Universidad Autónoma del Estado de México

Centro Universitario UAEM Ecatepec

Licenciatura en Informática Administrativa

Unidad de Aprendizaje:

Seminario de Ingeniería de Software

Autor:

Patricia Delgadillo Gómez

Semestre:

2016A

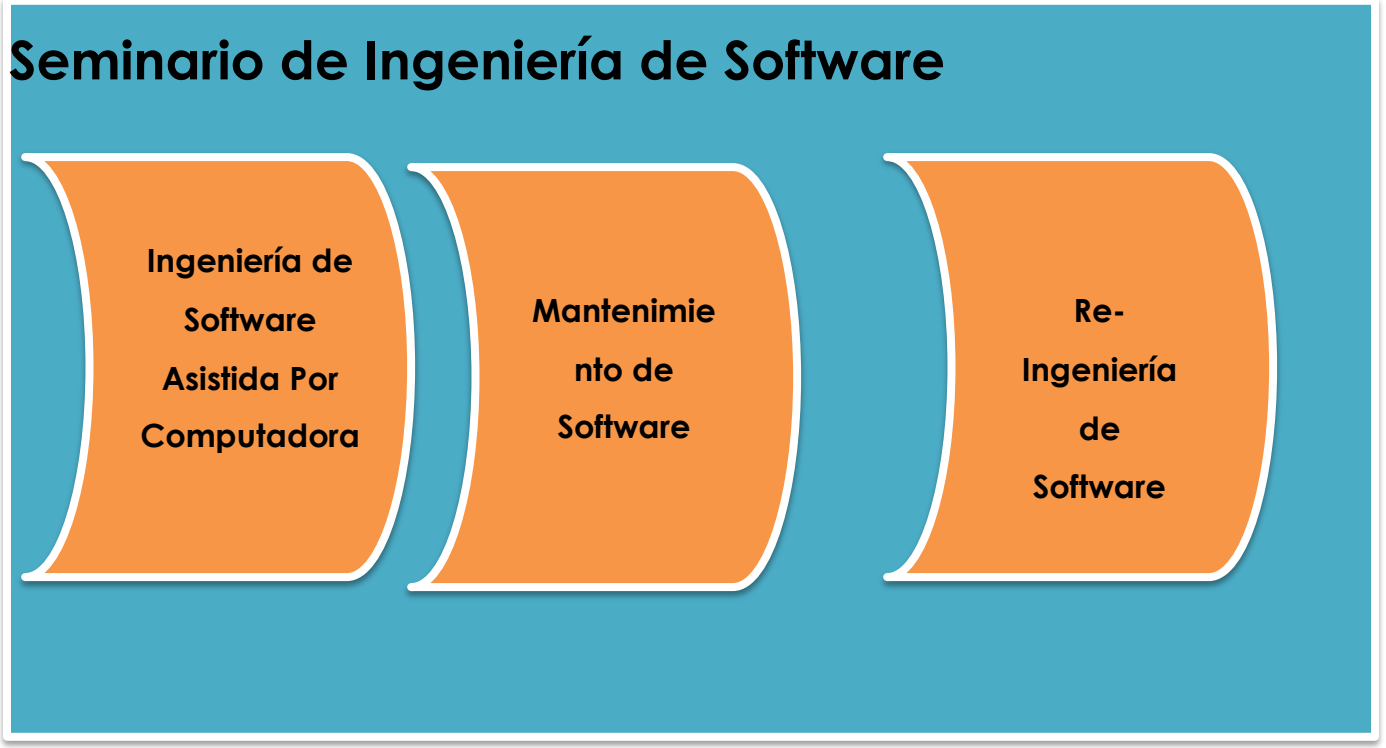
Guion Explicativo

El presente material didáctico esta diseñado como apoyo a la asignatura *Seminario de Ingeniería de Software* con la finalidad de que el alumno obtenga los conocimientos y herramientas necesarios para el desarrollo, mantenimiento y reingeniería de Software.

Objetivo General

Debido a los avances de la ciencia y a la creciente demanda de software cada vez más sofisticado y complejo surge la necesidad de contar con bases que permitan dar a los alumnos los conocimientos y herramientas para el desarrollo, mantenimiento y Reingeniería de Software.

Secuencia Didáctica



Contenido Temático

Unidad I: Ingeniería de Software Asistida Por Computadora

Objetivos de la Unidad I:

- Discutir las consecuencias relacionadas al CASE y la tecnología CASE.
- Sugerir una clasificación para los sistemas CASE.
- Discutir la integración de las herramientas CASE.
- Describir el ciclo de vida CASE.

Unidad I: Ingeniería de Software Asistida Por Computadora

¿Qué significa Computer-aided Software Engineering?

"CASE es la automatización del software"
CASE es una filosofía que se orienta a la mejor comprensión de los modelos de empresa, sus actividades y el desarrollo de los sistemas de información.

Esta filosofía involucra además el uso de programas que permiten:

- Construir los modelos que describen la empresa.
- Describir el medio en el que se realizan las actividades.
- Llevar a cabo la planificación.

- El desarrollo del Sistema Informático, desde la planificación, pasando por el análisis y diseño de sistemas, hasta la generación del código de los programas y la documentación.

- "La creación de sistemas software utilizando técnicas de diseño y metodologías de desarrollo bien definidas, soportadas por herramientas automatizadas operativas en el ordenador"

Objetivos del CASE

- Aumentar la productividad de las áreas de desarrollo y mantenimiento de los sistemas informáticos.

- Mejorar la calidad del software desarrollado.

- Reducir tiempos y costes de desarrollo y mantenimiento del software.

- Mejorar la gestión y dominio sobre el proyecto en cuanto a su planificación, ejecución y control.

- Mejorar el archivo de datos (enciclopedia) de conocimientos (know-how) y sus facilidades de uso, reduciendo la dependencia de analistas y programadores.

- Automatiza.

- El desarrollo del software

- La documentación

- La generación del código

- El chequeo de errores

- La gestión del proyecto

- Permitir

- La reutilización (reusabilidad) del software

- La portabilidad del software

- La estandarización de la documentación

- Integrar las fases de desarrollo (ingeniería del software) con las herramientas CASE.

- Facilitar la utilización de las distintas metodologías que desarrollan la propia ingeniería del software.

Clasificación del CASE

Las herramientas CASE se pueden clasificar por su función, por su papel como instrumentos para administradores o personal técnico, por su utilización en los distintos pasos del proceso de Ingeniería del Software, por la arquitectura de entorno (hardware y software) que les presta su apoyo, o incluso por su origen o su coste.

- ***Herramientas de la Ingeniería de la Información.***

  Al modelar los requisitos de información estratégica de una organización, las herramientas de la ingeniería de la información proporcionan un <<metamodelo>> del cual se derivan sistemas de información específicos. En lugar de centrarse en los requisitos de una aplicación específica, estas herramientas CASE modelan la información de negocios cuando esta se transfiere entre distintas entidades organizativas en el seno de una compañía.

  El objetivo primordial de las herramientas de esta categoría consiste en representar objetos de datos de negocios, sus relaciones y la forma en que fluyen estos objetos de datos entre distintas zonas de negocio en el seno de la compañía.

- ***Modelado de procesos y herramientas de administración***

  Si una organización intenta mejorar un proceso de negocios (o de software) lo primero que debe hacer es entenderlo. Las herramientas de modelado de procesos (que también se denominan herramientas de <<tecnología de procesos>> se utilizan para representar los elementos claves del proceso de modo que sea posible entenderlo mejor.

  Estas herramientas también pueden proporcionar vínculos con descripciones de procesos que ayuden a quienes estén implicados en el proceso de comprender las tareas que se requieren para llevar a cabo ese proceso.

  Además, las herramientas de administración de procesos pueden proporcionar vínculos con otras herramientas que proporcionen un apoyo para actividades de procesos ya definidas.

- ***Herramientas de planificación de proyectos***

  Las herramientas de estas características se concretan en dos áreas primordiales: estimación de esfuerzos de proyecto y de costes de software, y planificación de proyectos. Las herramientas de estimación calculan el esfuerzo estimado, la duración del proyecto y el número recomendado de personas.

Las herramientas de planificación de proyectos capacitan al administrador para definir todas las tareas del proyecto (la estructura de desglose de tareas), para crear una red de tareas (normalmente empleando una entrada gráfica), para representar las interdependencias entre tareas y modelar la cantidad de paralelismo que sea posible para ese proyecto.

- ***Herramientas de análisis de riesgos***

  La identificación de riesgos potenciales y el desarrollo de un plan para mitigar, monitorizar y administrar estos riesgos tiene una importancia fundamental en los grandes proyectos.

  Las herramientas de análisis de riesgos capacitan al administrador del proyecto para construir una tabla de riesgos proporcionando una guía detallada en la identificación y análisis de riesgos.

- ***Herramientas de administración de proyectos***

  La planificación del proyecto y el plan de proyecto deben de seguirse y de monitorizarse de forma continúa. Además, el gestor deberá de utilizar las herramientas que recojan métricas que en última instancia proporcionen una indicación de la calidad del producto del software.

  Las herramientas de esta categoría suelen ser extensiones de herramientas de planificación de proyectos.

- ***Herramientas de seguimiento de requisitos***

  Cuando se desarrollan grandes sistemas, el sistema proporcionado suele no satisfacer los requisitos especificados por el cliente.

  El objetivo de las herramientas de seguimiento de requisitos es proporcionar un enfoque sistemático para el aislamiento de requisitos, comenzando por la solicitud del cliente de una propuesta (RFP) o especificación.

Las herramientas de trazado de requisitos típicas combinan una evaluación de textos por interacción humana, con un sistema de gestión de bases de datos que almacena y categoriza todos y cada uno de los requisitos del sistema que se <<analizan>> a partir de la RFP o especificación original.

- ***Herramientas de métricas y gestión***

  Las métricas de software mejoran la capacidad de administrador para controlar y coordinar el proceso del software y la capacidad del ingeniero para mejorar la calidad del software que se produce.

  Las métricas y herramientas de medida actuales se centran en procesos, proyectos y características del producto.

  Las herramientas orientadas a la administración capturan métricas específicas del proyecto que proporcionan una indicación global de productividad o de calidad.

  Las herramientas orientadas técnicamente determinan métricas avanzadas mantienen una base de datos de medidas de la industria. Basándose en características de proyectos y de productos proporcionales por el usuario, estas herramientas <<califican>> los números locales frente a los valores medios de la industria (y frente el rendimiento local anterior) y sugieren estrategias para llegar a mejoras.

- ***Herramientas de documentación***

  Las herramientas de producción de documentos y de autoedición prestan su apoyo a casi todos. Los aspectos de la ingeniería del software, y representan una importante oportunidad de <<aprovechamiento>> para todos los desarrolladores de software.

  La mayor parte de las organizaciones dedicadas al desarrollo de software invierte una cantidad de tiempo considerable en el desarrollo de documentos, y en muchos casos el proceso de documentación en si resulto bastante deficiente.

  No es infrecuente que una organización de desarrollo de software invierta hasta un 20 o un 30 por ciento de su esfuerzo global de desarrollo de software de documentación. Por esta razón, las herramientas de documentación suponen una oportunidad importante para mejorar la productividad.

- ***Herramientas de software de sistema***

  CASE es una tecnología de estaciones de trabajo. Por tanto el entorno CASE debe adaptarse a un software de sistema en red de alta calidad, al correo electrónico, a los bolees electrónicos y a otras capacidades de comunicaciones.

- ***Herramientas de control de calidad***

  La mayor parte de las herramientas CASE que afirman que tienen como principal interés el control de calidad son en realidad herramientas métricas que hacen una auditoria en código fuente para determinar si se ajusta o no a ciertos estándares lenguaje.

  Otras herramientas extraen métricas técnicas un esfuerzo para extrapolar la calidad del software que se está construyendo.

- ***Herramientas de gestión de bases de datos***

  El software de gestión de bases de datos sirve como fundamento para establecer una base de datos CASE, que también se denominará base de datos del proyecto.

Dado el énfasis acerca de los objetos de configuración de herramientas de gestión de base de datos para CASE pueden evolucionar a partir de los sistemas de gestión de bases de datos relacionales (SGBDR) para transformarse en sistemas de gestión de bases de datos orientadas a objetos (SGBDOO).

- ***Herramientas de gestión de configuración de software***

  La gestión de configuración de software (GCS) se encuentran en el núcleo de todos los entornos CASE. Las herramientas pueden ofrecer su asistencia en las cinco tareas principales de GCS: identificación, control de versiones, control de cambios, auditoría y contabilidad de estados.

  La base de datos CASE proporciona un mecanismo para identificar todos los elementos de configuración y relacionarlo con otros elementos se pueden implementar con ayuda de herramientas especializadas; un acceso sencillo a los elementos de configuración individuales facilita el proceso de auditoría; y las herramientas de comunicación CASE pueden mejorar enormemente la contabilidad de estados (ofreciendo información acerca de los cambios a todos aquellos que necesiten conocerlos).

- ***Herramientas de análisis y diseño***

  Las herramientas de análisis y diseño capacitan al ingeniero del software para crear modelos del sistema que haya que construir. Los modelos contienen una representación de los datos, de la función y del comportamiento, así como caracterizaciones del diseño de datos, arquitectura, procedimientos e interfaz.

Al efectuar una comprobación de la consistencia y validez del modelo, las herramientas de análisis y diseño proporcionan al ingeniero del software un cierto grado de visión en lo tocante a la representación del análisis, y le ayudan a eliminar errores antes de que se propaguen al diseño, o lo que es peor, a la propia implementación.

- ***Herramientas PRO/SIM***

  Las herramientas PRO/SIM proporcionan al ingeniero del software la capacidad de predecir el comportamiento de un sistema en tiempo real antes de llegar a construirlo.

  Además, capacitan al ingeniero del software para desarrollar simulaciones del sistema de tiempo real que permitirán al cliente obtener ideas acerca de su funcionamiento, comportamiento y6 respuesta antes de la verdadera implementación.

- ***Herramientas de desarrollo y diseño de interfaz***

  Las herramientas de desarrollo y diseño de interfaz son en realidad un conjunto de primitivas de componente de programas tales como menús, botones, estructuras de ventanas, iconos, mecanismos de desplazamiento, controladores de dispositivos etc.

Sin embargo, estos conjuntos de herramientas se están viendo sustituidos por herramientas de generación de prototipos de interfaz que permiten una rápida creación en pantalla de sofisticadas interfaces de usuario, que se ajustan al estándar de interfaz que se haya adoptado para el software.

- ***Herramientas de generación de prototipos***

  Se puede utilizar toda una gama de herramientas de generación de prototipos. Los *generadores de pantallas* permiten al ingeniero del software definir rápidamente la disposición de la pantalla para aplicaciones interactivas.

  Otras herramientas de prototipos CASE más sofisticadas permiten la creación de un diseño de datos, acoplado con las disposiciones de la pantalla y de los informes simultáneamente. Muchas herramientas de análisis y diseño proporcionan extensiones que ofrecen alguna opción de generación de prototipos.

  Las herramientas PRO/SIM generan un esqueleto de código fuente en Ada y C para las aplicaciones de ingeniería. Por último, una gama de herramientas de cuarta generación poseen también características de generación de prototipos.

- ***Herramientas de programación***

  La categoría de herramientas de programación abarca los compiladores, editores y depuradores que están disponibles para prestar su apoyo en la mayoría de los lenguajes de programación convencionales.

  Además, los entornos de programación orientados a objetos (OO), los lenguajes de cuarta generación, los entornos de programación gráfica, los generadores de aplicaciones, y los lenguajes de consulta de bases de datos residen también en esta categoría.

- ***Herramientas de integración y comprobación***

  En su directorio de herramientas de comprobación de software, Software Quality Engineering define las siguientes categorías de herramientas de comprobación:

  - Adquisición de datos
  - Medida estática
  - Medida dinámica
  - Simulación
  - Administración de comprobaciones
  - Herramientas de funcionalidad cruzada

- ***Herramientas de análisis estático***

  Prestan su asistencia al ingeniero del software a efectos de derivar casos prácticos. Se utilizan tres tipos distintos de herramientas estáticas de comprobación en la industria: Herramientas de comprobación basadas en código, lenguajes de comprobación especializados, y herramientas de comprobación basadas en requisitos.

  Las *herramientas de comprobación basadas en código* admiten un código fuente como entrada, y efectúan un cierto número de análisis que dan lugar a la generación de casos de prueba.

  Los *lenguajes de comprobación especializados* capacitan al ingeniero de software para escribir detalladas especificaciones de comprobación que describirán todos los casos de prueba y la logística de su ejecución.

  Las herramientas de comprobación basadas en requisitos aíslan requisitos específicos del usuario y sugieren casos de prueba que ejerciten estos requisitos.

- ***Herramientas de análisis dinámico***

  Las herramientas de análisis dinámico interactúan con un programa que se esté ejecutando, comprueben la cobertura de rutas, comprueban las afirmaciones acerca del valor de variables especificas y en general instrumental el flujo de ejecución del programa.

  Las herramientas dinámicas pueden ser bien intrusivas, bien no intrusivas. Las *herramientas intrusivas* modifican el software que hay que comprobar mediante sondas que se insertan y que efectúan las actividades mencionadas anteriormente.

  Las *herramientas de comprobación no intrusivas* utilizan un procesador que contenga el programa que se está comprobando.

- ***Herramientas de gestión de comprobación***

  Se utilizan para comprobar y coordinar la comprobación de software para cada uno de los pasos principales de comprobación. Las herramientas de esta categoría administran y coordinan la comprobación de regresiones, efectúan comparaciones que determinan las diferencias entre la salida real y la esperada, y efectúan comprobaciones por lotes de programas con interfaces interactivas entre hombre y máquina.

Además de las funciones indicadas anteriormente, muchas herramientas de gestión de comprobaciones sirven también como controladores de comprobación genéricos.

Un controlador de comprobación lee uno o más casos de prueba de algún archivo de pruebas, da formato a los datos de prueba para que se ajusten a las necesidades del software que se está probando, e invoca entonces al software que sea preciso comprobar.

- ***Herramientas de comprobación cliente/servidor***

  El entorno C/S exige unas herramientas de comprobación especializadas que ejerciten la interfaz gráfica de usuario y los requisitos de comunicaciones en red para el cliente y el servidor.

- ***Herramientas de reingeniería***

  La categoría de herramientas de reingeniería se puede subdividir en las funciones siguientes:

  - *Herramientas de ingeniería inversa para producir especificaciones*
  - *Herramientas de restructuración y análisis de código*
  - *Herramientas de re ingeniaría para sistemas en línea*

Muchas de las herramientas anteriores están limitadas a lenguajes de programación específicos y requieren un cierto grado de interacción con el ingeniero del software.

CASE Integrado

Aun cuando se pueden obtener beneficios a partir de herramientas CASE individuales que abarquen las actividades de ingeniería del software por separado, la verdadera potencia de CASE solamente se puede lograr mediante la integración.

Los beneficios del *CASE Integrado (I-CASE)* incluyen una transferencia suave de información (modelos, programas, documentos, datos) entre una herramienta y otra, y entre un paso de ingeniería y el siguiente; una reducción del esfuerzo necesario para efectuar actividades globales tales como la administración de configuración de software, el control de calidad y la producción de documentos; un aumento del control del proyecto que se logra mediante una mejor planificación, monitorización y comunicación; y una mejor coordinación entre los miembros del personal que estén trabajando en grandes proyectos de software.

Ahora bien, I-CASE también presenta desafíos significativos. En cada acción exige unas representaciones consistentes de la información de la ingeniería del software, unas interfaces estandarizadas entre herramientas, un mecanismo homogéneo para la comunicación entre el ingeniero de software y todas sus herramientas, y un enfoque efectivo que capacite a I-CASE para desplazarse a lo largo de distintas plataformas de hardware y distintos sistemas operativos.

Aun cuando las soluciones de los problemas implícitos eb estos desafíos se han propuesto ya, los entornos I-CASE generales sólo empiezan a emerger en la actualidad.

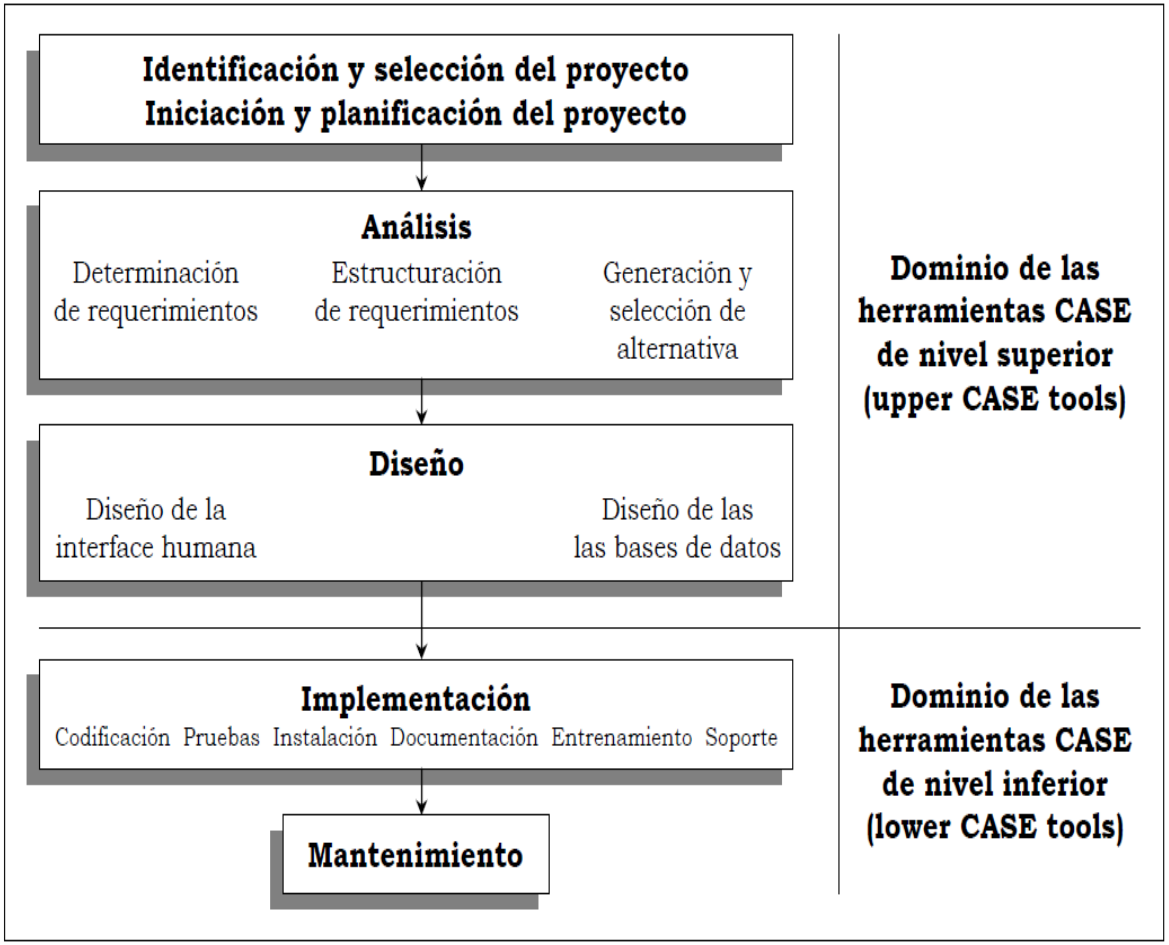
El término <<integración>> implica tanto la <<combinación>> como el <<cierre>>. La I-CASE combina todo una gama de herramientas diferentes y de informaciones distintas de tal modo que hace posible el cierre de la comunicación entre herramientas, entre personas, y entre procesos de software. Se integran las herramientas de tal modo que la información de ingeniería del software esté disponible para todas las herramientas que se necesiten; la utilización se integra de tal modo que se proporciones un aspecto común para todas las herramientas; y se integra una filosofía de desarrollo, implicando un enfoque de ingeniería del software estandarizado que aplique prácticas modernas y métodos ya probados.

Para definir *la integración* en el contexto del procesos del software, es necesario establecer un conjunto de requisitos para I-CASE. Un entorno CASE integrado debería de:

- Proporcionar un mecanismo para compartir la información de ingeniería del software entre todas las herramientas que estén contenidas en el entorno.
- Hacer posible que un cambio de un elemento de información se siga hasta los demás elementos de información relacionados.
- Proporcionar un control de versiones y una gestión de configuración general para toda la información de la ingeniería del software.
- Permitir un acceso directo y no secuencial a cualquiera de las herramientas contenidas en el entorno.
- Establecer un apoyo automatizado para un contexto de procedimientos para el trabajo de la ingeniería del software que integra las herramientas y los datos en una estructura de desglose de trabajo estandarizada.
- Capacitar a los usuarios de cada una de las herramientas para experimentar una utilización consistente en la interfaz hombre-máquina.
- Permitir la comunicación entre ingenieros del software.
- Recoger métricas tanto de gestión como técnicas que se puedan utilizar para mejorar el proceso y el producto.

Para alcanzar estos objetivos, cada uno de los bloques de construcción de una arquitectura CASE debe encajar con los demás sin costuras.

Ciclo de Vida del CASE



CONCLUSIONES

La finalidad de este material es proporcionar a los alumnos los conocimientos y herramientas para el desarrollo, mantenimiento y reingeniería de software con la finalidad de aplicarlos en las empresas privadas donde se desarrollan procesos de información automatizados con herramientas computacionales.

Referencias

- Braude, Eric J. Ingeniería de Software. Una Perspectiva Orientada a Objetos, 1ra edición Alfaomega, 2003.
- Gido, Jack and Clements, James P., Administración Exitosa de Proyectos, Thomson Editores, 1999.
- Humphery S. Watts, Introducción al Proceso Software Personal (PSP), Addison Wesley, 2001.
- Kendall y Kendall, Análisis y Diseño de Sistemas, 3ra de Pretince Hall, 1997.
- Medellin Serna Luis Antonio, Ingeniería de Software II y II, 2005.
- Pages-Jones, Meilir, The practical Guide to Structured System Design, Englewood Clifs, N.J., Yourdon Press 1988.
- Pressman Roger S., Ingeniería de Software. Un enfoque práctico, 5ta. Edición, McGraw Hill, 2004.
- Roger S. Pressman, Ingeniería de Software, Un Enfoque Práctico, Mac Graw Hill, 2004.
- Shari Lawrence Pfleeger Ingeniería de Software, Teoria y Práctica, Prentice Hall, 2002, Argentina.
- Sommerville Ian., Ingeniería de Software, 6ta. Edición. Adisson Wesley, 2002.
- Weitzenfeld Alfredo, Ingeniería de Software. Orientada a Objetos, 1ra Edición, Thomson, 2004.
- Yourdon, Ed, Análisis Estructurado Moderno, Prentice Hall, 1991.