UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CIENCIAS

PRÁCTICAS Y TÉCNICAS PARA EL TRABAJO EN LA INGENIERÍA DE SOFTWARE.

REPORTE DE ACTIVIDAD DOCENTE

QUE PARA OBTENER EL TÍTULO DE:

LICENCIADA EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA:

M I T Z I Z O R A I D A D E L A C R U Z P I M E N T E L

TUTORA:

M. EN C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ

2008

Agradecimientos

A los miembros del jurado: Dra. Hanna Oktaba, Dra. Amparo López Gaona, M. en C. C. Alejandro Alberto Ramírez Ramos y L.C.C Magdalena Manuela Dávila Muñoz, por la revisión del trabajo y sus comentarios.

A mi Maestra Lupita, por todo el apoyo y tiempo que me dedicó para la realización de este trabajo, siempre dispuesta a escucharme, resolver mis dudas, aconsejarme y ayudarme.

A todos mis profesores y entrenadores, que con sus regaños, malas o buenas prácticas, risas, enojos, exámenes, juegos, reuniones, triunfos, fracasos, fiestas, desvelos y clases, me hicieron crecer como alumna, jugadora y persona.

A mis alumnos, con los que platique, aprendí y disfrute de las ayudantias.

A mis amigos Ariadna, Belem, Hugo, Karina y Luis, que siempre estuvieron ahí cuando los necesite en los momentos buenos y malos.

Amilcar gracias……

A mi familia que siempre me apoyo, aconsejo y se preocupo por mí en todo momento.

A mi tía Lolita, que me ha apoyado siempre y que se preocupa por que siga creciendo como persona y profesionista.

A mi abuelita Lupita, que siempre me apoyo, se preocupo por que llegara hasta aquí, que ahora me ve, sonríe conmigo desde donde esta y me dice que siga adelante apoyando a mi familia.

A mi hermana, que ha estado conmigo en todo momento, por enseñarme tantas cosas, por seguir a mi lado a pesar de las dificultades y obstáculos que hemos vivido.

A mis padres, que con tanta dedicación y esfuerzo me han dado todo para poder llegar a este punto de mi vida. Gracias por su amor, sus “regaños”, sus consejos y su paciencia.

A la persona más especial de mi vida, que siempre estuvo, y estará conmigo, apoyándome y transmitiéndome su fe y esperanza, con la cual siempre llegamos a todos lados y que ahora me ve, sonríe conmigo desde donde esta, dándome su bendición para que yo siga adelante: Mi Abuelita Inesita.

Y a Dios que siempre esta conmigo en cada paso, en cada salón, en cada materia, en cada lugar me ayuda y guarda para que salga adelante.

Dedicatoria

A mis padres.

A mi hermana.

A mis abuelitas

A mis amigos.

A mi familia.

A Dios

Gracias Maestra Guadalupe Ibargüengoitia González y Doctora Hanna Oktaba

Índice **Página**

Introducción

Objetivos

Apoyar a la materia de Ingeniería de Software con las prácticas de laboratorio tanto para los alumnos como para los instructores del laboratorio. Las prácticas incluyen técnicas para trabajar en equipo durante el desarrollo de un software.

Dar un manual a los ayudantes de la materia de Ingeniería de Software, según las prácticas elaboradas en este trabajo.

Proporcionar técnicas para trabajar con equipos de desarrollo de software y técnicas para trabajar en un equipo de desarrollo de software.

Antecedentes

En cursos anteriores de Ingeniería de Software ya se contaba con prácticas, que fueron elaboradas por compañeras de la carrera hace algunos semestres, basadas en el proceso y libro de Humphrey TSPi (Team Software Process) en el cual también se basaba el curso. Algunas de las prácticas resultaban tediosas, repetitivas y de poco ayuda para los alumnos y para el curso.

En las prácticas realizadas se busca motivar a los alumnos a trabajar en equipo, proporcionándoles algunas técnicas que pueden usar para conformar un mejor equipo de desarrollo de software.

A lo largo de mi experiencia de alumna y ayudante del curso, me he dado cuenta que para trabajar con equipos y trabajar en un equipo de desarrollo de software, se requiere de cierta experiencia, no solo de programación sino de trabajar con o en equipo. Como es común uno no cuenta con dicha experiencia, ya que eso no se nos enseña en la carrera y muy difícilmente tomamos cursos relacionados con ese tema. Por estas razones mi interés en buscar técnicas para dicho trabajo.

Trabajo

En este trabajo busqué sintetizar y mejorar las prácticas con las que ya se contaba en la materia, basándome en mi experiencia como ayudante del curso durante tres semestres, proporcionando más información en cada práctica sobre el tema, haciendo ejemplos y ejercicios más dinámicos e ilustrativos para los alumnos, eliminando las prácticas redundantes y aumentando prácticas para que los alumnos trabajen más en equipo.

Se revisaron los documentos que a lo largo de varios semestres las profesoras han ido recopilando de los alumnos que cursaron la materia de Ingeniería de Software en la Facultad de Ciencias y en el Posgrado en Ciencias e Ingeniería de la Computación. De dichos documentos se seleccionaron los de mejor calidad y más completos para crear una lista de ejemplos en cada práctica según la fase. De esta forma se elaboró un listado de ejemplos y ejercicios para posteriores compañeros que trabajen como ayudantes de la materia, dando así opción a que ellos elijan sus ejercicios y ejemplos que darán a los alumnos durante las clases de laboratorio.

Tanto las prácticas como los ejemplos se encuentran en el sitio web de la materia (sección de prácticas), el cual fue elaborado por una compañera de la carrera hace aproximadamente dos años y se ha ido completando con diferentes trabajos como este para que tanto los alumnos como los instructores de la materia cuenten con un buen material de apoyo para el curso.

Usando como base de la información en las prácticas, el libro de la materia: Ingeniería de Software Pragmática [IBARGÜENGOITIA y OKTABA], que está disponible para los alumnos en el sitio del curso. En el libro se describe un proceso de desarrollo de software iterativo que se divide en dos ciclos y cada ciclo en ocho fases, elaborando un producto de Software en dos ciclos.

Estructura del trabajo

Este trabajo consta de doce capítulos en los cuales se explica a grandes rasgos cómo se trabaja en la materia de Ingeniería de Software y cómo es que las prácticas ayudan en general a la materia de Ingeniería de software. A continuación se da una breve explicación de los capítulos de este trabajo.

En el capítulo uno se habla de la materia de Ingeniería de Software en la Facultad de Ciencias, cómo se trabaja dentro del salón de clases y el laboratorio, y cómo los alumnos desarrollarán un software a lo largo del curso.

En el capítulo dos se dan técnicas para trabajar en equipo según las características del equipo y para cada rol que se maneja. También se dan técnicas para los instructores de cómo trabajar con los equipos de trabajo dentro del salón de clase.

A partir del capítulo tres y hasta el capítulo once se van desarrollando las prácticas por fase, que se elaboraron en este trabajo. A continuación se da una lista de lo que se presenta en cada práctica:

- Al inicio de cada práctica se establecen sus objetivos, la introducción, una breve explicación del tema o temas de la práctica complementando lo que ve en las clases.
- Un ejemplo del tema que será explicado por el ayudante durante la clase de laboratorio.
- El desarrollo, en el cual se encuentran los ejercicios e instrucciones para ser realizados por los alumnos, los resultados los deben entregar o enviar por correo electrónico, algunos al término de la clase y otros en el transcurso del día o para la siguiente clase según sea la dificultad del tema.
- Las conclusiones, son tres enunciados que se discutirán por todo el grupo según lo hecho en la clase.
- Finalmente una lista de documentos que deben leer y otra con documentos que se deben tener para la siguiente práctica, algunas prácticas contienen una lista de herramientas que los alumnos deben llevar la siguiente práctica.

El capítulo doce es una guía para los ayudantes de la materia, con sugerencias par cada práctica de cómo exponerla ante los alumnos, con las soluciones a los problemas, si es que estos los requieren y consejos que les pueden ayudar a trabajar con los equipos.

Finalmente se dan las conclusiones de éste trabajo.

Capítulo 1. La materia de Ingeniería de Software en la Facultad de Ciencias

El objetivo de este capítulo es presentar información sobre el curso de Ingeniería de Software en la Facultad de Ciencias, presentando la forma en que los alumnos practican los conceptos fundamentales, que las profesoras y ayudantes les presentan por medio de clases teóricas, practicas y teórico-practicas, haciendo uso del libro base de la materia.

En la Facultad de Ciencias de la UNAM se imparte la materia de Ingeniería de Software, materia obligatoria de séptimo semestre en la Licenciatura de Ciencias de la Computación y optativa de otras licenciaturas de la Facultad. El curso es impartido, por la Maestra en Ciencias Guadalupe Ibargüengoitia y la Doctora Hanna Oktaba.

La información para este capítulo se tomó principalmente de los guiones, temarios, problemas, sitios web y prácticas de cuatro semestres anteriores, recopilados durante mi transcurso de alumna y ayudante.

En este capítulo, se explican los objetivos de la materia, los requisitos para que el alumno la curse, la metodología usada por las profesoras, el método de evaluación para los alumnos, lo que los alumnos harán a lo largo del curso y finalmente lo que se espera que los alumnos aprendan en el curso.

1.1 Antecedentes de la materia

La materia de Ingeniería de Software se ha impartido en la Facultad de Ciencias desde principios de los 90's. Actualmente cuenta con información y material para los alumnos que se ha ido perfeccionando y renovando cada semestre que ha sido impartida.

Inicios: El curso en un principio se basaba en libros tradicional de Ingeniería de Software como son: "Software Engineering: A Practioner's Approach" [Pressman], "Software Engineering" [Sommerville] y "Software Engineering Theory and Practice" [Pfleeger].

En dichos cursos los alumnos realizaban prácticas y ejercicios de cada tema tradicional de la Ingeniería de Software, como son la especificación y análisis de requerimientos, el diseño, las pruebas y el mantenimiento del sistema. Ya que al desarrollaba un sistema completo los alumnos no alcanzaban a realizar las últimas fases (pruebas y mantenimiento del sistema y la administración de la configuración).

Por estas razones los alumnos veían las prácticas de Ingeniería de Software aburridas e innecesarias ya que todas las técnicas las realizaban de manera separada y no en un proyecto completo.

Uso de TSPi: Posteriormente en el año 2000, el curso de Ingeniería de Software se baso en el libro de TSPi (Introduction to Team Software Process) [Humphrey]. Dicho libro es una guía para los profesores de Ingeniería de Software, ya que se presenta de manera práctica el trabajo con equipos que desarrollarán un software, tomando en cuenta todas las fases de Ingeniería de Software de forma iterativa, a través de un proceso bien definido, con roles y sus responsabilidades, y guías para conformar un equipo de desarrollo de software efectivo.

En estos cursos el alumno conformaba un equipo de 5 integrantes, con el cual desarrollaba un proyecto de software elegido por ellos, de las dos opciones que presenta el apéndice del libro. A lo largo del desarrollo del curso, el alumno practicaba las mejores técnicas de la Ingeniería de Software, las cuales eran nuevas para ellos. Cabe mencionar el llenado de formas, el uso de métricas, prácticas relacionadas con la calidad del producto, entre otras.

El libro describe un proceso iterativo, con ocho fases por iteración o ciclo de vida. Cada fase requiere por lo menos una semana, para que los alumnos realicen las tareas y logren aprender y comprender la fase. Ya que el semestre consta de 16 semanas se realizaban dos ciclos, en los cuales los alumnos desarrollaban su sistema llevando a cabo dos veces cada fase.

Conforme se fue impartiendo el curso con dicha metodología, las profesoras se dieron cuenta que las fases de Implementación y Pruebas del sistema tomaba más de una semana a los alumnos, por lo cual en el primer ciclo

se comenzaron a utilizar dos semanas para estas dos fases. En el segundo ciclo, los alumnos ya conocían el proceso y contaban con la documentación realizada en el primer ciclo por lo tanto las fases se hacían en menos tiempo para de esta forma terminar el producto en dos ciclos durante un semestre.

Actualmente: Desde el 2006, se trabaja con el libro Ingeniería de Software Pragmática [IBARGÜENGOITIA y OKTABA] (disponible para los alumnos que cursen la materia en el sitio del curso). Dicho libro tiene el mismo enfoque y metodología del libro TSPi, con modificaciones que las profesoras han ido haciendo según su experiencia en la materia y en materias similares de la licenciatura y maestría en Ciencias e Ingeniería de la Computación, adaptándolo de esta forma para los alumnos mexicanos, anexando nuevas técnicas según lo que han aprendido con los alumnos de varios semestres.

Mejoras: En el tiempo de la fase, según la complejidad de la misma. En la documentación que los alumnos deben entregar. En las formas que se llenan. El proyecto es un problema real, redactado por las profesoras y ayudantes, que se entrega a los alumnos al inicio del curso. El libro esta en idioma español y disponible para los alumnos en el sitio del curso, cuenta con ejemplos interactivos en cada tema y un proyecto completo con toda la documentación que se generar.

1.2 Objetivos de la materia de Ingeniería de Software

Desde un principio, el objetivo de la materia ha sido que los alumnos aprendan diferentes técnicas de Ingeniería de software para trabajar en equipo, y así puedan desarrollar de la mejor forma y con la más alta calidad un producto de software en equipo.

Cumpliéndose este objetivo, los alumnos terminan el curso conociendo los métodos, técnicas y herramientas que la Ingeniería de Software proporciona para construir un software de calidad.

A lo largo del curso, se busca que los alumnos desarrollen las habilidades de un rol específico, se organicen para planear sus actividades tanto individuales como en equipo. Siguen un proceso disciplinado, para especificar los requerimientos, el diseño y la implementando de un producto, que se les solicita al inicio del semestre.

Para seguir este proceso, a lo largo del curso los alumnos utilizan técnicas de calidad, como son la revisión, inspección, pruebas unitarias, de integración y de sistema. Logrando que la calidad del producto sea la mejor. Aprenden a recolectar métricas de tiempo, tamaño y defectos en el trabajo personal y de equipo, llenando diversas formas que se les proporcionan en el curso.

1.3 Requisitos para la materia

La materia es de séptimo semestre, por lo tanto los alumnos ya han cursado la mayoría de sus materias obligatorias y cuentan con el conocimiento de programación en un lenguaje orientado a objetos, en particular java y conocimiento de bases de datos. Durante las fases de construcción, los alumnos cuentan con asesoría de los ayudantes para la parte de programación.

Si alguno de los equipos sugiere programar en otro lenguaje orientado a objetos, lo puede hacer siempre y cuando se haga responsable de sus dudas y den una clase al grupo, del lenguaje en el cual eligieron programar.

Ya que esta es una licenciatura de tiempo completo, los alumnos deben contar con tiempo suficiente fuera del salón de clases, para poder emplearlo en la materia, en equipo para realizar juntas semanales e individualmente.

1.4 Metodología de enseñanza

La forma en que los alumnos trabajan es dinámica y divertida, ya que al inicio del semestre se les pide formen equipos de cinco personas (cuatro o seis, según el número de alumnos en el grupo, o las relaciones entre ellos). Se asignan roles, los cuales son a elección de ellos, según aptitudes y gustos. En el equipo cada integrante es también llamado Ingeniero de desarrollo.

Al inicio del semestre se les presenta a los alumnos el curso, con una explicación breve que da la profesora encargada del grupo, se les proporciona el temario de la materia y un guión del curso.

1.4.1 Equipo de Trabajo: Conjunto de alumnos auto asignados para trabajar juntos durante todo el semestre, en el cual ellos se asignan un rol según sus habilidades y gustos, para cumplir los objetivos y metas de equipo e individuales. El equipo puede presentarse de diferentes formas, según el número de integrantes con el que cuenten.

1.4.2 Rol: Conjunto de responsabilidades, asignadas a un integrante de equipo. Varían según las actividades y perspectiva de cada rol. Los roles que se manejan están relacionadas al equipo de trabajo, el desarrollo de software, la planeación y calidad del proyecto, y al ambiente de trabajo.

El uso de roles en los equipos de desarrollo de software, ayuda a tener un equipo bien definido y organizado, ya que saben cuales serán sus tareas principales a lo largo del semestre. El libro contiene un capítulo por rol, en el cual se establecen sus responsabilidades. A continuación se da una breve explicación de cada rol (objetivos, responsabilidades y habilidades):

Líder del Equipo

Objetivo: Dirigir y mantener unido al equipo, asegurar que todos los integrantes reporten y terminen a tiempo las tareas que les corresponden y fomentar el trabajo en equipo.

Responsabilidades: Encargado de construir y motivar al equipo, verifica que cada integrante cumpla con sus responsabilidades y así el equipo cumpla. Dirigirá reuniones semanales generando los reportes de avance del equipo. Resolverá conflictos que se presenten dentro del equipo, con ayuda de sus compañeros. Informara al instructor los conflictos y avances que tenga el equipo.

Habilidades: Ser un líder nato en los grupos, no es conflictivo, es sociable y amistoso y sabe conciliar los intereses y motivar al equipo.

Administrador de Desarrollo

Objetivo: Ser el guía en el diseño y desarrollo del producto de software que entregaran.

Responsabilidades: Dirige al equipo en las fases de desarrollo, siguiendo los estándares propuestos por todos los integrantes del equipo. Aprovecha las habilidades de cada integrante al máximo. Al final integra el trabajo de todos los integrantes, para concretar el software con la mejor calidad.

Habilidades: Tener mucha experiencia con la programación orientada a objetos, conocimiento de herramientas auxiliares, lenguajes y ambientes de programación que pudieran llegar a necesitar.

Administrador de Planeación

Objetivo: Guiar al equipo en la planeación del producto, dar seguimiento al trabajo realizado por cada integrante.

Responsabilidades: Realiza un plan de trabajo para el equipo, asegura que todos los integrantes del equipo se comprometan a cumplir el calendario. Se encarga de recabar las mediciones, registrar y resolver los riesgos que se vayan presentando en el equipo.

Habilidades: Es una persona organizada en todo momento, conciliador e intolerante a las personas que suelen faltar a los compromisos que han establecido.

Administrador de Calidad

Objetivo: Establecer un plan de calidad, para toda la documentación que será generada y para el producto final que se entregará.

Responsabilidades: Establecer y cerciorarse que se cumplan los estándares, para que el trabajo realizado sea uniforme. Revisar los productos generados por sus compañeros, antes de ser entregados al instructor. Establecer un calendario de las revisiones entre colegas, dirigirlas y registrar los defectos encontrados. Se asegura que los defectos sean corregidos.

Habilidades: Sabe identificar las fuentes de defectos, es ordenado y meticuloso en la revisión de productos y se interesa extremadamente en la calidad de todos los productos que son entregados.

Administrador de Apoyo

Objetivo: Ayudar a conseguir y saber como se utilizan las herramientas que requiera su equipo.

Responsabilidades: Generar un plan de la configuración. Realiza la administración del proyecto, controla el cambio de las versiones de cada producto y avisa a sus compañeros cuando un cambio los puede afectar. Proporciona y explica como se usan las herramientas que el equipo solicita.

Habilidades: Ser bueno buscando y explicando nuevas herramientas que resuelvan las dificultades a lo largo del proceso, le gusta manejar las versiones de los productos.

Ingeniero de Desarrollo: Este rol lo llevan a cabo todos los integrantes del equipo.

Responsabilidades: Llenar forma semana personal, asistir a las reuniones semanales del equipo, realizar los productos que le corresponden con la mejor calidad, realizar las reuniones de revisión entre colegas cumpliendo con el calendario correspondiente, realizar sus documentos según el estándar y si sus productos requieren de correcciones las realiza.

Habilidades: Saber programar y conocer las bases y estructuras de datos.

1.4.3 Lenguaje de Modelado Unificado (UML): La herramienta más importante que aprenden a usar los alumnos es UML, para la mayoría de los alumnos es algo nuevo y muy útil.

Esta herramienta es muy útil para los alumnos, ya que es un lenguaje de modelado visual, que se usa en el desarrollo de software para realizar la documentación de un sistema de software. UML cuenta con un conjunto de herramientas, que permite modelar (analizar y diseñar) sistemas orientados a objetos. Algunos de los diagramas que los alumnos aprenden a usar son: Diagrama de casos de uso, de clases, de estados, de secuencias, entre otros.

1.4.4 Guión del Curso: Es un documento que se les proporciona a los alumnos al inicio del curso, en el cual se presenta una lista con la fase que se cubrirá cada semana, las actividades a realizar y los documentos a entregar en cada fase. Este documento incluye las fechas según las semanas del semestre, para que de esta forma ellos sepan como trabajarán a lo largo del semestre.

1.4.5 Proyecto: Al inicio del curso se les da un proyecto para desarrollar un software en equipo. Se les indica que trabajarán durante todo el semestre, realizando la documentación necesaria para su proyecto. Se les indica que el curso consta de dos ciclos, en los cuales tendrán que desarrollar su proyecto y entregarlo funcionando junto con la carpeta en cada ciclo.

1.4.6 Concurso: Para motivar a los alumnos, al inicio del semestre se les informa que al final del curso se hace un concurso, entre todos los equipos (de los dos grupos, si es el caso) de sus proyectos terminados.

El concurso consta en presentar su programa final, el cual todos los alumnos probarán y calificarán. Estas calificaciones se suman y el que obtenga mas alto puntaje gana un premio simbólico, que las profesoras le dan al equipo. En este concurso solo se toma en cuenta el programa terminado, la carpeta no se evalúa, ya que sería muy tardado para los alumnos calificar toda la documentación que ésta contiene. No influye en la calificación final del equipo.

1.5 Método de evaluación

La evaluación consta de tres aspectos y se evalúa el trabajo del alumno individual y de equipo:

1.5.1 Exámenes: Se aplican dos exámenes individuales (al final de cada ciclo). Con esto se busca verificar que los alumnos hayan aprendido los conceptos de Ingeniería de Software, con preguntas sobre los conceptos que se

han trabajando en el curso. El alumno lo resolverá fácilmente, si han trabajado individualmente y en equipo, siguiendo las indicaciones y el libro.

1.5.2 Prácticas: Se realizan y entregan durante la hora de laboratorio (dos horas a la semana), para que no tengan trabajo extra en casa de prácticas, una práctica por clase. Los alumnos aprenden a usar las herramientas que necesita, para su proyecto, a trabajar bajo presión. Las prácticas son individuales, en equipo o ambas y la calificación para el alumno es individual, según lo que entrego él, su equipo y cómo se observó su trabajo en el laboratorio.

Durante las prácticas, se realizan ejercicios sobre los temas que se vieron en la clase, para que practiquen antes de entregar los documentos de su proyecto.

1.5.3 Desarrollo del proyecto: Es la parte más importante tanto del curso como de la calificación final, ya que el objetivo principal del curso es que el alumno aprenda a trabajar en equipo desarrollando un software, en este caso se evaluara lo siguiente:

- Desempeño del alumno en su rol dentro del equipo, es decir se califica individualmente a cada alumno según su participación durante el desarrollo del sistema y la carpeta.
- El Funcionamiento del sistema, debe cumplir con los requerimientos y presentar la documentación correspondiente en la carpeta.
- La evaluación de las dos fases de cierre (primer y segundo ciclo). La fase de cierre, es la última fase de cada ciclo, en la que el alumno se autoevalúa y cada integrante evalúa a sus compañeros de equipo. Se evalúa la participación del rol, el trabajo realizado, la ayuda aportada a los demás integrantes, el espíritu de equipo, entre otras.

1.6 Productos a entregar

A lo largo del semestre los alumnos desarrollan un producto de software, en dos ciclos con ocho fases cada uno, según la metodología del libro *Ingeniería de software Pragmática*. La duración de cada fase en el primer ciclo es de una o dos semanas. En el segundo se realizan varias fases por semana, ya que los alumnos cuentan con experiencia y varios de los documentos que realizaron en el primer ciclo les servirán para las entregas de documentos del segundo ciclo.

En cada fase se explica a los alumnos cuales son los documentos que deben entregar, para qué sirve cada uno, cómo se hacen y cuál es el rol encargado de realizarlos. Los documentos son entregados a las profesoras quienes los revisan y regresan a los alumnos para que se realicen las correcciones correspondientes. Al realizar las correcciones cambia de versión el documento.

A lo largo del curso los alumnos llenan y entregan formas que les ayudan a conocerse más y a llevar un control del tiempo y de los productos que entregan, a continuación se da una lista de las formas:

1.6.1 Formas: Documentos que se proporcionan a los alumnos por los instructores. Los alumnos se encargan de llenarlos según la explicación de los instructores. El objetivo principal del uso de formas es que los alumnos realicen mediciones de tiempo y defectos, para que así logren un desarrollo de software con calidad.

El llenado de formas es parte de la metodología que se sigue en el curso y es un ejercicio que ayuda a los alumnos a crear hábitos de organización, planeación de tiempos, calidad, etc.

Las mediciones se toman de forma individual y se conjuntan en equipo, existen formas de llenado individual y formas de equipo. Los tipos de formas son:

Forma Semana personal: Se llena por persona diariamente, su objetivo es que cada integrante registre los productos que trabajó durante la semana, indicando el tiempo que dedicó, tamaño del producto y estado de avance del producto.

Forma semana del equipo: Aquí se conjunta la información de las formas semanales de cada integrante, que se entregan en las reuniones semanales de equipo, y el administrador de planeación la llena. Se hace y registra un

resumen de avance del trabajo de equipo, con la suma de los tiempos y productos registrados por cada integrante. Se entrega al instructor cada semana. Incluye una tabla para dar seguimiento a los riesgos encontrados en el equipo.

Forma Semana del equipo ciclo 2: La forma anterior cambia en el segundo ciclo, para que se aprovechen las mediciones del primer ciclo, tomando en cuenta el tiempo y tamaño de los productos. Para que se den cuenta cuánto tiempo han empleado y cuánto han ganado.

Forma de Registro de riesgos: El responsable de llenarla es el administrador de planeación, desde el inicio del proyecto. En un principio se detectan y registran los riesgos, conforme se avanza en el proyecto se pueden detectar y registrar más. En la forma semana del equipo se da el seguimiento a los riesgos.

Forma Estrategia: El responsable de llenarla es el administrador de desarrollo. Se documenta la estrategia que se seleccionó por todo el equipo, indicando las funcionalidades del producto que se desarrollarán en cada ciclo.

Forma de Solicitud de cambio: Forma del segundo ciclo. Para solicitar un cambio y se entrega al administrador apoyo. Si se acepta el cambio, en la misma forma se da seguimiento.

Forma de Informe del estado de la configuración: Forma del segundo ciclo. Llenada por el administrador de apoyo, para realizar el informe del estado de la configuración, con los documentos, última versión que se generaron y en qué dirección están almacenados.

Forma Semanal del estado de los cambios: El encargado es el administrador de apoyo. La mantiene disponible y actualizada semanalmente para que todos los integrantes del equipo puedan conocer y consultar el estado de los cambios propuestos.

Forma Registro de defectos: Se llena cuando se realizan las revisiones entre colegas, registrando los defectos encontrados en los documentos, cada revisor llena una forma. Contiene el total de productos que se revisaron y los defectos encontrados en esa revisión.

Forma de Evaluación del equipo y personal: Cada integrante se autoevalúa y evalúa a cada uno de sus compañeros de equipo, haciendo una revisión de los objetivos que se establecieron al inicio del ciclo, si se cumplieron o no y cómo fue el trabajo para cumplirlos.

Forma de Lecciones aprendidas y sugerencia de mejoras: Los responsables de llenarla son el líder del equipo y administrador de desarrollo. El equipo hace una revisión de lo que mejor les funcionó a lo largo del ciclo, de las mejores prácticas, las experiencias que les parecieron exitosas y los problemas a los que se enfrentaron para no repetirlos. Se proponen cosas para mejorar o cambiar para el siguiente ciclo.

Forma de Informe de mediciones: Los encargados son el administrador de calidad y de planeación. Se resumen las mediciones que se hicieron durante las fases, los tiempos en efectuar cada fase, el tiempo total en horas del ciclo, los tamaños de cada producto, el total de defectos que se encontraron, la productividad y calidad de los productos.

En el sitio del curso están disponibles las formas con ejemplos de cómo deben ser llenadas, los alumnos pueden darles el formato según su estándar de documentación.

1.6.2 Productos a entregar en todas las fases: Al inicio del curso (en el guión) se da una lista de los documentos que los alumnos entregarán cada semana y el responsable del documento según su rol, también esta disponibles en el sitio del curso.

Agenda: Se realiza al final de en las reuniones semanales. Es una lista con las actividades que se deben realizar para la siguiente reunión.

Minuta: Se realiza durante las reuniones semanales, con la lista de los puntos que se revisaron, el informe de los avances de cada integrante. Se entrega la forma semana personal y el registro de los problemas y las propuestas de solución que se encontraron.

Productos de la fase de Lanzamiento: Lo que se hace en esta fase, es organizar el equipo, definir objetivos y roles, establecer estándares para los documentos e identificar los riesgos a los que se pueden enfrentar durante el proyecto.

Productos	Responsable
Estándar de documentación.	Administrador de calidad.
Forma de Registro de riesgos	Administrador de planeación.

Productos de la fase de Estrategia: Se plantea la estrategia en dos ciclos, se hace una lista de necesidades del proyecto, se distribuyen en cada ciclo, de tal forma que en el primer ciclo se entregue una parte funcional del sistema y en el segundo el proyecto completo.

Productos	Responsable
Forma Estrategia	Administrador de desarrollo.
Repositorio de productos del equipo	Administrador de apoyo.
Plan de configuración.	Administrador de apoyo.
Forma del Informe del estado de la configuración	Administrador de apoyo.
Forma del Informe semanal del estado de los cambios	Administrador de apoyo.
Solicitudes de cambios	Cada Ingeniero de Desarrollo que solicite el cambio.

Productos de la fase de Planeación: Se planean las actividades del equipo, indicando cuándo se harán, qué se hará y quién lo hará.

Productos	Responsable
Plan del equipo.	Administrador de planeación.
Forma de Registro de defectos.	Administrador de calidad

Productos de la fase de Especificación de Requerimientos: Se inicia la construcción del producto, se interactúa con el cliente, se especifican claramente los requerimientos, se muestran al cliente un prototipo de interfaz de usuario, para que se verifique si es lo que quiere. Los ingenieros de desarrollo comienzan a trabajar con los diagramas de UML.

Productos	Responsable
Definición del problema	Administrador de desarrollo y Cliente
Glosario de términos	Administrador de desarrollo y Cliente
Especificación de requerimientos del software	Administrador de desarrollo
Plan de prueba del software	Administrador de desarrollo

Productos de la fase de Diseño: Se describen las partes de las cuales se va a componer el sistema, definiendo la arquitectura.

Productos	Responsable
Especificación del ambiente de implementación	Administrador de desarrollo y el de Apoyo
Estándar de documentación del diseño	Administrador de desarrollo y el de Apoyo

Arquitectura con diagrama de paquetes	Administrador de desarrollo
Diagrama de distribución	Administrador de desarrollo
Diagrama de clases	Administrador de desarrollo
Diagramas de secuencia	Administrador de desarrollo
Diagrama de estados	Administrador de desarrollo
Plan de prueba de integración	Administrador de desarrollo

Productos de la fase de Construcción: Se comienza a programar y se realizan las pruebas unitarias (cada ingeniero de software a su código).

Productos	Responsable
Diagramas detallados de clases	Todos los Ingeniero de desarrollo
Código para las clases y paquetes	Todos los Ingeniero de desarrollo
Plan de prueba unitaria de cada clase	Todos los Ingeniero de desarrollo
Pruebas de caja blanca y negra	Todos los Ingeniero de desarrollo

Productos de la fase de Prueba del sistema: Se integra el sistema, se prueba que cumpla con los requerimientos establecidos por el cliente y se elaboran los manuales.

Productos	Responsable
Reporte de las pruebas en la forma de Registro de defectos	Administrador de desarrollo y de apoyo
Informe de la prueba del sistema	Administrador de desarrollo e Ingeniero de desarrollo.
Manuales de usuario, Manual de operación o instalación, Manual de desarrollo y mantenimiento	Administrador de desarrollo e Ingeniero de desarrollo

Productos de la fase de Cierre: Se evalúa el desempeño del equipo, se registran las lecciones aprendidas y las sugerencias de mejora.

Productos	Responsable
Forma de evaluación del equipo y personal	Líder del equipo
Lecciones aprendidas y sugerencias de mejoras	Líder del equipo y Administrador de desarrollo
Informe de mediciones	Administradores de calidad y de planeación
Forma de Informe del estado de la configuración	Administrador de apoyo.

En el segundo ciclo, los alumnos utilizan los documentos que entregaron en el primer ciclo, completándolos según su estrategia, con los documentos que les hacen falta para el segundo ciclo.

1.7 Bibliografía

La bibliografía para el curso, por el momento consta de seis libros los cuales los alumnos pueden consultar en la biblioteca de la Facultad de Ciencias o en la biblioteca central.

El libro principal, en el cual se basa el curso es *Ingeniería de Software Pragmática*, el cual se encuentra disponible para los alumnos inscritos en la materia, en el sitio del curso **http://victoria.fciencias.unam.mx/cursois.**

A continuación se da una lista de los otros libros y su clasificación:

- Humprey W., *Introduction to Team Software Process*, SEI Series in Software Engineering, Addison Wesley, 2000. Disponible en: Biblioteca Central y en la facultad de Ciencias
    - No. de sistema 000854349
    - Clasificación QA76.758 H845
    - ISBN 020147719X
- Humprey W., *Introduction to Personal Software Process*, SEI Series in Software Engineering, Addison Wesley, 1997. Disponible en: Biblioteca de la facultad de Ciencias:
    - No. de sistema 000752111
    - Clasificación QA76.758 H84 1997
    - ISBN 020158097
- Booch G., Rumbaugh J., Jacobson I., "*The Unified Modeling Language. User Guide*", Second Edition, Addison-Wesley, 2005. Disponible en: Biblioteca de la facultad de Ciencias:
    - No. de sistema 000831691
    - Clasificación QA76.76 D47 B664
    - ISBN 0201571684
- Jacobson I., Booch G., Rumbaugh J. *The Unified Software Development Process*, Addison Wesley, 1999. Disponible en: Biblioteca Central y en la facultad de Ciencias:
    - No. de sistema 000837787
    - Clasificación QA76.76D47 J335
    - ISBN 0201571692
- Fowler M. Scott K. *UML gota a gota.* Pearson 1999. Disponible en: Biblioteca Central y en la facultad de Ciencias:
    - No. de sistema 000859817
    - Clasificación QA76.9035 F6918
    - ISBN 968-444-364-1

1.8 Herramientas de trabajo

Los alumnos cuentan con el Taller de Ingeniería de Software, con impresora, equipos de cómputo con el software necesario para que elaboren su documentación, un tapanco con mesas y sillas, en el cual pueden realizar juntas y trabajar en la documentación y todo lo relacionado con su proyecto.

Los alumnos cuentan con la asesoría de ayudantes dos días a la semana durante la hora de clases, fuera del horario de clases por medio de correos electrónicos, si es necesario se puede hacer una cita para que las resuelvan las dudas.

Cuentan con el material didáctico disponible en un sitio web del curso (**http://victoria.fciencias.unam.mx/cursois**). Para que los alumnos puedan acceder al sitio y usar todas las herramientas que en éste se presentan, deben estar inscritos en la materia y dados de alta en el sistema, lo cual se hace la segunda semana de clases, se les da un nombre de usuario y contraseña para que puedan acceder al sitio como alumnos del curso. A continuación se explica brevemente con lo que cuenta el sitio:

Apoyos al libro de Ingeniería de software Pragmática: Ejemplos de toda la documentación de un proyecto completo, artículos de revistas relacionados a cada fase, para que los alumnos se informen de cómo avanza la Ingeniería de Software en la vida real y cómo tanto papeleo si sirve en las grandes empresas.

Prácticas: Lista de documentos, que utilizarán en cada práctica y de las actividades que deben realizar antes de cada práctica y ejemplos para las prácticas.

Herramientas: Que los alumnos utilizaran durante el curso, cada herramienta cuenta con un tutorial con una breve reseña del software, una liga para bajar el programa, explicación de cómo instalarlo y de cómo usarlo.

Ligas de interés: Lista de varios sitios que les pueden ayudar a encontrar más información sobre los temas relacionados con la Ingeniería de Software.

Curso de IS: Para estudiantes e instructores, a la cual solo se puede ingresar con su nombre de usuario y contraseña.

- **Estudiantes:** Los alumnos pueden revisar sus calificaciones, guión, temario, enunciado del proyecto y libro completo de Ingeniería de Software Pragmática.
- **Instructores:** Los instructores pueden acceder al libro, subir el temario, guión o proyecto, dar de alta, baja o cambiar los datos de los alumnos, ver y asentar las calificaciones de las prácticas, exámenes y ciclos.

1.9 Al final del curso

Cuando finalice el curso se espera que los alumnos hayan aprendido las prácticas de la Ingeniería de Software.

Que los alumnos hayan trabajado como un equipo exitoso, siguiendo cada alumno el rol que le correspondía dentro de su equipo de trabajo.

Que los alumnos hayan logrado desarrollar un producto de software, de manera incremental en dos ciclos, con todos los requerimientos y necesidades que el problema les exige y la documentación completa y consistente que se les pide, de forma electrónica y física generando así una carpeta por equipo para su proyecto.

Que los alumnos hayan aprendido a utilizar las herramientas que se les proporcionaron a o largo del curso, como son el software con el que realizaron su documentación (diagramas, planeación, estrategias, etc.).

Capítulo 2. Técnicas para el trabajo en la Ingeniería de Software

En este capítulo se darán técnicas para que se logre conformar un buen equipo de desarrollo de software. Información de cómo llevar a cabo reuniones, cómo debe ser la comunicación entre ellos, cómo dividir las tareas, cómo evaluarse, cómo confrontar los problemas, etc.

La primera parte de este capítulo, esta basada en el libro "Formación de grupos de desarrollo de software" [Josefina]. Dicho libro habla de técnicas de trabajo con equipos que desarrollan software utilizando la psicología social.

Los equipos que se forman en la materia son de cinco personas, en algunos casos de cuatro o seis, dependiendo el número de alumnos que hay en el grupo. Son cinco integrantes, según lo propuesto en [Humprey], cada uno tiene un rol dentro del equipo, los roles:

- Líder del equipo LE
- Administrador de desarrollo AD
- Administrador de planeación AP
- Administrador de calidad AC
- Administrador de apoyo AA

Los alumnos eligen el rol, según su personalidad, conocimientos y gustos. En el libro [IBARGÜENGOITIA y OKTABA, capítulos 11 al 15] se explica lo que debe cumplir el encargado de cada rol, los documentos y actividades que debe realizar, a lo largo de cada ciclo. Como alternativa los alumnos pueden cambiar de rol en el segundo ciclo.

2.1 Psicología Social

La Psicología social, se interesa por las personas que ocupan un lugar en un equipo. Nos dice que toda persona siempre ocupa un lugar en un equipo cualquiera, la sociedad, ya que desempeña un conjunto de papeles dentro de la sociedad en que vive. De esta forma se ve a la sociedad como un equipo, que a su vez se divide en equipos menores (la familia, trabajo, amigos o compañeros, etc.). Por estas razones la Psicología social define a la sociedad como un conjunto de equipos que están interconectados entre sí.

Dentro de estos equipos entra el equipo de la escuela, los equipos de cada materia, en particular el de Ingeniería de Software. Es un equipo pequeño, en el cual se ayuda a los alumnos a aprender a trabajar en equipo.

Los alumnos integran durante todo un semestre, posiblemente su primer equipo formal de desarrollo de software, en el cual aprenderán a convivir, discutir, tomar dediciones, estresarse, etc. Desarrollando un software, con toda lo necesario para que sea un software de calidad.

Dentro de la psicología social, existen varias definiciones de equipos, estas pueden depender del número de integrantes o de los intereses por los cuales se formó el equipo. A continuación se da una definición de equipo.

"Equipo: Es un grupo que consta de un determinado número de miembros quienes, para alcanzar un objetivo común, se unen durante un periodo de tiempo que se extiende en un proceso relativamente continuo de comunicación e interacción y desarrollan un sentimiento de solidaridad. Son necesarios un sistema de normas comunes y una distribución de tareas según una diferenciación de roles específica de cada grupo" [Josefina].

Tomando la definición anterior y aplicándola a la Ingeniería de Software, daré una definición para los equipos que se forman en la materia:

Equipo de Ingeniería de Software: Grupo que puede ser conformado de cuatro a seis integrantes, que buscan alcanzar metas en común que ellos mismos se plantean, trabajan juntos durante un semestre desarrollando un software y aprendiendo las mejores prácticas de Ingeniería de Software. Plantean sus propias reglas y normas

dentro del equipo, llevando un control total de sus actividades y tiempo durante su trabajo en equipo, se asignan roles específicos a cada integrante y así cada uno sabe sus tareas específicas de rol.

Dentro de la psicología social los equipos se clasifican según su tamaño en dos tipos:

Macro-equipos o equipos grandes: Tienen más de treinta integrantes. Por ejemplo los equipos formados por todos los alumnos de la carrera de Ciencias de la computación.

El que nos interesa para la materia de Ingeniería de Software es el siguiente:

Micro-equipos o equipos pequeños: Pueden tener hasta treinta integrantes. Los equipos que pueden existir son de dos personas (pareja), de tres (tríada) y así sucesivamente hasta llegar a los equipos de treinta. Como sabemos en cualquier equipo existen problemas y este no es la excepción, a partir de tres personas comienzan a aparecer los celos y las coaliciones, en el equipo de tres integrantes puede darse el caso de que se forme una coalición de dos personas contra una.

Algunos autores indican que el número idóneo para un equipo pequeño es de siete integrantes y otros creen que es de cinco, en un equipo de treinta suelen surgir subgrupos mas pequeños.

Según la psicología social los equipos tienen ciertas características, a continuación se explican las importantes para los equipos de Ingeniería de Software.

2.2 Características de los Equipos de Ingeniería de Software

Interacción reciproca: Debe haber convivencia y relación entre todos los integrantes del equipo, de tal manera que se llegue a formar un equipo con buena comunicación, en el cual se apoyen mutuamente en el trabajo que deben entregar semanalmente. Esto ocurre, ya que todos los integrantes del equipo conocen las tareas de cada uno y si así lo requieren pueden solicitar ayuda de alguno de sus compañeros.

Existencia de objetivos, metas, actividades y sentimientos compartidos: En cualquier equipo la existencia de objetivos, metas, actividades y sentimientos es la parte más importante, ya que gracias a éstos se logra tener y mantener al equipo unido.

Dentro del equipo de Ingeniería de Software los alumnos plantean sus objetivos tanto personales, como de rol y del equipo con respecto al proyecto a desarrollar. Para que los alumnos definan dichos objetivos existe una práctica específica, al realizar esta práctica los alumnos comienza a integrarse como equipo, se van conociendo entre ellos y saben cuales son sus prioridades sin dejar de considerar al equipo como lo más importante.

Normas: Es muy importante que dentro del equipo existan normas, ya que los integrantes no deben comportarse de cualquiera forma. Se deben comprometer a ciertas normas, que entre todos plantean. Se sugiere que al inicio se haga una lista de las normas que se seguirán dentro del equipo, para que éste funcione correctamente. Dentro de las normas se deben establecer algunas que controlen la conducta de cada integrante, premiando y castigando las buenas y malas conductas. Al terminar de crear la lista de normas, todos los integrantes del equipo deben estar enterados y de acuerdo con ellas, con los castigos y premios, para que así funcionen correctamente.

Estabilidad y duración: En el caso del equipo de Ingeniería de Software, la duración es de un semestre. La estabilidad se va dando conforme el equipo avanza en sus tareas, tomando en cuenta los objetivos, metas, tareas y normas que se plantean al inicio y sobre todo de cómo es la interacción entre todos. Al final del primer ciclo tienen la oportunidad de evaluarse así mismos, a cada integrante de su equipo y sobre todo de evaluar sus métodos de trabajo, de esta forma se pueden dar cuenta la estabilidad de su equipo y si lo requieren, hacer cambios.

Conciencia de grupo: Cuando todos los integrantes del equipo se ven como un “nosotros”, presumen su equipo a los demás, por decirlo de alguna forma, demostrando así que son un equipo unido, capaces de ayudarse entre sí y resolver juntos cualquier dificultad que se les presente.

Reconocimiento como tal: Esto es el complemento del punto anterior, el equipo es reconocido por las personas del exterior, por otros equipos, por el profesor, ya que muestran a las demás personas que son un equipo unido, capaz de lograr sus objetivos y metas, que pueden superar los obstáculos que se les presente.

2.3 Estructuras de los equipos

Para que un equipo logre una buena estructura, se requiere de una buena organización, cada integrante del equipo debe tener uno o varios roles dentro del mismo y asumir sus responsabilidades y tareas para lograr las metas propuestas. Deben tener iniciativa de colaboración, para con los demás integrantes del equipo y una buena comunicación entre todos los integrantes del equipo.

Si un equipo de Ingeniería de Software toma en cuenta los tres puntos que a continuación se explican, logrará ser un equipo exitoso durante el curso.

Roles

En la vida diaria todos tomamos diferentes roles, ya sea como parte de una familia, como parte de algún grupo social, con nuestro grupo de amigos, etc. Estos roles pueden ser impuestos o tomados por nosotros, de cualquier forma debemos cumplir con las tareas que se nos asignan según el rol.

En la materia de Ingeniería de Software existen cinco roles, uno por integrante y el rol que todos deben tomar que es el de ingenieros de desarrollo. Estos roles son asignados al inicio de cada ciclo, por los mismos integrantes de cada equipo, después de haber leído y comprendido que habilidades requiere cada rol, las tareas que debe realizar y los gustos individuales. En el segundo ciclo pueden o no cambiar de rol.

Una parte muy importante dentro de la estructura de los equipos, son los roles que cada integrante lleva a cabo dentro del equipo. Cada rol tiene tareas específicas ya establecidas para que el equipo funcione correctamente y así todos cumplan con sus objetivos tanto personales como individuales.

Colaboración

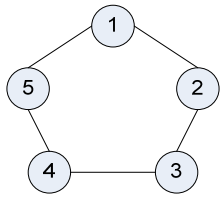
Dentro de la estructura del equipo, entra la colaboración de cada integrante de equipo, la cual hace que el equipo sea más unido y actúe como tal, llegando juntos a las metas propuestas.

Comunicación

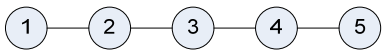
Esta podría ser la parte más importante del éxito de un equipo. Sin comunicación los integrantes del equipo no saben lo que sus compañeros están haciendo. La comunicación ayuda a los equipos a discutir las ideas individuales y llegar al mejor acuerdo, para que el equipo funcione, trabajando de la mejor forma. También enriquece y refuerza la confianza del equipo en cada uno de sus integrantes.

A continuación se dan ejemplos de diferentes formas de comunicación en equipos de cinco personas:

- **Comunicación circular:** La comunicación sólo se da entre vecinos y cada integrante tiene la misma oportunidad de comunicación con su vecino. Como se muestra en la siguiente gráfica: el 1 se puede comunicar únicamente con sus vecinos que son 5 y 2, etc.

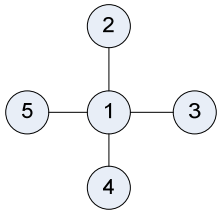


- **Comunicación de cadena:** No es muy conveniente y no se recomienda, ya que existen desigualdades. Los integrantes que se encuentren en cada extremo de la cadena solo se pueden comunicar con un integrante del equipo, en cambio los que están en la parte central pueden comunicarse con dos integrantes diferentes cada uno.

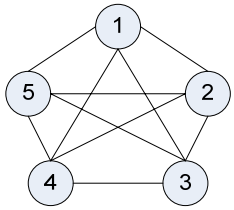


- **Comunicación de rueda:** Cuatro de los integrantes solo se pueden comunicar con un solo integrante y éste integrante tiene la oportunidad de comunicarse con los otros cuatro. Puede ser usada, para cuando

hay un encargado principal de una tarea y este la reparte a los demás. En la gráfica, vemos que 1 es el que se puede comunicarse con los otros integrantes.



- **Comunicación de "all channel":** Variante de la comunicación en círculo, los integrantes se pueden comunicar con todos los demás, como se puede ver en la gráfica, y así resolver cualquier problema de manera eficaz y rápida de forma organizada. Es la forma más usada por los alumnos, cuando éstos están trabajando en equipo. Como se muestra en la siguiente gráfica todos tienen canal de comunicación con todos.



Existen diferentes estructuras de equipo, formal o institucional y equipos formados por estructura afectiva o informal, a continuación se explica la diferencia entre ellos.

- **Estructura formal o institucional:** Equipos formados por lo general en las empresas, según los criterios técnico-administrativos, para realizar diferentes actividades según la tarea que se realizará, el tiempo de vida del equipo, entre otros aspectos. Su relación es únicamente para y en la empresa, realizando las tareas que les fueron asignadas como equipo. Dentro de estos grupos pueden o no existir roles asignados por alguien, pero lo que si existe son roles que se dan a conocer como son el más listo, el chistoso, el enojón, etc.
- **Estructura afectiva o informal:** Equipos basados en los gustos, simpatías, rechazos, atracciones personales, etc. de cada uno de los integrantes del equipo. Los equipos se van formando conforme las personas se van conociendo y conviviendo. Es fácil que los integrantes logren distinguir las habilidades de cada uno y de esta forma si se requiere asignar roles. Es común que se formen en el ámbito escolar, a nivel secundaria, preparatoria, licenciatura, etc.

En el curso de Ingeniería de Software, para los equipos que se forman, ellos seleccionan las personas, la única restricción es que sean cinco integrantes por equipo y que estén concientes que será el equipo con el que trabajarán durante todo el semestre. Tomando las estructuras citadas en la parte de arriba, los equipos que se forman en la materia, tendrían una estructura afectiva o informal con roles establecidos y todas las características antes mencionadas. En ocasiones también existe la posibilidad que los alumnos no se conozcan entre sí y formen un equipo de 4 o 6 integrantes, en este caso puede ser una estructura formal o institucional, no tan estricta como la de una empresa.

2.4 Fomentar el trabajo en equipo en el curso

Evaluación: En el curso, a los alumnos se les califica tanto de forma individual como en equipo, esto ayuda a que el equipo se fortalezca como tal. Los alumnos se preocupan por que las tareas de equipo sean correctas y de calidad, de esta forma se fomenta más el trabajo en equipo.

Normas: Las normas que se plantean de forma externa para los equipos son:

- **Entregas:** En cada fase, los equipos entregan sus productos. El líder de cada equipo es el encargado de realizar esta entrega, debe reunir los documentos que realizaron cada uno de los integrantes y entregarlos a la profesora.
- **Recibir Documentos:** Se les indica a los alumnos el día que se les regresaran sus documentos revisados. Se le entregan al líder, quien revisa si requieren de correcciones o no.
- **Prácticas:** En las prácticas se les pide a los alumnos que trabajen en equipo y de forma individual según sea el caso, los ayudantes especifican las tareas que deben realizar individualmente o en equipo.
- **Normas de equipo:** Este punto es muy importante, se les pide a los alumnos que establezcan sus propias normas de equipo, ya que esto les ayudará a tener más confianza entre ellos y saber los límites que hay dentro del equipo. Se realizan en una práctica, que es benéfica y refuerza al equipo. Para que estas normas realmente ayuden al equipo, se debe asegurar en ellas que el equipo las acatará sin problema alguno, que el comportamiento de los integrantes pueda ser pronosticado según las normas establecidas, evitar conflictos ya sean personales o de equipo entre los integrantes y por supuesto que las normas reflejen los valores fundamentales del equipo.

2.5 Evaluación

Existen diferentes tipos de evaluación dentro de un equipo y para el equipo. En cada evaluación se debe dar una pequeña explicación de para qué servirá dicha evaluación. A continuación se nombran las básicas y se dan ejemplos de preguntas para dichas evaluaciones en la formación de grupos de desarrollo de software:

Evaluación al final de cada reunión de equipo: Servirá para que las reuniones del equipo sean mejores cada vez, y si existiera alguna opinión de mejora de las reuniones en esta evaluación se dirá:

- ¿Qué te pareció la reunión? ¿A qué lo atribuyes?
- ¿Señala el mayor acierto que encontraste en la reunión? ¿Por qué?
- ¿Señala lo que te disgustó y el por que no lo hablaste en la reunión?
- ¿Que sugieres para las próximas reuniones?

Evaluación del equipo en las prácticas: Para el ayudante, para que éste se auto evalúe y evalúe al los equipos, para mejorar las clases, información y ejercicios en las prácticas:

- ¿Cómo funcionan cada equipo en las prácticas?
- ¿Cuál es la reacción de cada equipo si se les presenta una nueva situación?
- ¿Los equipos esperan las indicaciones para realizar las tareas?
- ¿Los integrantes de los equipos esperan la reacción del líder ante algo para que ellos actúen?
- ¿Cómo se comporta cada equipo si su líder esta ausente?

Evaluación de las necesidades del equipo: Para saber qué es lo que necesita el equipo para seguir creciendo como tal. Aquí se toman diferentes puntos a evaluar dentro del equipo, como son:

Motivación

- ¿Los temas que se tratan en el equipo te hacen sentirte motivado?
- ¿Tu equipo te hace sentir motivado a seguir con el?
- ¿Alguno de tus compañeros en especial, hace que te motive el seguir en el equipo?
- ¿Existe alguna otra cosa o factor que te motive dentro de tu equipo?

Objetivos: Ya que los objetivos que se plantean son de forma individual, por rol, por equipo y para el proyecto, las preguntas son relacionadas con los objetivos de equipo y del proyecto.

- ¿Los objetivos del proyecto/equipo fueron definidos con claridad? ¿Son realistas? ¿Te parecieron acertados?
- ¿Crees que todos los integrantes del equipo estuvieron de acuerdo con todos los objetivos?
- ¿El equipo toma en cuenta los objetivos y los mantiene presentes durante la realización de las tareas?
- ¿Crees que los integrantes de tu equipo y el equipo en general está conciente de adónde van?
- ¿Las técnicas que están practicando para cumplir con los objetivos, crees que sean las indicadas?

Trabajo: Se evaluará cómo ha sido el trabajo de equipo en lo que lleva de vida (prácticas, clases, etc.) Se puede realizar al terminar una fase, un ciclo, después de un mes.

- ¿Cómo juzgas el trabajo de tu equipo?
- ¿Crees que las actividades que hay dentro de tu equipo, son las indicadas y las necesarias para lograr las metas planteadas?
- ¿Crees que se pierde el tiempo en actividades no relacionadas al proyecto?
- ¿Participan todos los integrantes del equipo en las tareas?
- ¿Se comparte la responsabilidad del trabajo por parte de todos los integrantes?
- ¿Crees que las funciones de cada integrante están bien delimitadas?
- ¿Dentro de tu equipo, existe la oportunidad para que todos los integrantes participen?

Comunicación: Ayuda a que la comunicación sea mejor, sin comunicación no existe equipo.

- ¿Crees que existe buena comunicación dentro de tu equipo?
- ¿Existe falta de comunicación de alguno de los integrantes de tu equipo?
- ¿Con la comunicación que existe en tu equipo, crees que se cubra la interacción que se necesita?
- ¿Hay oportunidades para que todos los integrantes de tu equipo se comuniquen?

Libertad: Los alumnos se dan cuenta qué tan libres son dentro de su equipo y les ayudará a tomar decisiones posteriores.

- ¿Es democrático ó autocrático tu equipo? **Democrático:** La toma de decisiones es por democracia. **Autocrático:** El líder decide sin importarle la opinión de los demás.
- ¿Crees que el esquema de tu equipo es rígido?
- ¿Se delega la autoridad?
- ¿Si las normas no se siguen, si se incurre en alguna falta o incumplimiento existen sanciones?
- ¿Se les da la oportunidad de discutir si algo no les parece?

Auto evaluación: Ayuda a los alumnos a conocerse a si mismos y saber qué papel es el que desempeñan en su equipo.

- ¿Cómo es tu participación en tu equipo?

- ¿Cuál es el papel que desempeñas con mayor frecuencia? Para dar esta respuesta puedes guiarte en la tabla 1 marcando la frecuencia que creas que aplica en ti

Comportamiento	Siempre	Con frecuencia	Alguna vez	Nunca
Iniciador				
Animador				
Orientador				
Colaborador				
Alborotador				
Conciliador				
Activador				
Avaluador				
Sintetizador				
Divagador				
Dominador				
Obstructor				
Informador				
Ausente				
Agresor				
Adulador				
Intransigente				

Las evaluaciones anteriores son sugerencias y ayuda para reforzar al equipo, a los alumnos e instructores, no es forzoso que las realicen. A continuación se listan las evaluaciones que son forzosos para los equipos, en la fase de cierre (final de cada ciclo) los alumnos realizan las evaluaciones y en la práctica que deben evaluar a sus instructores.

- **Evaluación a los instructores**
- **Evaluación del proyecto**
- **Evaluación de los integrantes del equipo**

Estas evaluaciones ayudan a los alumnos a conocer un poco mas de los otros integrantes de su equipo y a hacerles ver lo que les está fallando. Con respecto al proyecto les ayuda a tomar las mejores prácticas para el segundo ciclo o para su vida laborar.

Capítulo 3. Introducción a la Ingeniería de Software y al trabajo en equipo

Son las primeras prácticas, solo se introduce a lo que es la Ingeniería de Software y al trabajo en equipo, proporcionándoseles una serie de problemas, no deberán programar ni usar la computadora, son para que se conozcan y se integren como equipo.

3.1 Práctica 1

Esta práctica tiene como objetivos, que los alumnos conozcan la historia, ventajas y otros aspectos de la Ingeniería de Software.

Para esto se les proporciona un artículo llamado “Ingeniería de Software”, en el cual se describen diferentes aspectos de la Ingeniería de Software, con los cuales los alumnos comenzarán a interesarse por la materia, realizando su primer trabajo en equipo.

3.2 Práctica 2

Esta práctica tiene como objetivos, que los alumnos formen equipos dentro del grupo, que conozcan la importancia de trabajar en equipo, motivar la reflexión personal sobre las habilidades de cada uno, para escoger un rol dentro de un equipo y finalmente que hagan una tarea en equipo.

Para esto se les explican las características de un equipo de trabajo y que deben tomar en cuenta para resolver un problema que se les da.

3.3 Práctica 3

Esta práctica tiene como objetivos, que los alumnos integren su equipo definitivo y estén concientes de querer trabajar con el durante todo el semestre, poniendo nombre, logo y lema al equipo, fijando día y hora de reuniones semanales, que se conozcan más entre ellos, solucionando los problemas que se les presentaran a lo largo de la práctica, siguiendo las normas que ellos mismos establecerán y asignando roles a cada integrante, que tengan conciencia de leer y comprender las instrucciones que se les dan.

Se les explican diferentes problemas que pueden llegar a tener para conformar un buen equipo, dándoles soluciones a dichos problemas.

ALUMNO:
FECHA:
FASE: **Introducción a la Ingeniería de Software y al trabajo en equipo**

PRÁCTICA 1 INGENIERÍA DE SOFTWARE

OBJETIVOS

- Conocer la historia de la Ingeniería de Software.
- Conocer ventajas de la Ingeniería de Software.

INTRODUCCIÓN

Como se ha visto en las clases, la Ingeniería de Software surgió por la necesidad de realizar software con la mejor calidad, después de una gran crisis que se presentó durante muchos años dentro del desarrollo de software.

La Ingeniería de Software consta de un proceso bien definido, para dicho proceso existen diferentes formas de planearlo y diferentes técnicas que han ido evolucionando y mejorando conforme se avanza en los estudios de la misma.

DESARROLLO

Reunirse en equipos de cinco personas, para realizar las siguientes tareas en equipo.

- Leer el artículo de Manuel Cota Aguilar y el capítulo 1 del libro de Ingeniería de Software Pragmática (Introducción a la Ingeniería de Software) que se encuentran disponibles en la página del curso.
- Formar equipos y repartirse el trabajo.
- Discutir lo leído entre todo el equipo y escribir los puntos que les parecieron mas importantes.
- Discutir los puntos que anotaron con el resto del grupo, pasando a exponerlo ante el grupo.
- Entregar los puntos que anotaron al ayudante.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron las tareas realizadas.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 2**
    - Leer capítulo 2 del libro Ingeniería de Software Pragmática (Introducción a la Ingeniería de Software).

- Leer práctica 2.

- **Documentos impresos que se utilizarán en la práctica 2**
  - Práctica 2 (Individual).
- **Herramientas que se utilizarán en la práctica 2 (por cada herramienta que no lleven será un punto menos)**
  - Un cronómetro (Individual).
  - Una hoja (Individual).

ALUMNO:
FECHA:
FASE: **Introducción a la Ingeniería de Software y al trabajo en equipo**

PRÁCTICA 2 COMENZANDO A TRABAJAR EN EQUIPO

OBJETIVOS

- Que los alumnos formen equipos dentro del grupo, haciendo que conozcan la importancia de trabajar en equipo.
- Motivar la reflexión personal sobre las habilidades de cada alumno, para seguir un rol dentro de un equipo de trabajo.
- Hacer una tarea en equipo.

INTRODUCCIÓN

Las nuevas tendencias laborales llevaron a las empresas a pensar en los equipos como una forma de trabajo habitual. Alcanzar y mantener el éxito en las empresas requiere talentos prácticamente imposibles de encontrar en un solo individuo, es por esto que el trabajo en equipo se ha vuelto algo fundamental. Actualmente las estructuras de las empresas, requieren una interacción mayor entre las personas que laboran en ellas, que sólo puede lograrse con una actitud cooperativa y no individualista, es decir trabajando en equipo.

Para lograr tener un buen ambiente laboral dentro de un equipo de trabajo es necesario saber la diferencia entre equipo de trabajo y trabajo en equipo:

- **Equipo de trabajo:** Es el conjunto de personas asignadas o auto asignadas a un rol, de acuerdo a sus habilidades y gustos específicos, para cumplir determinados objetivos bajo la conducción de un líder.
- **Trabajo en equipo:** Es la serie de estrategias, procedimientos y metodologías que utiliza un equipo de trabajo para lograr los objetivos propuestos.

Para lograr tener un buen equipo de trabajo, cada integrante debe tomar en cuenta lo siguiente:

1. Ser capaces de establecer relaciones satisfactorias con todos los integrantes del equipo.
2. Ser leales consigo mismos y con los demás.
3. Tener espíritu de equipo, de autocrítica y de crítica constructiva.
4. Tener sentido de responsabilidad para cumplir con los objetivos personales y de equipo.
5. Tener capacidad de autodeterminación, optimismo, iniciativa, comunicación y tenacidad.
6. Tener inquietud de perfeccionamiento, para la superación personal y de equipo.

Un equipo de trabajo puede estar formado por gente con formas de pensar y actuar distintas o iguales, sin embargo debemos tomar en cuenta que es mejor que sean distintas, ya que de esta forma cada uno de los integrantes aportará ideas diferentes para que las decisiones que se tomen en el equipo sean las mejores. Cuando hay diferencias y discrepancias surgen propuestas y soluciones más creativas. Esto lo podemos ver en los

equipos de fútbol, básquetbol o voleibol, en los cuales cada uno ocupa un puesto diferente (defensa, volante, etc.), pero todos dirigen sus energías ha lograr un objetivo. Algo importante dentro del equipo es que no hay lugar para el intolerante.

DESARROLLO

Reunirse en equipos de cinco personas, para realizar las siguientes tareas en equipo.

- El trabajo del equipo consistirá en analizar la información de un problema
- Llegar en equipo a una respuesta colectiva
- Tomar el tiempo que les lleva resolver el problema desde ahora hasta que lo resuelvan.
- Entregar al ayudante la hoja con las respuestas individuales y la hoja con las respuestas del equipo.

Sugerencia: Dentro de los equipos se pueden asignar roles a cada integrante para llevar acabo un mejor trabajo en equipo.

Problema

EL EDEN

Todos los integrantes del equipo deben imaginar que se encuentran en el cielo, después de su muerte, son acogidos por un ángel que les explica que tendrán que pasar tres años juntos en las siguientes condiciones:

- Ocuparán un terreno que mide 20 Km. de largo, donde:
  - 1/4 es un lago rico en peces.
  - 1/4 es de tierra cultivable.
  - 1/4 es un bosque selvático.
  - 1/4 es un terreno sin cultivar.
- Tendrán las mismas necesidades humanas y condiciones de vida que tenían mientras estaban vivos.
- EI clima del lugar es veraniego, con una temperatura constante de 30°C durante el día, y 20°C durante la noche. Solo llueve 30 días al año.
- No se puede ir a otros lugares durante esos tres años.
- No se sabe que pasará con las personas al finalizar los tres años.

- Las únicas personas con las que se estará en relación durante esta estancia, serán los demás miembros del equipo.

Deben elegir por consenso un máximo de 10 objetos (lista de abajo), sin restricciones de volumen o dimensiones, los objetos estarán a disposición de todos durante su estancia allí. La lista de objetos se debe enumerar de la siguiente manera:

- En forma individual, ordene los objetos de acuerdo con su importancia. Colocando el número 1 en el objeto que considere mas importante, el 2 en el siguiente, etc.
- Con su equipo, debe tomarse una decisión acerca del orden de importancia de los objetos, la cual se discute en equipo para llegar a un consenso grupal.

Objetos	Elección Personal	Elección Equipo
Un equipo completo de pesca		
Dos palas y dos picos de jardinería		
Una vaca y un toro		
Tres raquetas de tenis y 20 pelotas		
Dos guitarras		
Veinte trajes de baño		
EI cuadro de La Gioconda		
Un proyector cinematográfico, pilas y 10 películas eróticas		
Una capilla totalmente equipada		
Un kilogramo de marihuana		
Cien cajas de conservas surtidas		
Cien libros de literatura		
Cien botellas de bebidas alcohólicas		
Un jeep nuevo (sin gasolina)		
Una barca de remos		
Diez barras de metal		
Cien cajas de fósforos		
Un caballo de seis años		
Una buena cantidad de penicilina		
Cien paquetes de cigarros		
Un gato siamés		
Tras mazos de naipes		
Material de tocador y de belleza		
Semillas de diversas clases		
Una maquina de escribir		
Cinco guardarropas completos		
Veinticinco fotografías de personas conocidas		
Cinco mil hojas de papel para escribir		
Un fusil y 100 balas		
Treinta tubos de pintura al óleo		
Cien discos y un tocadiscos con pilas		
Un cadillac y 4000 litros de gasolina		
Cinco walkie-talkies		
Dos tiendas de campaña de tres plazas cada una		
Tres camas grandes		
Cien cajas de píldoras anticonceptivas		
Varios juegos de salón		

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvió el problema.
- Cómo fue la participación de tu equipo.
- Lograron llegar a la solución con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 3**
  - Leer capítulo 2 (Desarrollo de Software en Equipo).
  - Leer práctica 3.
- **Documentos impresos que se utilizarán en la práctica 3**
  - Práctica 3 (Individual).
  - Tabla 2: Habilidades por rol (Capítulo 2, Individual).
- **Herramientas que se utilizarán en la práctica 3 (por cada herramienta que no lleven será un punto menos)**
  - Un cronometro (Individual).
  - Una regla (Individual).
  - Dos hojas reciclables (Individual).
  - Un lápiz (Individual).
  - Color verde, azul, rojo, morado, rosa y negro (Individual).

ALUMNO:
FECHA:
FASE: **Introducción a la Ingeniería de Software y al trabajo en equipo**

PRÁCTICA 3 FORMANDO EL EQUIPO DE TRABAJO

OBJETIVOS

- Formar el equipo definitivo y que los alumnos estén concientes de querer trabajar con ese equipo durante todo el semestre siguiendo normas que ellos mismos establecerán.
- Que los alumnos se conozcan cada entre ellos y conformen un buen equipo de trabajo. Solucionando los problemas que se les presentarán a lo largo de la práctica.
- Que los alumnos se creen una conciencia de leer y comprender las instrucciones que se les dan al inicio de cada tarea sin pena a preguntar y realizar la tarea en equipo.
- Ponerle nombre, hacer un logo y un lema para el equipo, fijar día y horario de reuniones semanales y asignar roles a cada integrantes del equipo.

INTRODUCCIÓN

Como se mencionó en la práctica anterior en la actualidad en las grandes empresas se requiere del trabajo en equipo. En los equipos, en ciertas ocasiones surgen problemas o diferencias entre los integrantes ocasionados interna o externamente.

Para iniciar con el trabajo de equipo es poner el nombre, logo y lema a su equipo, se deben asignar por acuerdo de todos los integrantes y así comenzar la comunicación y el trabajo en equipo.

Para asignar los roles a los integrante del equipo, deben conocer las habilidades y gustos de cada integrante y las habilidades que requieren los roles. Para que así cada integrante haga su trabajo de la mejor manera y en conjunto el equipo cumpla con los objetivos. Los roles son para que cada integrante sepa que debe hacer. Dentro de la Ingeniería de Software cada rol tiene habilidades y tareas para que se pueda cumplir adecuadamente. Los roles que se asignarán son:

- Líder del equipo LE
- Administrador de desarrollo AD
- Administrador de planeación AP
- Administrador de calidad AC
- Administrador de apoyo AA

Las reuniones de trabajo constituyen uno de los distintivos del trabajo en equipo más importantes, estas se llevan acabo con frecuencia. Sirven para debatir y decidir, para que todos los integrantes tengan un conocimiento exacto de la situación del proyecto, fijar criterios, estandarizar ideas, compartir opiniones, intercambiar puntos de vista, además de que facilita la comunicación y ayudan a la integración del equipo. Para que estas reuniones sean efectivas se debe acordar:

- Fecha, hora y duración máxima de cada reunión.
- Asuntos a tratar en una agenda de reunión que se distribuye por adelantado al equipo, asegurándose que se cumpla la agenda en todos sus puntos.

- La coordinación de la reunión es llevada a cabo por el líder del equipo.
- Establecer el compromiso de asistencia de todos los integrantes.
- Dejar por escrito los compromisos acordados en la minuta, que se envía a los integrantes.

Problemas en un equipo de trabajo

Ocasionados internamente: Es muy común que ocurran, en cualquier equipo cuando comienza a formarse. Puede ser que sea la primera vez que trabajan juntos o que ni siquiera se conozcan. A continuación se explican los principales factores por los que pueden surgir problemas de este tipo:

- **La comunicación:** La causa más común es la falta o mala comunicación entre los integrantes del equipo, ya que no cuentan con la confianza para expresar su opinión ante los demás. Si este problema no se soluciona a tiempo, va generando un equipo apático, individualista y desunido, que termina por separarse o no logra terminar sus tareas. Para resolver este problema es necesario que la persona que tomó el rol de líder hable con sus compañeros de equipo, los haga sentirse en confianza para que estos puedan expresar su opinión o desacuerdos y de esta forma el equipo comienza a unirse.
- **Falta de Material:** Cuando a un equipo se le proporciona el material que requiere para el trabajo y no lo administra correctamente, pueden comenzar a surgir los problemas en el equipo. También puede ser causado externamente.
- **Diferentes puntos de vista:** Ya que se logró una comunicación dentro del equipo comienzan a surgir diferentes puntos de vista de los integrantes. Cuando esto ocurre puede ser que todos tengan un punto de vista diferente y ninguno quiere ceder o que se formen coaliciones. Esto se puede resolver si la persona que tiene el rol de líder es capaz de llevar a su equipo a un acuerdo con el cual todos estén satisfechos.

Ocasionados Externamente: Los factores que causan los problemas, no dependen de los integrantes del equipo, sino de las personas que formaron el equipo (empresas, profesores, etc.) o de los encargados de proporciona al equipo lo necesario para que éste cumpla con sus tareas. A continuación se dan las causas básicas de estos problemas.

- **Asignación de Roles:** En ocasiones los roles los asigna otra persona, puede ser que los integrantes no estén de acuerdo con el rol que les fue asignado a ellos o a sus compañeros, creando un ambiente de trabajo con envidias o superioridad. Este tipo de conflictos se pueden resolver viendo cuales son las tareas que les serán asignadas a cada rol y si es necesario hablar con la persona que les asignó el rol, para que se puedan cambiar si así lo requieren.
- **Ingreso de un integrante nuevo al equipo:** Cuando el equipo ya esta formado y trabajando, puede y se les une un nuevo integrante que conozcan o no, con un rol superior o inferior, trae problemas, si el equipo ya estaba organizado se tendrá que reorganizar lo cual puede afectarlo, por tener que hacer nuevos calendarios de trabajo o retrabajar. Para resolver estos conflictos, antes de que entre un nuevo integrante, se debe hablar con el equipo, entre todos hacer un análisis de los pros y contras y el por qué se le asignará tal rol.

Normas del equipo

En un equipo de trabajo es muy importante que existan normas. Ayudarán a tener buena comunicación y buen ambiente de trabajo. Deben establecerse por todos los integrantes del equipo.

Se propone se haga una junta, cada integrante de una o varias normas que crea son necesarias para generar un buen equipo. Posteriormente se haga una votación de equipo, en la que todos lo integrantes participen y se llegue a un acuerdo para establecer sus normas y las sanciones si estas normas no son cumplidas, por alguno de los integrantes. Finalmente todos los integrantes deben estar concientes de seguir esas normas y de la sanción correspondiente a la falta.

DESARROLLO

Reunirse en equipos de 5 personas, tomando en cuenta que deben trabajar con el equipo definitivo.

Deben reflejen las mejor características de un equipo de trabajo ideal y lo que han aprendido en las prácticas anteriores. Para esto todos los equipos deben pasar por una serie de pruebas en el menor tiempo posible, realiza las siguientes actividades en una hoja por equipo, tomando el tiempo que les lleva realizar cada actividad y escribiendo este tiempo debajo de cada actividad.

- Asignarle nombre a su equipo. (Escribirlo con color verde)
- Hacer una lista de los nombres completos de los integrantes. (Escribirlos con color azul)
- Hacer un logo y lema para su equipo. (Dibujar el logo con los colores y escribir el lema con color rojo)
- Asignar los roles a cada integrante, según habilidades, capacidades y gustos. (Escribirlos con color morado)
- Llegar a un acuerdo del día y hora de la reunión semanal. (Escribirlo con color rosa).

Realizar una lista de normas y sanciones por no cumplir las normas, según lo que aprendieron en esta práctica, tomando en cuenta los problemas que han surgido durante esta y las prácticas anteriores y que puedan surgir, entregarla.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvió el problema.
- Cómo fue la participación de tu equipo.
- Lograron llegar a la solución con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 4**
    - Leer capítulo 2 (Desarrollo de Software en Equipo).
    - Leer práctica 4.
- **Documentos impresos que se utilizarán en la práctica 4**
    - Forma Semana Personal (Individual)
    - Forma Semana del Equipo (Individual).
    - Práctica 4 (Individual).
- **Herramientas que se utilizarán en la práctica 4 (por cada herramienta que no lleven será un punto menos)**
    - Un cronometro (Individual).
    - Una hoja (Individual).

Capítulo 4. Práctica para el Llenado de Formas Básicas

Primera práctica de desarrollo de software en equipo. Los alumnos se comienzan a familiarizar con las formas, las cuales usarán a lo largo de todo el curso, con las que aprenderán a llevar un control de su tiempo.

4.1 Práctica 4

Esta práctica tiene como objetivos, que los alumnos comiencen a llevar el control del registro del tiempo tanto personal y como del equipo y aprender a usar las formas básicas.

Se les da ejemplos de cómo llenar las formas de semana personal y semana del equipo y se les pide realizar ciertas tareas con las que llenaran sus formas.

ALUMNO:
FECHA:
FASE: **Llenado de Formas Básicas**

PRÁCTICA 4 LLENADO DE FORMAS BÁSICAS

OBJETIVOS

- Comenzar a llevar el control del registro del tiempo personal y del equipo.
- Aprender a usar las formas básicas que usaran durante todo el semestre.

INTRODUCCIÓN

Dentro de la Ingeniería de Software se usa el llenado de formas, en esta ocasión aprenderemos a llenar las formas básicas, en las cuales se lleva el control de los tiempos que se ocupan en la realización del proyecto de forma personal y en equipo. En estas formas se anota el tiempo que lleva efectuar cada una de las tareas de desarrollo de software y tamaños de los productos que se van generando a lo largo del proceso de desarrollo del proyecto.

El control del tiempo es una de las mejores prácticas que ofrece la ingeniería de software, ya que teniendo un registro de tiempo de proyectos anteriores podemos hacer un estimado en proyectos futuros y de esta forma ser más formales en la planeación de los proyectos.

A continuación se dan ejemplos de la Forma Semana Personal y la Forma Semana del Equipo llenas:

Forma Semana personal: Se llena por persona diariamente, su objetivo es que cada integrante registre los productos que trabajó durante la semana, indicando el tiempo que dedicó, tamaño del producto y estado de avance del producto. Con esta forma los alumnos aprenderán a llevar un registro de su tiempo durante una semana, para que posteriormente se tomen las métricas del equipo.

Forma semana del equipo: Aquí se conjunta la información de las formas semanales de cada integrante, que se entregan en las reuniones semanales de avance del equipo, y el administrador de planeación la llena. Se hace y registra un resumen de avance del trabajo de equipo, con la suma de los tiempos y productos registrados por cada integrante. Se entrega al instructor cada semana. Incluye una tabla para dar seguimiento a los riesgos encontrados en el equipo. Para tener así métricas del tiempo que puedan usar para el siguiente ciclo o proyectos posteriores.

Forma Semana del equipo ciclo 2: Es una variante de la forma anterior para el segundo ciclo, se aprovechan las mediciones del primer ciclo, tomando en cuenta el tiempo y tamaño de los productos que se realizaron en el ciclo 1. Para que se den cuenta cuánto tiempo han empleado y cuánto han ganado.

- **Forma Semana Personal**

Nombre Miguel Ángel Téllez R. Fase 9 Equipo FairuSoft ®
Fecha 30/07/2007 Ciclo 1 Semana IMPlementación

Productos de desarrollo generados	Horas dedicadas	Tamaño del producto	Estado actual del producto
Reunión del Viernes 21 de Mayo	02:00:00	————	Terminado
Reunión de Inspección, Lunes 24 de Mayo	01:30:00	————	Terminado
Implementación de clases	06:45:00		Proceso
Implementación de métodos	01:00:00		Proceso
Plan de pruebas unitario	03:15:00	————	Terminado
Totales	**14:30:00**	**535 ldc**	**Terminado**

- **Semana 2, Estrategia**

Equipo	FairuSoft ®	Fase	ESTRAT	Instructor	Guadalupe Ibargüengoitia
Fecha	14/04/2004	Ciclo	1	Semana	3

Datos semanales	**Actual**
Horas dedicadas al proyecto en esta semana	**29:47:00**
Horas dedicadas al proyecto en este ciclo hasta esta semana	**56:41:00**
Horas dedicadas a las tareas terminadas en esta fase en esta semana	**29:47:00**

Tareas de desarrollo efectuadas esta semana	**Horas dedicadas**	**Tamaño del producto**	**Estado actual del producto**
Elaboración de criterios de estrategia	05:15:00	2 hojas	Terminado
Elaboración del registro de riesgos	01:12:00	2 hojas	Terminado
Creación del nombre y logo del producto	01:15:00	1 logotipo	Terminado
Definición del plan de configuración	02:30:00	2 hojas	Terminado
Lectura del capitulo 4 y apéndice A	07:40:00	168 hojas	Terminado
Reunión del 02/04/2004	02:40:00	1 minuta	Terminado
Reunión del 06/04/2004	06:08:00	1 minuta	Terminado
Reunión del 12/04/2004	03:15:00	1 minuta	Terminado
Totales	**29:47:00**	**178 páginas**	**Terminado**

Seguimiento de asuntos o riesgos	**Estatus**
Copias de seguridad de los documentos	Terminado
Escasa papelería	--pendiente--

DESARROLLO

Reunirse con su equipo, tomar el tiempo de todas las actividades que realicen durante esta práctica, anotarlo en su forma semana personal. Posteriormente juntarlas en la Forma semana del equipo y entregar las dos formas al ayudante.

Actividades:

- Revisar la lista de normas y sanciones que hicieron la práctica pasada, si están de acuerdo todos con esas normas, el día y hora de la reunión semanal del equipo, firmarla cada integrante de equipo.
- Hacer, en equipo, una lista de las ventajas y desventajas que tiene el tomar y registrar el tiempo en cada actividad que realicemos.

- Revisar el logo, nombre y lema que realizaron la práctica pasada y llegar a un acuerdo si serán los definitivos y si no lo son realizar los definitivos.
- Revisar si todos quedaron de acuerdo con el rol que se asignaron.
- Entregar todos los documentos anteriores al ayudante.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron las tareas realizadas.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 5**
  - Leer capítulo 3 del libro de Ingeniería de Software Pragmática (Fase de Lanzamiento).
  - Leer práctica 5.
- **Documentos impresos que se utilizarán en la práctica 5**
  - Tabla 1: Responsabilidades de los roles (Capítulo 2, Individual).
  - Forma Registro de Riesgos (Por equipo).
  - Práctica 5 (Individual).

Capítulo 5. Fase de Lanzamiento

5.1 Fase de Lanzamiento

- **Primer ciclo:** En este ciclo esta fase se realiza en una semana completa. Entre sus objetivos se encuentran el que se definan los objetivos del equipo, del proyecto, de cada rol y en particular que cada alumno defina sus objetivos, que irán cumpliendo conforme se avance en el primer ciclo, que los equipos establezcan un estándar para su documentación, que en cada equipo se identifiquen los posibles riesgos que se pueden presentar a lo largo del curso y comiencen a realizar las reuniones semanales de equipo haciendo uso de la agenda y minuta.

  La documentación la empezaran a entregar a partir de esta fase y ya debe estar en el estándar que establecieron.

- **Segundo ciclo:** En el segundo ciclo esta fase se junta con las siguientes dos fases, ya que los alumnos cuentan con el conocimiento y documentación del primer ciclo.

  Los alumnos deben hacer una revisión a conciencia sobre los objetivos que tuvieron en el primer ciclo y verificar si fueron cumplidos o no.

Se les pide a los alumnos que lean el capítulo 3 del libro, en el que viene todo lo relacionado con la fase de lanzamiento, ellos tienen dos clases de laboratorio. En una realizarán la práctica 5 y en la otra se les dirá que realicen su reunión semanal de equipo, para que de esta forma los ayudantes se den cuenta como están trabajando los equipos y de ser necesario corregir algunos aspectos que lo necesiten sin presionar de ninguna forma.

5.2 Práctica 5

Los objetivos de esta práctica son que los alumnos establezcan los objetivos del equipo para su proyecto, que definan los estándares de su documentación y finalmente identificar y documentar los posibles riesgos que se les irán presentando conforme avance el proyecto.

Para lograr dichos objetivos se les proporcionan listas de objetivos, para que ellos selecciones los que consideran sean sus objetivos durante el semestre con la opción de anexar algunos, se les presentan ejemplos del registro de riesgos y de algunos estándares.

ALUMNO:
FECHA:
FASE: **Lanzamiento**

PRÁCTICA 5 ESTABLECER OBJETIVOS, ESTÁNDARES Y RIESGOS

OBJETIVOS

- Establecer los objetivos del proyecto.
- Definir los estándares de documentación.
- Identificar y documentar los riesgos.

INTRODUCCIÓN

Para poder definir los objetivos del proyecto, los equipos necesitan haber asignado los roles a cada integrante. Los objetivos se establecen unos por consenso de equipo y otros de forma individual.

En equipo primero se establecen los objetivos del equipo y después los del producto. Posteriormente de forma individual, cada integrante establece sus objetivos personales y según su rol.

En un equipo es importante establecer un estándar de documentación, el cual tiene como objetivo primordial homogenizar la documentación, que será elaborada por cada integrante para entregar un producto de calidad y para tener una fácil identificación y orden a la hora de revisarlo. El estándar de documentación puede incluir:

- Software a utilizar (editor de texto).
- Tipo y tamaño de hoja y de letras.
- Tabulaciones (sangrías) e interlineados.
- Encabezados de los documentos (incluirá: título del documento, nombre del equipo, responsable, fecha, versión y paginación).
- Nombres de los archivos, etc.

En un equipo de trabajo se presentan riesgos los cuales pueden tener graves consecuencias, crear conflictos dentro del equipo o ser solucionados. Por estas razones se debe crear una lista de riesgos que puedan surgir a lo largo del proyecto y así tener una solución anticipada a dicho riesgo. Se les deben asignar la probabilidad de que sucedan, estimar sus consecuencias y así buscar estrategias para manejarlos y en todo caso solucionarlos. Los riesgos son identificados por todos los integrantes del equipo de común acuerdo.

Forma Registro de Riesgos: La lista de los riesgos que se encuentren, se manejará en esta forma, de la cual es responsable el administrador de planeación y se debe llenar de forma conjunta por todo el equipo. A continuación se da un ejemplo de una forma de Registro de Riesgos llena:

FairuSoft®
Desarrollo de Software empresarial

México, D.F. a 31 de Julio de 2007

Registro de Riesgos RR

Equipo: FairuSoft® Fecha: 8-04-2004
Ciclo: 1

Riesgo	Probabilidad de ocurrencia	Impacto	Estrategia para manejar el riesgo.
Falta de herramientas para la impresión.	Baja	Medio	Transferirlo; imprimir en otro lado.
Que no se tenga conocimiento específico de ciertas herramientas.	Media	Alta	Prevenirlo; buscar al administrador de apoyo. Mitigarlo; aprender el uso de dichas herramientas.
Fallas de equipo de cómputo.	Baja	Alta	Contenerlo; ir al otro lugar.
Problemas personales o enfermedad.	Baja	Media	Mitigarlo; repartir las actividades con el resto del equipo.
Fallas de energía eléctrica.	Baja	Baja	Aceptarlo; en la medida que se pueda mitigarlo.
Deserción de algún miembro del equipo.	Baja	Alta	Transferirlo; repartir las actividades con el resto del equipo.

FAIRUSOFT_Registroriesgosv1,2.doc Versión: 1,0 Página 2 de 2

DESARROLLO

Reunirse con su equipo, guiados por el líder del equipo y con ayuda del instructor, definirán los objetivos del equipo y del producto por consenso.

A continuación se da una lista de objetivos, deben tachar los que elijan para su equipo o escribir otro objetivo en otros. Al terminar entregar una copia a los integrantes del equipo y una al ayudante.

OBJETIVOS DEL EQUIPO

- Mantener un ambiente de trabajo agradable en el equipo.
- Formar un equipo comprometido a realizar sus tareas puntualmente y así entregar a tiempo y con la mejor calidad los productos.
- Adquirir experiencia como equipo en el desarrollo de software.
- Tener una administración efectiva logrando un buen equipo que sea responsable, organizado y efectivo, fomentando la formalidad y profesionalismo para con el cliente (instructor) y con nosotros mismos.
- Llevar a cabo todas las actividades y prácticas encargadas por el cliente (instructor), de acuerdo con las especificaciones y en el tiempo planteado.
- Afrontar los problemas que se presentarán a lo largo del curso, con el fin de que nos acerquemos más a la realidad laboral.

- Que todos los integrantes del equipo con ideas, características y conocimientos diferentes se comuniquen, piensen, convivan y trabajen juntos con un objetivo común, logrando la mejor comunicación y transparencia entre todos los integrantes.
- Conocer las habilidades de cada integrante del equipo para explotarlas y asignar las tareas basándonos en este parámetro, sin descuidar los roles.
- Tener tolerancia entre nosotros y hacer efectiva la coordinación y planeación dentro del equipo.
- Lograr el mejor software que cumpla con las expectativas del cliente.
- Conocer las bases para producir un Software de calidad trabajando en equipo.
- Realizar un buen proyecto para que el cliente (instructor), quede satisfecho con el producto.
- Obtener una buena calificación como equipo.
- Aprender a llevar un control sobre nuestro trabajo como equipo y de forma individual.
- Otros:

OBJETIVOS DEL PRODUCTO

- Generar un producto de calidad que satisfaga los requisitos, sea amigable y costo-efectivo.
- Usar lo más posible tecnología de punta sin sacrificar estabilidad y compactabililidad, generando así un producto que sea mantenible y extendible.
- Generar un producto que sea modular, hecho con la mayor parte de componentes reusables.
- Que el código fuente sea entendible y este bien documentado.
- Respetar los tiempos de desarrollo y cada una de las fases por las que va a pasar nuestro producto y terminándolo para la fecha establecida.
- Que el programa no tenga errores.
- Que la interfaz con el usuario sea sencilla.
- Que la estructura del programa y sus componentes no tengan problemas de comunicación.
- Otros:

A continuación, cada integrante del equipo define sus objetivos personales y los del rol que se le asignó.

Abajo se muestra una lista de posibles objetivos deben tachar los que elijan o escribir otro objetivo en otros. Al terminar deben entregar una copia al administrador de planeación y una al ayudante.

OBJETIVOS PERSONALES PARA EL PRIMER CICLO

- Desempeñarme con responsabilidad en el equipo de trabajo, siendo ordenado y disciplinado.
- Aprender y adquirir experiencia en el desarrollar de software de calidad y en las técnicas de la ingeniería de software para poder aplicar aquellas que considere útiles en proyectos futuros y para tener una buena formación profesional.
- Obtener 10 como calificación final y no faltar a clases.
- Tener una buena relación con la maestra, ayudantes y mis compañeros de grupo y de equipo.
- Lograr integrarme a mi equipo haciendo que el equipo se sienta satisfecho con mi trabajo.

- Ser responsable con el trabajo asignado para poder entregar las cosas a tiempo, completas y con la mejor calidad, para tener la carpeta y el producto en la fecha indicada.
- Ayudar a que el producto satisfaga las necesidades del cliente.
- Dedicar el mayor tiempo posible para el trabajo en equipo.
- Aprender a trabajar en equipo y valorar el trabajo de otras personas.
- Dedicar el tiempo necesario a las actividades de la materia.
- Identificarme con mis compañeros de equipo.
- Establecer una buena comunicación y relación con todos, identificando los puntos fuertes y débiles de nuestra comunicación y escuchando las opiniones de cada integrante
- Coordinarme con todos los integrantes de mi equipo para trabajar como uno solo.
- Convertirme en una persona más responsable aprovechando el compromiso con el equipo.
- Aprender a desenvolverme en ambientes y situaciones similares (trabajar con gente que no conocía respetando diferentes puntos de vista).
- Contribuir a que lo entregado en el primer ciclo sea satisfactorio.
- Ser tolerante mis compañeros de equipo y de grupo.
- Contribuir a la armonía del equipo, ya que es fácil que se surjan problemas dentro del mismo.
- Otros:

OBJETIVOS DEL ROL

Líder

- Mantener el orden y la disciplina en el equipo con un ambiente agradable, para lograr resultados satisfactorios como equipo.
- Realizar correctamente las actividades correspondientes a cada fase.
- Apoyar a cada uno de los integrantes para evitar problemas.
- Mantener al equipo con el mismo ritmo de trabajo y que su efectividad incremente.
- Motivar a los integrantes de mi equipo a ser un equipo efectivo, en el que ningún integrante entren en conflictos, manteniendo la comunicación con todos y entre todos y desarrollar un excelente producto.
- Que el instructor esté satisfecho con mi trabajo manteniéndolo informado de nuestro progreso.
- Que el equipo esté satisfecho por como cumplo mi rol.
- Guiar al equipo de manera democrática.
- Crear un ambiente de confianza hacia mis compañeros.
- Hacer que todos los temas concernientes al producto se discutan entre todos los integrantes del equipo y así llegar a un acuerdo entre todos.
- Resolver problemas que surjan en el equipo (una discusión, una falla en algún programa, etc.)
- Otros:

Administrador de desarrollo.

- Lograr que el producto funcione como se espera.
- Entender el proceso de desarrollo.
- Conocer y hacer uso de las habilidades de desarrollo de cada integrante del equipo, distribuyendo de la mejor manera las partes a realizar del proyecto.
- Que todo el código del programa este debidamente comentado.
- Que se haga una programación limpia y sin desperdicio de recursos.
- Unir los pedazos de código en uno sólo en el cual no haya una distinción de dueño.
- Proponer estrategias útiles que proporcionen resultados.
- Asegurar que la calidad del producto sea la mejor.
- Dirigir al equipo en la producción de la estrategia de desarrollo, en estimar el tiempo para producir el proyecto y en producir un diseño de alto nivel.
- Dirigir el desarrollo y diseño de las especificaciones requeridas por el software.
- Dirigir al equipo en la implementación del software, construcción, integración y aplicación de los planes de prueba y en la producción de la documentación del usuario del producto.
- Participar en la producción del reporte del ciclo del desarrollo.
- Actuar como un desarrollador de producto.
- Otros:

Administrador de planeación.

- Organizar y sincronizar el trabajo de todos los integrantes del equipo verificando el guión, para terminar a tiempo los productos de cada fase del ciclo.
- Hacer que la planeación sea de común acuerdo con el todos los integrantes del equipo.
- Registrar correctamente los tiempos de todos los integrantes del equipo, para planear con mayor precisión la fecha de entrega de los productos.
- Asistir a todas las reuniones del equipo, y procurar lograr los objetivos de las mismas.
- Desarrollar la Agenda Electrónica de manera eficiente, para minimizar las pérdidas de tiempo y mantener informados a todos los miembros del equipo, de los sucesos.
- Establecer los días más propicios para las reuniones y para la entrega de los documentos.
- Actuar como ingeniero de desarrollo.
- Asistir a todas las reuniones del equipo.
- Hacer las tareas que corresponden a mi rol y ayudar en las tareas comunes del equipo así como en las que pueda contribuir fuera de mi rol especificado.
- Otros:

Administrador de calidad.

- Verificar que todos cumplan con el estándar definido por todos los integrantes del equipo y los que nos proporciona la Ingeniería de Software y así realizar un producto de calidad.
- Hacer revisiones de los productos a tiempo, mejorar documentos de los miembros del equipo y verificar que se corrijan los defectos encontrados para que el producto tenga la mejor calidad.
- Lograr el mejor nivel de calidad en el desarrollo del sistema.
- Que los miembros del equipo utilicen correctamente los procesos de la Ingeniería de Software.
- Realizar y reportar correctamente todas las inspecciones al equipo.
- Reportar todas las reuniones del equipo e incluirlas en la carpeta.
- Otros:

Administrador de apoyo.

- Abogar por la reutilización dentro del equipo.
- Establecer y darle mantenimiento al glosario del sistema.
- Llevar una buena administración de la carpeta, del depósito electrónico, es decir, llevar un control de los cambios (versiones del producto) e informar de éstos a los integrantes del equipo
- Proporcionar al equipo el apoyo y herramientas necesarias para que el desarrollo del producto se realice en las mejores condiciones técnicas, con la mejor herramienta y soporte.
- Crear la tabla del control de la configuración y administrar el sistema de control de cambios.
- Participar en el desarrollo del reporte del ciclo.
- Otros:

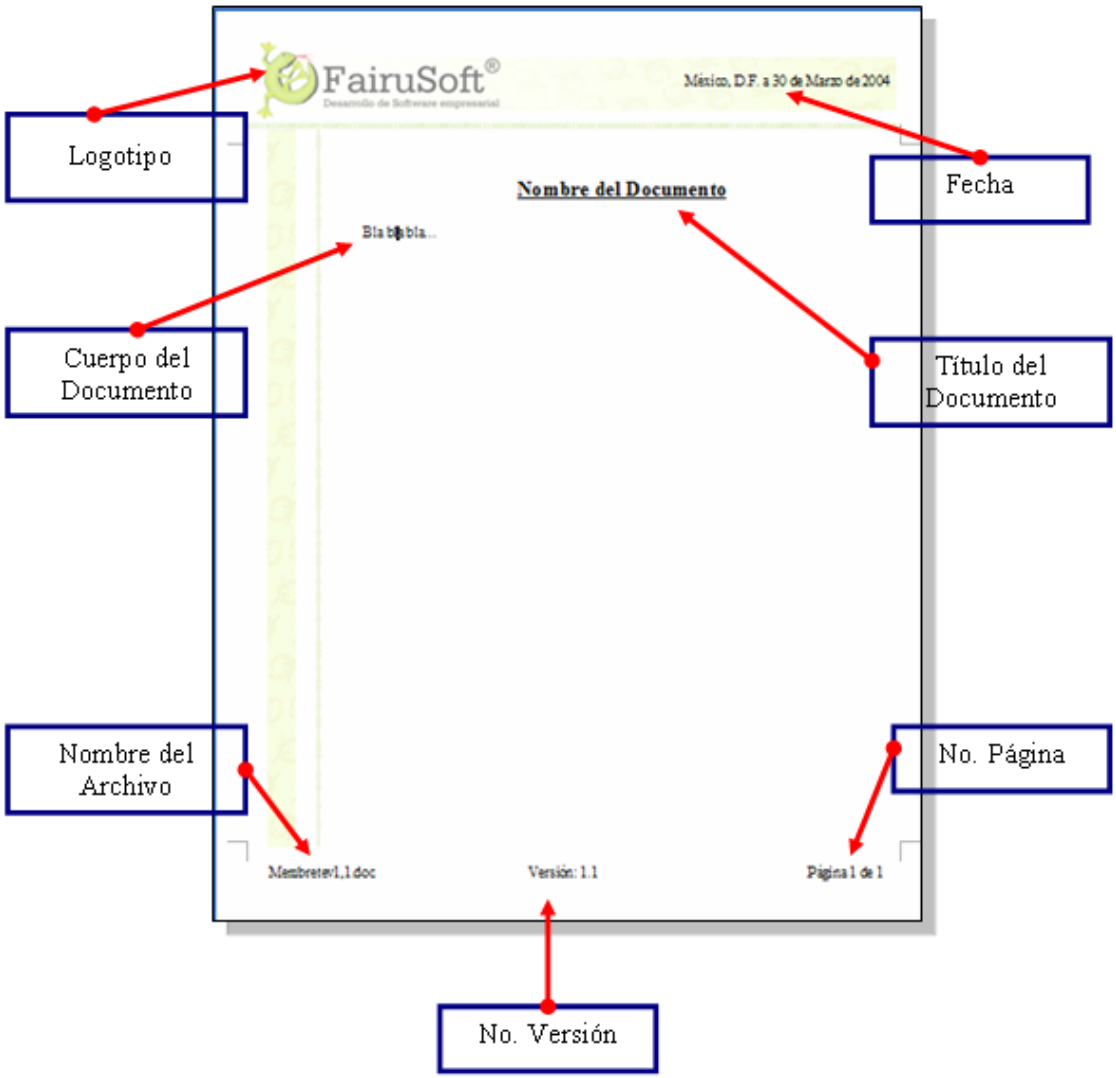
Ingenieros de Desarrollo.

- Actuar como ingeniero de desarrollo.
- Asistir a todas las reuniones del equipo.
- Contribuir a la armonía del equipo, ya que es fácil que se den problemas dentro del mismo.
- Hacer las tareas que corresponden a mi rol y ayudar en las tareas comunes del equipo así como en las que pueda contribuir fuera de mi rol especificado.
- Otros:

ESTANDAR DE DOCUMENTACIÓN

Entre todos los integrantes del equipo definirán el estándar para la documentación de su proyecto. A continuación se presentan ejemplos de estándares de documentación:

Ejemplo 1:



- Software: Este documento está generado en Microsoft Word®.
- Tipo y tamaño de hoja: Se usará hoja tamaño carta en color blanco membretada como se muestra en la figura de arriba.
- Tipo y tamaño de letras: Todos en color negro,
    - Título: Times New Roman 18, Negrita, Subrayado
    - Subtítulo: Times New Roman 14, Negrita
    - Cuerpo del Texto: Times New Roman 12
- Interlineado: Será sencillo entre renglones y doble entre párrafos.
- Nombre del archivo: FAIRUSOFT_XXXVN,n, en donde:

Donde: XXX son las siglas de la fase a la que pertenece el documento generado, N es el número del ciclo en el que estamos trabajando, n es el número de la versión en que va el documento, la primera vez es 0, si hay correcciones sólo se pasa a la versión n+1. Si es de la fase de LANzamiento del primer ciclo y en la primera versión, sería FAIRUSOFT_LANV1,0.doc.

Ejemplo 2:

Estándar de Documentación.

Del editor de textos.

- Word de Microsoft Office 2003 SP2.

Del nombre del archivo.

- El nombre del archivo deberá ser S*nn*V*nnd*.doc, donde se codificará en el siguiente orden el número de semana, la versión y la descripción del documento.
  - *nn* se refiere a un número de dos dígitos.
  - *d* se refiere a la descripción del documento – el título sin artículos, en mayúsculas y en CammelBack.
- Se usarán sólo los siguientes caracteres ASCII: A-Z, a-z, 0-9.

De la hoja.

- Hoja tamaño carta, color blanco, con los siguientes márgenes:
  - Los márgenes izquierdo y derecho medirán 2cm.
  - El margen superior medirá 1.5cm.
  - El margen inferior medirá 1.8cm.
  - Los márgenes del encabezado y del pie de la página serán de 1.25cm.

Del formato.

- La carátula tendrá el siguiente formato en orden descendente:
  - Logotipo del equipo.
  - Línea en blanco de 26pts.
  - Fase con la forma de título en tamaño 26.
  - Línea en blanco de 26pts.
  - Semana con la forma de texto normal, centrada y en tamaño de 20pts.
  - Ciclo con la forma de texto normal, centrada y en tamaño de 20pts.
  - Línea en blanco de 20pts.
  - Fecha con forma de texto normal, centrada y en tamaño de 12pts.
  - La versión v*n* (*n* es el número de la versión), con la forma de título en tamaño de 12pts.
  - Sin número de página.
  - Sin puntos al final
- Las hojas normales tendrán el siguiente formato:
  - En la esquina superior izquierda el título enunciado en la carátula con forma de título en tamaño 10 sin negritas.
  - En la esquina superior derecha el logotipo del equipo con 1cm. de alto y el ancho correspondiente en proporción.
  - La línea bajo el logotipo y el título será negra de 1pt. de grosor.
  - En la siguiente línea el nombre del autor del documento.
  - En el pie de página los siguiente elementos:
    - A la derecha la fecha con formato dd/mm/aaaa.

        - Al centro la versión del documento.
        - A la izquierda el número de página.
    - Los títulos, subtítulos y secciones llevarán un punto al final.
    - Los títulos, subtítulos y secciones tendrán una línea en blanco antes y después del tamaño al menos de la forma correspondiente.
    - Los títulos siempre deberán comenzar una página nueva.
    - La línea siguiente a un título debe ser en blanco del mismo tamaño.

De la letra.

- Fuente color negra.
- Forma de títulos.
    - Fuente Lucida Sans Unicode.
    - Tamaño de 18pts.
    - Centrado.
    - Forma de vérsales y negritas.
    - Todas las palabras a excepción de artículos y preposiciones.
- Forma de subtítulos.
    - Fuente Tahoma.
    - Tamaño de 12pts.
    - Alineado a la izquierda.
    - Forma de vérsales y cursiva.
- Forma de texto normal.
    - Fuente Tahoma.
    - El tamaño de 10pts.
    - Alineación justificada.
    - Sangría de 1cm. excepto en el primer párrafo de cada subsección (texto entre títulos y subtítulos).
    - Interlineado sencillo.
    - Sin espacio adicional entre párrafos.

De las viñetas y sangrías.

- 1cm. de margen para el texto y 0.5cm. de margen para el carácter.
- El carácter para el primer nivel será: ❖
- El carácter para el segundo nivel será: ♦
- El carácter para el tercer nivel será: *
- Las sangrías no llevan carácter.

De las tablas.

- Pueden ser horizontales (con texto escrito de izquierda a derecha) o verticales (con texto escrito de abajo hacia arriba).
- Las tablas horizontales serán centradas horizontalmente en la página y ocuparán a lo más 98% de la anchura de la página.
- La anchura de las columnas y la altura de los renglones serán variables y uniformes.

- El título de cada columna y renglón, en caso de ser necesario, será en forma de texto normal en negritas centrado horizontalmente y verticalmente.
- El texto de las celdas será en forma de texto normal centrado horizontalmente y verticalmente.
- Los bordes serán sólidos de color negro.
- Los bordes exteriores y de las celdas titulares serán de 1.5pts. de espesor.

Los bordes interiores serán de 0.5pts. de espesor.

IDENTIFICACIÓN DE LOS RIESGOS

Entre todos los integrantes del equipo identificarán los riesgos haciendo una lista, entregar una copia al ayudante. A continuación se muestran ejemplos de riesgos:

Riesgo	Probabilidad de ocurrencia	Impacto	Estrategia para manejar el riesgo
⌗ Que se traspapelen los documentos	Media	alto	Prevenirlo, tener todos los documentos del proyecto en un mismo sitio
⌗ No tener acceso al laboratorio	Baja	medio	Transferirlo, a los laboratoristas.
⌗ Que algún integrante se enferme	baja	alto	Mitigarlo, repartir su trabajo entre los demás
⌗ No tener los recursos para el desarrollo del producto	baja	medio	Prevenirlo, tener el equipo necesario tanto en el laboratorio como en nuestras casas
⌗ Que haya días de asueto	baja	baja	Mitigarlo, recorrer las fechas de los trabajos
⌗ No cumplir con los estándares de la documentación	Baja	Medio	Mitigarlo, siguiendo el estándar
⌗ Falta de comunicación	Media	Alto	Prevenirlo, tratando de comunicarse y asistiendo a las reuniones
⌗ Inexperiencia en manejo de tecnología	Alta	Alto	Evitarlo, tomar asesoría técnica de las tecnologías
⌗ Expulsión de un integrante	Baja	Alto	Prevenirlo, tomando mayor compromiso
⌗ Paro de labores	Baja	Alto	Contenerlo, vías alternas para mantener el curso
⌗ Falta de tiempo por otra materia	Media	Alto	Mitigarlo, administrando mejor el tiempo
⌗ Perdida de información accidental	Baja	Alto	Prevenirlo, haciendo respaldos
⌗ Falta de un integrante	Baja	Alto	Aceptarlo
⌗ Otros:			

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron las tareas realizadas.
- ¿Cómo fue la participación de tu equipo?
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 6**
    - Leer capítulo 4 del libro de Ingeniería de Software Pragmática (Fase de Estrategia).
    - Leer práctica 6.
- **Documentos impresos que se utilizarán en la práctica 6**
    - Forma Estrategia (Por equipo).
    - Práctica 6 (Individual).

Capítulo 6. Fase de Estrategia

6.1 Fase de Estrategia

- **Primer ciclo:** En este ciclo la fase de estrategia se junta con la fase de planeación que es la siguiente fase, ya que dichas fases no son difíciles ni muy laboriosas para los alumnos, lo cual se ha comprobado a lo largo de varios semestres.

  Entre los objetivos de esta fase están que cada integrante del equipo sugiera una estrategia para resolver una parte del problema en este ciclo, posteriormente el equipo discutirá y se elegirá la estrategia con la cual llevarán a cabo su proyecto, en cada ciclo.

  Otro objetivo, es que en cada equipo se cree un depósito de documentos, en el que guardarán su documentación, llevando un control sobre las versiones de cada documento guardado.

  La tarea principal que se explica es el análisis del problema, para llegar a la selección de la mejor estrategia terminando el proyecto en dos ciclos. Los alumnos deben leer el capítulo 4 del libro que es sobre la fase de estrategia. Una de las dos clases de laboratorio de esta semana se usa para que los alumnos realicen la práctica 6.

- **Segundo ciclo:** El objetivo de esta fase en este ciclo es que los alumnos hagan la Administración de la Configuración, para lo cual realizan un plan de configuración y un informe de estado de la configuración. Con lo que cada equipo podrá controlar las versiones de los productos y los cambios que se generan, informando a todos los integrantes del equipo del estado de los cambios.

  La tarea principal que se explica es la realización de la administración de la configuración por medio del plan de la configuración y el control de cambios y versiones. Realizan la última práctica la práctica 17 y se les sugiere que en las siguientes clases de laboratorio llevan a cabo sus reuniones de equipo.

6.2 Práctica 6

Los objetivos de esta práctica, son que los alumnos aprendan a crear y documentar una estrategia para el desarrollo de un producto de software en dos ciclos.

Para lograrlo se les explica lo que es una gráfica de dependencias entre necesidades, se les da un problema de ejemplo con una estrategia ya planteada y se les pide que mejoren dicha estrategia creando una nueva gráfica, de esta forma los alumnos comienzan a pensar en esto y les surgen dudas las cuales se resuelven en la clase para que posteriormente ellos puedan realizar la estrategia para su proyecto.

6.3 Práctica 17

Los objetivos de esta práctica son que los alumnos aprendan hacer el Plan de la Configuración de su software, hacer el Informe del estado de la configuración y a dar seguimiento a los cambios que se presentan en el desarrollo del software.

ALUMNO:
FECHA:
FASE: **Estrategia**

PRACTICA 6 ESTRATEGIA DE DESARROLLO

OBJETIVO

- Aprender a crear y documentar una estrategia para desarrollar el producto en dos ciclos.

INTRODUCCIÓN

La estrategia nos sirve para dividir el alcance del proyecto en ciclos, en nuestro caso lo dividiremos en dos ciclos. En el primero se diseña, implementa y evalúa una primera versión funcional del sistema; en el segundo ciclo se incrementan las funcionalidades del producto para generar una versión final, la cual deberá cubrir todas las necesidades del cliente.

Cada estrategia tiene sus méritos de acuerdo con la situación que se considere dentro del equipo. Sin importar cuál sea la estrategia utilizada, el equipo debe procurar que la versión del sistema en el primer ciclo proporcione un subconjunto de funcionalidades del producto final, que se encuentre libre de problemas y listo para que el cliente pueda usarlo cubriendo algunas de sus necesidades.

La técnica que usaremos para obtener las diferentes alternativas para la estrategia es la “Gráfica de dependencias entre necesidades” [OKTABA y ORTEGA]

Gráfica de Dependencias entre Necesidades

Es una gráfica dirigida con vértices que representan las funcionalidades y describen las relaciones de dependencia total o parcial entre las necesidades funcionales de un problema. Se construye a partir de los siguientes pasos:

1. Se identifican las necesidades independientes, colocándose como nodos terminales de la gráfica. Una necesidad *a* se considera independiente si su satisfacción no depende de la satisfacción de ninguna otra necesidad.
2. Se analizan, una por una, el resto de las necesidades, revisando si su satisfacción tiene dependencia total de la satisfacción de otra necesidad previamente colocada en la gráfica. Una necesidad *a* depende totalmente de otra necesidad *b* si la satisfacción de *a* requiere previamente de la satisfacción completa de la necesidad *b*. Si este es el caso, la necesidad *a* se coloca en la gráfica como un nuevo nodo, unido por un vértice dirigido hacia el nodo de la necesidad *b* de la cual depende totalmente.
3. Se repite el paso anterior hasta colocar como vértices todas las necesidades.
4. La gráfica resultante se analiza, a fin de identificar los nodos cuyas necesidades se satisfagan parcialmente a partir de la satisfacción de otros nodos. Una necesidad *a* depende parcialmente de otra necesidad *b*, si *a* puede satisfacerse de modo incompleto mediante la satisfacción completa de la necesidad *b*. En estos casos, la necesidad *a* se conecta mediante un vértice dirigido (trazado como una línea discontinua) hacia el nodo de la necesidad *b*, de la cual depende parcialmente.

Es importante hacer notar que el análisis se realiza a partir de una descripción de las necesidades funcionales en lenguaje natural, por lo que es susceptible a distintas interpretaciones. En consecuencia, es de esperar que este tipo de análisis pueda generar más de una gráfica representativa del problema.

Justificación de la Estrategia

En el primer ciclo se deben elegir las necesidades que son más importantes, para presentarle al cliente una versión del proyecto con la que pueda comenzar a trabajar si es que así lo desea y se dejan para el segundo ciclo las necesidades complementarias, para en el segundo ciclo hacer mejoras y extensiones al proyecto.

A continuación se da un ejemplo de una Estrategia, con una opción de la lista de necesidades y de la gráfica de dependencias:

Ejemplo:

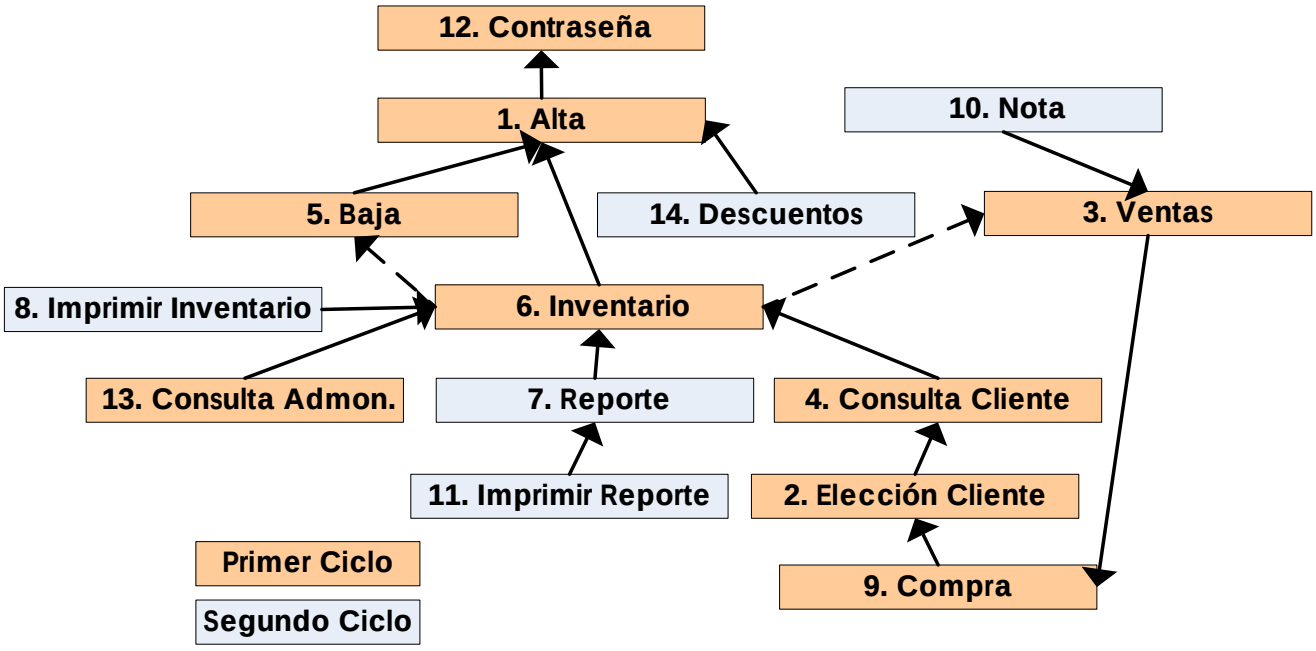
Problema: Librería Electrónica

Desarrollar un sistema de software para apoyar al administrador de una librería por Internet a consultar, dar de alta y baja los artículos que venden. El sistema lo podrá usar también algún posible cliente que quiera consultar los productos de la librería a través de Internet. La librería vende libros, CD de música, y DVD de películas. El administrador deberá poder identificarse para entrar al sistema. Las consultas podrán hacerse por diversos campos, por ejemplo título, autor o editorial. Se podrá sacar el inventario de los artículos en existencia y hacer reportes de faltantes de algún artículo cuando se terminen las existencias. El sistema se desarrollará en java.

Lista de las Necesidades funcionales del problema

Funcionalidades		**Ciclo**
N1.	El sistema será capaz de dar de alta los artículos que se venden en la librería (**Alta**).	1
N2.	El cliente podrá elegir que artículo quiere comprar (**Elección del Cliente**).	1
N3.	El sistema podrá hacer ventas según la solicitud del cliente (**Ventas**)	1
N4.	El cliente dará datos al sistema de algún artículo (libros, CD de música o DVD de películas) del que requiera información para su compra y el sistema le ofrecerá distintos artículos según lo que solicito el cliente (**Consulta Cliente**).	1
N5.	El sistema será capaz de dar de baja los artículos que se venden en la librería (**Baja**).	1
N6.	El sistema podrá presentar un inventario de los artículos existentes (**Inventario**).	1
N7.	El sistema podrá presentar un reporte artículos vendidos (**Reporte**).	
N8.	El sistema permitirá imprimir el inventario (**Imprimir Inventario**).	
N9.	El cliente podrá decidir que artículo o artículos comprará (**Compra**).	1
N10.	El sistema podrá imprimir la compra de cada usuario (**Nota**).	
N11.	El sistema permitirá imprimir el reporte (**Imprimir Reporte**).	
N12.	El sistema deberá permitir que el administrador entre al sistema por medio de una contraseña (**Contraseña**).	1
N13.	El administrador podrá consultar los artículos de la librería (**Consulta Admón.**)	1
N14.	El administrador deberá poder establecer descuentos a los artículos por medio del sistema (**Descuentos**).	

Gráfica de Dependencias entre Necesidades



DESARROLLO

- Individualmente, lee nuevamente el problema del ejemplo
- Entre todo el equipo, revisen y discutan la lista de necesidades del problema planteada y la gráfica.
- Planteen una gráfica de dependencias según el acuerdo al que se llego en el equipo y llenen la forma estrategia. Entregar la Forma llena al instructor.
- Posteriormente todo el grupo junto con el instructor, discutir las diferentes gráficas por equipo y llegar a un acuerdo de cual es la mejor opción, según las agrupaciones de las funcionalidades que se harán en el ciclo 1 y cuales en el ciclo 2.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- ¿Cómo fue la participación de tu equipo?
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 7**
  - Leer capítulo 4 del libro de Ingeniería de Software Pragmática (Fase de Estrategia).
  - Leer práctica 7.

- **Documentos impresos que se utilizarán en la práctica 7**
    - Dos formas Registro de Defectos (Individual).
    - Práctica 7.
- **Herramienta que se utilizarán en la práctica 7**
    - Un software que realice diagramas de gantt.

ALUMNO:
FECHA:
FASE: **Estrategia 2**

PRÁCTICA 17 ADMINISTRACIÓN DE LA CONFIGURACIÓN

OBJETIVOS

- Aprender hacer el Plan de la Configuración del software.
- Aprender hacer el Informe del estado de la configuración
- Aprender a dar seguimiento a los cambios del software.

INTRODUCCIÓN

La administración de la configuración es una práctica importante dentro de la Ingeniería de Software, ayuda a tener el control de los cambios y las versiones del producto, con esto llevar acabo un software con mejor calidad. El encargado de la administración de la configuración es el administrador de apoyo. Para la administración de la configuración, se llevan a cabo las siguientes actividades:

1. Plan de la configuración.
2. Llevar el control de los cambios.
3. Informe del estado de los cambios.

Plan de configuración

Incluye el conjunto de actividades para tener el control y consistencia de las partes del sistema a lo largo de su ciclo de vida. Para realizarlo se definen los productos del sistema, dándoles nombre y definiendo los responsables de cada producto, quien se encargará de aceptar y realizar los cambios, si el producto lo requiere.

Encargado: Administrador de apoyo, realiza la lista de productos, tomando los documentos que se han generado durante el proyecto, y define el responsable de cada uno.

En la tabla 3 o capítulo 4 del libro, se da una lista de componentes y sus responsables, en la cual se puede basar el administrador de apoyo para realizar el plan de la configuración.

Control de Cambios

Para llevar a cabo una buena administración de la configuración, en equipo se debe llegar a un acuerdo de cómo será el procedimiento para el control de cambios.

Un integrante del equipo crea un producto (responsable del producto), lo entregará al administrador de calidad para su revisión. Una vez revisado, en caso de requerir correcciones se regresa al responsable para que las realice y ya hechas pasa al administrador de calidad nuevamente, si lo aprueba, lo envía al administrador de apoyo para archivarlo en la carpeta.

Solicitud de cambios

En la elaboración de software, los componentes de los productos (documentación, código, etc.) cambiarán a través del ciclo de vida, por esto es que existe la solicitud de cambios.

Para llevar acabo un cambio a cualquier producto, en equipo deben llegar a un acuerdo de cómo será el procedimiento para la solicitud de algún cambio, que puede ser de la siguiente forma:

Procedimiento para efectuar cambios

- Un integrante del equipo solicita un cambio a un documento ya resguardado por medio de la solicitud de cambio.
- Esta solicitud es examinada por el Comité de Control de Cambios (el cual es definido por todos los integrantes del equipo) para su aprobación o rechazo.
- Si el cambio es aprobado, el responsable del documento realiza los cambios en una nueva versión del documento y lo envía al administrador de calidad para su revisión.
- El administrador de calidad revisa los cambios en el documento, si no requiere de correcciones se lo hace llegar al administrador de apoyo para que sea archivado en las carpetas.

Informe del estado de la Configuración

Con este documento cada integrante podrá saber los documentos que fueron generados, última versión y dirección en la que están disponibles. De esta forma todos están enterados cual es la última versión aprobada para que sea usada en el segundo ciclo.

Encargado: Administrador de apoyo, toma en cuenta las formas semanales del estado de los cambios, que sus compañeros han llenado a lo largo del semestre. Usa la forma de Informe del estado de la configuración, en la tabla escribe los nombres de los documentos ya generados la versión y la dirección en la que están guardados.

DESARROLLO

- En equipos hacer el Informe del Estado de la Configuración de su proyecto.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- ¿Cómo fue la participación de tu equipo?
- Lograron llegar acuerdos con facilidad.

Capítulo 7. Fase de Planeación

7.1 Fase de Planeación

- **Primer ciclo:** Su objetivo principal es hacer el plan de actividades del equipo para realizar el proyecto.

  Se explica a los alumnos lo que es la calidad, dándoles como técnica las revisiones entre colegas, las cuales realizarán en los equipos durante todo el semestre, utilizando la forma de registro de defectos encontrados.

- **Segundo ciclo:** Se utilizan las mediciones y experiencia adquirida en el primer ciclo, para que los alumnos realicen la planeación del segundo ciclo tomando en cuenta lo aprendido y los documentos que ya se tienen generados.

Algunas de las tareas que deben realizar durante esta fase son: definir el plan de trabajo para el primer ciclo, en el cual los alumnos tendrán que hacer una lista con las actividades que realizarán a lo largo de este ciclo, usando los diagramas de Gantt, para lo cual existen herramientas de software.

Se explica lo que son las revisiones entre colegas, que es una técnica de verificación de calidad. La usarán los alumnos a lo largo del semestre, para que los productos que generen sean de calidad.

7.2 Práctica 7

Los objetivos de esta práctica son que los alumnos aprendan a planear un proyecto e identificar y registrar los defectos en la forma Registro de defectos. Se les explican los conceptos y se les dan ejemplos para realizar diagramas de gantt, se les da una lista de software y su dirección en la red para que ellos los puedan bajarlas e instalarlas. Se les pide realizar un diagrama de Gantt personal y realizar una revisión entre colegas, llenando su forma de registro de defectos (se les da un ejemplo de cómo llenar la forma registro).

ALUMNO:
FECHA:
FASE: **Planeación**

PRÁCTICA 7 PLANEACIÓN DEL PROYECTO Y REGISTRO DE DEFECTOS

OBJETIVOS

- Aprender a planear un proyecto.
- Aprender a identificar y registrar defectos utilizando la forma de Registro de defectos.

INTRODUCCIÓN

En la metodología que se sigue en la materia, la planeación es un proceso iterativo incremental que termina cuando el proyecto se completa. Los objetivos del equipo son un factor importante que debe considerarse cuando se formula el plan.

Para cada ciclo, el instructor da una fecha de inicio y de entrega, con base en estas fechas los equipos realizan el plan del ciclo. En el segundo, los equipos usan la experiencia adquirida en el primero y mejoran la planeación. Para hacer el plan del proyecto se usan los diagramas de Gantt.

A lo largo de todo el proyecto, se realiza el seguimiento del plan. La información de los tiempos invertidos, que se registra en la forma semanal, se actualiza en el plan. En el transcurso del primer ciclo se va comparando el esfuerzo planeado con el real, se señalan las diferencias en tiempos, y se consideran en el segundo ciclo para una mejor planeación.

Dentro de la ingeniería de software, la calidad es algo muy importante ya que el software que se desarrollará debe ser de la mejor calidad. Para alcanzar este objetivo, es necesaria una actitud de compromiso por parte de todos los integrantes del equipo en todas las fases.

Calidad: Conjunto de propiedades y características de un producto, que le confiere una aptitud para satisfacer necesidades explícitas o implícitas del cliente. Dentro de la calidad del software, entra la claridad y exactitud de la documentación y de todos los productos que se entregarán a lo largo de los ciclos. Los autores de cada documento pueden encontrar sus defectos y corregirlos.

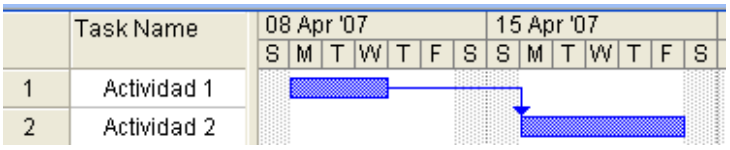
Revisión entre colegas: Técnica para entregar un producto sin defectos, un integrante diferente al autor de los productos a revisar, los revisa y encuentra los defectos, de esta forma se entrega un producto con la mejor calidad y le menor cantidad de defectos posible.

Defecto: Es una característica no conforme con los requerimientos de la especificación del producto, que puede afectar negativamente la calidad de éste.

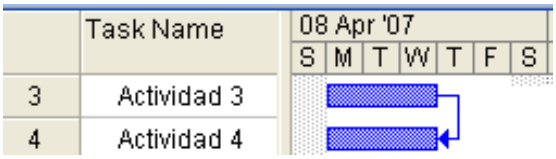
Diagrama de Gantt: Herramienta para la planeación de actividades, distribuye las actividades conforme a un calendario, se puede ver el periodo de duración de cada una, fechas de inicio y fin, el tiempo total para cada actividad, proporciona el porcentaje de avance de cada actividad, el adelanto o retraso conforme a lo previsto.

Cada actividad se representa mediante un bloque rectangular, cuya longitud indica su duración. La posición de cada bloque, indica el inicio y fin de la actividad a la que corresponde. Las actividades pueden o no depender de otras. A continuación se muestran las dependencias existentes:

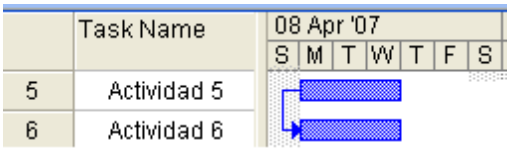
Fin-inicio: Se representan alineando el final del bloque de la actividad predecesora (actividad1) con el inicio del bloque de la actividad dependiente (actividad2).



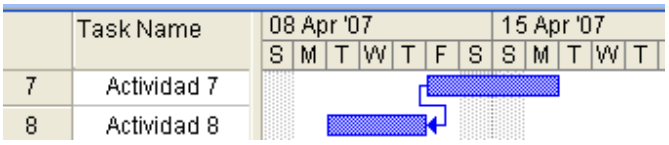
Final-final: Se representan alineando los finales de los bloques de las actividades predecesora (actividad3) y dependiente (actividad4).



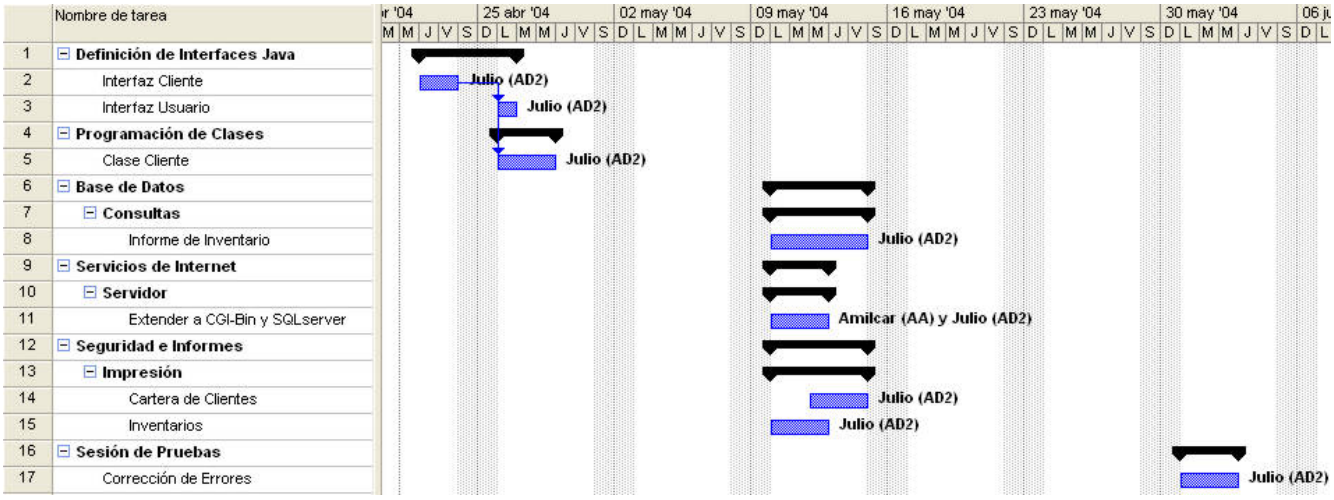
Inicio-inicio: se representan alineando los inicios de los bloques de las actividades predecesora (actividad5) y dependiente (actividad6).



Inicio-fin: se representan alineando el inicio del bloque de la actividad predecesora (actividad7) con el fin del bloque de la actividad dependiente (actividad8).



Ejemplo: De un diagrama de Gantt.



Los diagramas de Gantt son una técnica común para representar actividades en Ingeniería de Software, por lo que existen herramientas comerciales y de licencia libre para representarlos.

Software libre para trabajar con diagramas de Gantt

- TaskJuggler, gestión de proyectos para KDE
  http://www.taskjuggler.org/
- GanttProject, es una aplicación open source escrita en Java
  http://ganttproject.biz/
- Planner, una aplicación para Linux
  http://www.simpleprojectmanagement.com/planner/home.html
- dotProject, software de gestión de proyectos multiplataforma
  http://www.simpleprojectmanagement.com/planner/home.html

Software Comercial para trabajar con diagramas de Gantt

- Project, actualmente Microsoft Project es líder en diagramas de Gantt
  http://office.microsoft.com/es-mx/project/default.aspx

En la página del curso, sección herramientas, se encuentran los enlaces a páginas con software para esta fase. En la sección herramientas encontraran una lista con las fases, en la fase de planeación encontrar un tutorial de GanttProject que les ayudará a instalar y usar el programa.

Reunión de Revisión entre colegas

En estas reuniones cada integrante tiene diferentes actividades según su rol:

Administrador de calidad:

- Establece fecha y hora de la revisión, selecciona participantes, recibe el producto a revisar, preparar la lista de verificación.
- Dirige la reunión, asegura que todos hicieron las revisiones individuales a los productos, discute los defectos encontrados los registra en la forma de Registro de defectos.
- Asegura que se corrija el producto y realiza la revisión final.

El equipo de revisión:

- Revisa de forma individual el producto apoyándose en la lista de verificación.
- Registra los defectos en la forma de Registro de defectos.

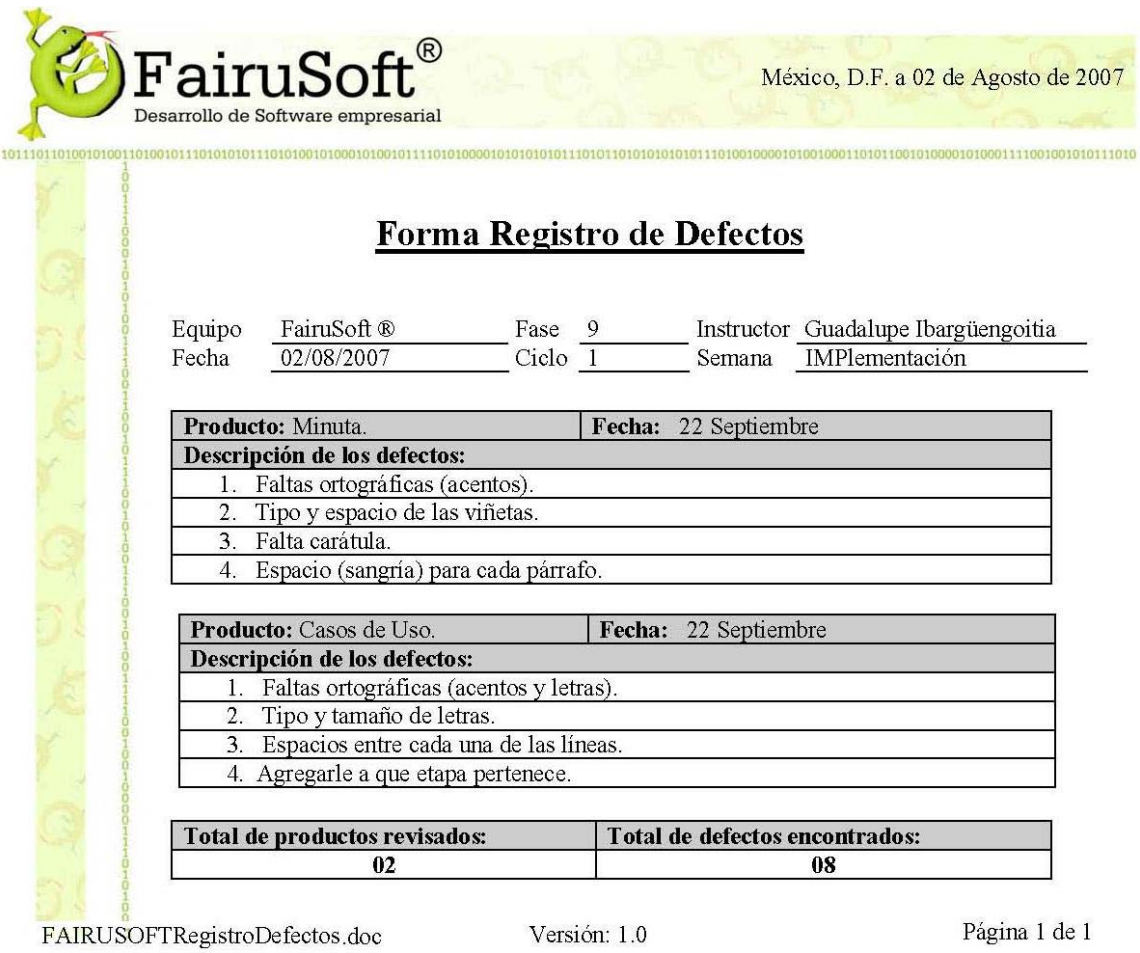
El propietario del producto:

- Discute, analiza y corrige los defectos encontrados en la reunión de revisión.
- Entrega el producto corregido al administrador de calidad.

Forma Registro de Defectos

Como su nombre lo dice, sirve para registrar los defectos encontrados en los documentos, durante las revisiones entre colegas y en general al revisar algún documento. Se anotan los defectos en dicha forma, en revisiones posterior se verifica que hayan sido corregidos satisfactoriamente.

A continuación se da un ejemplo de una forma de registro de Defectos llena:



FairuSoft®
Desarrollo de Software empresarial

México, D.F. a 02 de Agosto de 2007

Forma Registro de Defectos

Equipo	FairuSoft ®	Fase	9	Instructor	Guadalupe Ibargüengoitia
Fecha	02/08/2007	Ciclo	1	Semana	IMPlementación

Producto: Minuta.	**Fecha:** 22 Septiembre
Descripción de los defectos:	
1. Faltas ortográficas (acentos).	
2. Tipo y espacio de las viñetas.	
3. Falta carátula.	
4. Espacio (sangría) para cada párrafo.	

Producto: Casos de Uso.	**Fecha:** 22 Septiembre
Descripción de los defectos:	
1. Faltas ortográficas (acentos y letras).	
2. Tipo y tamaño de letras.	
3. Espacios entre cada una de las líneas.	
4. Agregarle a que etapa pertenece.	

Total de productos revisados:	**Total de defectos encontrados:**
02	**08**

FAIRUSOFTRegistroDefectos.doc Versión: 1.0 Página 1 de 1

DESARROLLO

- De manera individual hacer un diagrama de Gantt, con todas las actividades que hasta este momento saben que deben hacer a lo largo de este semestre en la materia de Ingeniería de Software, según el guión del curso.
- Reunirse con un colega de su equipo y llevar acabo una revisión entre colegas del diagrama de Gantt que realizaron, registrar los defectos encontrados en la forma registro de defectos y corregirlos.
- Enviar los diagramas de gantt por correo al instructor y entregarle las formas de registro de defectos llenadas.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 8**

    - Leer capítulo 6 (Fase Captura de Requerimientos).
    - Leer práctica 8.
- **Documentos impresos que se utilizarán en la práctica 8**
    - Práctica 8.
- **Herramientas que se utilizarán en la práctica 8**
    - Un software para modelar con UML.

Capítulo 8. Fase de Especificación de Requerimientos

8.1 Fase de Especificación de Requerimientos

- **Primer ciclo:** La fase requiere de dos semanas, ya que los conceptos deben quedar claros para los alumnos y deben tener tiempo suficiente para crear los documentos que se requieren.

  Sus objetivos son establecer los requerimientos del sistema. Con estos requerimientos los ingenieros de desarrollo y el cliente se entenderán más entre sí, logrando una mayor comunicación. Se realiza también el plan de pruebas del sistema según los requerimientos que fueron especificados y el prototipo de interfaz de usuario.

- **Segundo ciclo:** Los alumnos utilizan el trabajo realizado en el primer ciclo, complementándolo con las funcionalidades que ellos mismos decidieron dejar para el segundo ciclo.

Para que tanto el cliente como los ingenieros de desarrollo se entiendan mejor, se les explica a los alumnos diferentes documentos que se realizarán, entre los cuales está realizar un glosario con los términos que resulten desconocidos para el cliente, un documento con el texto del problema que realizarán, etc.

Se explica como elaborar los casos de uso (diagrama general y detallado), el prototipo de interfaz de usuario con el cual el cliente podrá entender de una mejor manera lo que el programa realizará y el plan de pruebas del sistema, en el que se especifica el comportamiento esperado del sistema en casos normales y excepcionales, para que al final del ciclo se pueda probar el sistema.

8.2 Práctica 8

Los objetivos de esta práctica son que los alumnos aprendan a realizar los diagramas de casos de uso (general y detallados). Se les explica a los alumnos que es UML, se les da un ejemplo de un problema con el diagrama general y un caso de uso detallado. Ellos realizan el diagrama de casos de uso general y algunos detallados del problema de su proyecto.

UML: Lenguaje de modelado, utilizado para expresar un diseño gráfico estandarizado. Proporciona herramientas en las cuales se realizas diagramas de casos de uso, de clases, de estados, de secuencias, de actividades, de colaboraciones, de distribución, etc. Se utiliza en las grandes empresas que realizan software.

8.3 Práctica 9

El objetivo de esta práctica es que los alumnos aprendan a realizar su prototipo de interfaz de usuario usando HTML. Se les da la opción a los alumnos de utilizar alguna otra herramienta que ellos conozcan. Se les explica que es y como realizar el prototipo, algunas herramientas que pueden utilizar y un ejemplo con líneas de código y el resultado de esas líneas. Al final deben realizar el prototipo de los caso de uso que hicieron en la práctica 8.

8.4 Práctica 10

El objetivo de esta práctica es que los alumnos aprendan a especificar los requerimientos no funcionales y a desarrollar el plan de pruebas del sistema según sus requerimientos. Se les explica el concepto y se les da ejemplos de requerimientos no funcionales y de plan de pruebas del sistema.

ALUMNO:
FECHA:
FASE: **Especificación de Requerimientos**

PRÁCTICA 8 DIAGRAMA DE CASOS DE USO

OBJETIVOS

- Aprender hacer los diagramas de casos de uso
- Aprender a definir de manera detallada los casos de uso

INTRODUCCIÓN

La fase de Especificación de Requerimientos cumple un papel primordial en el proceso de producción de software, se enfoca en la definición de lo que se desea producir. Una de las tareas consiste en la generación de especificaciones del comportamiento del sistema, descritas con claridad, sin ambigüedades y en forma consistente y compacta; de esta manera se pretenden minimizar los problemas relacionados al desarrollo de sistemas.

La técnica para captura de requerimientos funcionales que se usara son los diagramas de Casos de Uso de UML, herramienta muy útil para esta fase. Los diagramas de casos de uso son sencillos de entender para el usuario, pero también muy generales, por esto se deben detallar, expresando con claridad lo que los clientes desean y necesitan, de esta forma el usuario entienda el comportamiento del sistema. Una vez identificadas las principales funcionalidades en los casos de uso, se detallan cada una, haciendo un diagrama amplificado de cada caso de uso general que lo amerite.

Lenguaje de Modelado Unificado (UML)

Cuenta con un conjunto de modelos, que permite analizar y diseñar sistemas orientados a objetos. Si nos imaginamos una caja de herramientas con su martillo, pinzas, clavos y destornillador, es semejante a la caja de herramientas de UML que contiene varios tipos de diagramas:

- Diagrama de casos de uso
- Diagrama de clases
- Diagrama de estados
- Diagrama de secuencias
- Diagrama de actividades
- Diagrama de colaboraciones
- Diagrama de distribución

Siguiendo con la analogía, si vamos a colgar un cuadro no usamos todas las herramientas de la caja, sólo usamos el martillo y clavo. Lo mismo pasa con UML, dependiendo de la tarea, se usa la herramienta adecuada. En esta práctica aprenderemos a usar los diagramas de Casos de Uso y a detallarlos. A continuación se presentan los elementos de un diagrama de casos de uso.

Diagrama de Casos de uso

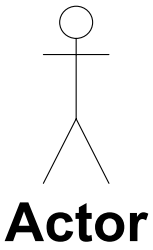
Permiten definir los límites del sistema y las relaciones entre el sistema y el entorno, dividen el conjunto de necesidades atendiendo a la categoría de usuarios que participan en el mismo. Están basados en el lenguaje natural, ya que usan el lenguaje del usuario.

Los elementos de los diagramas de casos de uso son “muñecos y óvalos”. Los muñecos representan a los usuarios y los óvalos o las funciones del sistema. Se dibuja un muñeco por cada usuario y una óvalo por cada caso de uso y se enlazan con líneas.

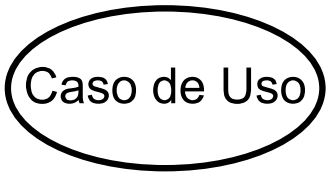
Actores (Muñecos): Rol que un usuario juega con respecto al sistema, no necesariamente representa a una persona, representa la labor que realiza frente al sistema. Tipos de actores:

- **Usuarios:** Son las personas que usan el sistema.
- **Otros sistemas:** Son los sistemas con los que el sistema interactúa.
- **Tiempo:** Puede ser un momento en específico.

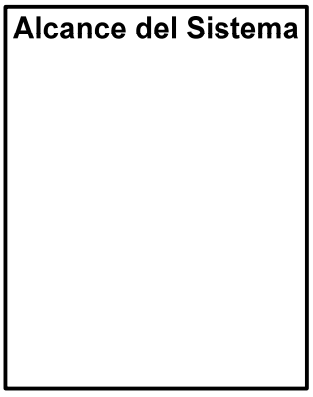
Un mismo actor puede tener variantes, el nombre del actor describe el papel desempeñado:



Casos de Uso (Óvalos): Funcionalidades del sistema y regresan un valor o servicio al actor. El proceso de desarrollo estará dirigido por los casos de uso. Se denota por un óvalo, con el nombre del caso de uso. Se relaciona cada actor con sus casos de uso por una (una línea):



Alcance del sistema: Casos de uso que se desarrollarán para el sistema en cada ciclo. Se denota por un cuadrado que abarca los casos de uso dentro el diagrama de casos de uso:



A continuación se da un ejemplo de un Diagrama de Casos de Uso. Retomando el ejemplo de la práctica 6, se da el diagrama de casos de uso general, otros casos de uso, que se explican más adelante con el Detalle de los Casos de Uso.

Ejemplo:

Problema: Librería Electrónica

Supón que tienes que desarrollar un sistema de software para apoyar al administrador de una librería por Internet a consultar, dar de alta y baja los artículos que venden. El sistema lo podrá usar también algún posible cliente que quiera consultar los productos de la librería a través de Internet. La librería vende libros, CD de música, y DVD de películas. El administrador deberá poder identificarse para entrar al sistema. Las consultas podrán hacerse por diversos campos, por ejemplo título, autor o editorial. Se podrá sacar el inventario de los artículos en existencia y hacer reportes de faltantes de algún artículo cuando se terminen las existencias. El sistema se desarrollará en java.

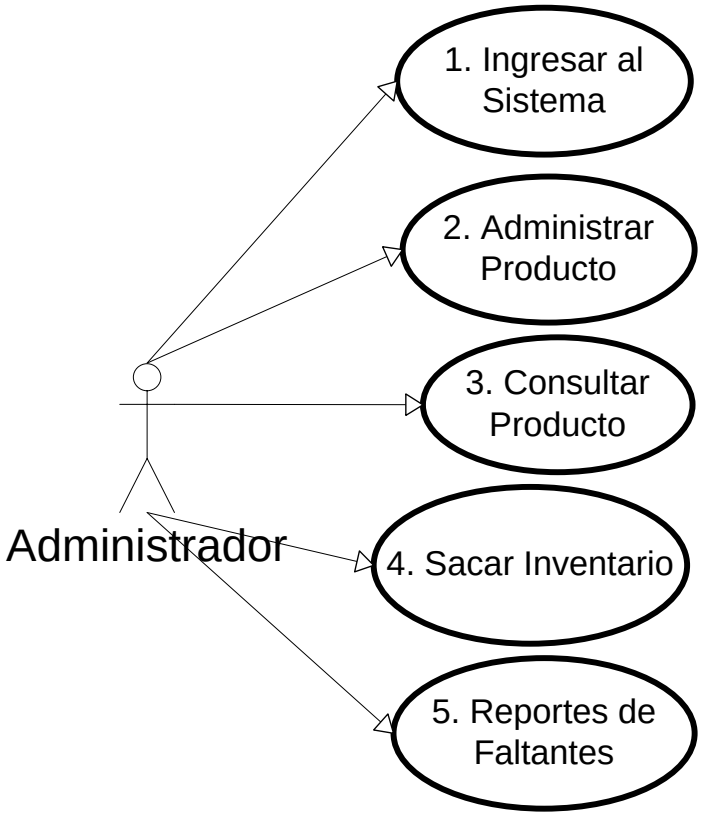
Paso 1: Analicemos el problema, existen dos actores, administrador y cliente. El administrador podrá:

- Ingresar al Sistema
- Hacer consultas por productos para verificar la existencia de algún producto.
- Dar de alta y baja algún producto, es decir administrar los productos.
- Sacar el inventario de los artículos existentes
- Emitir reportes de faltantes.

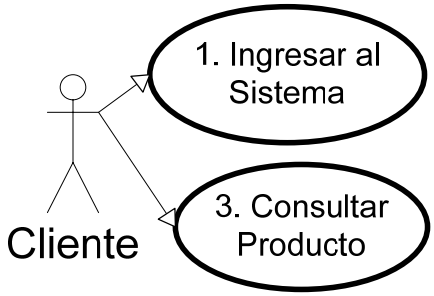
El cliente podrá:

- Ingresar al Sistema
- Consultar la existencia de algún producto.

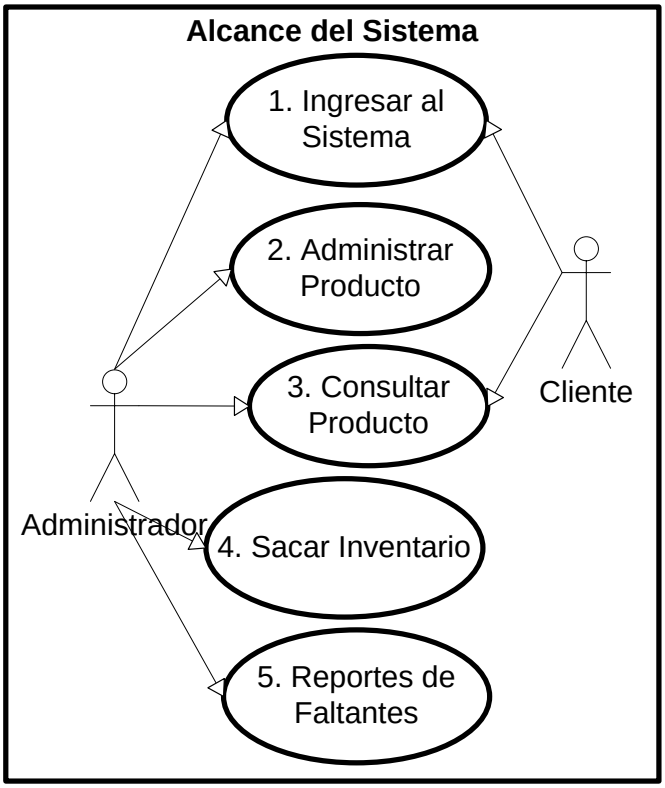
Paso 2: Construir los diagramas de caso de uso. Diagrama de Casos de Uso del Administrador:



- Diagrama de Casos de Uso del Cliente:



- Diagrama Generar de Casos de Uso del Alcance del Sistema de librería Electrónica



Detalle de los Casos de Uso

La descripción del caso de uso puede incluir lo siguiente:

Identificador: Número que le corresponde al caso de uso, ya que cada uno se numera.

Nombre: Por lo general es un verbo en infinitivo, representativo de la funcionalidad del caso de uso.

Actor: Nombre del actor encargado de iniciar la acción o que recibe el resultado del casos de uso.

Diagrama detallado: Un diagrama del caso de uso indicando el actor y sus variantes.

Descripción: Texto breve describiendo la acción, reseña del caso de uso, explica actividades que tienen lugar en el caso de uso.

Precondiciones: Estado que el sistema debe presentar antes de que el caso de uso sea ejecutado.

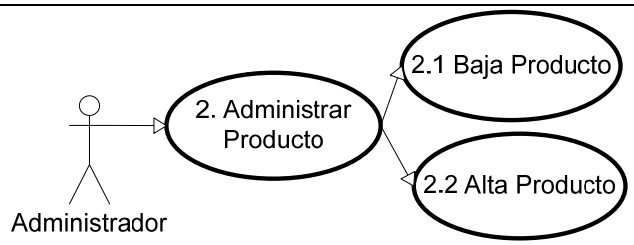
Flujo de Eventos: Texto que describe de forma breve cómo inicia el caso de uso, considerando que empieza la interacción entre el actor y el sistema. Existen dos tipos de eventos:

- **Eventos normales**: Se describen las actividades que ocurren entre sistema y actor durante el caso de uso, en condiciones ideales, para que se obtenga algún resultado para el actor.

- **Eventos alternativos o excepcionales**: Se describen las actividades que ocurren en situaciones anormales o excepcionales, entre sistema y actor durante el caso de uso. No necesariamente tienen que existir flujos alternativos.

Poscondiciones: Define el estado del sistema después de la terminación exitosa del caso de uso.

Ejemplo: Diagrama Detallado del caso de uso Administrar Producto.

<table>
<tr><td>Identificador y nombre:</td><td>2. Administras Producto</td></tr>
<tr><td>Actor:</td><td>Administrador</td></tr>
<tr><td>Diagrama detallado:</td><td> 
</td></tr>
<tr><td>Descripción:</td><td>El administrador podrán administrar el producto, dándolo de baja o alta</td></tr>
<tr><td>Precondiciones:</td><td>Haber entrado al sistema con la clave de administrador.</td></tr>
<tr><td>Flujo:</td><td></td></tr>
</table>

Actor		**Sistema**		
Paso	Acción	Paso	Acción	Excepción
1	Haber seleccionado la opción de administrar producto de la página principal.	2	Se muestra la pantalla con las opciones de dar de alta o baja el producto.	E1

Excepciones:

ID	Nombre	Acción
E1	No se muestran las opciones de alta o baja producto.	El administrador no entró con la clave correcta.

Poscondiciones:	El administrador ha dado de alta o baja un producto.

Tips para casos de Uso

- Describa solamente los eventos visibles para el actor: Lo que el actor hace, lo que el sistema hace en respuesta.
- Haga que los casos de uso proporcionen un resultado de valor para el usuario.
- Los casos de uso pueden tener distintos niveles de precisión. Detalle hasta que todos los involucrados tengan un conocimiento común desde su propia perspectiva.
- Utilice términos y vocabulario común y un lenguaje preciso.

Existen muchas herramientas para el modelado en UML algunas de las cuales son libres:

- StartUML http://staruml.sourceforge.net/en/
- ArgoUML http://argouml.tigris.org/

Software Comercial para modelar con UML:

- Visio http://office.microsoft.com/es-mx/visio/default.aspx

En la página del curso, sección herramientas, se encuentran enlaces a páginas con software pera el modelado con UML. En la sección se herramientas, en fase de diseño hay un tutorial de ArgoUML que les ayudará a instalar y usar el programa.

DESARROLLO

- Reunirse con sus equipos, leer y discutir el problema anterior y los diagramas realizados para el ejemplo y preguntar al ayudante si tienen dudas.
- Con tu equipo realiza el diagrama de casos de uso general y dos diagramas de casos de uso detallados, del problema que se te dio al inicio del semestre.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 9**
  - Leer capítulo 6 (Fase de Captura de Requerimientos).
  - Leer práctica 9.
  - Leer un tutorial de HTML
    - http://html.conclase.net/tutorial/html/
    - http://www.webestilo.com/html/cap4a.phtml
    - http://html.conclase.net/w3c/html401-es/index/list.html#f
    - Libro Creating Web Pages All-In-One Desk Reference For Dummies, 2ed (Wiley, 2004) (disponible en la página del curso).
- **Documentos impresos que se utilizarán en la práctica 9**
  - Tutorial de HTML (no necesario si ya sabes html o alguna otra herramienta con la que harás tu sistema)
  - Práctica 8 resuelta (casos de uso)
  - Práctica 9.
- **Herramientas que se utilizarán en la práctica 9**
  - Un software para realizar su prototipo de interfaz de usuario.

ALUMNO:
FECHA:
FASE: **Especificación de Requerimientos**

PRÁCTICA 9 PROTOTIPO DE INTERFAZ DE USUARIO

OBJETIVO

- Aprender a realizar un prototipo de interfaz de usuario usando HTML.

INTRODUCCIÓN

En un proyecto de software, es muy importante que el cliente esté seguro de qué es lo que quiere y necesita, así como de que el producto que se le entregará cumplirá con todas sus expectativas.

Cuando la descripción del proyecto, casos de uso y detalles, no son lo suficientemente claros y buenos para expresar los requisitos, se usa un prototipo de interfaz de usuario, el cliente entenderá completamente lo que hará el software y si cumplirá sus expectativas. Se construye con la participación del usuario final, esta es la mejor forma de desarrollar la interfaz ya que así los usuarios están implicados en la construcción del software.

Prototipo de Interfaz

Modelo del sistema, se construye para comprender mejor el problema y sus posibles soluciones, como son el evaluar mejor los requisitos y probar opciones de diseño. En los prototipos se pueden incorporar algunas o todas las características del sistema final, dependiendo del alcance en cada ciclo. Con el prototipo el cliente podrá darse cuenta de lo que su software será capaz de hacer y así estar convencido de qué es lo que requiere.

El prototipo se presenta al cliente para ayudarlo a establecer claramente los requisitos, necesidades y deseos. También ayuda a los ingenieros de desarrollo a validar la corrección de las especificaciones, conocer problemas que se puedan presentar durante la implementación del sistema, mejorar la calidad del producto y examinar viabilidad y utilidad de la aplicación.

Pasos para desarrollar interfaces de usuario

1. Conocer al usuario.
2. Aplicar los principios del buen diseño de pantallas que son:
    a. Crear una pantalla de inicio y en cada una de las demás pantallas poner una opción para poder regresar al inicio
    b. Incluir una pantalla de acceso si se requiere identificar al usuario
    c. Evitar las interfaces innecesarias.
3. Seleccionar el tipo adecuado de ventanas para cada caso de uso algunos tipos son: información, captura, etc.
4. Desarrollar los menús del sistema según el diagrama general de casos de uso. Para cada actor y caso de uso, crear las ventanas que permitan cubrir sus funcionalidades.
5. Organizar y distribuir la pantalla.

6. Elegir los colores adecuados.
7. Crear íconos fáciles de entender.
8. Crear ventanas y mensajes para los errores y situaciones excepcionales descubiertas en los detalles de los casos de uso.

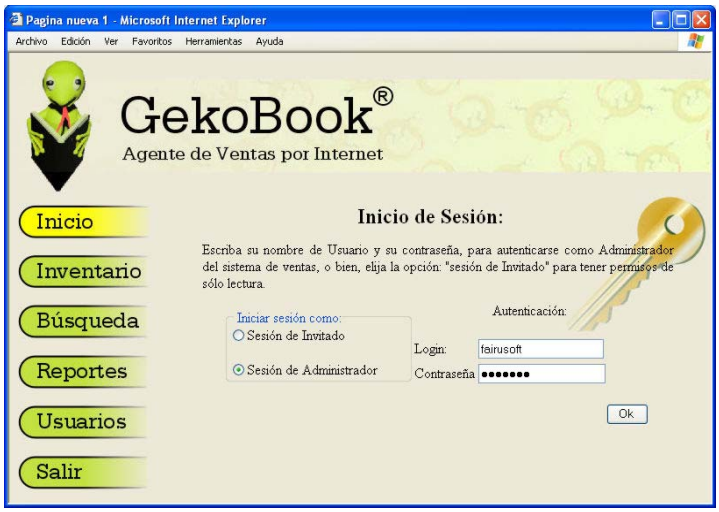
Herramientas para desarrollar el prototipo de interfaz

- Editores de páginas Web
- Bocetos en papel
- Aplicaciones de dibujo (PowerPoint, Saint, CorelDraw, etc.)

Ejemplo: Prototipo de la Página Principal, del ejemplo de la práctica 6, en código HTML y pantalla.

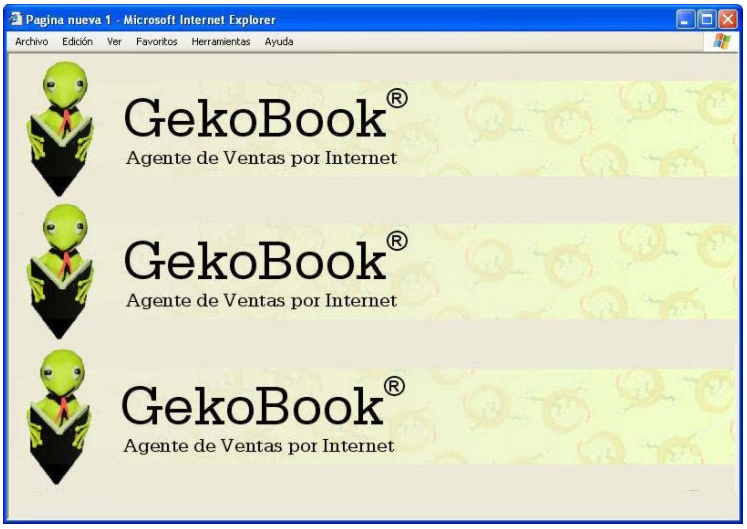
Código principal: Se divide la pantalla principal en varias zonas (frames), cada zona puede tener un contenido independiente de otra, cada zona es asimismo un frame. Cada frame tiene su código.

```
<html>
   <frameset rows="157,*" NORESIZE BORDER=0>
      <frame name="titulo" src="titulo.html" scrolling="auto">
      <frameset cols="179,*" NORESIZE BORDER=0>
         <frame name="indice" src="indice.html" scrolling="no">
         <frame name="principal" src="principal.html">
      </frameset>
   </frameset>
</html>
```



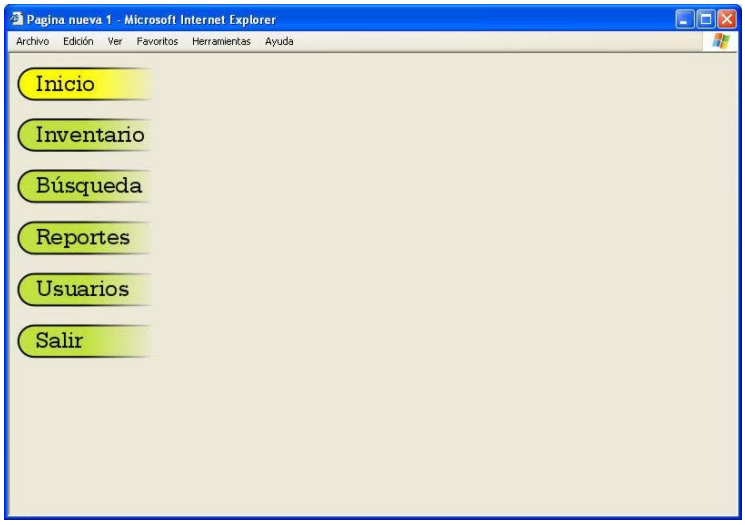
Código del frame titulo.html

```
<html>
   <body bgcolor="#ECE9D8" background="imagenes/titulo.jpg">
   </body>
</html>
```



Código del frame indice.html

```
<html>
   <head>
      <script language="JavaScript"> </script>
   </head>
   <body bgcolor="#ECE9D8">
      <p><a href="principal.html" alt="Página de Autenticación" target="principal">
            <img id="img" border="0" src="imagenes/inicio1.jpg" alt="inicio" border="0"
              onmouseover="enfoca(0)" onmouseout="desenfoca(0)" onclick="fija(0)"
              width="150" height="38"></a></p>
      <p><a href="inventario.html" alt="Página de Autenticación" target="principal">
            <img id="img" border="0" src="imagenes/inventario1.jpg" alt="inventario"
             border="0" onmouseover="enfoca(1)" onmouseout="desenfoca(1)"
             onclick="fija(1)" width="150" height="37"></a></p>
      <p><a href="busqueda.html" alt="Página de Autenticación" target="principal">
            <img id="img" border="0" src="imagenes/busqueda1.jpg" alt="busqueda"
             border="0" onmouseover="enfoca(2)" onmouseout="desenfoca(2)"
             onclick="fija(2)" width="150" height="38"></a></p>
      <p><a href="reportes.html" alt="Página de Autenticación" target="principal">
            <img id="img" border="0" src="imagenes/reportes1.jpg" alt="reportes"
             border="0" onmouseover="enfoca(3)" onmouseout="desenfoca(3)"
             onclick="fija(3)" width="150" height="38"></a></p>
      <p><a alt="Página de Autenticación" target="principal" href="usuarios.html">
            <img id="img" border="0" src="imagenes/usuarios1.jpg" alt="usuarios"
            border="0" onmouseover="enfoca(4)" onmouseout="desenfoca(4)"
            onclick="fija(4)" width="150" height="38"></a></p>
      <p><a href="javascript: top.document.write('<HTML>Adiós</HTML>')" alt="Página de
            Autenticación" target="principal"> <img border="0" src="imagenes/salir.jpg"
            border="0" width="150" height="38"></a></p>
   </body>
</html>
```



Código del frame principal.html

```
<html>
   <body bgcolor="#ECE9D8">
      <p><h2 align="center">Inicio de Sesión:<img border="0" src="imagenes/llave.gif"
         align="right" style="position:absolute; top:2; left:450" hspace="0" width="167" height="160"></h2>
         <p style="position:absolute; left:41; top:56; width:538; height:57"
          align="justify">Escriba su nombre de Usuario y su contraseña, para
          autenticarse como Administrador del sistema de ventas, o bien, elija la
          opción: &amp;quot;sesión de Invitado&amp;quot; para tener permisos de sólo
          lectura. </p> </p>
      <form method="POST" action="_derived/nortbots.htm" style="position:absolute;
          left:42; top:125; width:535; height:137"
          onSubmit="location.href='_derived/nortbots.htm';return false"
          webbot-onSubmit webbot-action="--WEBBOT-SELF--">
         <input TYPE="hidden" NAME="VTI-GROUP" VALUE="0"><div align="center">
            <center><table border="1" cellpadding="0" cellspacing="0"
                        style="border-collapse: collapse; border-width: 0"
                        bordercolor="#111111" width="90%" id="AutoNumber1">
               <tr><td width="44%" style="border-style: none; border-width: medium">
                     <fieldset style="padding: 2"> <legend>Iniciar sesión como:</legend> <input
                        type="radio" value="V1" name="R1">Sesión de Invitado<p> <input type="radio"
                        name="R1" value="V2" checked>Sesión de Administrador</p>
                  </fieldset></td>
                  <td width="74%" style="border-style: none; border-width: medium">
                     <p align="center">Autenticación: </p>
                     <table width="179"> <tr><td width="37">Login:</td><td width="159">
                        <input type="text" name="T1" size="20"></td>
                        <tr><td width="37">Contraseña</td><td width="159">
                        <input type="password" name="T2" size="20"></td> </table>
                  </td> </tr> <tr>
                  <td width="44%" style="border-style: none; border-width:
                  medium">&amp;nbsp;</td>
                  <td width="74%" style="border-style: none; border-width: medium"><br>
                  <input type="button" value="  Ok   " name="B1" style="float: right"></td>
               </tr> </table> </center> </div> </form> <p>
&amp;nbsp;</p> </body> </html>
```

DESARROLLO

- Con sus equipos de trabajo, hacer un prototipo de interfaz de usuario con HTML, para los casos de uso que hiciste la práctica anterior y enviar el documento al ayudante.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 10**
  - Leer capítulo (Fase de Captura de Requerimientos).
  - Leer práctica 10.
- **Documentos impresos que se utilizarán en la práctica 10**
  - Práctica 8 y 9 resuelta (casos de uso y prototipo de interfaz de usuario)
  - Tabla 4. Plan de Pruebas del Sistema
  - Práctica 10.

ALUMNO:
FECHA:
FASE: **Especificación de Requerimientos**

PRÁCTICA 10 REQUERIMIENTOS NO FUNCIONALES Y PLAN DE PRUEBAS DEL SISTEMA

OBJETIVOS

- Aprender a encontrar los requerimientos no funcionales de un sistema.
- Aprender a desarrollar el plan de pruebas de un sistema según sus requerimientos.

INTRODUCCIÓN

Los requerimientos funcionales y no funcionales se deben tener presente durante todo el desarrollo del proyecto, muchas veces solo se les toma en cuenta en esta fase, en ocasiones esto conlleva al fracaso de los proyectos, por retardo en la entrega, aumento en el presupuesto, etc. Por esto se deben capturar los requerimientos no funcionales y cumplir con las demandas del cliente y dejarlo satisfecho con el producto que solicitó.

También se vera el plan de pruebas del sistema. Las pruebas tienen como objetivo, verificar que el sistema que se entregará al cliente, cumple los requerimientos. De esta forma el plan de pruebas del sistema tiene como objetivo identificar las pruebas que se harán para verificar el sistema.

Requerimientos no funcionales

Requerimientos de equipo y software para el desarrollo del proyecto, las restricciones, cualidades y aspectos visibles para el usuario final, que el cliente pide que tenga el diseño del sistema. No prestan funcionamiento real al sistema, es decir, no están relacionados de forma directa con el comportamiento funcional del sistema, algunos de estos son:

- Usabilidad (basada en los usuarios esperados)
- Rendimiento (velocidad, tamaño, seguridad, disponibilidad)
- Requisitos operacionales (ambiente esperado de uso)
- Mantenibilidad y portabilidad (tolerancia a cambios)
- Seguridad (seguridad, confidencialidad, integridad)
- Documentación (documentación requerida, destinatarios, tipo de documentación técnica)
- Consideraciones de hardware (compatibilidad con otro hardware, existencia de otro sistema)
- Características de ejecución (usuarios concurrentes, carga de trabajo)
- Gestión de errores y excepciones
- Cuestiones de calidad (fiabilidad, disponibilidad, robustez)
- “Look and feel” (apariencia deseada, que las pantallas sean de un color x, que muestre en pantalla quien está en sesión en el sistema, o que permita ver los usuarios conectados, etc.)

Ejemplo: Requerimientos no funcionales según los ejemplos anteriores:

Requerimientos no funcionales

Look and feel

- Fondo: Color gris tenue con el logotipo del "Gueko" de Fairusoft como fondo de agua.
- Tipo de letra: Times New Roman.
- El nombre del sistema aparecerá en la parte superior de las pantallas centrado.
- El menú principal estará en la parte izquierda de la pantalla como un Frame.

Requerimientos de usabilidad

- El sistema estará orientado hacia personas que tengan conocimientos básicos de computación como el uso de un Sistema Operativo, el uso de un navegador de Internet, etc.

Requerimientos operacionales

- Sistemas Operativos: Windows, Linux, Unix, en general cualquier sistema operativo que soporte la ejecución de alguno de los navegadores más populares, así como soporte para la Maquina Virtual de Java. Para el caso del Sistema Operativo de la computadora que funciona como servidor será el un sistema Windows.
- Equipo físico:
  - Cualquier computadora que pueda ejecutar un navegador para Internet.
  - Para el caso de la computadora que funciona como servidor, recomendamos una maquina con las siguientes características.
    - Windows XP
    - Un manejador de bases de datos como SQL Server
    - Disco Duro 80 GB.
    - 512 Mb en RAM.
    - Java Virtual Machina.
    - Tarjeta de Red.
    - Conexión a Internet.

Requerimientos de ejecución

- El tiempo de respuesta no es critico y dependerá de la velocidad de conexión entre la maquina que funcione como servidor y la maquina cliente. Para la búsqueda de algún producto en la base de datos el tiempo de respuesta, también será afectado por la velocidad de hacer búsquedas del manejador de bases de datos.

Requerimientos de Portabilidad.

- Dado que el sistema se desarrollara en Java, será altamente portable tanto como Java.

Extensibilidad:

- Dado que el sistema será desarrollado en Java nos permitirá programar orientado a objetos lo que hace al sistema altamente Extensible.

Seguridad:

- La seguridad del sistema consistirá en pedir un nombre y una contraseña para que un administrador pueda ejecutar funciones que solo le corresponden a él.

Plan de Pruebas del sistema

Consta de la definición de casos para probar el sistema cuando esté terminado, posteriormente se usarán para aplicar las pruebas al sistema. Se hace una tabla por cada caso de uso, cada tabla consta de dos columnas:

- **Caso de prueba o entradas:** En esta columna se escribe la acción en la cual interactúa el sistema y el usuario.
- **Resultado esperado:** Aquí van los resultados que se esperan que tenga la prueba, los cuales permitirán evaluar si la prueba se superó con éxito o no.

Ejemplo: Plan de Pruebas según los casos de uso de la práctica 6:

PLAN DE PRUEBA DE SISTEMA

Ciclo 1

Caso de prueba PS1

Caso de uso	Entradas	Resultados esperados
Inicio		
Administrador	Seleccionar administrar	Se muestra la pantalla de Registro de Administrador
	Nombre y contraseña correctos	Entra al sistema y manda a la página principal del sistema
	Nombre correcto y contraseña incorrecta	Error, la contraseña no es válida
	Nombre incorrecto y contraseña incorrecta	Error, la contraseña no es válida
	Nombre incorrecto	Error, la contraseña no es válida
Mantenimiento de artículos		
Alta de Artículo	Seleccionar Alta de artículo	Presenta la pantalla de alta
	Ingresar datos para alta del artículo correctamente	Guarda el artículo asignándole una clave
	Ingresar datos incorrectamente	Error, "El tipo de dato no es correcto."
	Ingresar datos que ya existen	Error, "Ya existe artículo"
Baja Artículo	Seleccionar Baja de artículo	Presenta pantalla de baja
	Ingresar clave del artículo correctamente	Elimina el artículo
	Ingresar incorrectamente la clave del artículo	Error, "La clave no existe."
Cambio Artículo	Seleccionar Cambio Artículo	Presentar la pantalla de cambio
	Ingresar clave del artículo correctamente	Mostrar pantalla donde se ingresarán los nuevos datos del artículo
	Ingresar clave del artículo incorrectamente	Error, "No existe el artículo"
	Ingresar nuevos datos del artículo correctamente	Cambia los datos del artículo.
	Ingresar nuevos datos del artículo incorrectamente	Error, "El tipo de Dato no es correcto"o "No se lleno la casilla"
Consulta Artículo	Seleccionar Consulta Artículo	Presenta pantalla de consulta
	Ingresar clave del artículo correctamente	Muestra en pantalla los datos del artículo

	Ingresar clave del artículo incorrectamente	"No existe el artículo"
Salir	Seleccionar Salir	Sale de la pantalla al menú principal
Mantenimiento de Cliente		
Alta Cliente	Seleccionar Alta Cliente	Presenta pantalla de alta Cliente
	Ingresar datos para alta de cliente correctamente	Guarda al nuevo cliente
	Ingresar datos incorrectamente	Error, "El tipo de dato no es correcto."
	Ingresar datos que ya existen	Error, "Ya existe ese cliente"
Baja Cliente	Seleccionar Baja Cliente	Presenta pantalla de baja Cliente
	Seleccionar al cliente correctamente	Elimina al cliente
	Seleccionar al cliente incorrectamente	Error, "No existe el cliente"
Cambio Cliente	Seleccionar Cambio Cliente	Presentar la pantalla de cambio Cliente
	Seleccionar al cliente al que se le quieren hacer los cambios correctamente	Mostrar pantalla donde se ingresarán los nuevos datos del cliente
	Seleccionar al cliente incorrectamente	Error, "No existe el cliente"
	Ingresar nuevos datos del cliente correctamente	Cambia los datos del cliente
	Ingresar nuevos datos del cliente incorrectamente	Error, "El tipo de Dato no es correcto"o "No se lleno la casilla"
Consulta cliente	Seleccionar Consulta Cliente	Presenta pantalla de consulta
	Ingresar al cliente correctamente	Muestra en pantalla los datos del cliente
	Ingresar al cliente incorrectamente	"No existe el cliente"
	Seleccionar Salir	Sale de la pantalla al menú principal
Salir del Sistema		
Cerrar Sesión	Responder SI a la pregunta, ¿Desea salir?	Sale del sistema
	Responder NO a la pregunta, ¿Desea salir?	Regresa a la pantalla de entrada
Salir	Seleccionar Salir	Presentar la pantalla de Salida y redirige a otra página

DESARROLLO

En equipo realiza las siguientes actividades, entregar al instructor los documentos generados de realizar las actividades:

- Según el prototipo y los casos de uso que has hecho en las prácticas anteriores, da una lista de los requerimientos no funcionales que tú consideres apropiados.
- Realizar el plan de pruebas adecuado para los casos de uso según el prototipo y los requerimientos anteriores.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron las tareas realizadas.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 11**
    - Leer el capítulo 7 del libro de Ingeniería de Software Pragmática (Fase de Diseño)
    - Leer práctica 11.
- **Documentos impresos que se utilizarán en la práctica 11**
    - Práctica 8 y 9 resuelta (casos de uso y prototipo de interfaz de usuario)
    - Práctica 11.

Capítulo 9. Práctica para la fase de Diseño

9.1 Fase de Diseño

- **Primer ciclo:** En este ciclo esta fase requiere de dos semanas, ya que los alumnos deben realizar una gran cantidad de diagramas que se les explicarán.

  Entre sus objetivos se está el que se establezcan la arquitectura del sistema, en la cual se definen las partes de las que se compondrá el sistema y la relación entre ellas. Se hace el plan de pruebas de integración, según la arquitectura que se realizó. Finalmente se establecen las clases con las que se construirá el sistema.

- **Segundo ciclo:** Los alumnos reusan el trabajo realizado en el primero, como se ha hecho en las fases anteriores, complementándolo con las funcionalidades que ellos mismos decidieron dejar para el segundo ciclo.

Se explica como especificar la arquitectura del sistema, según la arquitectura en tres capas, estableciendo los componentes del software, la relación entre ellos y sus propiedades. Si es necesario los alumnos deberán realizar el diagrama de distribución. Para modelar la arquitectura del sistema se utilizarán los diagramas de Paquetes.

En el plan de pruebas de integración, se específica el orden en el que se integrará cada componente. Se enseñará a los alumnos a distinguir cuáles son los componentes más importantes del sistema, para modelar la vista estática y dinámica, con los diagramas de clase y secuencia y de estados respectivamente.

9.2 Práctica 11

Los objetivos son, que los alumnos aprendan hacer la arquitectura usando el diagrama de paquetes, la arquitectura en tres capas y el Plan de prueba de integración según la arquitectura.

9.3 Práctica 12

El objetivo de esta práctica es que los alumnos aprendan hacer los diagramas de clases, para la vista estática del software y los diagramas de secuencia, para la vista dinámica.

ALUMNO:
FECHA:
FASE: **Diseño**

PRÁCTICA 11 DISEÑO DE LA ARQUITECTURA Y PLAN DE PRUEBAS DE INTEGRACIÓN

OBJETIVO

- Aprender hacer el diseño de la arquitectura usando el diagrama de paquetes.
- Aprender hacer el Plan de prueba de integración.

INTRODUCCIÓN

En la fase de Diseño, se determina la arquitectura general del sistema que mejor satisface los requerimientos. El resultado obtenido de la etapa de diseño facilita enormemente la implementación del sistema, pues proporciona la estructura básica del mismo y cómo los diferentes componentes actúan y se relacionan entre ellos.

La elección de la arquitectura interna del sistema, es un punto básico de la fase de diseño así como del proyecto. En este curso aprenderemos a usar la arquitectura en capas. Esta arquitectura es descrita por medio de los diagramas de paquete, poniendo un paquete por cada capa.

El plan de pruebas de integración, ayuda a darnos cuenta de cómo se integraran los componentes del sistema después de haberlos construido.

Arquitectura en tres capas

Capa: Abstracción, toma el resultado de la capa inferior, efectúa su función y entrega su resultado a la capa superior. Las capas del modelo que utilizaremos son:

- **Capa de Presentación o interfaz (interfaz con el usuario):** Aquí se ponen los elementos que interactuarán directamente con el usuario, que forman la interfaz de usuario. Esta capa es responsable de presentar información y de interactuar con la capa inferior.
- **Capa Lógica de la aplicación (reglas del negocio):** Se definen los elementos que implementan las funcionalidades.
- **Capa de Almacenamiento (bases de datos):** En esta capa se encuentran los elementos que guardan la información, los elementos persistentes.

Diagramas de paquete

Ya definida la arquitectura del sistema, será representada con diagramas de paquete, otra herramienta de UML. Muestran como un sistema está dividido en agrupaciones lógicas, mostrando las dependencias entre ellas. Normalmente un paquete está pensado como un directorio, por lo que los diagramas de paquetes suministran una descomposición de jerarquía lógica de un sistema.

Los paquetes están organizados para maximizar la coherencia interna dentro de cada paquete, y minimizar el acoplamiento externo entre ellos. Cada paquete puede asignarse a un individuo o a un equipo, y las dependencias

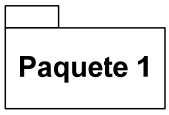
entre ellos pueden indicar el orden de desarrollo requerido. Los paquetes pueden incluir clases, interfaces, otros paquetes, código HTML, etc. Cada paquete tiene un nombre.

Usando la arquitectura de tres capas, tenemos tres paquetes, uno para cada capa:

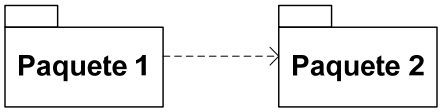
- **Paquete de Interfaz de usuario**: Puede tener uno o varios paquetes que contengan la interfaz de usuario.
- **Paquete de lógica de la aplicación:** Puede contener varios paquetes, uno para cada caso de uso importante en el sistema, otro para los servicios que necesitan todos estos paquetes como es la conexión a la base de datos. Este paquete es responsable de implementar las operaciones solicitadas por los clientes.
- **Paquete de almacenamiento**. Este paquete puede contener varios paquetes, contienen la base de datos en uno o más paquetes y es responsable de la manipulación de datos.

Elementos del Diagrama de Paquetes

Paquete: Representa un grupo de elementos (casos de uso, clases, componentes, otros paquetes) relacionados según algún criterio. Representado por un fólder, con el nombre del paquete.

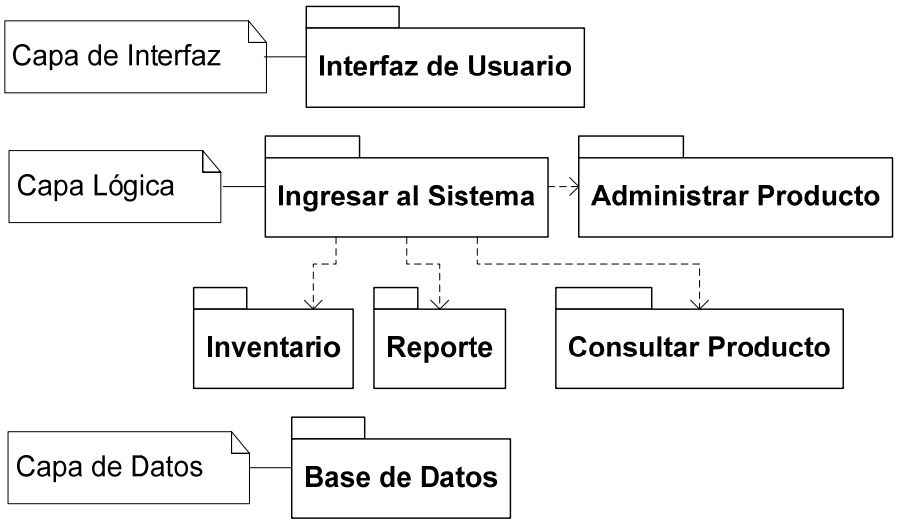


Dependencia entre paquetes: Por lo menos un elemento de un paquete depende de algún elemento del otro. Si un cambio en el paquete 1 afecta al paquete que depende de él (Paquete 2). Por ejemplo, cuando una clase de un paquete llama a una clase de otro. Se representa por medio de una línea discontinua con flecha.



Asociación entre paquetes: Cuando un paquete requiere de otro paquete y se relacionan entre sí. Se representan mediante una línea que los une.

Ejemplo: Diagrama de Paquetes.



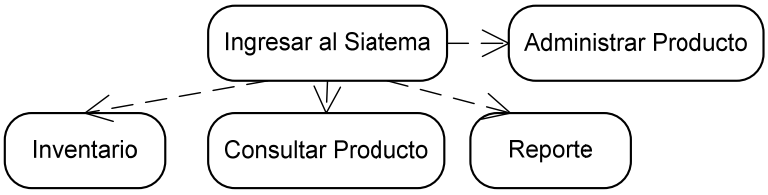
Plan de pruebas de integración

Es usado para verificar que los componentes interactúan apropiadamente con otros. Para realizarlo, se considera el diagrama de paquetes que se hizo en la arquitectura, para indicar cómo se irán integrar los paquetes, se hace una tabla con el nombre de cada uno de los componentes que se usaron en el diagrama.

Orden de Integración de paquetes

- En primer lugar se integrará el paquete que no dependa de otros.
- En el segundo lugar se ponen los paquetes que dependen del primer paquete.
- Así, sucesivamente, se van poniendo los paquetes que dependen de los anteriores en los siguientes lugares hasta integrar todos los paquetes del sistema.

Ejemplo: De plan de pruebas de integración, según el diagrama anterior de paquetes, la tabla contiene cada uno de los componentes que se usaron en el diagrama anterior, están colocados primero el que no depende de ningún otro (ingresar al sistema) y posteriormente los que dependen de éste:



DESARROLLO

Reunirse con sus equipos y entregar al ayudante los siguientes documentos:

- Hacer la arquitectura del sistema y el diagrama de paquetes, para los diagramas de casos de uso de las prácticas anteriores.
- Ahora según la arquitectura de sistema que realizaron en el ejercicio anterior, realizar el plan de pruebas de integración.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 12**
    - Leer el capítulo 7 del libro de Ingeniería de Software Pragmática (Fase de Diseño)
    - Leer práctica 12.

- **Documentos impresos que se utilizarán en la práctica 12**
    - Práctica 8 y 9 resuelta (casos de uso y diagramas de paquete).
    - Práctica 12.

ALUMNO:
FECHA:
FASE: **Diseño**

PRÁCTICA 12 DIAGRAMAS DE CLASES Y DE SECUENCIA

OBJETIVOS

- Aprender hacer los diagramas de clases, para la vista estática del software.
- Aprender hacer los diagramas de secuencia, para la vista dinámica del software.

INTRODUCCIÓN

Ya que se tiene la arquitectura del sistema, se deben identificar los componentes de cada paquete. Para esto dentro del diseño orientado a objetos tenemos dos vistas:

Vista estática: Muestra al sistema sin funcionar, es decir, sabemos su estructura y sus elementos pero no sabemos como interactúan. Aquí usaremos los diagramas de clase.

Vista dinámica: Muestra la interacción entre las clases mediante los diagramas de secuencia. Nos damos cuenta de la forma como se comunicarán los objetos y sus estados en cada momento.

Diagramas de clases

Definen las características de cada una de las clases del sistema, así como sus relaciones. A continuación vemos los elementos de un diagrama de clases:

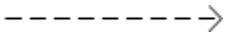
Clase: Está representada por un rectángulo que dispone de tres apartados:

- **Nombre:** Cada clase tiene un nombre único, que la diferencie de las otras clases.
- **Atributos:** Representa alguna propiedad de la clase que se encuentra en todas las instancias de la clase. Pueden representarse mostrando nombre, nombre y tipo, e incluso su valor por defecto.
- **Métodos u Operaciones:** Es una función de la clase, que muestra un comportamiento común a todos los objetos.

Ejemplo: La clase con nombre: *usuario*. Tres atributos: *Nombre* que es public (+), *dirección* que es protected (#) y *teléfono* que es private (-). Tres métodos *Entrar*, *Salir* y *Trabajar*.

Relaciones entre clases: Tres tipos de relaciones entre clases: dependencias, generalización y asociación. Dos tipos de clases: Clase origen, desde la que se realiza la acción de relacionar, es decir, desde la que parte la flecha. Clase destino, es la que recibe la flecha.

- **Dependencias:** Es relación de uso. Una clase usa a otra que la necesita para su cometido. Se representa con una flecha discontinua, va desde la clase utilizadora a la clase utilizada. Con la dependencia mostramos que un cambio en la clase utilizada, puede afectar al funcionamiento de la clase utilizadora, pero no al contrario.



- **Herencia:** Generalización o especialización. Una o varias clases padre o superclase, y una o varias clases hijas o subclases. Línea con un triangulito hacia la superclase.

- **Asociación:** Especifica que los objetos de una clase están relacionados con los elementos de otra clase. Se representa con una línea continua que une las dos clases.

- **Agregación:** Una clase está compuesta por otras o una clase es parte de otra. Se representa por un rombito en la clase que contiene a las otras clases.

- **Multiplicidad:** Cardinalidad de la relación, es decir, cuántos objetos de esa clase pueden participar en la relación.

Ejemplo: Diagrama de clases, se han creado cuatro clases. Superclase *Usuario*, con dos clases hijas *UsuarioADM* y *UsuarioINF*.

El *usuario* mantiene una relación de *asociación* con la clase *Clave*, se indica que es *propietario* de una *clave*.

El *usuario* mantiene una relación de *dependencia* con la clase *Fichero*, es decir las instancias de *usuario* contendrán como miembro una instancia de *Fichero*.

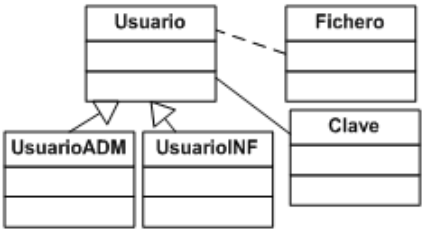


Diagrama de secuencia

Sirve para modelar interacción entre objetos en un sistema. Se hace un diagrama de secuencia para el comportamiento normal de cada caso de uso y uno para el comportamiento excepcional. Este diagrama incluye los objetos de las clases que se usan para implementar un escenario, y un mensaje pasado entre los objetos.

Se debe examinar la descripción del caso de uso, para determinar qué objetos son necesarios para la implementación del escenario. Si se tiene modelada la descripción del caso de uso, como una secuencia de varios pasos, entonces puedes "caminar sobre" esos pasos, para descubrir qué objetos son necesarios para que se puedan ejecutar esos pasos. Sus elementos más importantes son:

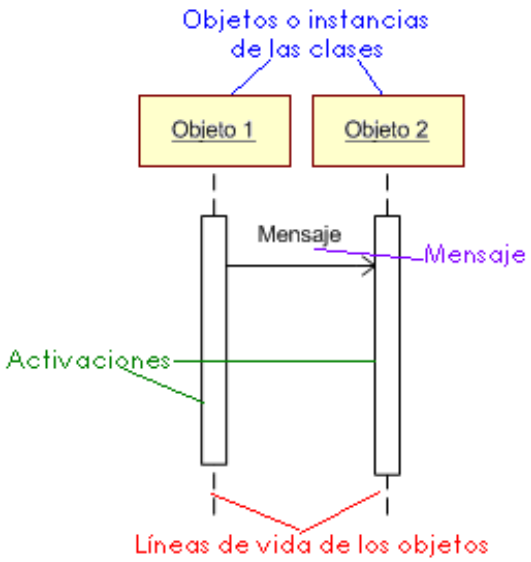
Actor: Elemento que inicia las acciones del caso de uso, envía un mensaje a una clase de interfaz identificada para este escenario.

Objetos o instancias de las clases: Segundo elemento del encabezado, es un objeto de una clase de interfaz. Se representa como un rectángulo con el nombre del objeto y el de su clase, en un formato *nombreObjeto: nombreClase* o solo la clase a la que pertenece ese objeto. Enseguida se pone un objeto de la clase de la capa de la lógica y si es necesario, un objeto de la clase de la capa de almacenamiento, es decir de la base de datos.

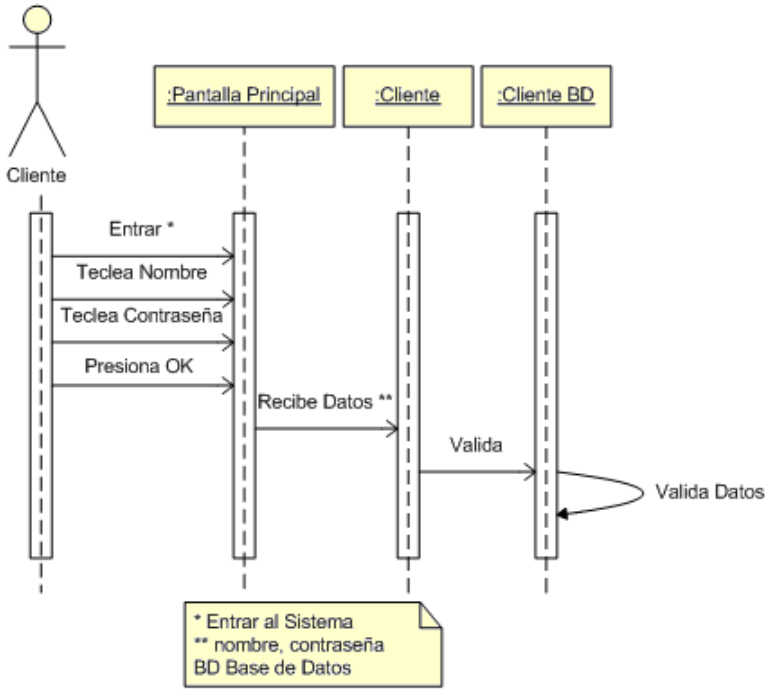
Línea de vida de un objeto o lifeline: Representa la vida del objeto durante la interacción. Se representa como una línea vertical punteada.

Activación: Muestra el período de tiempo en el cual el objeto se encuentra desarrollando alguna operación. Se denota como un rectángulo delgado sobre la línea de vida del objeto.

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.



Ejemplo: Diagrama de secuencia, el actor *Cliente* entra *Entrar** a la *:Pantalla Principal* del sistema ingresa sus datos: *Nombre, Contraseña* y presiona el botón *OK*. Se reciben los datos *Recibe Datos*** en un objeto *:Cliente* que lo *Valida* en un objeto de la clase *:ClienteBD* y el cliente entra a la pantalla cliente.



DESARROLLO

Reunirse con sus equipos y entregar al ayudante los siguientes documentos:

- Usando los diagramas de paquete de la práctica anterior, realizar los diagramas de clases indicados por el ayudante.
- Ahora usando los diagramas de caso de uso de la práctica 8, realizar los diagramas de secuencia indicados por el ayudante.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 13**
  - Leer el capítulo 8 (Fase de Construcción)
  - Leer práctica 13.
- **Documentos impresos que se utilizarán en la práctica 13**
  - Práctica 8 (casos de uso).
  - Práctica 13.

Capítulo 10. Práctica para la fase de Construcción

10.1 Fase de Construcción

- **Primer ciclo:** Entre sus objetivos se encuentran hacer el diseño detallado del sistema, comenzar la construcción del código del sistema de manera individual, cada ingeniero de desarrollo realiza su pedazo de código, al cual le realiza las pruebas unitarias.

  En esta fase los alumnos trabajan de manera individual, en cada equipo se reparten las partes del sistema que realizarán cada uno. Cada uno debe construir su parte de código y entregarlo al resto del equipo funcionando correctamente.

- **Segundo ciclo:** Se utiliza el trabajo realizado en el primer ciclo, como en las fases anteriores, los alumnos deben estar seguros que el trabajo que realizaron en el primer ciclo funciona correctamente, para poder completar y terminarlo en este ciclo.

Se explica a los alumnos como hacer el diseño detallado del sistema. Los alumnos aprenden a completar los diagramas de clase realizados en la fase anterior, con los atributos, tipos, métodos, parámetros y resultados. De esta forma pueden realizar la codificación fácilmente.

Los alumnos construyen el código de su sistema, según los estándares que establecieron. Se les explica la técnica de programación entre pares, para que realicen la codificación del sistema, utilizando las revisiones entre colegas.

Se explica cómo realizar y aplicar el plan de pruebas unitarias a su sistema, incluyendo las pruebas de caja blanca y de caja negra. Al realizar estas pruebas también registran los defectos que encontraron.

10.2 Práctica 13

El objetivo de esta práctica, es que los alumnos aprendan hacer el plan de pruebas unitarias, usando pruebas de caja negra y de caja blanca.

ALUMNO:
FECHA:
FASE: **Construcción**

PRÁCTICA 13 PLAN DE PRUEBAS UNITARIAS

OBJETIVO

- Aprender hacer el plan de pruebas unitarias, usando pruebas de caja negra y de caja blanca.

INTRODUCCIÓN

En esta fase del proyecto, se realizará el plan de pruebas unitarias, para probar cada unidad del código. La finalidad de realizar pruebas en unidades de código, es mostrar la presencia de defectos, para así demostrar que el sistema es satisfactorio y si no lo es, determinar en qué parte no lo es.

Las pruebas unitarias, las pueden realizar personas diferentes a las que desarrollaron el código, es más fácil que alguien ajeno al código que se está probando descubra los defectos.

Prueba unitaria

Forma de encontrar en dónde no es correcto el funcionamiento de una unidad de código. Para que una prueba unitaria sea buena, se recomienda que la prueba sea:

- **Completa**: Que deben cubrir la mayor cantidad de código.
- **Independiente**: Que la ejecución de una prueba no debe afectar a la ejecución de otra.
- **Profesional**: Que la prueba debe ser considerada igual que el código, con la misma profesionalidad, documentación, etc.

Ventajas: Como el objetivo de las pruebas unitarias es aislar cada parte del código en unidades y mostrar que dichas unidades no tienen errores, sus ventajas son:

- **Fomentan el cambio**: Facilitan que el programador cambie el código para mejorar su estructura, asegurando que los cambios no han introducido errores.
- **Simplifica la integración**: Permiten llegar a la fase de integración con un grado mayor de seguridad de que el código funciona correctamente. Y se facilitan las pruebas de integración.
- **Los errores están acotados en unidades y son más fáciles de localizar**: Dado que tenemos pruebas unitarias que pueden desenmascararlos.

Aunque las pruebas unitarias tienen las ventajas anteriores, también es importante darse cuenta de que no descubrirán todos los errores del código. Por definición, sólo prueban las unidades por sí solas. Por lo tanto, no descubrirán errores de integración, problemas de rendimiento y otros problemas que afectan a todo el sistema en su conjunto.

Además, puede no ser trivial anticipar todos los casos especiales de entradas que puede recibir la unidad de programa que se esta probando. Las pruebas unitarias sólo son efectivas si se usan en conjunto con varias técnicas de pruebas de software. Existen dos tipos de pruebas unitarias: De caja blanca y de caja negra (se explicarán mas adelante).

Plan de Pruebas Unitarias

Se escriben casos de prueba para cada función no trivial o método de cada clase, de forma que cada caso sea independiente del resto.

Se definen los casos de prueba para cada clase, el tipo de prueba que se hará y el resultado que se espera de dicha prueba. Para documentarlo se puede hacer una tabla con 5 columnas:

1. Unidad que se probará.
2. Tipo de prueba que se le aplicará (caja negra o caja blanca).
3. Caso de prueba (con valores de variables que se requerirán para probar la clase u operación).
4. Resultado esperado
5. Resultado obtenido (si el resultado obtenido coincide con el esperado, la unidad se aprueba y si no, se registran los defectos en la forma de Registro de defectos y se corrigen).

Pruebas de caja blanca

Pruebas para analizar el código. Usan la estructura del código para derivar los casos de prueba, se busca ejecutar al menos una vez cada instrucción del programa. La idea de estas pruebas es confeccionar casos de prueba que garanticen que se verifican todos los caminos de control del programa, diseñando datos de pruebas que los recorran.

Camino de control: Trayectoria que introduce por lo menos un valor de condición.

Complejidad ciclomática de McCabe: Se usara para saber cuántos casos de prueba son necesarios para tener una buena cobertura de las instrucciones de una unidad. La idea es derivar casos de prueba a partir de un conjunto dado de caminos de control. Para obtener la complejidad ciclomática de McCabe tomamos el número de condiciones en la unidad +1 éste es el número de casos de prueba que se deben definir.

Procedimiento para hacer el Plan de pruebas unitarias de caja blanca:

1. Determinar complejidad ciclomática de McCabe de la unidad a probar.
2. Definir los casos de prueba y los resultados esperados.
3. Efectuar la prueba.
4. Comparar los dos resultados y aprobar o registrar los errores encontrados.

Ejemplo: De una prueba de caja blanca

Seudocódigo:

```
1 private String valorAtributoV(int i){
2       if (i==0) return "area";
3       if (i==1) return "zona";
4       if (i==2) return "sueldo";
5       return "horario";
6 } //end valorAtributoV
```

Complejidad ciclomática: 3 + 1 = 4

Solo se tienen 3 if’s (líneas 2,3 y 4), entonces se necesitan 4 casos de pruebao 4 caminos de control diferentes.

Casos de prueba:

1. No cumpliéndose la condición de los tres if's. Por lo tanto el dato de prueba es i = 3.
    - Camino1: Como no entra en ningún if, se ejecuta la línea 5, la cual solo debe regresar la cadena "horario".
2. Si se cumplirá la condición de if de la línea 2. Por lo tanto el dato de prueba es i = 0.
    - Camino2: Como entra en el if de la línea 2, se ejecuta esta, la cual solo debe regresar la cadena "area".
3. Si se cumplirá la condición de if de la línea 2. Por lo tanto el dato de prueba es i = 1.
    - Camino3: Como entra en el if de la línea 3, se ejecuta esta, la cual solo debe regresar la cadena "zona".
4. Si se cumplirá la condición de if de la línea 2. Por lo tanto el dato de prueba es i = 2.
    - Camino4: Como entra en el if de la línea 4, se ejecuta esta, la cual solo debe regresar la cadena "sueldo".

Plan de pruebas:

Código	Tipo de Prueba	Caso de prueba	Resultado esperado	Resultado obtenido
Seudocódigo anterior	Caja blanca	i = 3	Regrese "horario".	
		i=0;	Regrese "area".	
		i=1;	Regrese "zona".	
		i=2;	Regrese "sueldo".	

Pruebas de caja negra

Se centran en los requisitos funcionales del software. Se toma un caso de uso y se diseñan los casos de prueba, según los datos de entrada del caso de uso, se especifican las respuestas esperadas. Esta prueba intenta encontrar errores basados en lo qué debe hacer el programa. El número de casos de prueba, depende del conjunto de condiciones necesarias para que se ejecute cada acción y de lo que deben regresar esas acciones.

Tablas de decisión: Para diseñar los casos de prueba de caja negra. Se propone construir tablas de decisión para cada caso de uso. Una tabla de decisión contiene:

- **Condiciones de entrada:** Se define la lista de condiciones de las entradas y la lista de combinaciones de condiciones posibles. Si se tienen n condiciones, existen 2^n combinaciones de condiciones, esto es un conjunto significativo que se tendrán que probar.
- **Acciones a realizar:** Se define la lista de acciones a realizar, y la lista de las combinaciones de acciones para los valores de las condiciones de entrada.

Procedimiento para hacer el Plan de pruebas de caja negra:

- **Determinar casos de prueba de la unidad:** Condiciones de las variables de entrada que se definen y las posibles combinaciones de esas condiciones (se escriben en la sección de condiciones, de la tabla de decisiones).
- **Resultados esperados:** Combinaciones de acciones (se escriben en la sección de acciones a realizar, de la tabla de decisiones).

Ejemplo: De una prueba de caja negra.

Caso de uso: Registrar Usuario.

Actor: Usuario.

Descripción: El usuario registra sus datos.

Precondiciones: El usuario no esta registrado en el sistema.

Flujo de eventos normales:

ACTOR		SISTEMA		
Paso	**Acción**	**Paso**	**Acción**	**Excepción**
1	Selecciona la opción de registrarse en el sistema.	2	Muestra un formulario que pide los siguientes datos: Nombre, dirección, teléfono y contraseña.	
3	Al terminar el llenado envía los datos al sistema.	4	Una vez aceptados los datos por el sistema, éste muestra las opciones que puede utilizar del sistema.	E1

Flujo de eventos excepcionales:

Id	Nombre	Acción
E1	Falta de datos obligatorios o incorrectos durante el registro.	El sistema recarga la página con un mensaje que indica los datos obligatorios faltantes.

Tabla de decisiones:

Condiciones				
Enviar datos	dirección	nombre	teléfono	contraseña
Recibir datos	dirección	nombre	teléfono	contraseña
Acciones				
UsuarioRegistradosetNombre()		X		
UsuarioRegistradosetDireccion()	X			
UsuarioRegistradosetTelefonos()			X	
UsuarioRegistradosetContraseña()				X

Plan de pruebas del sistema:

Clase / Método	Tipo de Prueba	Caso de Prueba	Resultado Esperado	Resultado Obtenido
UsuarioRegistrado setNombre()	Caja Negra	miNombre	Guarda el nombre	
UsuarioRegistrado setDireccion()	Caja Negra	miCalle1 miColonia1 MiDelegacion1234	Guarda los campos de la dirección	
UsuarioRegistrado setTelefonos()	Caja Negra	(55) 55-52-49-06 (33) 7-68-11-11 01-800-123-4567	Guarda los 3 teléfonos	
UsuarioRegistrado setContraseña()	Caja Negra	myContraseña	Guarda la contraseña	

DESARROLLO

Reunirse con sus equipos y entregar al ayudante los siguientes documentos:

- Usando el siguiente código, realicen el plan de pruebas de caja blanca.

```
1  private String[] datosVistaV;
2  Almacena tmp = new Almacena();
3
4   public Vacante[] buscarVacante(){
5       LinkedList listaPosicion = noNulos(datosVistaV);
6       Vacante[] resultado = tmp.buscarVacante(listaPosicion, datosVistaV);
7       return resultado;
8   } //end buscarVacante
9
10    /*
11     * Metodo para verificar que patrones se usaran para buscar
12     */
13  private LinkedList noNulos(String datos[]){
14      LinkedList listaPosicion = new LinkedList();
15      for (int i = 0; i < 4; i++){
16          if (datos[i] !=  ""){
17              listaPosicion.add(i);
18          } //end if
19      } //end for
20      return listaPosicion;
21  } //end noNulos
```

- Usando el diagrama detallado de caso de uso que les indique el instructor, realizar el plan de pruebas de caja negra.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 14**
  - Leer el capítulo 9 del libro (Fase de Prueba del sistema).
  - Leer práctica 14.
- **Documentos impresos que se utilizarán en la práctica 14**
  - Código de su sistema (por equipo, no impreso).
  - Sus Planes Pruebas de Integración y del Sistema.
  - Práctica 14.

Capítulo 11. Prácticas para la fase de Pruebas del Sistema

11.1 Fase de Pruebas del Sistema

- **Primer ciclo:** Los objetivos son, que los alumnos integren su sistema y lo prueben, verificando que cumple con los requerimientos establecidos por ellos y por el cliente y que realicen los manuales del sistema.
- **Segundo ciclo:** Deben terminar su sistema, con todas las funcionalidades y entregarlo al cliente, actualizar los manuales que realizaron en el primer ciclo, según la última versión del software.

Se explica a los alumnos como deben realizar la integración del sistema, en que orden integraran cada componente y como probarán que integraron correctamente.

Realizan la integración del sistema, utilizando el plan de integración que establecieron en la fase de diseño, completándolo según la implementación que realizaron.

Prueba que la integración haya sido correcta, con la técnica que las profesoras explican en clase llamada tablas cruzadas.

Los alumnos aprenden a realizar las pruebas del sistema. Deben revisar los planes de pruebas del sistema que realizaron en la fase de captura de requerimientos y de diseño actualizándolos y realizando las pruebas al sistema para verificar que cumplen con los requerimientos establecidos.

Aprenden a realizar los manuales del sistema, dentro de estos manuales se encuentran el de usuario y de instalación.

11.2 Práctica 14

Los objetivos de esta práctica son que los alumnos aprendan a integrar los componentes del sistema, hacer las pruebas de integración y a probar el sistema.

11.3 Práctica 15

Los objetivos de esta práctica son que los alumnos conozcan los tipos de manuales que existen para los sistemas, y que aprendan a construirlos manuales.

ALUMNO:
FECHA:
FASE: **Pruebas del Sistema**

PRÁCTICA 14 PRUEBAS DE INTEGRACIÓN Y DEL SISTEMA

OBJETIVOS

- Aprender a integrar los componentes del sistema.
- Aprender a hacer las pruebas de integración.
- Aprender a probar el sistema.

INTRODUCCIÓN

En la fase de prueba del sistema se realizan entre otras actividades, la integración del sistema. En esta práctica aprenderán a integrar los componentes del sistema y a realizar las pruebas de integración y del sistema, para esto verificaran que tengan todos los productos, que funcionen bien juntos y que el sistema haga lo que se espera de él.

Integración del Sistema

Proceso de ensamblar todos los componentes que se desarrollaron por los ingenieros de software. Para lograrlo, se deben hacer las especificaciones de la misma. Es recomendable integrar por casos de uso e ir incorporando los que ya están.

Especificación de la Integración del Sistema

Para realizar la integración del sistema se debe:

Juntar todos los componentes desarrollados que estén en la última versión revisada y corregida, para construir el sistema. De la siguiente manera:

- Cada ingeniero de desarrollo hace la lista de los componentes que desarrolló y se aprobaron.
- Basándose en la arquitectura del sistema, se hace una lista con todos los componentes que se generaron con los datos del dueño de cada uno.

Con las listas anteriores se revisa que se tengan todos los componentes, comprobando que estén en la última versión aprobada.

Se revisa el Plan de pruebas de integración (realizado en fase de diseño), se agrega el orden de integración (según el diseño realizado).

Se define un caso de prueba por caso de uso y las pruebas que se aplicarán para comprobar que todos los componentes interactúen correctamente entre sí.

Ya que tenemos todo lo anterior, se comienza la integración del sistema.

Se elige una de las estrategias vistas en clase, para hacer la integración del sistema. Ya integrado el sistema, se le aplican las pruebas indicadas en el Plan de Pruebas de Integración revisado.

Pruebas de Integración

Consisten en realizar pruebas para verificar, que el conjunto de partes del sistema funcionan correctamente juntas. Se realizan las pruebas con los casos de pruebas asociados, efectuando el correspondiente análisis e informe de los resultados de cada prueba, y generando un registro.

Se comparan los resultados de las pruebas de integración obtenidos con los esperados, se identifica el origen de cada defecto detectado, para poder remitirlo a quien corresponda, se hacen las modificaciones correspondientes y se vuelven a realizar las pruebas ya sea de forma total o parcial, y se verifica si será necesario contemplar nuevos casos de prueba no considerados anteriormente.

Pruebas del Sistema

Su objetivo es probar el sistema en su totalidad. Es importante comprobar la cobertura de los requerimientos funcionales y no funcionales, dado que su incumplimiento puede comprometer la aceptación del sistema por el cliente.

Estas pruebas se realizan conforme al plan de pruebas del sistema, establecido en fase de captura de requerimientos, se agregan nuevos casos de prueba, tomando en cuenta la instalación, arranque y usabilidad del sistema. El sistema lo deben probar usuarios reales (personas que no hayan estado involucrados en el desarrollo del producto), no solo los ingenieros de desarrollo.

Se prueba todo el sistema, se registran los defectos encontrados en la forma de registro de defectos, se corrigen y se prueba nuevamente el sistema

Tablas Cruzadas

Se usan para documentar las pruebas del sistema usaremos. Tablas que nos ayudarán a comprobar que el sistema cumpla con todos los requerimientos (funcionales y no funcionales), verificando que se cubran todas las funcionalidades. Constan de tantos renglones como requerimientos a probar y de tantas columnas como componentes.

Se va colocando una X si el requerimiento de ese renglón está cubierto por el componente de esa columna, así sucesivamente hasta que cada requerimiento tenga al menos un componente marcado.

Ejemplo: Tabla cruzada. Se coloca una X si el requerimiento de ese renglón está cubierto por el componente de esa columna, cada requerimiento debe tener al menos un componente.

REQUERIMIENTOS \ COMPONENTES	C1	C2	C3	C4	C5	C6	C7	C8
La empresa podrá dar de alta sus datos.	X			X		X		
La empresa podrá dar de baja sus datos.		X		X				X
La empresa accederá a su espacio en la página mediante una clave y una contraseña.	X			X				
La empresa podrá consultar el Currículo Vitae de las personas.		X		X				
El acceso a la página será por Internet.				X				
El sistema verificará si una clave ya está en uso y lo notificará para evitar que se dupliquen.					X			
El sistema contará con una interfaz gráfica fácil de usar.				X				

C1. En la ventana de inicio se ofrece esta opción.

C2. La Página Inicial de Empresa ofrece esta opción.

C3. La Página Inicial de Aspirante ofrece ésta opción.

C4. La interfaz se realizará con JSP's.

C5. Se cuenta con una clase ABD en la cual existe un método llamado "yaexisteclave".

C6. Se cuenta con una clase ABD en la cual existe un método llamado "guardaprimeraBDEmp".

C7. Se cuenta con una clase ABD en la cual existe un método llamado "guardaprimeraBDAsp".

C8. Se cuenta con una clase ABD en la cual existe un método llamado "darDeBajaEmp".

DESARROLLO

- Reunirse con sus equipos y comenzar a realizar la integración de su sistema.
- Realizar las pruebas de integración y del sistema.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 15**
    - Leer el capítulo 9 del libro (Fase de Prueba del sistema).
    - Leer práctica 15.
- **Documentos que se utilizarán en la práctica 15**
    - Su sistema y la documentación de éste.
    - Práctica 15.

ALUMNO:
FECHA:
FASE: **Pruebas del Sistema**

PRÁCTICA 15 MANUALES DEL SISTEMA

OBJETIVOS

- Conocer los tipos de manuales que existen.
- Aprender a construir los manuales del sistema.

INTRODUCCIÓN

En esta fase se realizan los manuales que se requieren para el sistema, en el lenguaje adecuado según el tipo de lector que tendrán (cliente que solicito el sistema o cualquier otro usuario del mismo).

A lo largo del desarrollo del sistema en cada fase, se genera documentación del sistema, la cual forma parte de los manuales para los ingenieros de desarrollo. Pero se requieren manuales para las otras personas involucradas, como son el cliente que solicitó el sistema, las personas que lo van a usar, las personas que lo van a instalar, etc.

Los manuales pueden estar de forma electrónica o impresa, tener índice, glosario, explicación de los casos excepcionales del sistema, de los posibles errores con explicación de cómo resolverlos.

A continuación se explican los tipos de manuales y se dan ejemplos.

Manual de usuario

Contiene la información necesaria para obtener del sistema los resultados deseados. Describe como se usa el software con base en la interfaz de usuario. Se redacta en términos comprensibles para los usuarios. Explica que pasa en la interfaz, en cada campo, botón u opción que tenga dicha interfaz, qué hacer en caso de que se presente algún problema en el momento de estar usando el programa.

Ejemplo: Manual de Usuario.

ÍNDICE

1. Menú principal del sistema

a) Conociendo el menú principal del sistema

Dentro del sistema se tienen varias opciones a realizar, para realizarlas se coloca el cursor sobre la opción y se presiona el botón izquierdo del mouse:

- *La opción de Registrar alumnos.*
- *La opción de Registrar médicos.*
- *La opción de Registrar citas.*
- *La opción de Consultar sistema.*

2. Registrar alumnos

Al haber seleccionado la opción de Registrar alumnos desde el menú principal se obtiene la siguiente vista:

Al colocar el cursor sobre la opción y presionando el botón izquierdo del mouse se podrá realizar la opción elegida.

b) Dar de alta un alumno

Al haber seleccionado la opción de Dar de alta un alumno a partir del menú principal de Registrar alumnos se obtiene la siguiente vista:

Para agregar un alumno se pide el número de cuenta de éste, y se puede saber si el alumno se encuentra en el sistema o no.
Si el alumno ya se encuentra registrado, se mostrará una vista parecida a la siguiente:

Si el alumno no se encuentra registrado, a continuación se le pedirán los datos restantes como son: nombre, fecha de nacimiento, semestre vigente y carrera, introduciendo cada dato y al finalizar colocar el cursor sobre el botón aceptar y presionar el botón izquierdo del mouse, así como lo muestra la siguiente vista:

b) Editar datos de un alumno

Al haber seleccionado la opción de Editar datos de un alumno a partir del menú principal de Registrar alumnos se obtiene la siguiente vista:

Para editar los datos de un alumno se piden el número de cuenta de éste, y se puede saber si el alumno se encuentra o no.

Si el alumno ya se encuentra registrado, se mostrarán los datos del alumno y se puede elegir editarlos o no, seleccionando el campo que se desea editar y modificarlo, al terminar las modificaciones colocar el cursor sobre el botón aceptar y presionar el botón izquierdo del mouse, así como lo muestra la siguiente vista:

Si el alumno no se encuentra registrado, se mostrará una vista parecida a la siguiente:

c) Eliminar el registro de un alumno

Al haber seleccionado la opción de Editar Eliminar el registro de un alumno a partir del menú principal de Registrar alumnos se obtiene la siguiente vista:

Para dar de baja un alumno se pide el número de cuenta de éste y se puede elegir eliminarlo o no.
Si el alumno no se encuentra registrado, se mostrará una vista parecida a la siguiente:

Si el alumno ya se encuentra registrado se mostrarán sus datos y se pedirá la confirmación de que desea eliminar ese alumno

Nota: Para regresar al menú anterior de cada página se debe de utilizar el botón de regreso del navegador o las opciones localizadas en la esquina inferior derecha.

Manual de operación o instalación

Contiene la información necesaria para instalar y administrar el sistema. Establece los parámetros iniciales, las indicaciones para preparar el sistema para su uso y la versión que se está instalando, la información que se requiere sobre el hardware, indicando capacidades y velocidades mínimas que se requieren del equipo en el que se instalará y la información del software que se requieren (versiones mínimas del sistema operativo, etc.). Forma de comunicarse con alguien que pueda resolver los problemas que se presenten, durante la instalación del sistema.

Ejemplo: Manual de operación o instalación.

ÍNDICE

1. Requerimientos para el sistema

Para llevar a cabo la Instalación de Javonario1.0 se necesitará tener previamente los siguientes componentes:

- *Unidad lectora de CD*
- *Fedora Core 2*
- *Postgresql 7.4.5*
- *Tomcat 4 o posterior*
- *J2EE 1.4 Sun Java System Application Server Platform Ed. 8*
- *5MB de memoria virtual*
- *128Mb de memoria RAM*
- *Goshtview 4.6*
- *Mozilla 1.7*

2. Instalación del sistema

Al localizar el archivo Javonario.tgz dentro del disco de Instalación, desde la línea de comandos del sistema operativo se teclearán las siguientes instrucciones:

- *Descompresión del archivo fuente: Con permisos de super usuario (ROOT) teclear:* ***cd <JAKARTA>/webapps***
  Donde <JAKARTA> es la dirección en donde se encuentra instalado jakarta-tomcat.
  Si no se tiene instalado jakarta-tomcat, debe ser instalado ya que es necesario para la Instalación y Ejecución del sistema.
  Nota: Para la Instalación de jakarta-tomcat puede consultar la siguiente dirección: ***http://jakarta.apache.org/***
  Una vez colocados en el directorio webapps de jakarta-tomcat se debe de teclear el siguiente comando para la descompresión:
  tar xvzf <direccion donde esta el disco>/Javonario.tgz
  Donde <direccion donde esta el disco> es la dirección donde se encuentra la unidad de CDROM.

3. Ejecución del sistema

Para ejecutar el sistema, se debe de contar con una base de datos llamada medicoCita creada en POSTGRES, así como el usuario de POSTGRES llamado 'javonez', el cual debe de contar con permisos de lectura/escritura sobre la base de datos medicoCita.
Si no se tiene instalado POSTGRES, debe ser instalado ya que es necesario para la
Ejecución del sistema.
Nota: Para la Instalación de POSTGRES puede consultar la siguiente dirección: ***http://www.postgresql.org/***
Además de contar con la Instalación de jakarta-tomcat y de POSTGRES, se debe de contar con la correcta Instalación de Java, ya que debe de estar configurado apropiadamente para trabajar con estos.
Nota: Si no se cuenta con java, debe de ser instalado ya que es necesario para Ejecución del sistema, se encuentra disponible en la siguiente dirección: ***http://java.sun.com/***
Para la Ejecución del sistema se ejecutara un navegador de internet y se colocará la dirección:

http://<midominio>:<puerto>8080/javonario/

Donde <midominio> es el dominio o IP del servidor donde se encuentre instalado el sistema Javonario, y donde <puerto> es el puerto del servidor por donde se realizan peticiones a través de tomcat. Ejemplo:

http://topanga.fciencias.unam.mx:8080/javonario/index.html

4. Contáctanos

Si tienes comentarios o dudas, escríbenos. Te atenderemos a la brevedad posible, en: javonario@yahoo.com.mx

Manual de administración del proyecto

Contiene la información necesaria sobre la administración del proyecto. Debe aparece por primera vez el nombre del sistema, el cual es único (posteriormente aparece en todos los documentos), información sobre el equipo que lo realizó, personal encargado del análisis y diseño del sistema. Un resumen administrativo, con información de los documentos de las fases de lanzamiento, estrategia, planeación y la evaluación en el cierre de cada ciclo, elaborados durante el proceso del software.

Manual de desarrollo y mantenimiento

Contiene la información necesaria para mantener operando un sistema durante su ciclo de vida, describe la Configuración de Software, ambiente usado para el desarrollo y pruebas (compiladores, herramientas de análisis y diseño, construcción y pruebas). Incluye documentos de las fases de requerimientos, diseño, implementación y pruebas (se va desarrollando al igual que el sistema). Usa lenguaje comprensible para el personal de mantenimiento.

Revisión de los manuales

Para estar seguros de que los manuales harán su función correctamente, se hacen prueban con colegas y usuarios, de esta forma nos aseguramos que la escritura es clara, precisa, completa y entendible. Después de aplicar las pruebas se hacen preguntas a las personas que usaron el manual, para saber cual fue su impresión si fue claro o si recomienda cambios para mejorarlo.

DESARROLLO

- Reunirse con sus equipos y comenzar a realizar los manuales de su sistema.

CONCLUSIONES

Junto con el instructor y todo el grupo discutir lo siguiente:

- Para qué les sirvieron los ejercicios realizados.
- Cómo fue la participación de tu equipo.
- Lograron llegar acuerdos con facilidad.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 16**
    - Leer el capítulo 10 del libro de Ingeniería de Software Pragmática (Fase de Cierre).
    - Leer práctica 16.
- **Documentos que se utilizarán en la práctica 16**
    - Forma de Lecciones aprendidas y sugerencias de mejoras (por equipo).
    - Forma Evaluación del equipo y personal (individual).
    - Práctica 16 (individual).

Capítulo 12. Práctica para la fase de Cierre

12.1 Fase de Pruebas del Sistema

- **Primer ciclo:** Se hace un informe de mediciones del proceso y del estado de la configuración del sistema, lo cual servirá para realizar el trabajo del segundo ciclo.

  Se evalúa el trabajo realizado, se identifican lecciones aprendidas y propuestas de mejora.

  Finalmente entregan la parte del sistema que ellos decidieron entregar para finales del primer ciclo, es una parte funcional, en la cual el usuario pueda comenzar a trabajar con algunas de las funcionalidades.

- **Segundo ciclo:** Los alumnos entregan la segunda versión del sistema y la carpeta completa, en la que vendrá toda la documentación que realizaron durante los dos ciclos.

  Finalmente reflexionan sobre lo que aprendieron en el curso, tomando en cuenta las mejores prácticas que tuvieron tanto individualmente como en equipo y desechan las malas prácticas o las practicas que no les servirán en un futuro.

Se explica a los alumnos, cómo realizar la evaluación de equipo e individual. Evalúan si cumplieron o no los objetivos que plantearon al inicio (fase de lanzamiento) para el rol, personales, de equipo y del sistema.

Aprenden a registrar lecciones aprendidas, buscando mejoras en el equipo para el segundo ciclo, eliminan los problemas que se tuvieron.

Finalmente se les explica como realizar los informes de mediciones del proceso y el informe del estado de la configuración, para que lo puedan usar durante el siguiente ciclo.

12.2 Práctica 16

Los objetivos de esta práctica son, que los alumnos evalúen el desempeño individual de los miembros del equipo y el desempeño como equipo, las prácticas, a los ayudantes y lo aprendido en las clases. Que hagan el informe de lecciones aprendidas y de las mediciones del ciclo, proponiendo mejoras en el equipo para el siguiente ciclo.

ALUMNO:
FECHA:
FASE: **Cierre**

PRÁCTICA 16 EVALUACIÓN, LECCIONES APRENDIDAS, PROPUESTAS DE MEJORAS Y MEDICIONES

OBJETIVOS

- Evaluar el desempeño individual de los miembros del equipo y el desempeño como equipo.
- Evaluar las prácticas, a los ayudantes y lo aprendido en las clases.
- Hacer el informe de las lecciones aprendidas y de las mediciones del ciclo.
- Proponer mejoras en el equipo.

INTRODUCCIÓN

De la misma forma en que evaluarán a sus compañeros de equipo y así mismos, en esta práctica corresponde evaluar a los ayudantes, las clases de laboratorio y las prácticas. Con esto ayudarán a mejorar la calidad de las clases.

Después de haber realizado sus evaluaciones, se darán cuenta de prácticas que les hayan servido otras que no y escribirán que es lo que les gustaría se cambiara, mejorara o eliminara.

Ya que esta es una evaluación muy importante tanto para ustedes como para los ayudantes, deben contestarla lo más sinceramente posible, lo pueden hacer de forma anónima si así lo quieren.

Evaluación del equipo y personal

Cada integrante de equipo se analiza así mismo y a los otros integrantes de su equipo, señalan las dificultades que tuvieron durante el primer ciclo. Evaluarán el trabajo que realizaron como equipo y la experiencia que lograron sacar del mismo, evaluaran su rol y cómo se sintieron en su trabajo y evaluarán a sus compañeros de equipo en su rol y en su aportación al equipo. Llenan la forma de Evaluación personal y del equipo.

Lecciones aprendidas y sugerencias de mejoras

Se hace una revisión de todo lo aprendido durante el ciclo, se ven las cosas buenas y malas, para que puedan mejorar individualmente y equipo. Llenan la forma de Lecciones aprendidas y sugerencias de mejora, documentando las experiencias exitosas y se hacen sugerencias para mejorar la calidad del equipo en todos los aspectos que crean necesarios.

Informe de mediciones

Se llena la forma de Mediciones, haciendo un recuento de lo hecho en el primer ciclo en cada fase, el tiempo que invirtieron, tamaño de los productos que fueron hechos, los defectos que se encontraron, sacan la productividad y calidad de los productos, tomando en cuenta los defectos y productos.

Evaluación a las ayudantías

Es para que los ayudantes puedan mejorar en las clases, la información que les transmiten y la forma en que se las transmiten.

A continuación se dan listas de puntos a evaluar, para los ayudantes, las prácticas y las clases de laboratorio.

Instrucciones: Elegir la opción adecuada, según tu opinión y experiencia en las clases de laboratorio.

Evaluación de los ayudantes

Ayudante 1

	Siempre	**Casi siempre**	**A veces**	**Casi nunca**	**Nunca**
Tu ayudante fue puntual					
Tu ayudante asistió con regularidad a dar la clase.					
Su forma de explicar fue entendible.					
La accesibilidad que mostró fue buena (forma de contactarlo).					
El trato a ti como alumno fue bueno.					
Otros:					
Da un comentario general sobre lo que no te gusto de tu ayudante y una sugerencia de mejora:					

Ayudante 2

	Siempre	**Casi siempre**	**A veces**	**Casi nunca**	**Nunca**
Tu ayudante fue puntual					
Tu ayudante asistió con regularidad a dar la clase.					
Su forma de explicar fue entendible.					
La accesibilidad que mostró fue buena (forma de contactarlo).					
El trato a ti como alumno fue bueno.					
Otros:					
Da un comentario general sobre lo que no te gusto de tu ayudante y una sugerencia de mejora:					

Evaluación a las prácticas

	Siempre	**Casi siempre**	**A veces**	**Casi nunca**	**Nunca**
Los ejemplos fueron útiles para entender el tema que se estaba explicando.					
La redacción de las prácticas fue buena.					
El formato de las prácticas te gusto.					
La información que en las prácticas les servía para la resolución de su proyecto.					
Otros:					
Da un comentario general sobre lo que menos te gusto de las prácticas y una sugerencia de mejora:					

Evaluación de las clases en laboratorio

	Siempre	Casi siempre	A veces	Casi nunca	Nunca
Las clases fueron divertidas.					
Las clases fueron buenas.					
El tiempo de clases fue suficiente.					
El número de clases por fase fue suficiente.					
Otros:					
Da un comentario general sobre lo que menos te gusto de las prácticas y una sugerencia de mejora:					

DESARROLLO

- Con tu equipo haz un análisis de las cosas que aprendieron durante éste primer ciclo, exponiendo cada uno, su experiencia y lo que les gustaría que se cambiara y vayan llenando la forma de Lecciones aprendidas y sugerencias de mejoras.
- Individualmente llena las tablas que se encuentran en la parte de arriba para la evaluación de las ayudantías y escribe las cosas que te gustaría se mejoraran, cambiaran o eliminaran en la clase, prácticas y con respecto a los ayudantes.
- Ahora haz la evaluación tanto de tu equipo como personal, llenando la forma correspondiente .

Entrega lo anterior al líder de tu equipo y éste debe de entregar las prácticas de todos los integrantes de tu equipo al ayudante.

CONCLUSIONES

Entre todos los alumnos del grupo discutir sobre las clases, prácticas o lo que les gustaría se cambiara, mejorara o quitara.

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica 17**
  - Leer práctica 17.
- **Documentos impresos que se utilizarán en la práctica 17**
  - Forma de Informe del Estado de la Configuración.
  - Depósito de documentos de su proyecto (electrónico).

Capítulo 13. Manual para los ayudantes de la materia de Ingeniería de Software

En este capítulo se da una lista de recomendaciones para los próximos ayudantes de la materia. Las recomendaciones las elaboré, según mi experiencia como ayudante, y después de haber trabajado un semestre con las prácticas que en este mismo trabajo se presentan. Son recomendaciones que pueden ayudar a los instructores a dar una mejor clase a los alumnos.

12.1 Para todas las prácticas los ayudantes deben

- Leer con los alumnos la sección de próxima práctica, dejándoles en claro que si no lo hacen o no llevan el material se les bajará un punto.
- Dependiendo como vean el trabajo en cada equipo, discutir las conclusiones ya sea por separado con cada equipo o en grupo, según el avance de trabajo.
- Si notan que hay problemas en algún equipo, acercarse a él y preguntar de forma conciliadora y así poder ayudar a resolver el problema. Si persiste el problema hablarlo con la profesora.
- Pedirle a los alumnos lean las instrucciones de la práctica y preguntar si tienen alguna duda, si es que la tienen responderla y decirles que comiencen a resolverla.
- Diez minutos antes de terminar la clase, pedir a los alumnos los resultados de la práctica. Hacer lo mismo 5 minutos antes, para que los alumnos tengan la conciencia del tiempo en la hora de laboratorio.
- Preguntar si leyeron el capítulo y la práctica correspondiente antes de comenzar la clase. Si no lo hicieron recordarles que lo deben hacer, ya que es una buena práctica.
- Dar confianza a los equipos y en particular a cada integrante, para que si tienen dudas o problemas puedan acercarse a ellos y ayudarlos a solucionarlos.

12.2 Reglas para las prácticas

Al inició del semestre los ayudantes deben explicarle a los alumnos las reglas que se seguirán a lo largo del semestre en las prácticas.

- Todos los ejercicios se entregarán o enviarán al instructor dentro de la hora de clase, algunas veces serán en equipo y otras individuales.
- Las actividades en equipo se entregarán con los nombres de los integrantes que estuvieron presentes en la clase haciéndolas.
- Al final de cada práctica, hay una lista de actividades a preparar antes de la siguiente práctica. Si no sé realizan antes de la práctica, tendrán dudas que no podrán resolverse por falta de tiempo.
- En algunas prácticas al final hay documentos que se deben llevar impresos para la siguiente práctica. Estos documentos son obligatorios de llevar, ya que sin ellos no podrán realizarla.
- Si después de alguna práctica quedara alguna duda sobre la actividad realizada, debes decírselo al ayudante de forma personal o enviando un correo ese mismo día, para que se imparta otra clase con más ejemplos sobre el tema.

12.3 Prácticas de Introducción a la Ingeniería De Software y al trabajo en equipo

Los ayudantes deben estar concientes de que es muy probable que los alumnos no sepan lo que es trabajar en equipo, por lo tanto deben ser pacientes y amigables con ellos para poder ayudar en la formación de los equipos.

12.3.1 Práctica 1 Ingeniería de Software

- Que los equipos sean de exactamente cinco alumnos, de esta forma los alumnos comenzarán a formar su equipo con el cual trabajarán durante todo el semestre, aclarándoles desde luego que tendrán otras prácticas para formar el equipo definitivo.
- Si existen grupos de más de cinco amigos y existen personas sin equipo se debe sugerir que alguno de los amigos se cambie de equipo, que lo decidan por volado, votación, etc.
- La actividad de leer el articulo es de a lo mas 15 minutos.
- La actividad de leer el capítulo 1 del libro es de 30 minutos.
- Posteriormente pasen a exponer los puntos que les parecieron importantes y haya la discusión entre todos los equipos.
- Si son muchos equipos que se haga un sorteo para que solo pasen a exponer algunos equipos, según el tiempo y número de equipos formados.
- Dar la opción a los alumnos de elaborar un mapa mental del artículo o del capítulo, en lugar de un resumen, según les parezca más divertido y entendible para cada equipo y alumno.
- Leer con los alumnos la sección de próxima práctica, dejándoles en claro que si no lo hacen o no llevan el material se les bajará un punto.
- Recordarle a los alumnos que deben entregar la respuesta antes de que termine la clase.

12.3.2 Práctica 2 Formando El Equipo de Trabajo

Para el problema del edén que es el que está actualmente en ésta práctica se sugiere a los ayudantes:

- Se debe sugerir a los alumnos que retomen el equipo con el que trabajaron la práctica anterior. Si no lo desean preguntar la razón y hacer que se conformen nuevos equipos.
- Tomarles el tiempo a los alumnos para que trabajen de forma individual a lo más 15 minutos. Posteriormente darles la instrucción que es tiempo para que trabaje en equipo y tomarles el tiempo nuevamente a lo más 15 minutos, sin especificarles que se les está tomando el tiempo.
- Ya que estén trabajando en equipo, verificar si ellos se están tomando el tiempo y si no es así anotar los equipos que no lo hacen para posteriormente hacerles énfasis en que es algo importante, explicándoles que deben leer las instrucciones completas y preguntar si tienen dudas y que tomarse el tiempo es una forma de práctica útil en la Ingeniería de Software.
- Verificar si están tomando en cuenta la sugerencia de asignarse roles dentro del equipo y si no es así explicarles al final la importancia de asignarse roles.
- Recordarle a los alumnos que deben entregar la respuesta antes de que termine la clase.
- Darles máximo 15 minutos para que realicen su lista con las normas de equipo.
- Recordarle a los alumnos que deben entregar la respuesta del problema y la lista de las normas de equipo antes de que termine la clase.

12.3.3 Otros ejercicios

A continuación se dan otros ejercicios, para que el ayudante pueda aplicar a los alumnos en lugar de los que tiene la práctica 2, para el segundo ciclo en el que ya no hay prácticas ó para los días que no hay prácticas, dándoles así puntos extra o algún premio para los que lo contesten más rápido o en alguna dinámica que se haga dentro del grupo. Solo debe cambiar la sección desarrollo.

12.3.3.1 Ejercicio 1

DESARROLLO

Reunirse en equipos de cinco personas, para realizar las siguientes tareas en equipo:

- Con las herramientas que se les pidieron para esta práctica resolver el problema entre todo el equipo que a continuación se da.
- Con el cronómetro tomar el tiempo a partir de ahora hasta que terminen la práctica, anotarlo en una hoja por equipo con color verde.
- Con la calculadora hacer las cuentas que requieran, anotarlas en la misma hoja de equipo con color azul.
- Cualquier otra anotación u observación que hagan anotarlas en la misma hoja con color rojo.
- Si tuvieron algún problema anotarlo con color negro en la misma hoja.
- Finalmente anotar los nombres de cada integrante con color morado.

Problema

LOS CABALLOS

Un señor un día se compró un caballo a $665.95, al rato lo vendió a $753.63. Poco después, en el mismo mercado, volvió a comprar el mismo caballo a otra persona en $892.24; finalmente lo vendió en $ 806.96.

En equipo contestar las siguientes preguntas:

¿Gano o perdió?

¿Cuánto perdió o cuánto ganó?

¿Cómo les pareció la asignación de roles?

En la sección de próxima práctica de la práctica anterior a la que pongan este ejercicio debe poner:

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica N**
  - Leer la práctica N.
- **Documentos impresos que se utilizarán en la práctica N**

- Práctica N.

- **Herramientas que se utilizarán en la práctica N (por cada herramienta que no lleven será un punto menos)**
  - Un cronometro (Individual).
  - Una calculadora (Individual).
  - Una hoja (Individual).
  - Color verde, azul, rojo, morado y negro (Individual).

Las sugerencias para este ejercicio son:

- Cuando los alumnos ya estén trabajando en equipo, verificar que todos los equipos estén tomando el tiempo y si no es así no recordarles. Hacer la misma verificación con el uso de los colores y las anotaciones.
- Si existe algún equipo que no sea de cinco personas y alumnos sin equipo integrarlos a dicho equipo y posteriormente pedirles que hagan anotaciones sobre el ingreso de ese nuevo integrante.
- Asignar a cada integrante un rol, a continuación se da una lista de posibles roles para esta práctica, recordándoles la práctica anterior en la cual trabajaron sin roles y hubo dificultades. Mencionarles los siguientes roles pasar a cada equipo a asignarlos nosotros, si darles oportunidad de elegir a ellos.
  - Secretario (el que anotará lo que se pide en la práctica)
  - Líder (el que coordinara al equipo y lo guiara para que se realice una buena práctica)
  - Intransigente (El que se encargará de que todo salga mal), los otros integrantes no deberán saber cual es el rol de éste.
  - El que lleva el tiempo
  - El que leerá el problema.
- Hacerles ver o comprender lo importante de leer y seguir las instrucciones que se nos dan antes de cualquier tarea que se asigne.
- Verificar nuevamente si tomaron el tiempo o no, y decirles que esta vez si se les bajará puntos por no haber tomado en cuenta todas las instrucciones.
- La respuesta puede ser relativa, en cuestión de dinero gano $2.4, pero puede ser que los alumnos lo tomen en cuestión de tiempo el cual perdió al estar vendiendo y comprando al caballo.

Recordarles a los alumnos que deben entregar la respuesta antes de que termine la clase.

12.3.3.2 Ejercicio 2

DESARROLLO

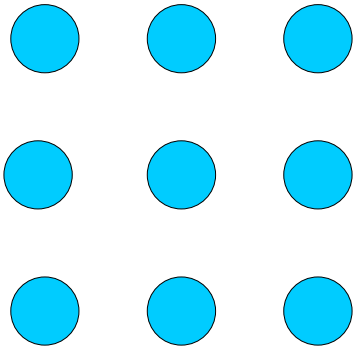
Reunirse en equipos de cinco personas, si es posible retomar el equipo que han estado formando las prácticas pasadas, si lo desean cambiar de equipo tomando en cuenta que la siguiente práctica es la última que tienen para formar el equipo definitivo. Con las herramientas que se les pidieron para esta práctica resolver el problema que a continuación se da entre todo el equipo y realizar las siguientes tareas:

- Con el cronometro tomar el tiempo a partir de ahora hasta que terminen la práctica, anotarlo en una hoja por equipo con color verde.
- Con la regla hacer las líneas que se requieran con lápiz (intentarlo las veces que sean necesarias para encontrar la solución).

- Cualquier otra anotación u observación que hagan anotarlas en la misma hoja con color rojo.
- Si tuvieron algún problema anotarlo con color negro en la misma hoja.
- Ya que encontraron las dos (mínimo) respuestas que se les piden trazar una con color azul y la otra con color morado, usando diferentes gráficas para cada respuesta.
- Finalmente anotar los nombres de cada integrante con color rosa.

Problema

LOS NUEVE PUNTOS



Entre todos los integrantes del equipo deben encontrar por lo menos dos de las opciones que hay para unir los nueve puntos arriba dibujados, usando a lo más cuatro líneas rectas, sin levantar el lápiz de la hoja ni repasar ninguna línea. Siguiendo las instrucciones arriba escritas sobre como deben usar los colores.

En la sección de próxima práctica de la práctica anterior a la que pongan este ejercicio debe poner:

PRÓXIMA PRÁCTICA

- **Actividades a realizar en casa antes de la práctica N**
  - Leer la práctica N.
- **Documentos impresos que se utilizarán en la práctica N**
  - Práctica N.
- **Herramientas que se utilizarán en la práctica N (por cada herramienta que no lleven será un punto menos)**
  - Un cronometro (Individual).
  - Una regla (Individual).
  - Dos hojas reciclables (Individual).
  - Un lápiz (Individual).
  - Color verde, azul, rojo, morado, rosa y negro (Individual).

Las sugerencias para este ejercicio son:

Para el problema de los nueve puntos que es el que esta actualmente en ésta práctica se sugiere a los ayudantes:

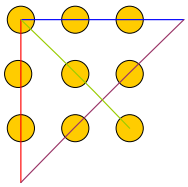
- Cuando los alumnos ya estén trabajando en equipo, verificar que todos los equipos estén tomando el tiempo. Hacer la misma verificación con el uso de los colores y las anotaciones. Recordándoles al final que no siguieron las instrucciones lo cual les bajara puntos.

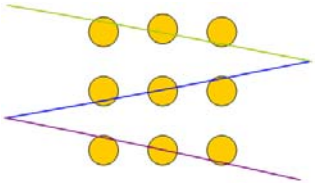
- Si existe algún equipo que no sea de cinco personas y alumnos sin equipo integrarlos a dicho equipo y posteriormente pedirles que hagan anotaciones sobre el ingreso de ese nuevo integrante.
- Hacerles ver o comprender a los alumnos lo importante de leer y seguir las instrucciones que se nos dan antes de cualquier tarea que se nos asigne, si es que aún no lo hacen.
- Dar máximo 30 minutos para realizar el problema de los nueve puntos.

Soluciones del problema de los nueve puntos:

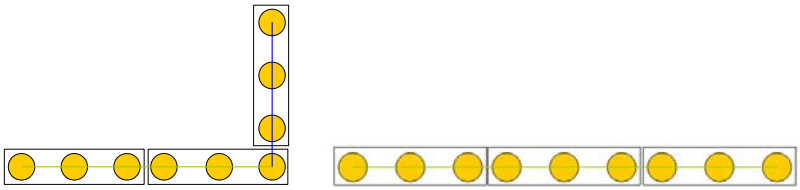
- **Restricciones:**
  - Encontrar por lo menos dos soluciones.
  - Usar a lo más cuatro líneas rectas.
  - No levantar el lápiz de la hoja.
  - No repasar ninguna línea.

- **Soluciones**





Ya que el alumno no tiene más restricciones, éste puede cortar el dibujo sin cortar los puntos y encontrar así más soluciones, a continuación se muestran soluciones cortando el papel.



12.3.4 Práctica 3 Comenzando a trabajar en equipo

- Especificarles que el logo, nombre y lema del equipo puede ser no los definitivos de su equipo pero que ya deben tener la idea o las ideas y eso será lo que entregarán.
- Tomarles el tiempo y recordarles que deben entregar todo antes de terminar la clase, dándoles a lo más 10 minutos por actividad.
- Darles su hoja con la lista de normas que realizaron la clase anterior para que la revisen y mejoren según lo que se les explico de normas y los problemas que les puedan surgir, sugerir normas que les puedan ayudar, como son: Puntualidad en las reuniones, cumplimiento a tiempo de las tareas, buen manejo de las métricas, es decir que se tomen correctamente el tiempo, etc.

12.4 Práctica para comenzar el desarrollo de software en equipo.

Ya que los alumnos han trabajado en equipo, los ayudantes deben comenzar a guiar a los alumnos a forman el equipo de desarrollo de software con el cual desarrollaran su proyecto. Los ayudantes deben explicar a los alumnos y hacerlos concientes de que trabajarán con ese equipo el resto del semestre.

12.4.1 Práctica 4 Establecer el equipo definitivo y llenado de formas básicas

- Dejarles claro a los alumnos que el equipo con el que trabajarán en esta práctica será el equipo definitivo durante todo el semestre en el desarrollo del software.
- Verificar que todos traigan las formas que se les pidieron para que las entreguen llenas, y preguntar si tienen dudas de cómo llenarlas.
- Entregarles los documentos que hicieron la semana pasada (logo, nombre, lema, nombres de los integrantes del equipo, asignación de roles) para que decidan si son los definitivos o no.
- Entregar a los alumnos la lista de normas y el acuerdo de su reunión semanal, para que los rectifiquen y los firmen estando consientes de que es un acuerdo de todo el equipo, que si no cumplen serán sancionados dentro de su equipo.
- Cerciorarse que tanto la lista de sanciones como de normas estén firmadas por todos los integrantes del equipo.

12.5 Práctica para la fase de Lanzamiento

En esta fase los ayudantes deben establecer sus propios objetivos para con ellos mismos, con los alumnos y con las profesoras, ya que de esta forma también se podrán evaluar a final de ciclo y de semestre.

12.5.1 Práctica 5 Establecer objetivos, estándares y riesgos.

- Esta práctica la pueden terminar los alumnos en casa o durante su reunión de equipo.
- Sugerir a los equipos que la reunión semanal de equipo la realicen el viernes durante su hora de laboratorio ya que no hay práctica para ese día.
- Verificar que los alumnos están llenando de forma conciente los objetivos y riesgos del equipo, acercarse a los equipos y preguntarles por qué tacharon alguno.
- Darles la opción de que el estándar lo manden por correo ya que hayan realizado su junta.

12.6 Prácticas para la fase de Estrategia

Los ayudantes deben comenzar a interactuar más con los alumnos viéndolos o tratándolos ya como un equipo que desarrollará un software, motivándolos a seguir el proceso que se plantea en el curso y eliminando las dudas de los alumnos de que es un proceso tedioso.

12.6.1 Práctica 6 Estrategia de desarrollo

- Leer el problema de la librería electrónica con los alumnos y realizar la estrategia que se encuentra en la práctica con los alumnos.
- Darles máximo 40 minutos a los alumnos para que realicen la nueva estrategia.
- Que un integrante de cada equipo pase a explicar a los otros equipos la estrategia que realizaron y la justificación de ella.
- Que entre todo el grupo se decida cual fue la mejor estrategia y por que o de todas las que se realizaron hacer una nueva en la que todos o la mayoría estén de acuerdo.
- Recordar a los alumnos en particular a los administradores de apoyo, que deben conseguir un software para realizar diagramas de gantt para la próxima práctica.

- Pedir a cada equipo que entregue la estrategia que hizo.

12.6.2 Práctica 17 Administración de la configuración

- Recordarle a los alumnos una práctica antes que deben traer su repositorio si es que no está en línea, para que realicen el Informe del Estado de la Configuración.
- Acercarse con cada equipo por separado y preguntarles si tienen alguna duda sobre esa práctica o sobre los documentos en general, especificándoles que es la última práctica.
- Pedir que se envié la práctica por correo electrónico.
- Recordarles que es la última práctica que llevaran a cabo durante el semestre e invitarlos a realizar sus reuniones de equipo a la hora de la clase de laboratorio para ser guiados y apoyados por los ayudantes.

12.7 Práctica para la fase de Planeación

En esta fase los alumnos comienzan a utilizar software nuevo para su proyecto. El ayudante debe estar familiarizado con él. Para esto debe saber que en la página del curso existen enlaces y tutoriales del software que se usará a lo largo del semestre.

Para esta fase en la parte de herramientas, está la parte de fase de Planeación, para la cual existen dos enlaces. El que se propone usar con los alumnos es el de GanttProject para el cual existe le tutorial en la página.

12.7.1 Práctica 7 Planeación del proyecto y registro de defectos

- Preguntarles a los alumnos si consiguieron algún software en especial y si ya lo saben usar, si es así decirles que con ese software realicen la práctica.
- Si no lo consiguieron, recordarles que en la página del curso existe un tutorial de GanttProject en el cual aprenderán a usarlo e instalarlo.
- Darles máximo 30 minutos para que realicen los diagramas de gantt.
- Darles a lo más 15 minutos para que realicen las revisiones entre colegas.
- Pedir que entreguen la forma de registro de defectos antes de irse y que envíen los diagramas en formato de imagen por correo electrónico.

12.8 Práctica para la fase de Especificación de Requerimientos

En esta fase los ayudantes deben ser conscientes que es difícil para los alumnos y en algunos casos muy tediosa, ya que los alumnos no están acostumbrados a realizar documentación de un programan y en la mayoría de los casos tampoco han usado UML. Es por esto que los ayudantes deben tener mucha paciencia con los alumnos y disposición para apoyarlos en la realización de los diagramas de su proyecto. Mostrándoles varios ejemplos y programas para que ellos decidan cual es la mejor opción y trabajen con ella.

Explicarles que hay diagramadores de UML que ya cuentan con la opción de generar el código que son un poco más difíciles de usar, y que existen otros que no generan el código pero que son más fáciles de usar, que el código que se genera es solo el esqueleto del programa. Con esta información los alumnos estarán concientes de las ventajas y desventajas de los diagramadores y podrán elegir.

En esta fase los ayudantes deben ser conscientes que los alumnos en muchas ocasiones no han construido un prototipo de interfaz de usuario y por lo tanto no cuentan con los conocimientos de alguna herramienta. Se

propone que los alumnos utilicen HTML para que construyan la interfaz, si los alumnos conocen alguna otra herramienta se recomienda que se les permita usar dicha herramienta.

12.8.1 Práctica 8 Diagrama de casos de uso

- Leer y realizar con los alumnos el ejemplo que se presenta en la práctica, realizar un ejemplo del detalle de un caso de uso, pasar algún alumno a que realice otro caso de uso.
- Dejar claro que el ejemplo que se presenta del detalle muestra otros dos casos de uso y que no necesariamente todos los detalles deben presentar esos dos casos de uso, si es necesario dar más ejemplos a los alumnos.
- Explicar un poco sobre el modelado de UML y el software, recomendar alguno a los alumnos y especificar que lo usarán en las prácticas.
- Dar máximo 30 minutos para realizar su diagrama general de casos de uso.
- Pedir que envíen los diagramas que se elaboraron en formato de imagen así como las tablas para los detalles de los casos de uso.
- Recordar a los alumnos que la siguiente práctica deberán llevar alguna herramienta para realizar la interfaz de usuario, se les explicará HTML y si ellos quieren otra la deben llevar.

12.8.2 Práctica 9 Prototipo de interfaz de usuario

- Preguntar si alguno desea usar otra herramienta que lo puede hacer siempre y cuando se haga responsable de sus dudas.
- Después de haber explicado los ejemplos que vienen en la práctica, explicar algún otro que pueda ser continuado en las siguientes prácticas.
- Darles máximo 30 minutos para que realices sus prototipos y pasar a revisarlos a cada equipo.
- Pedir que envíen lo que tengan cuando termine la clase en formato de imagen.

12.8.3 Práctica 10 Requerimientos no funcionales y Plan de Pruebas del Sistema

- Leer los ejemplos con los alumnos y si es necesario explicarlos con más ejemplos de los requerimientos no funcionales.
- Darles no más de 20 minutos para que realicen la lista de requerimientos no funcionales y pedir que la entreguen.
- Posteriormente pedir que realicen el plan de pruebas del sistema en no más de 20 minutos y que al terminarlo lo entreguen o lo envíen.

12.9 Práctica para la fase de Diseño

En esta fase los alumnos realizarán otro plan de pruebas para su sistema según la arquitectura que ellos mismos van planteando. Se van adentrando cada vez más a lo que será el sistema en sí y se van dando cuenta de que todo lo que han hecho les servirá para que se realice un buen trabajo de software.

EL ayudante debe contar con los conocimientos de UML que se requieren en esta fase para que pueda explicar correctamente a los alumnos y se debe apoyar en los diversos tutoriales que están en la página del curso.

12.9.1 Práctica 11 Diseño de la arquitectura plan de pruebas de Integración

- Explicar a los alumnos los ejemplos que están en la práctica y llevar otros ejemplos también para explicar.
- Dar máximo 40 minutos para las actividades y pasar a cada equipo pidiendo que expliquen por qué la hicieron de tal forma verificando si les quedó claro o no.

12.9.2 Práctica 12 Diagramas de clases y de secuencia

- Explicar a los alumnos los ejemplos que están en la práctica y llevar otros ejemplos también para explicar.
- Verificar que los alumnos traigan el material que se les pidió e indicar cuales diagramas deben elaborar.
- Dar máximo 40 minutos para las actividades y pasar a cada equipo pidiendo que expliquen por que la hicieron de tal forma verificando si les quedo claro o no.
- Si los alumnos no avanzan rápido pedir solo un diagrama de secuencia y uno de clases y que los demás los manden por correo.

12.10 Práctica para la fase de Construcción

En esta fase los alumnos comienzan con la construcción del sistema. Si los equipos utilizarán java para realizar su sistema, el ayudante debe tener el conocimiento necesario para brindarles el apoyo y si utilizarán algún otro lenguaje del cual no se tiene conocimiento se les debe dejar claro es bajo su riesgo.

En esta fase los alumnos están muy estresados con la realización del sistema, por lo tanto se les debe apoyar para que salgan adelante como equipo, brindándoles soluciones sencillas con las cuales logren terminar el alcance de su proyecto en este ciclo.

12.10.1 Práctica 13 Plan de pruebas unitarias

- Que un alumno de cada equipo explique los que son los diferentes tipos de pruebas, ya sea con los ejemplos que están en la práctica realizando nuevos ejemplos.
- Llevar más ejemplos para explicar a los alumnos.
- El resultado del ejercicio uno es:

Código:

```
1     private String[] datosVistaV;
2   Almacena tmp = new Almacena();
3
4    public Vacante[] buscarVacante(){
5    LinkedList listaPosicion = noNulos(datosVistaV);
6    Vacante[] resultado = tmp.buscarVacante(listaPosicion, datosVistaV);
7    return resultado;
8    } //end buscarVacante
9
10    /*
```

```
11   * Metodo para verificar que patrones se usaran para buscar
12   */
13  private LinkedList noNulos(String datos[]){
14  LinkedList listaPosicion = new LinkedList();
15  for (int i = 0; i < 4; i++){
16     if (datos[i] !=  ""){
17              listaPosicion.add(i);
18     } //end if
19  } //end for
20  return listaPosicion;
21  } //end noNulos
```

Código	Tipo de Prueba	Caso de prueba	Resultado esperado	Resultado obtenido
VistaVacante.buscarVante()	Caja blanca	i = 4	listaPosicion tenga 4 elementos o menos que representan las valores no nulos de datos y ya no se agreguen mas y ejecutar la línea 6	listaPosicion con exactamente 4 elementos o menos
Capa de Control		i=0; datos[0]=””	listaPosicion = Vacía ; incrementar i ; verificar el siguiente dato y una vez que i = 4 ejecutar la línea 6	listaPosicion = Vacía
		i=0; datos[0]=”b”	listaPosicion debe contener el valor de i, incrementar i; verificar el siguiente dato y una vez que i = 4 ejecutar la línea 6	listaPosicion con un elemento el cual es i

Complejidad ciclomatica: 2 + 1 = 3

En línea 15 hay una condición para el for, y un if con otra condición. Por lo tanto se necesitan 3 casos de prueba, es decir 3 caminos diferentes.

Casos de prueba:

1. Entrar al for donde no se cumplirá la condición, es decir i = 4 o i >=4. Por lo tanto el dato de prueba será i = 4.
    a. Camino 1: Pasa por las líneas 1, 2, 4, 5, 13, 14, 15, 19, 20, 21, 6, 7 y 8
2. Entrar al for y al if donde no se cumplirá la condición, es decir datos[i] = a la cadena vacía. Por lo tanto el dato de prueba será i = 0 y datos[0] = ””.
    a. Camino 2: Pasa por las líneas 1,2,4,5,13,14,15,16,18,19,20,21,6,7 y 8
3. Entrar al for y al if, es decir se cumplirán ambas condiciones. Por lo tanto los datos de prueba son i = 0 y datos[0] = “b” ; donde “b” es la cadena b.
    a. Camino 3: Pasa por las líneas 1,2,4,5,13,14,15,16,17,18,19,20,21,6,7 y 8.

Plan de Prueba:

En la línea 6, el método *buscarVacante(LinkedList lista, String[] datos)* de la clase Almacena de la capa de Datos, hace lo siguiente:

- Si lista = {0,1,2,3} o Vacía y datos es el arreglo de parámetros de búsqueda, regresa todas las vacantes registradas.
- Si lista.size() >= 1 y lista.size() < 4, regresa las vacantes que coincidan con los parámetros de búsqueda según datos.

El camino 1, las líneas 1,2,4,5,13,14,15 se ejecutan bien, pero como i = 4, ya no cumple con la condición del for con lo que no se agregan mas elementos y se regresa la lista para continuar con la ejecución de la línea 6.

El camino 2, las líneas 1,2,3,4,5,13,14,15,16 se ejecutan bien, ya que entra al for por que i=0, pero no al if ya que datos[0] = "", por lo tanto no se agrego nada a la lista. Al terminar el for, se regresa la lista y se ejecuta la línea 6

El camino 3, las líneas 1,2,3,4,5,13,15,16,17 se ejecutan bien, ya que cumplen ambas condiciones (i = 0 y datos[0]= "b"), la instrucción de la línea 17 agrega un elemento a la lista justamente el valor de i, y continua su ejecución normal.

- El resultado del ejercicio dos dependerá del proyecto de los alumnos o del caso de uso que se indique.
- Dar máximo 20 minutos por ejercicio para poder ver como avanzan y si entendieron los alumnos.

12.11 Prácticas para la fase de Pruebas del Sistema

En esta fase se supone que los alumnos ya tienen su código, lo integraron y al que le deben realizar las pruebas de integración y las pruebas al sistema. Ya que realizaron dichas pruebas deben realizar los manuales del sistema para los cuales ya cuentan con la documentación que han ido realizando a lo largo de cada fase y que está integrada en una carpeta tanto electrónica como impresa solo les queda realizar dos manuales.

12.11.1 Práctica 14 Pruebas de integración y del sistema

- Recordarle a los alumnos una clase antes que deben traer su sistema o su parte del sistema para que comiencen con la integración o pruebas del mismo.
- Verificar que los alumnos estén trabajando con su sistema y si lo requieren ayudarles.

12.11.2 Práctica 15 Manuales del sistema

- Recordarle a los alumnos una clase antes que deben traer su sistema y la documentación del mismo.
- Si es posible llevar a los alumnos manuales de otros sistemas y carpetas de otros cursos para que se den una idea de cómo lo deben hacer.

12.12 Prácticas para la fase de Cierre

En esta fase los alumnos han terminado el sistema y solo les resta realizar las evaluaciones y las métricas que deberán tomar en cuenta en el siguiente ciclo.

12.12.1 Práctica 16 Evaluación, lecciones aprendidas, propuestas de mejoras y mediciones

- Explicar a los alumnos la importancia de la evaluación a sus equipos, a sí mismos y a las clases y ayudantes.
- Darles máximo 15 minutos para que platiquen y reflexionen como equipo y llenen la forma Lecciones aprendidas y sugerencias de Mejora.
- Posteriormente darles máximo 10 minutos para evaluar a los ayudantes, las clases y las prácticas.
- Dar máximo 15 minutos para que se evalúen y evalúen a su equipo.

Platicar con todo el equipo dando la confianza para que den su opinión a todo el grupo de cómo les pareció su trabajo como equipo, personal y en el salón de clases. Preguntando opiniones sobre las clases y proponiendo cambios si se requiere.

Conclusiones

Durante la realización de este trabajo, confirme la importancia de tener un buen material didáctico para impartir una clase, de desarrollar un software con calidad, de seguir una lista de requerimientos, de tomar métricas, de tener un buen equipo de trabajo, un buen ambiente de trabajo, buena comunicación, de saber trabajar con equipos de desarrollo de software y finalmente la importancia de saber transmitir el conocimiento a los alumnos.

Durante el primer semestre que di clases como ayudante de la materia, me di cuenta que los alumnos y ayudantes necesitábamos prácticas más dinámicas y actualizadas, que apoyaran a los alumnos con su proyecto final y para trabajar en equipo. Se contaba con una serie de prácticas que debían evolucionar tanto en información como en didáctica, al igual que la materia lo había hecho. Mi objetivo principal fue renovar estas prácticas, de tal manera que los alumnos pudieran obtener y refirmar conocimiento con ellas y los ayudantes explicar mejor los temas.

La primera versión de las prácticas, que en este trabajo se presentan, fueron terminadas antes del semestre 2007-1, para que se utilizarán durante ese semestre y de esta forma mejorarlas terminando el semestre y realizar el manual para los ayudantes de la materia.

Una de las cosas que se cambiaron, fueron las instrucciones en las prácticas, ya que los alumnos tienen la costumbre de no trabajar en equipo y de no leer las instrucciones. Se escribió en cada práctica, una sección en la que se da la información a los alumnos de lo que deben hacer y llevar para la siguiente práctica, dicha información también se escribió para los alumnos en la página del curso en la liga de cada práctica.

Al finalizar cada clase o práctica con los alumnos, anotaba las dudas o dificultades que se me presentaban y de esta forma realice el manual para los ayudantes de la materia. En dicho manual agrupo las sugerencias para cada práctica, menciono sugerencias de cómo hacer para que los alumnos sigan las instrucciones, realicen los ejercicios, planteen sus preguntas, etc.

Por sugerencia de los alumnos se eliminaron algunos ejercicios de las prácticas, ya que era mucho trabajo para ellos realizar su proyecto y aparte un semi-proyecto en las prácticas, esta información la saque de las evaluaciones que ellos realizaron a las prácticas, en la fase de cierre del primer ciclo y de las evaluaciones que las profesoras realizaron a los alumnos al termino del semestre.

Las prácticas tienen ejercicios individual o en equipo de un problema sencillo para realizar un software, con el cual se trabaja a lo largo de todas las prácticas, el ayudante realiza los ejemplos que están basados en dicho problema con ayuda de los alumnos, en algunas prácticas los alumnos realizan una parte solos, para que posteriormente lo realicen en su proyecto, de esta forma les surgen dudas a los alumnos antes de comenzar el trabajo de su proyecto.

En ocasiones es necesario que los ayudantes presenten más ejemplos para que los alumnos comprendan por completo los conceptos. Para esto se incluyeron en la página del curso una serie de ejemplos (según la práctica) que fueron recopilados de proyectos de semestres anteriores, pueden ser tomados por los ayudantes para ejemplificar más las prácticas.

Los alumnos también cuentan con un ejemplo de un proyecto completo, que se realizó como apoyo al libro, al cual también pueden acceder desde la página web.

Con esto se cumplió mi principal objetivo de aportar a la materia un material didáctico tanto para los alumnos como para los ayudantes, como son las prácticas de la materia y el manual para los ayudantes de la misma.

Otro de los objetivos de este trabajo surgió después de haber cursado la materia, ya que en ninguna materia se me había presentado la oportunidad de tener un equipo de trabajo con las características que requiere la Ingeniería de Software, fue cuando me surgió la inquietud de investigar más sobre el trabajo en equipo y en particular el trabajo en equipo para la realización de un software.

De esa forma fue que realice un capítulo sobre las técnicas para trabajar en equipo y con equipos de desarrollo de software. Dicho capítulo también se llevo a la práctica durante el semestre 2007-1, con lo que me di cuenta la

falta que hace, que se nos inculque la cultura del trabajo en equipo, no solo para la materia de Ingeniería de Software si no en general en la vida diaria.

Durante el semestre 2007-1 utilice la primera versión de las prácticas. Al inicio del semestre, se les aplicó un ejercicio a los alumnos para comenzar a integrar su equipo y a utilizar las técnicas de trabajo en equipo, me di cuenta que deben realizar otro ejercicio, para que se integren completamente y posteriormente comenzar con el desarrollo de software en equipo, por lo que agregue una práctica con otro ejercicio en equipo.

Llevando a cabo las técnicas y los ejercicios que se presentan en las prácticas se comienza a crear la conciencia de equipo en los alumnos desde el primer día de clases para que estos se den cuenta de que es lo más importante para poder realizar un buen proyecto. Con esto se cumplió mi objetivo de proporcionar técnicas para el desarrollo de software en equipo.

Me queda claro que el trabajo que realice debe seguir creciendo y evolucionando por lo menos cada semestre que es impartida la materia, ya que las prácticas en algún tiempo deberán ser actualizadas, para que continúen siendo un buen apoyo para la materia. De la misma forma, las técnicas de trabajo en equipo deben actualizarse con más ejercicios que apoyen el trabajo en equipo.

Bibliografía

Libros

- **[HUMPHREY]**

  Humphrey Watts S.

  "*Introduction to the Team Software Process"*.

  SEI Series in Software Engineering. Addison-Wesley. 2000.

- **[IBARGÜENGOITIA y OKTABA]**

  Ibargüengoitia G. Guadalupe; Oktaba Hanna

  "*Ingeniería de Software Pragmática"*.

  Sitio del curso http://victoria.fciencias.unam.mx/cursois/index.htm.

- **[JOSEFINA]**

  Josefina Rodríguez Jacobo

  "*Formación de grupos de desarrollo de software: Un enfoque psicosocial"*.

  Ediciones Yoltéotl. 2004

- **[BOOCH]**

  Booch G.; Rumbaugh J.; Jacobson I.

  "*The Unified Modeling Language User Guide*".

  Adisson-Wesley. 2005.

- **[PRESSMAN]**

  Roger S. Pressman.

  "Software Engineering: A Practioner's Approach".

  Boston; Mexico City: McGraw-Hill, c2001.

- **[SOMMERVILLE]**

  Sommerville, Ian.

  "Software Engineering"

  Harlow: Addison-Wesley, 2000.

- **[PFLEEGER]**

  Shari Lawrence Pfleeger

  "Software Engineering Theory and Practice"

Upper Saddle River, New Jersey : Prentice Hall, 1998.

Artículos

- **[IBARGÜENGOITIA y OKTABA]**

  Ibargüengoitia G. Guadalupe; Oktaba Hanna

  Una alternativa práctica en la enseñanza de la Ingeniería de Software

- **[OKTABA y ORTEGA]**

  Hanna Oktaba y Jorge L. Ortega Arjona

  Una Propuesta de Análisis de Necesidades mediante las Gráficas de Dependencia en la fase de Estrategia de TSPi

  Memorias del Workshop Iberoamericano de Ingeniería de Requisitos y Ambientes de Software IDEAS 2002, Universidad de la Habana, La Habana, Cuba, 23 al 26 de abril de 2002.

- **[IBARGÜENGOITIA y OKTABA]**

  Hanna Oktaba y Guadalupe Ibarguengoitia G.

  Calidad en procesos de software, ejemplo de TSPi.

  Capítulo en libro “Calidad en el desarrollo y mantenimiento del software, Mario G. Piattini y Félix O. García, RA-MA, 2003.

- **[IBARGÜENGOITIA y OKTABA]**

  Guadalupe Ibarguengoitia G. y Hanna Oktaba.

  Material didáctico de Ingeniería de Software para principiantes, memorias del Taller de Ingeniería de Software del VI Encuentro Internacional de Ciencias de la Computación ENC 2005, Benemérita Universidad Autónoma de Puebla, Puebla, Puebla, septiembre de 2005.

Referencias electrónicas

- **Página del curso de Ingeniería de Software.**
  http://victoria.fciencias.unam.mx/cursois