la estructura o contenido del producto.

2. *Terminología del manual.*

¿El manual presume conocimiento que el usuario probablemente no tenga?. Debe definir cualquier palabra en el diccionario.

3. *Contenido del manual.*

¿El manual cubre el material requerido?.

4. *Precisión.*

¿El desarrollo de los métodos y procedimientos es de acuerdo a su descripción?.

5. *Legibilidad.*

¿Es fácil leer el manual? Lea el manual verificando que es fácil decir lo que está escrito.

6. *Entendimiento.*

¿Las personas entienden lo que está escrito? Generalmente, esta pregunta es la más difícil de responder. Las mejores pruebas consiste en solicitar a alguien que no ha utilizando el sistema que lo haga. Observe sus reacciones, registre los problemas y modifique el manual de acuerdo a los problemas identificados.

MATERIAL :

- El sistema desarrollado en curso.

DESARROLLO:

Generar en borrador del manual de usuario

- Generar el borrador del manual de usuario para ciclos posteriores.
- Revisar , corregir y generar el manual de usuario.

Generar en borrador del manual de instalación

PREGUNTAS DE CONTROL:.
1.¿Cuál es la importancia de generar manuales para un producto?
Escribe tus respuestas aquí.
2.¿Qué tipos de manuales existen?
Escribe tus respuestas aquí.
3.¿Por qué es útil actualizar los manuales ?
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para tí el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Watts S. Humphrey. Introduction to the Team Software Process. Cap9, 2000.
- Eric J Braude. INGENIERÍA DE SOFTWARE "Una perspectiva Orientada a Objetos", 2003.

CAPÍTULO IX FASE POSTMORTEM (Práctica #22)

9.1 INTRODUCCIÓN

El objetivo para realizar la práctica en la fase de postmortem y que es la última fase en el proceso de TSPi, es realizar la recolección de métricas del proceso (práctica #21) que se encuentra en el proyecto de Daniela Avalos y se revisa que el equipo de trabajo haya terminado todas las tareas y registrado todos los datos requeridos (práctica #22). Proporciona una manera estructurada de aprendizaje y mejora, donde se evalúa el desempeño personal y del equipo. Cada ciclo de desarrollo de TSPi finaliza con un postmortem.

9.2 EVALUACIÓN DE ROLES EN EL EQUIPO

En esta práctica se hace un reporte, donde se describe lo que se produjo, el proceso que se utilizó y los roles desempeñados; lo que se pudo aplicar y lo que no fue conveniente, se pueden hacer sugerencias que puedan aplicarse en el siguiente ciclo. Se debe describir el desempeño que cada integrante tuvo respecto al rol que desempeñó, de acuerdo a los indicados por TSPi.

Deberá escribirse un reporte breve y objetivo.

El reporte del ciclo deberá tener el siguiente contenido:

- Índice.
- Resumen.
- Reporte de roles, en cuanto a:
  - Liderazgo.
  - Desarrollo.

- Planeación.
- Proceso.
- Calidad.
- Apoyo o soporte.
- Reporte Personal. Cada quien describe el desempeño personal de acuerdo a lo planeado y la calidad de trabajo que desarrollo.

El objetivo de este reporte consiste en proporcionar una guía para desempeñar de mejor manera cada rol o que sea utilizado por quien posteriormente tenga que desempeñar cada rol tomando en cuenta las recomendaciones escritas.

PRÁCTICA # 22

NOMBRE DEL ALUMNO:

INSTRUCTOR:

FECHA:

FASE: **POSTMortem**

INGENIERÍA DE SOFTWARE

EVALUACIÓN DE ROLES EN EL EQUIPO.

OBJETIVO :

- Revisar el desempeño de los miembros del equipo.

INTRODUCCIÓN

Para que un grupo trabaje verdaderamente en equipo es muy importante que sus miembros desempeñen ciertos roles conforme lo requieren las circunstancias. No se pretende que cualquier miembro sea capaz de ejercer todos los roles positivos. Esto es muy difícil, si no imposible. Se trata de que entre todos los miembros se logre un adecuado ejercicio de los roles. Unos miembros serán más aptos para ciertos roles y otros lo serán para otros roles.

Para ello se hará uso de las formas de TSPi personal y en equipo:

La forma ***Semana Personal***, como su nombre lo indica es individual y resume los tiempos dedicados al proyecto de esa semana y se forma del resumen que cada persona hace, apuntando en su REGT de la semana. Consta de los siguientes campos:

- Tareas de desarrollo efectuadas.- Es la tarea que se está realizando.
- Horas dedicadas.- Cuánto tiempo se le dedicó a determinada tarea en esta semana.
- Tamaño del producto.- Si ya está terminado el producto se pone el tamaño que tuvo, puede ser en páginas, LOC, etc.
- Total.- Es la suma de horas y tamaños. (FIGURA 9.1)

"Semana Personal"

Nombre _____ Equipo _____ Instructor _____

Fecha _____ Ciclo _____ Semana # _____

Tareas de desarrollo efectuadas	Horas dedicadas	Tamaño del producto	Estado del producto
Totales			

FIGURA 9.1 Semana Personal [Humphrey, 2000]

La forma ***Semana para el equipo***, es el reporte que se prepara cada semana y que resume el trabajo de todos los integrantes del equipo. Cada integrante del equipo entrega en la forma semana personal, el trabajo realizado durante la última semana. Y en la del equipo se resume el trabajo realizado en la semana por todo el equipo. (FIGURA 9.2)

"Semana para el equipo"

Equipo _____ Fase _____ Instructor _____

Fecha _____ Ciclo _____ Semana _____

Datos semanales	**Actual**
Horas dedicadas al proyecto en esta semana	
Horas dedicadas al proyecto en este ciclo hasta esta semana	
Horas dedicadas a las tareas terminadas en esta fase en esta semana	

Tareas de desarrollo efectuadas esta semana	**Horas dedicadas**	**Tamaño del producto**	**Estado actual del producto**
Totales			

Seguimiento de asuntos o riesgos	**Estatus**

FIGURA 9.2 Semana para el equipo. [Humphrey, 2000]

❖ Primera parte

DATOS SEMANALES

- *Horas dedicadas al proyecto en esta semana*.- Cuántas horas se le dedicó al proyecto en la semana. Se suman todos los totales de los integrantes.

- *Horas dedicadas al proyecto en este ciclo hasta esta semana.*- Es la suma de las horas que lleva el equipo en este ciclo desde que se comenzó.
- *Horas dedicadas a las tareas terminadas en esta fase en esta semana.*- Cuantas horas se llevaron, de las tareas terminadas en cada fase hasta el día de hoy.

❖ Segunda parte

TAREAS DE DESARROLLO EFECTUADAS ESTA SEMANA

- *Horas dedicadas.*- cuántas horas le dedicaron a determinada tarea todos los miembros del equipo.
- *Tamaño del producto.*- Puede ser en páginas, LOC, etc.
- *Estado actual del producto.*- Si está terminado o no.
- *Total.*- Es sacar la suma de cada uno.

❖ Tercera parte

SEGUIMIENTO DE ASUNTOS O RIESGOS

- *Estatus* .- Resumir el estado de asuntos y riesgos que preocupen al equipo, así como cualquier cambio importante acontecido en la semana.

MATERIAL

- Reunirse en equipo.
- Tener las formas de evaluación personal y del equipo.

DESARROLLO

PARTE I.- Individualmente contesta las siguientes preguntas:

❖ ¿Cómo prefieres que tu equipo se relacione contigo?
❖ ¿Qué hacías con la información que encontrabas y no te correspondía buscar?
❖ ¿Prefieres tomar tus decisiones?

PARTE II.- Contesten en equipo:

❖ Llenar la forma de evaluación personal y del equipo.

- Conjuntar los resultados y saca conclusiones sobre el desempeño de cada uno del equipo.
- Plantear cambio de roles para el siguiente ciclo.

PREGUNTAS DE CONTROL:.
1.- ¿Qué es lo que mas te agrado de tu equipo?
Escribe tus respuestas aquí.
2.¿Qué calificación te pondrías en el rol que jugaste?
Escribe tus respuestas aquí.
3.¿Qué recomiendas para el siguiente curso ?
Escribe tus respuestas aquí.

CONCLUSIONES Brinda una opinión de lo que significó para ti el desarrollo de la práctica, además de especificar los conocimientos adquiridos en la realización de la misma.
Escribe tus respuestas aquí.

BIBLIOGRAFÍA :

- Daniels, Aubrey. Gerencia del desempeño. McGraw Hill, 1993.
- Watts S. Humphrey. Introduction to the Team Software Process. Cap10, 2000.

CAPÍTULO X. CONCLUSIONES

10.1 CONCLUSIONES

En este trabajo se pudo apreciar como las prácticas ayudan a los estudiantes a la construcción de un software, aplicando los principios de la IS y que son un auxiliar didáctico en el aprendizaje de ingeniería de software.

Al realizar prácticas para la materia de Ingeniería de software, fue muy interesante ver como los alumnos aprendieron a convivir como equipo y ver qué les agradaba uno del otro, donde el objetivo común fue: el término de su producto, pero al mismo tiempo tenían que entregar la práctica de esa fase, allí es donde se veía si el alumno ponía interés y mejoraba en la presentación de sus documentos.

Los alumnos cada semana tenían que entregar sus documentos y llenar las formas correspondientes a dicha fase. Las prácticas les ayudaban a generar los documentos y llenar las formas. La mayoría de los alumnos entregaban sus prácticas a tiempo. En las primeras prácticas no hubo ningún descontrol, pero conforme pasaban las semanas ellos se iban presionando un poco más en el desarrollo del producto, para llegar a la fase de implementación ellos ya tenían corregidos los documentos de fases anteriores, para empezar la programación que es donde les llevó un poco de tiempo. Después de eso, siguió la fase de pruebas, donde veían que tal funcionaba su producto. Y por último tenían que evaluarse como equipo e individualmente dependiendo del rol que cada uno desempeñó.

El apoyo de las prácticas ocasionó que se entregaran mejores productos que en semestres anteriores en donde no había este apoyo. Se llegó a este resultado al ver que la calidad de trabajo de los alumnos mejoró a la hora de entregar documentos, en la

presentación de sus manuales, y sobre todo el interés que ellos manifestaron en la materia.

Algunos alumnos comentaban que las prácticas les ayudo mucho, pero que necesitaban más tiempo para practicar y conocer el software que se les facilitó para el desarrollo de sus documentos.

El haber compartido estas actividades con los alumnos, también le sirve a uno para reforzar cada día más los conocimientos adquiridos con anterioridad. Y a no olvidar que todos necesitamos de un interés o apoyo para poder realizar y terminar nuestras metas en la vida. Al término de este proyecto me he dado cuenta que existe una gran relevancia al aplicar prácticas en ingeniería de software.

Agradezco a Microsoft la donación de Microsoft Project y Visio, que fueron de gran utilidad en el curso, para el desarrollo de las prácticas de planeación, diseño e implementación.

BIBLIOGRAFÍA

Álvarez de Zayas, Carlos M.

Fundamentos Teóricos de la dirección del proceso docente Educativo en la Educación Superior Cubana. La Habana: MINED, 1989.

Aubrey ,C. Daniels

Gerencia del desempeño, McGRAW-HILL, 2000.

Borges, Jorge Luis

Diccionario Enciclopédico, Grijalbo, 1995.

Braude J. Eric .

INGENIERÍA DE SOFTWARE "Una perspectiva orientada a objetos", 2003 .

Conallen, J.

Modeling Web Applications with UML, 1999.

Ejiogo, Lem O.

Software Engineering with formal metrics, QED technical Publishing Group

Humphrey, Watts S.

A Discipline for Software Engineering, Addison-Wesley, 1995.

Humphrey, Watts S.

A discipline for Software Engineering, Addisson-Wesley.

Humphrey, Watts S.

Introduction to the Personal Software Process, Addison-Wesley, 2000.

Humphrey, Watts S.

Introduction to the Team Software Process, July .1999

Humphrey, Watts S.

Introduction to the Team Software Process, July .2000

Jones, Gregory W.

Software Engineering, Jhon Wiler & Sons, 1990.

Leach, Ronald J.

Introduction to Software Engineering, CRCpress 2000.

Microsoft Press.

Microsoft Project 1998 paso a paso; McGraw-Hill Interamericana, 1998.

Rumbaugh J , M. Blaha, W. Premerlani, F. Eddy y W. Lorensen.

Object-oriented modeling and design. Prentice- HallBraude,

Silberchatz Abraham, Henry F. Korth.

Database System Concept , McGraw -Hill, Third Edition.

Sommerville, Ian.

Software Engineering, Addison-Wesley.

http://es.tldp.org/Postgresql-es/web/navegable/todopostgresql/part-user.html

http://es.tldp.org/Tutoriales/doc-modelado-sistemas-UML/multiple-html/x219.html

http://support.microsoft.com/?pr=prj98

http://oce.spsu.edu/cseet2002/Anthony-Lattanze.pdf

http://www-306.ibm.com/software/rational/uml/

http://www.acm.org/crossroads/espanol/xrds7-4/onpatrol74.html

http://www.fing.edu.uy/inco/grupos/gris/

http://www.frlp.utn.edu.ar/materias/info2/ManualProject98.htm

http://www.lugmen.org.ar/eventos/san-martin-2004/conferencias/03mdsl/material/mdsl.html

http://www.mercado.com.ar/mercado/mo/lazzati/l25-1097/htm/L25-1097.asp#IV

http://www.sei.cmu.edu/tsp/

http://www.softwaretechnews.com/stn3-4/teamspi.html

APÉNDICE A. FORMAS DEL PRIMER CICLO

En el presente anexo se incluyen algunas formas de TSPi; las cuales se consideraron de uso básico, y por esta razón se decidió incluirlas en el presente trabajo.

Forma Semana del equipo

Equipo _____ Fase _____ Instructor _____

Fecha _____ Ciclo _____ Semana _____

Datos semanales	Actual
Horas dedicadas al proyecto en esta semana	
Horas dedicadas al proyecto en este ciclo hasta esta semana	
Horas dedicadas a las tareas terminadas en esta fase en esta semana	

Tareas de desarrollo efectuadas esta semana	Horas dedicadas	Tamaño del producto	Estado actual del producto
Totales			

Seguimiento de asuntos o riesgos	Estatus

Forma Semana Personal

Nombre _____ Equipo _____ Instructor _____
Fecha _____ Ciclo _____ Semana # _____

Tareas de desarrollo efectuadas	Horas dedicadas	Tamaño del producto	Estado del producto
Totales			

Forma ESTRATegia

Nombre _____ Fecha _____
Equipo Ciclo

Funcionalidades	**Ciclo**	
	1	**2**
Total en cada ciclo		

Forma Registro de Tiempo, REGT

Nombre:	Fecha:
Equipo:	Ciclo:

Fecha	Inicio	Fin	Tiempo de interrupción	Tiempo total	Producto o actividad	Tamaño o Comentarios

Forma Registro de Defectos, REGD

Nombre _____ Fecha _____
Equipo Ciclo

Fecha	Número	Descripción del defecto	Estado del defecto

Total de defectos	Encontrados	Corregidos

Forma Registro de Riesgos RR

Equipo _____ Fecha: _____
Ciclo:

Fecha	
Riesgo o asunto	
Descripción	
Gravedad B,M, A	
Asignado a	
Fecha de resolución	

Fecha	
Riesgo o asunto	
Descripción	
Gravedad B,M, A	
Asignado a	
Fecha de resolución	

Fecha	
Riesgo o asunto	
Descripción	
Gravedad B,M, A	
Asignado a	
Fecha de resolución	

Forma de Solicitud de cambio

Nombre _____ Fecha _____
Equipo Ciclo

Información del componente

Nombre_____Dueño_____
Dirección del respaldo_____

Información del cambio

Razones para hacer el cambio

_____
_____
_____
_____

Beneficio del cambio

_____
_____
_____
_____

Impactos del cambio

_____
_____
_____
_____

Descripción del cambio

_____
_____
_____
_____

Estado del cambio

Aprobado ℜ Rechazado ℜ

Información adicional

_____
_____
_____

Aprobado por

Dueño del producto _____ fecha
Adm. de calidad _____ fecha
MCC _____ fecha

Informe de la Configuración al final del ciclo____

Nombre _____ Equipo _____

Fecha _____ Ciclo _____

Elementos de la configuración del sistema _____ al finalizar el ciclo ___:

(Requerimientos, diseños, código fuente, materiales de prueba y resultados de las pruebas, estándares, etc.)

Elemento de la configuración (dirección o identificador)	Versión	Estado de los cambios propuestos	Estado de la implementación

Forma de informe del estado de la configuración (IEC)

Nombre _____ Equipo _____ Instructor _____
Fecha _____ Ciclo _____ Semana _____

Informe de la actividad de control de cambios

Formas de Solicitud de cambios:	Esta semana	A lo largo del ciclo
Enviadas		
Aprobadas		
Rechazadas		
En reversa (cambio que debe deshacerse)		
En proceso		

Estado de la configuración en esta semana:

Elemento de la configuración	Versión	Estado de los cambios propuestos	Estado de la implementación

Volumen en control de la configuración:

Producto	Esta semana	A la fecha
Páginas		
Modelos		
LOC		
Casos de prueba		
Páginas de material de prueba		
Páginas de resultados de pruebas		
Tablas		
Otros		

Comentarios:

Forma de evaluación del equipo y personal

Nombre: _____ Equipo: _____ Instructor: _____
Rol: _____ No. ciclo: _____ Fecha: _____

Evaluación del equipo:

Evaluación del equipo sobre los siguientes criterios. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Espíritu de equipo	1	2	3	4	5
Efectividad del equipo	1	2	3	4	5
Experiencia adquirida	1	2	3	4	5
Productividad del equipo	1	2	3	4	5
Calidad del proceso	1	2	3	4	5
Calidad del producto	1	2	3	4	5

Evaluación de cada rol:

Para cada rol, evalúe el porciento de trabajo requerido y la dificultad durante este ciclo		
Rol	Trabajo requerido	Dificultad en el rol
Líder de Equipo		
Administrador de Desarrollo		
Administrador de Planeación		
Administrador de Calidad y Proceso		
Administrador de Apoyo		
Contribución total (100%)		

Evalúe la contribución de cada rol. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5
Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Evalúe cada rol de acuerdo al apoyo y ayuda proporcionado. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5
Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Evalúe cada rol de acuerdo a su desempeño. Encierre un número desde el 1 (bajo) hasta el 5 (alto)					
Líder de Equipo	1	2	3	4	5

Administrador de Desarrollo	1	2	3	4	5
Administrador de Planeación	1	2	3	4	5
Administrador de Calidad y Proceso	1	2	3	4	5
Administrador de Apoyo	1	2	3	4	5

Resumen ciclo ___

Equipo _____ Instructor _____
Fecha _____ Ciclo _____

Tiempos por fase

Fases	Resumen de tiempos
Lanzamiento	
Estrategia	
Planeación	
Requerimientos	
Diseño	
IMPlementación	
Prueba	
Tiempo total del ciclo	

Tamaños de los productos

Productos	Tamaños	Unidades

Defectos encontrados por producto

Producto	Número de Defectos
Total de defectos	

Productividad y calidad de los productos

Producto	Tamaño del código / tiempo total de desarrollo	Número de defectos / tamaño de producto específico

APÉNDICE B. FORMAS DEL SEGUNDO CICLO

Forma SEMANA para el equipo

Equipo _____ Fase _____ Instructor _____

Fecha _____ Ciclo _____ Semana # _____

Datos semanales	Actual	Planeadas
Horas dedicadas al proyecto en esta semana		
Horas dedicadas al proyecto en este ciclo hasta esta semana		
Horas dedicadas a las tareas terminadas en esta fase en esta semana		

Tareas de desarrollo	Horas planeadas	Horas actuales	Tamaño	Tamaño planeado
Totales				

Tiempo ganado para esta semana	
Tiempo ganado en este ciclo a la fecha	

Seguimiento de asuntos o riesgos	Estatus

Forma SEMANA personal

Nombre _____ Equipo _____ Instructor _____
Fecha _____ Ciclo _____ Semana # _____

Tareas de desarrollo	Horas planeadas	Horas actuales	Tamaño	Tamaño planeado
Totales				

Forma ESTRATEGIA (ciclo n)

Nombre _____ Fecha _____
Equipo _____ Instructor _____
Parte _____ Ciclo N = _____

Referencia	Funcionalidades/ Requerimiento	Ciclo n	
		Tamaño Estimado	Horas Planeadas
Totales			

Forma de informe del estado de la configuración (IEC)

Nombre _____ Equipo _____ Instructor _____
Fecha _____ Ciclo _____ Semana _____

Informe de la actividad de control de cambios

Formas de Solicitud de cambios:	**Esta semana**	**A lo largo del ciclo**
Enviadas		
Aprobadas		
Rechazadas		
En reversa (cambio que debe deshacerse)		
En proceso		

Estado de la configuración en esta semana:

Elemento de la configuración	**Versión**	**Estado de los cambios propuestos**	**Estado de la implementación**

Volumen en control de la configuración:

Producto	**Esta semana**	**A la fecha**
Páginas		
Modelos		
LOC		
Casos de prueba		
Páginas de material de prueba		
Páginas de resultados de pruebas		
Tablas		
Otros		

Comentarios:

APÉNDICE C. GUIÓN PARA EL DESARROLLO DE PROCESO DE SOFTWARE EN EQUIPOS (Team Software Process- TSPi)

Objetivo		**Guiar al equipo en el desarrollo de un producto de software**
Criterio de entrada		• El instructor guía y apoya a uno o más equipos de 5 estudiantes. • El instructor tiene materiales, facilidades y recursos para apoyar a los equipos. • El instructor guía a los alumnos en la definición de los requerimientos generales del producto.
General		El proceso de TSPi está diseñado para: 1. Desarrollar el producto de software de tamaño pequeño a mediano en 2 o 3 ciclos de desarrollo. 2. Desarrollar un producto más pequeño en cada ciclo. 3. Producir los elementos del producto como especificación de requerimientos, de diseño, plan de pruebas, etc. en cada ciclo.
Semana	**Paso**	**Actividades**
0	**Introducción al curso**	• Entrega de temarios • Introducción a al ingeniería de software • Introducción a procesos • Leer articulo Manuel Cota
1	**Principios de TSPi**	• Principios de TSPi • Equipos y Roles • Mediciones y su papel en TSPi • Leer el cap. 1 y 2 del libro
2	**LAN1**	• Revisar y entender el proyecto a desarrollar • Revisar elementos del proceso de LAN1 • Formar equipos y asignar roles • Leer el cap. 3 del libro y uno de los capítulos 11-15
3	**ESTRA1**	• Revisar elementos del proceso de ESTR1 • Estrategia y estimaciones • Administración de la configuración • Leer el capítulo 4 y apéndice B
4	**PLAN1**	• Producir planes de equipo y de ingenieros para el primer ciclo. • Plan de calidad • Inspecciones • Introducción al Proceso Unificado y UML • Leer el capítulo 5 y el Apéndice C

5	**REQ1**	• Construir diagramas de casos de uso • Detalle de los casos de uso • Leer el capítulo 6
6	**REQ1**	• Hacer el prototipo del sistema • Plan de prueba del sistema
7	**DIS1**	• Construir e inspecciona el diseño del ciclo 1. • Diseño de la arquitectura del sistema • Diagramas de clases y de secuencia • Leer el capítulo 7
8	**IMP1**	• Implementar e inspeccionar el ciclo 1 • Diseño detallado del sistema • Diagramas detallados de clases y de secuencia • Producir el plan de pruebas unitarias y el material de soporte • Leer el capítulo 8
9	**IMP1**	• Implementar e inspeccionar el ciclo 1 • Hacer las pruebas unitarias
10	**TEST1 (Pruebas)**	• Aplicar las pruebas unitarias, de integración y sistema para el ciclo 1. • Producir la documentación para el usuario para el ciclo 1 • Leer el capítulo 9
11	**PM1**	• Conducir la evaluación postmortem y se escribir el reporte final del ciclo1. • Producir la evaluación de roles y de equipos para el ciclo 1. • Leer el capítulo 10 • **Primer examen**
12	**LAN2** **ESTRAT2** **PLAN2**	• Leer los capítulos 16, 17 y 18 • Reorganizar el equipo y asignar roles para el ciclo 2. • Leer los capítulos 11-15 si se cambia de roles • Producir la estrategia y el plan para el ciclo 2. • Evalúar riesgos.
13	**REQ2** **DIS2**	• Ajustar los requerimientos y el plan de pruebas para el ciclo 2. • Hacer un diagrama de estados en los requerimientos • Producir e inspeccionar el diseño de alto nivel para el ciclo 2. • Actualizar el plan de pruebas de integración para el ciclo 2.
14	**IMP2**	• Implementar e inspeccionar el ciclo 2, producir el plan de pruebas unitarias.

	TEST2 (PRUEBAS	• Aplicar las pruebas de unidad, integración y sistema para el ciclo 2. • Producir la documentación de usuario para el ciclo 2.
15	**PM2**	• Conducir la evaluación postmortem y escribir el reporte final del ciclo 2. • Producir la evaluaciones de roles y de equipos para el ciclo 2. • Revisar los productos generados y el proceso utilizado • Identificar las lecciones aprendidas y propuestas para la mejora de proceso • **Segundo examen**
Criterio de éxito		• Un producto terminado con documentación para el usuario • La carpeta de proyecto completa y actualizada. • Evaluaciones de equipos y los reportes de ciclos documentados.

BIBLIOGRAFÍA

W.S. Humphrey, Introduction to the Team Software Process, Addisson-Wesley, 2000.

GLOSARIO

Actividad Es una parte del proyecto que se lleva acabo durante un periodo de tiempo.

Administración de proyectos Proceso de satisfacer la responsabilidad de una terminación exitosa de un proyecto.

Algoritmo Un algoritmo es la especificación, detallada y libre, de un proceso, es decir, un conjunto de pasos que hay que seguir para llegar a un cierto fin.

Análisis de Requerimientos Obtener una declaración escrita, completa de la funcionalidad y desempeño que requiere cierta aplicación.

Archivo Conjunto de elementos de información (en forma numérica o alfabética) almacenados en algún medio adecuado, generalmente en un disco o CD.

Arquitectura de Software Diseño global de una aplicación que incluye su descomposición en partes.

Artefacto Cualquier tipo de datos, que es producida o usada por un desarrollador durante el proceso de desarrollo, se usa para describir el proceso de desarrollo de software unificado (Unifed Software Development Process).

Atributo Representa una propiedad de una entidad. Cada atributo de un objeto tiene un valor que pertenece a un dominio de valores determinado

Bases de datos Se puede utilizar como sinónimo de banco de datos y se refiere a un conjunto de archivos organizados de tal forma que permitan guardar y extraer información útil por medio de la ejecución de programas especiales.

Cascada Proceso de desarrollo de software en el cual primero se recolectan los requerimientos, se desarrolla un diseño, el diseño se convierte en código y después se prueba.

Casos de uso Secuencia de acciones, pueden ser realizadas por el usuario o por la aplicación.

Clase Es un constructor que define la estructura y comportamiento de una colección de objetos.

Código Se llama a un programa que está escrito en **lenguaje de máquina.**

Compilador Programa que sirve como traductor entre un lenguaje de programación y el lenguaje máquina de una computadora .

Computadora Sistema que, a razón de varios millones de veces por segundo, ejecuta los deseos, caprichos de los programadores , así como errores y aciertos.

Diagrama de clases Un diagrama de clases modela la estructura estática muestra del sistema como un conjunto de clases importantes que forman parte de un sistema

Diagramas de secuencia Un diagrama de secuencia muestra las interacciones entre objetos ordenadas en el tiempo.

DLL Permiten crear y definir nuevas bases de datos, campos e índices.

DML Permiten generar consultas para ordenar, extraer datos de la base de datos.

Evento Ocurrencia que afecta a un objeto, iniciando desde el exterior del objeto.

Ensamblador Es un programa que recibe programas escritos en un lenguaje y los traduce a lenguaje de máquina .

Integración Fusión de módulos que van a formar la aplicación.

Interacciones Acción entre dos o más objetos.

Hardware Conjunto de elementos y sistemas electrónicos que forman un sistema de cómputo .

Interfaz La interfaz para un sistema es la especificación de un conjunto de funciones que proporciona el sistema,

Lenguaje de alto nivel Se les llama así los lenguajes de programación que están por "encima" del lenguaje máquina y del el lenguaje ensamblador.

Lenguaje de máquina La única manera de comunicarse con el procesador de una computadora es por medio de un programa directamente ejecutable, mismo que debe estar escrito en este lenguaje.

Lenguaje de programación Nombre genérico que se aplica a cualquier lenguaje disponible para escribir programas para una computadora.

Mantenimiento Reparar y mejorar una aplicación que ya se entregó.

Mensaje En un entorno orientado a objetos no implica el uso de mensaje físico ,si no que hace referencia al intercambio de solicitudes entre los objetos

Métricas Especificación de cómo medir un artefacto de Ingeniería de Software.

Microsoft Project Es una aplicación de gestión de proyectos, que ayuda a organizar un proyecto en diferentes tareas y en un tiempo determinado.

Modelo Es un panorama del diseño desde una perspectiva en particular, como la combinación de sus clases, o su comportamiento manejado por los eventos.

Objeto Es aquel que tiene estado (propiedades),comportamiento (acciones y reacciones) e identidad (propiedad que los distingue de los demás objetos).

Orientado a objetos(OO) Es un método de implementación en el cual los programas son organizados como grupos cooperativos de objetos.

Personal Software Process(PSP) Proceso desarrollado por Watts Humphrey en el Software Engineering Institute, para mejorar y medir las capacidades de la Ingeniería de Software de los ingenieros individuales.

Plan de pruebas del software (PPS) Documentación que establece que partes de una aplicación deberán probarse y el programa de tiempos para realizar esas pruebas.

Procesador Es el encargado de ejecutar las instrucciones de un programa , escritas en lenguaje máquina . dispositivo electrónico diseñado para efectuar miles de operaciones en un segundo.

Proceso Es un conjunto de pasos parcialmente ordenados con el propósito de alcanzar una meta. Conjunto de instrucciones escritas en lenguaje máquina que ya están listas para ser ejecutadas por el procesador. El proceso es el resultado final del complejo conjunto de acciones que comienzan cuando un programador pone manos a la obra y escribe un programa.

Pruebas de humo Pruebas que se aplican en periodos regulares y frecuentes.

Pruebas de integración Proceso de probar el éxito de la fusión de los módulos.

Pruebas de unidades Proceso para probar una parte de una aplicación aislada del resto de la aplicación.

Prueba del Sistema Es la culminación de las pruebas de integración, se realizan mientras la aplicación se ejecuta en su entorno.

Software Es la suma total de programas ,reglas de documentación, documentos y datos necesarios para la operación de un sistema computacional.

Técnicas Es el conjunto de procedimientos que tienen como objetivo un resultado determinado.

Team Software Process (TSP) Proceso de desarrollo por Watts Humphrey para evaluar el desempeño de los equipos de desarrollo de software.

Unified Modeling Language (UML) Lenguaje de modelado unificado o notación gráfica que sirve para expresar los diseños orientados a objetos.