DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SISTEMA DE GESTIÓN DE COMPETENCIAS EN LA FACULTAD DE INGENIERÍA DE SISTEMAS F.I.S. UNAB DESDE LA CONCEPCIÓN DE COMPETENCIA Y EVALUACIÓN DEL SISTEMA DE ASEGURAMIENTO DE LA CALIDAD EN LA EDUCACIÓN SUPERIOR

Juan Francisco Acevedo Maza
CRISTIAN JAVIER GUTIÉRREZ TORRES

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
DIVISION DE CIENCIAS NATURALES E INGENIERÍA
BUCARAMANGA
2005

DISEÑO E IMPLEMENTACIÓN DE UN PROTOTIPO DE UN SISTEMA DE GESTIÓN DE COMPETENCIAS EN LA FACULTAD DE INGENIERÍA DE SISTEMAS F.I.S. UNAB DESDE LA CONCEPCIÓN DE COMPETENCIA Y EVALUACIÓN DEL SISTEMA DE ASEGURAMIENTO DE LA CALIDAD EN LA EDUCACIÓN SUPERIOR

JUAN FRANCISCO ACEVEDO MAZA
CRISTIAN JAVIER GUTIÉRREZ TORRES

Tesis de Grado

Director
JORGE ANDRICK PARRA VALENCIA

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA
FACULTAD DE INGENIERÍA DE SISTEMAS
DIVISION DE CIENCIAS NATURALES E INGENIERÍA
BUCARAMANGA
2005

GLOSARIO

.NET Es una plataforma de desarrollo tecnológico exclusiva de entorno Windows, .NET permite el uso de los diferentes lenguajes incluidos en Visual Studio .Net (Visual Basic.NET, C++.NET y C#)

CIBERNÉTICA ORGANIZACIONAL Es una disciplina que se preocupa por aclarar el enfoque de la *viabilidad* como un objetivo principal de todas las organizaciones que se están desarrollando en un entorno rápido y en continuo cambio

COMPETENCIA Son habilidades y destrezas que adquieren las personas por medio de un proceso de aprendizaje que esta en continua evaluación.

DIAGRAMAS CASOS DE USO Son muy útiles durante todo el proceso de desarrollo del software puesto que ayuda a comprender todas las actividades que se van a desarrollar en las fases de diseño, implementación y pruebas del sistema.

DIAGRAMAS DE CLASES Presenta un conjunto de clases, interfaces y colaboraciones y las relaciones entre ellas. Un diagrama de clases son los diagramas más comunes en el modelado de sistemas orientados a objetos.

DIAGRAMAS DE COLABORACIÓN Es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes.

DIAGRAMAS DE PAQUETES Es un mecanismo de propósito general para organizar elementos en grupos, gráficamente un paquete se representa como una carpeta

DIAGRAMAS DE SECUENCIA Es un diagrama de interacción que resalta la ordenación temporal de los mensajes. Un diagrama de secuencia presenta un conjunto de objetos y los mensajes enviados y recibidos por ellos, en si los diagramas de secuencia se utilizan para describir la vista dinámica del sistema.

ECAES Examen de Estado de Calidad de la Educación Superior.

F.I.S Facultad de Ingeniería de Sistemas.

ICFES Instituto Colombiano para el Fomento de la Educación Superior

INDICADORES DE GESTIÓN Los indicadores de gestión permiten observar a una organización todas las variables fundamentales sobre las que requiere garantizar su comportamiento, visualizando su influencia sobre la organización y los cambios internos y externos.

INDICADORES EXTERNOS Sirven para medir el rendimiento del sistema.

INDICADORES INTERNOS Sirven para medir que tanto la solución propuesta resuelve el problema.

ÍNDICES Ayudan a observar el comportamiento de la organización en el corto, mediano, largo plazo y a visualizar qué sucede bajo ciertas circunstancias si se mejora el comportamiento de los indicadores en el tiempo.

INGENIERÍA DE SOFTWARE Es la disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados

J2EE Esta plataforma desarrollo tecnológico es un estándar para el desarrollo de aplicaciones empresariales multicapa, en base al cual se pueden adquirir productos que proporcionan servicios añadidos, desarrollados por diversos proveedores.

MSV Es una de las herramientas principales que proporciona a la Cibernética la capacidad de mejorar la organización en la administración moderna, permitiendo que esta sea analizada en forma dinámica y sin rigidez.

PEI Proyecto Educativo Institucional

RUP Es un proceso de ingeniería de software; proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo.

SISTEMA Es una combinación de medios como personas, materiales, equipos, software, instalaciones, datos, etc.; integrados de tal forma que puedan desarrollar una determinada función en respuesta a una necesidad concreta.

SOFTWARE Son programas de computadora, estructuras de datos y su documentación que sirven para hacer efectivo el método lógico, procedimiento o control requerido.

STATUS QUO Estado actual

UML Es un lenguaje de entorno gráfico el cual se utiliza para observar, especificar, construir y documentar todas las partes que componen el desarrollo de software.

RESUMEN

El diseño de este proyecto tratará dicha problemática desde la concepción de la gestión de competencias, en un proceso cibernético organizacional, con el fin de poder llevar un seguimiento detallado del rendimiento de los alumnos, en base a las competencias establecidas para la carrera por el Sistema Nacional de Aseguramiento de la Calidad en la Educación Superior, logrando de esta forma dar posibles soluciones o planteando estrategias a seguir como medidas correctivas para el fortalecimiento académico de los estudiantes, según como sean las debilidades que se muestren en los resultados de los simulacros de exámenes tipo ECAES que realizaran los estudiantes de la facultad de ingeniería de sistemas, así podría conocerse cómo es que el proyecto educativo de la institución opera sobre el desarrollo de las competencias de los estudiantes, permitiendo definir y hacer seguimiento de políticas de mejoramiento curricular desde la concepción de dicho Sistema Nacional de Aseguramiento de la Calidad en la Educación Superior.

El proyecto “Diseño e Implementación de un Prototipo de un Sistema de Gestión de Competencias en la Facultad de Ingeniería de Sistemas F.I.S. UNAB desde La Concepción de Competencia y Evaluación del Sistema Nacional de Aseguramiento de la Calidad en la Educación Superior.” beneficiara directamente a la Facultad de Ingeniería de Sistemas, ya que asiste la gestión de competencias, lo que permitirá hacer seguimiento no solo del desarrollo de las competencias en los estudiantes, sino también la evaluación de las competencias definidas y de las estrategias propuestas para el logro de dichas competencias a través de indicadores de gestión.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

LISTA DE ANEXOS

INTRODUCCIÓN

Es problemático para la administración curricular de la Facultad de Ingeniería de Sistemas tanto el diagnosticar el desarrollo de competencias en los estudiantes, como la evaluación de la eficiencia de las políticas concebidas para el diagnostico y mejoramiento de dichas competencias, desde la perspectiva de la evaluación de competencias del Sistemas Nacional de Aseguramiento de la Calidad en la Educación. De esta forma, sería posible articular la interpretación de los resultados de las pruebas de estado con el desarrollo de los estudiantes por áreas de conocimiento según la definición del Examen de Estado de Calidad de la Educación Superior ECAES, así podría conocerse cómo es que el proyecto educativo de la institución opera sobre el desarrollo de las competencias de los estudiantes, permitiendo definir y hacer seguimiento de políticas de mejoramiento curricular desde la concepción de dicho Sistema Nacional de Aseguramiento de la Calidad en la Educación.

El Ingeniero de Sistemas como primera medida debe entender la situación actual del problema, después de haber entendido perfectamente el problema el Ingeniero de Sistemas debe saber que tipo de software se va a desarrollar y además debe conocer las metodologías de software para escoger la más viable o la que mejor se adapte al problema. En este proyecto el tipo de software a desarrollar es un software de gestión, ya que contiene indicadores de evaluación y por tal motivo esta estrechamente relacionado con la Cibernética Organizacional; es por eso que en este documento se tratan temas que a simple vista no tienen relación pero para el desarrollo ingenieril es fundamental hablar de estos temas.

1. ANTECEDENTES

El Sistema de Aseguramiento de la Calidad de la Educación Superior tiene un objetivo principal, que es "desplegar en la sociedad un proceso de ilustración sobre el sentido, el valor y los niveles de calidad de la educación superior"[1]. El Sistema de Aseguramiento de la Calidad de la Educación Superior debe mostrar a la sociedad modelos y niveles de mejoramiento obtenidos en la universidad, haciendo entender que la misión de los universitarios no es únicamente responder a las demandas sociales, también es transformar y orientar tales demandas. Teniendo en cuenta los conceptos de "autonomía, rendición de cuentas, garantía social ante los estándares básicos, y acreditación de alta calidad"[2], se podría relacionar que los conceptos y procesos que estructuran un modelo de aseguramiento de la calidad de la educación superior en Colombia pueden ser de esta manera:

Figura 1. Modelo de aseguramiento de la calidad en Colombia

ROA VALERO, Alberto. Hacia un modelo de aseguramiento de la Calidad en la Educación Superior en Colombia. Página 12.

[1] ROA VALERO, Alberto. Hacia un modelo de aseguramiento de la Calidad en la Educación Superior en Colombia: estándares básicos y acreditación de excelencia, Miembro del Concejo Nacional de Acreditación. 2003, Pág. 11.
[2] Lbid., Pág. 12.

Por este motivo el gobierno Colombiano ha diseñado un Examen de Estado de Calidad de la Educación Superior ECAES para evaluar a los futuros profesionales del país y así mejorar el rendimiento y ser más competitivos.

En la educación superior y como mecanismo de aseguramiento de la calidad a este nivel educativo el Plan Nacional de Desarrollo abarca la aplicación obligatoria de los Exámenes de la Calidad de la Educación Superior ECAES a todos los programas que se den en el país, "Con la expedición del decreto 1413 de 2001 por el cual se modifica la estructura del Ministerio de Educación Nacional, se crea la Dirección de Educación Superior, asignándole funciones, que de alguna manera ha venido desempeñando el ICFES, específicamente en el campo de Monitoreo y Vigilancia de la Educación Superior"[1], este proceso de chequeo y vigilancia de la educación superior esta a cargo del Ministro de Educación que es nombrado por el presidente de la República y el ICFES. El programa de mejoramiento de la calidad superior no es tan solo obtener excelentes resultados sino que también se ofrezcan buenos programas de educación, por este motivo se han creado diversos mecanismos de control como por ejemplo la creación de un observatorio laboral y el fortalecimiento del Sistema Nacional de Información de la Educación Superior; así nace un proyecto de la necesidad de conocer como se encuentran los estudiantes de la Facultad de Ingeniería de Sistemas en la UNAB con respecto al desarrollo de las competencias adquiridas durante su educación superior luego de su proceso de formación definido por el PEI (Proyecto Educativo Institucional), además este sistema desea asistir aquellas falencias que tienen los estudiantes en ciertas áreas del conocimiento para que los resultados en el ECAES sean óptimos. Esta investigación es una iniciativa de la Facultad de Ingeniería de Sistemas con miras a extenderse a toda la Universidad Autónoma de Bucaramanga.

[1] Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Transformación Del ICFES.2003, Pág.8.

2. ESTADO DEL ARTE

Se encontraron dos tipos de sistemas de gestión de competencias las cuales son:

El 28 de agosto de 2003 se creó un prototipo de simulador en Internet para los exámenes de estado ICFES[1], este simulador ha sido utilizado aproximadamente 250.000 veces e incluso ha sido utilizado en el exterior; este programa arroja resultados inmediatos para que puedan ser analizados por los propios estudiantes, a fin de poder realizar los respectivos correctivos y tener un resultado positivo en el momento de realizar la prueba de estado, además cuenta con un cronómetro para cada cuestionario, lo que permite medir el tiempo que se demora cada estudiante en contestar las preguntas que se formulan. Este simulador cuenta actualmente con banco de 1.500 preguntas con sus respectivas respuestas que sirven para realizar una prueba muy similar a la que se presenta formalmente.

También la Caja de Compensación Familiar de Antioquia COMFAMA[2] en convenio con Corpoeducación (Corporación para el desarrollo de la educación básica) realizó un proyecto empresarial de competencias con el ánimo de mejorar el desempeño de los trabajadores. El software consta de tres partes, una prueba empresarial de competencia: matemática, lectura y escritura que se diseño con el fin de medir los niveles de los empleados de base en las competencias de matemáticas, lectura y escritura y pueden ser aplicadas para selección de personal ó para identificar necesidades de formación de los trabajadores; consta también de un curso de formación en competencias laborales generales con el fin de que los trabajadores puedan mejorar sus habilidades como empleados basados en los principios de la formación por competencias; y por último consta de una evaluación de competencias laborales generales que responde a las

[1] MINISTERIO DE EDUCACIÓN, Ecaes, www.presidencia.gov.co/cne/2003/agosto/28/1528 2003.htm
[2] COMFAMA (Caja de Compensación Familiar de Antioquia)
www.comfama.com.co/contenidos/servicios/Educaci%C3%B3n/Competencias%20laborales/competenciaslaborales.asp

necesidades detectadas en las organizaciones para poder evaluar el desempeño del trabajador siguiendo un enfoque de competencias.

3. MARCO TEÓRICO

3.1 SISTEMA NACIONAL PARA LA EDUCACIÓN SUPERIOR EN COLOMBIA

La decisión de realizar exámenes a los estudiantes universitarios fue una propuesta del Plan Nacional para la Educación Superior en Colombia que data del año 1966; en 1981 de acuerdo con la reforma hecha a la educación superior, se hace la propuesta de estos exámenes para complementar y evaluar la calidad de los programas académicos de pregrado, para esta propuesta se realizaron múltiples debates hasta el año de 1989. En el año de 1990, en los planes de gobierno se comenzó a incluir la evaluación de los programas académicos como política del sector educativo, estos exámenes son encaminados a mejorar la calidad y nitidez de la educación superior. En el año de 1999 el ICFES junto con la Asociación Colombiana de Ingenieros ACIEM inició la elaboración de una prueba para ingenieros mecánicos la cual fue aplicada por primera vez a un grupo de estudiantes en febrero del año 2000. Un año después se realizó la primera evaluación oficial a estudiantes de pregrado en los programas de ingeniería mecánica, "reglamentado por el decreto 2233 del 23 de octubre de 2001 e igualmente en medicina, reglamentado por el decreto 1716 del 24 de agosto de 2001"[1]; estas pruebas fueron apoyadas por ACOFI (Asociación Colombiana de Facultades de Ingeniería) y ASCOFAME (Asociación Colombiana de Facultades de Medicina).

En el 2003 se convoco a las instituciones públicas y privadas de educación superior a nivel nacional, a agrupaciones de profesionales y a asociaciones de facultades con el fin de presentar proyectos orientados al diseño y a la elaboración

[1] Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Informe Nacional de Resultados.2003, Pág.6.

de Exámenes de Estado de Calidad de la Educación Superior ECAES, en solicitud a las metas propuestas por el gobierno para el año 2003.

Se elaboraron las pruebas para 26 carreras de acuerdo con los términos de la convocatoria, para poder realizar estas pruebas se basaron en talleres a nivel regional y nacional, donde se recogieron las preguntas y el marco conceptual que aportaron los docentes de cada carrera.

En junio de 2003 se reglamentan los Exámenes de Calidad de la Educación Superior ECAES por medio del decreto 1781 en el cual fue expedido por el Gobierno Nacional. A partir del 1 de noviembre de 2003 ECAES se empezó a aplicar a 27 distintos programas de pregrado, a 58974 estudiantes de últimos semestres de carrera y egresados en 41 ciudades del país.

El proyecto *diseño e implementación de un prototipo de sistema de gestión de competencias en la F.I.S. UNAB desde la concepción de competencia y evaluación del sistema de aseguramiento de la calidad en la educación superior* nace por la necesidad de conocer la evolución de los estudiantes de ingeniería de sistemas con respecto a las competencias de las áreas de conocimiento del Examen de Estado de Calidad de la Educación Superior ECAES luego de pasar el proceso de formación orientado por el PEI.

Según el "decreto 1781 de 2003 los ECAES deberán ser presentados por los estudiantes que se encuentren cursando el último año de los programas académicos de pregrado y también pueden ser presentados por los egresados que deseen autoevaluarse"[1].

[1] Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Informe Nacional de Resultados.2003, Pág.7.

Los programas que han sido evaluados son los siguientes: Arquitectura, Derecho, Enfermería, Fisioterapia, Fonoaudiología, Ing. Agrícola, Ing. Agronómica y Agronomía, Ing. Ambiental, Ing. Civil, Ing. de Alimentos, Ing. de Materiales, Ing. de Minas, Ing. de Sistemas, Ing. de Telecomunicaciones, Ing. Eléctrica, Ing. Electrónica, Ing. Geológica, Ing. Industrial, Ing. Mecánica, Ing. Metalúrgica, Ing. Química, Medicina, Nutrición y Dietética, Ontología, Optometría, Psicología, y Terapia Ocupacional.

Uno de los objetivos más importantes de los Exámenes de Calidad de la Educación Superior, ECAES, es "comprobar el grado de desarrollo de competencias de los estudiantes que cursan el último año de los programas académicos de pregrado que ofrecen las instituciones de educación superior"[1]; las áreas a evaluar fueron definidas por las entidades que estuvieron a cargo de la elaboración de las pruebas según cada programa, teniendo en cuenta los requerimientos mínimos que deben conocer los estudiantes en su proceso de formación, además de las competencias requeridas para poder ejercer profesionalmente.

Los resultados conseguidos en este examen tienen como intención principal brindar una valiosa información referente al desempeño de los estudiantes evaluados, tanto en el examen completo, como en los diferentes grupos de preguntas y las áreas del conocimiento en las cuáles esta dividida el examen para cada carrera. Se genera un informe de resultados individual que contiene el puntaje general y los resultados por grupos de preguntas. El informe arroja datos institucionales y nacionales que corresponden al programa o carrera del estudiante evaluado ó egresado, también se genera un informe de resultados institucional que se genera por cada carrera ó programa, basándose en las jornadas y semestres, y se realiza teniendo en cuenta los resultados obtenidos por

[1] Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Informe Nacional de Resultados.2003, Pág.8.

los estudiantes en cada institución; el informe institucional está formado por cinco reportes distintos, en donde cada reporte presenta los datos nacionales como guía el cual servirá para caso de estudio. Los dos tipos de resultados tienen datos cualitativos, cuantitativos y resultados nacionales que servirán de referente para una óptima interpretación.

"En general son tres tipos de resultados que se procesan para los ECAES"[1]:

- **El Puntaje General.** es el resultado que se obtiene a partir de la valoración de las respuestas dadas a todas las preguntas del examen, se expresa en una escala de 0 a 100, aproximadamente, siendo 0 el punto más bajo de la escala. Para las instituciones se reporta el promedio y la desviación estándar de los puntajes generales de los estudiantes.

- **Resultados por Grupos de Preguntas.** Se obtienen a partir de la valoración de las respuestas dadas a las preguntas de una misma área, de un tópico o de una problemática en particular dentro del examen. Este resultado se presenta de manera cuantitativa en escala de 0 a 100 y cualitativa en desempeño de categorías alto, medio, bajo. A las instituciones se les reporta la distribución de los resultados por grupos de preguntas de sus estudiantes en cada categoría de desempeño.

- **Porcentaje de Respuestas por Opción.** Es un resultado reportado sólo a las instituciones y que describe cómo se distribuyen las respuestas de los estudiantes frente a cada una de las opciones de respuesta que tienen las preguntas del examen. Para cada pregunta se indica la posición ocupada dentro del cuadernillo de prueba y la clave, es decir, la opción considerada como la respuesta correcta.

[1] Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Informe Nacional de Resultados.2003, Pág.9.

Se cuantifican también los casos de “multimarca” (marcar más de una opción de respuesta en la misma pregunta) y de “omisión” (no responder a la pregunta).

3.2 COMPETENCIAS DE INGENIERÍA DE SISTEMAS

ACOFI en convenio con el ICFES diseño el examen ECAES para Ingeniería de Sistemas teniendo en cuenta ciertas competencias básicas.

“Estas competencias son”[1]:

- Tener la habilidad de **recordar** información, datos y terminología, y evocar principios físicos.

- Tener la habilidad de **comprender** al traducir ideas con expresiones diferentes, interpretando información para darle una nueva configuración, extrapolando información para concluir e identificar consecuencias de sus decisiones.

- Tener la habilidad de **aplicar** principios físicos y naturales para resolver problemas, especificar fronteras de las soluciones para pasar de lo general a lo particular, explicar nuevos fenómenos a partir de principios conocidos, predice situaciones y justifica soluciones.

- Tener la habilidad de **analizar** al identificar los elementos que componen un sistema y explicita las relaciones entre si, identifica principios de organización de una situación dada.

- Tener la habilidad de **sintetizar**, produciendo nueva información, diseñando soluciones creativas a los problemas de Ingeniería, y formula juicios evaluando y

[1] ACOFI. Exámenes De La Calidad De La Educación Superior ECAES.2003.Pág.6.

proponiendo nuevos enfoques y soluciones técnicas a problemas que aquejan la sociedad bajo condiciones restringidas, con base en los criterios que adquiere en su vida como estudiante y más adelante en su vida como profesional de la Ingeniería.

El Examen de Estado de Calidad de la Educación Superior para Ingeniería de Sistemas ECAES/IS tiene como fin fundamental el evaluar a los estudiantes por áreas y sub-áreas de conocimiento básico que se deben haber impartido durante el estudio de la carrera para la formación de Ingenieros de Sistemas / Informáticos. Este examen tiene como objetivo evaluar lo que se considera básico en la formación de un Ingeniero de Sistemas, para que pueda ejercer exitosamente profesionalmente, sin tener en cuenta temas muy especializados y particulares que se den en las diferentes facultades de Ingeniería de Sistemas en el país; "La estructura del examen es de la siguiente forma:"[1]

- Áreas de evaluación

- Matemáticas: 16 preguntas.
- Física: 10 preguntas.
- Humanidades: 10 preguntas.
- Económico / Administrativa: 5 preguntas.
- Ciencias básicas de Ingeniería: 9 preguntas.
- Matemáticas Discretas: 10 preguntas.
- Programación y Algorítmica: 13 preguntas.
- Informática Teórica: 10 preguntas.
- Arquitectura y Funcionamiento del Computador: 8 preguntas.
- Redes y Comunicaciones: 5 preguntas.
- Administración de Información: 6 preguntas.

[1] Convenio ACOFI – ICFES. Especificaciones De Los Exámenes De Estado De Calidad De La Educación Superior En Ingeniería De Sistemas / Informática.2003, Pág.7.

- Sistemas y Organizaciones: 6 preguntas.
- Ingeniería de Software: 12 preguntas.

- Subáreas de evaluación.

Matemáticas

- Álgebra.
- Trigonometría.
- Geometría Analítica.
- Álgebra Lineal.
- Calculo Integral.

Física

- Mecánica y Ondas.
- Electricidad y Magnetismo.

Humanidades

- Comunicación.
- Cultura General.
- Ingles.
- Cívica y Democracia.

Económico / Administrativa

- Fundamentos de Economía.
- Análisis Financiero.

Ciencias Básicas de Ingeniería

- Análisis Numérico.
- Probabilidad y Estadística.
- Investigación de Operaciones

Matemáticas Discretas

- Funciones, Relaciones y Conjuntos.
- Lógica.
- Conteo.
- Grafos.
- Ecuaciones de Diferencia.

Programación y Algorítmica

- Estructuras de Datos.
- Algorítmica.
- Algoritmos Clásicos.
- Verificación de Programas.

Informática Teórica

- Autómatas.
- Lenguajes Formales.
- Paradigma Orientado a Objetos.

Arquitectura y funcionamiento del Computador

- Circuitos Lógicos.
- Representación de Datos.
- Arquitectura de Hardware Básica.
- Sistemas Operativos.

Comunicaciones y Redes

- Redes.
- Web.

Administración de Información

- Bases de Datos.
- Modelado.

Sistemas y Organizaciones

- Conceptos de Sistemas de Información.
- Organizaciones.
- Sistemas Socio-Técnicos.
- Fundamentos de Administración.
- Metodologías de Intervención.

Ingeniería de Software

- Diseño de Software.
- Procesos Básicos de Software.
- Especificación de Software.
- Validación de Software.

- Administración de Proyectos de Software.

3.3 SISTEMA

"Un sistema es una combinación de medios como personas, materiales, equipos, software, instalaciones, datos, etc.; integrados de tal forma que puedan desarrollar una determinada función en respuesta a una necesidad concreta"[1], Según Pressman un sistema es "Un conjunto o disposición de elementos que están organizados para realizar un objetivo predefinido procesando información"[2].

Los sistemas están catalogados en sistemas físicos o conceptuales, sistemas naturales o artificiales, sistemas estáticos o dinámicos y sistemas abiertos o de lazos cerrados. Los sistemas pueden cambiar dependiendo de la forma en que se adecuen ó funcionen, y podríamos tratar como ejemplo una red de comunicaciones capaz de distribuir información a nivel mundial, el cuerpo humano, entre otros. Un sistema en general está formado por elementos, y éstos se pueden descomponer en componentes más pequeños; estos sistemas más pequeños comúnmente se les denominan como subsistemas. Si alguno de los subsistemas llegara a fallar podría ocasionar un mal funcionamiento de todo el sistema o en el peor de los casos que no llegara a funcionar.

"Es esencial definir el sistema considerado especificando claramente sus límites y fronteras"[3].

3.4 INGENIERÍA DE SISTEMAS

La ingeniería de sistemas integra las actividades y los medios apropiados de forma oportuna y eficaz durante un proceso evolutivo que comienza en la

[1] BLANCHARD, Benjamín S. Ingeniería de Sistemas. Isdefe - c/ Edison, 1996. Pág.12.
[2] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág.166.
[3] Lbid., Pág. 13.

identificación de los requisitos ó necesidades de los usuarios y finaliza con la entrega del sistema, este proceso es repetitivo en cuanto a la definición de los requisitos, análisis, optimización, diseño, prueba y evaluación.

Durante la evolución del proceso de ingeniería de sistemas, que inicia con los detalles funcionales y los requisitos del diseño, tiene como fin obtener el equilibrio perfecto entre los factores económicos y logísticos. La mejor manera de llegar a este objetivo es realizando un esfuerzo en conjunto que esté dirigido al diseño.

Los sistemas actuales cada vez son más complejos, a medida que aparecen nuevas tecnologías en el mundo; actualmente el tiempo que se gasta en el desarrollo de un nuevo sistema operativo es más largo porque las necesidades cada vez son más complicadas, haciendo que se aumenten los costos relacionados con el desarrollo, producción, utilización y apoyo de los sistemas. Al mismo tiempo, los recursos se van diminuyendo y la competencia va aumentando en todo el mundo.

"***La ingeniería de sistemas*** puede definirse como la aplicación de técnicas científicas y de ingeniería para"[1]:

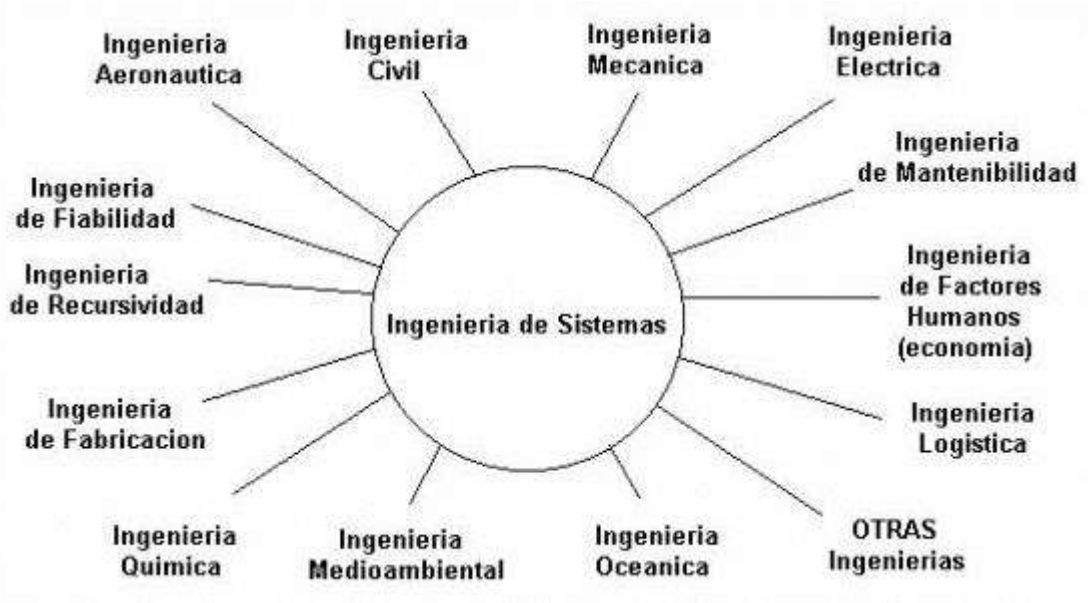
1. Transformar una necesidad operativa en la descripción de los parámetros de prestaciones de un sistema y en su configuración mediante la utilización de un proceso iterativo de definición, síntesis, análisis, diseño, prueba y evaluación.

2. Integrar los parámetros técnicos relacionados y asegurar la compatibilidad de todas las interrelaciones físicas, funcionales y del programa de forma que se consiga la mejor definición y diseño del sistema completo.

[1] BLANCHARD, Benjamín S. Ingeniería de Sistemas. Isdefe - c/ Edison, 1996.Pág.19.

3. Integrar los aspectos de fiabilidad, mantenimiento, seguridad, supervivencia, de personal y otros similares en el proceso global de ingeniería para conseguir los objetivos técnicos, de costos y de calendario fijados

Figura 2. Integración de Disciplinas de Ingeniería



BLANCHARD, Benjamín S. Ingeniería de Sistemas. Isdefe - c/ Edison, 1996.Pág.20.

3.4.1 Características de la Ingeniería de Sistemas

Actualmente la ingeniería de sistemas no es considerada como una ingeniería tradicional, como lo es la ingeniería eléctrica, la ingeniería mecánica, la ingeniería industrial, la ingeniería civil, o cualquier otra orientada al diseño. Su ejecución no requiere de muchos recursos económicos. Básicamente, la ingeniería de sistemas al ser aplicada es más bien la realización de un proceso intelectual. Para algunas personas la ingeniería de sistemas requiere de un gran cambio de mentalidad o más bien un cambio de cultura. Cabe señalar que la Ingeniería de Sistemas se caracteriza por:

- La necesidad de utilizar un enfoque de arriba hacia abajo, observando al sistema como un todo. En el pasado los trabajos de ingeniería tenían muy buenos diseños de los diferentes componentes de un sistema mostrando simplemente una trayectoria de abajo-arriba, carecían de la visión global y de la comprensión del sistema para poder integrarse eficazmente todos los componentes entre sí.

Figura 3. Elementos de un Sistema

BLANCHARD, Benjamín S. Ingeniería de Sistemas. Isdefe - c/ Edison, 1996.Pág.23

- Es preciso examinar todo el ciclo de vida del sistema, teniendo en cuenta sus fases, que incluyen diseño, desarrollo, la producción, su distribución, apoyo y mantenimiento durante su vida operativa, su baja y retirada.

- Se requiere de un mejor y más completo esfuerzo en cuanto a la definición de los requisitos del sistema.

- Se necesita realizar un esfuerzo de trabajo en equipo, durante el proceso de diseño y desarrollo de un sistema, consiguiendo todos los objetivos del diseño de una manera óptima. Para lograr este alcance es necesario conocer las diferentes disciplinas de diseño en su totalidad además de sus relaciones mutuas, así como los métodos y técnicas o herramientas que pueden aplicarse para facilitar el desarrollo del proceso de la ingeniería de sistemas de forma efectiva.

3.5 EL SOFTWARE

"El software son programas de computadora, estructuras de datos y su documentación que sirven para hacer efectivo el método lógico, procedimiento o control requerido"[1].

En nuestra sociedad actualmente el software es algo básico para generar servicios y suplir necesidades pero no se puede ignorar que sus costos de desarrollo y de adquisición son cada vez más fuertes si se compara con el hardware sobre el que son ejecutados, Al observar detenidamente nuestro entorno se ve cómo los sistemas de software son cada vez más importantes, ya que son los responsables de que sistemas que se basan en ellos y empresas que los construyen o utilizan, tengan éxito o fracaso.

"El software se caracteriza y diferencia del hardware por"[2]:

- **El software se desarrolla, no se fabrica en un sentido clásico.** En el software, la calidad se obtiene por medio de un buen diseño al igual que sucede con el hardware, pero el software en la fase de construcción no tiene problemas de calidad o si los tiene son muy fáciles de corregir, mientras que en el hardware no. Estas dos actividades tienen un enfoque muy diferente y no se puede

[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág.166.
[2] Lbid., Pág.5.

gestionar un proyecto de software como un proyecto de fabricación ya que los costes del software se encuentran en la ingeniería.

- **El software no se estropea.** El software falla durante las primeras etapas de su vida por defectos no detectados, pero a medida que pasa el tiempo estos defectos son corregidos hasta que el software queda libre de errores, logrando que el promedio de fallos se reduzca llegando a un funcionamiento ideal; pero el software durante su vida sufre muchos cambios como por ejemplo cuando se somete a mantenimiento, en donde se pueden insertar nuevos errores haciendo que el software se vaya deteriorando.

- **Aunque la industria tiende a ensamblar componentes, la mayoría del software se construye a la medida.** El software actualmente esta comenzado a aplicar la reutilización de componentes, esta reutilización ayuda a agilizar el proceso de desarrollo, ya que el ingeniero dispondrá de tiempo para concentrarse en un diseño más innovador y crear las nuevas partes a partir de estos componentes que ya están desarrollados.

"El software puede aplicarse en cualquier situación en la que se haya definido previamente un conjunto específico de pasos procedimentales, es decir un algoritmo"[1]. Las aplicaciones potenciales del software están divididas en las siguientes áreas:

- **Software de sistemas.** Este software consiste en programas que le sirven a otros programas como por ejemplo los compiladores.

- **Software de tiempo real.** Este software esta conformado por varios componentes, estos son: un componente de adquisición de datos, un componente

[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002, Pág.6.

de análisis, un componente de control/salida y un componente de monitorización; un ejemplo de software de tiempo real son los que usan los cajeros automáticos.

- **Software de gestión.** Este software hace parte del área mas grande de aplicaciones que tiene el software porque accede a múltiples bases de datos con el fin de facilitar operaciones tales como la toma de decisiones, además realiza cálculos interactivos.

- **Software de ingeniería y científico.** Este software se caracteriza por el manejo complejo de números, con múltiples aplicaciones como la astrología y la vulcanología, entre otros.

- **Software empotrado.** Este software se encuentra en memoria de solo lectura, es muy sencillo ya que ejecuta funciones muy limitadas que se encuentran dentro de los productos inteligentes.

- **Software de computadoras personales.** Este software se caracteriza porque tener cientos de aplicaciones de uso personal como por ejemplo procesadores de texto, hojas de cálculo, etc.

- **Software basado en web.** Este tipo de software se basa en instrucciones ejecutables y datos en la Web. Este software da recursos casi ilimitados para que los usuarios puedan acceder desde cualquier sitio teniendo un modem.

- **Software de inteligencia artificial.** Este tipo de software no utiliza algoritmos numéricos, es decir el cálculo no sirve para solucionar estos problemas complejos, algunos ejemplos de este tipo de software IA son: sistemas de reconocimiento de patrones, juegos, etc.

Los diseñadores actualmente están destinados a correr con algunos problemas que se remontan al proceso de desarrollo con una visión limitada hacia los

programas necesarios y se deja de lado el pensamiento sistémico del desarrollo del mismo y del contexto social, humano y técnico que abarca la ejecución. “La ingeniería de sistemas de software es, ante todo, una ingeniería”[1] Para lograr entender un sistema de software es necesario aplicar las diferentes perspectivas que complementan en su totalidad el sistema y que permiten comprender su funcionamiento y la tecnología que se necesita para desarrollarlo, Estas perspectivas son:

- **Infraestructura de ejecución.** Los sistemas de software en general requieren de un procesador que interprete y ejecute todas sus instrucciones y el apoyo necesario para otros programas que le ofrezcan unos servicios útiles como los ofrecidos por el sistema operativo.

- **Proceso de desarrollo.** Los sistemas de software no se obtienen y se desarrollan todos de la misma manera. El desarrollo de un sistema, desde que se tienen las primeras ideas sobre lo que este debería hacer o solucionar, hasta su entrega al usuario final, se somete a una serie de fases; Estas fases y todas las que continúan desde la entrega del producto hasta que este es retirado del mercado es denominado ciclo de vida.

- **Evolución.** Cuando un sistema de software es entregado, no se termina el trabajo sobre el, es más, comienza un largo proceso de mantenimiento y adaptación a posibles necesidades, reparaciones o complementos para mejorar la funcionalidad requerida, así como también realizar verificaciones sobre el sistema para saber si sigue siendo útil y suple todas las necesidades requeridas por los usuarios.

- **Dominio de la aplicación.** La mayoría de los sistemas de software tienen una problemática igual a la de otros sistemas con características similares o que

[1] LEÓN SERRANO, Gonzalo. Ingeniería de Sistemas de Software. Isdefe - c/ Edison, 1998.Pág.16.

pertenecen al un mismo tipo de aplicación, es decir características que obligan a utilizar determinadas técnicas durante el proceso de construcción dependiendo de el tipo de sistema. La tecnología empleada por el software no se puede observar superficialmente después de que el sistema se ha finalizado, aunque la calidad del sistema final depende de la tecnología empleada durante el proceso de desarrollo.

3.6 INGENIERÍA DE SOFTWARE

Según ***Fairley*** "La Ingeniería de Software es la disciplina tecnológica y de administración que se ocupa de la producción y evolución sistemática de productos de software que son desarrollados y modificados dentro de los tiempos y costos estimados"[1].

Según ***Ghezzi*** "la Ingeniería de Software es el campo de la ciencia de la computación que trata con la construcción de sistemas de software que son tan grandes o complejos que son construidos por un equipo o equipos de ingenieros"[2].

3.7 METODOLOGÍAS DE INGENIERÍA DE SOFTWARE

3.7.1 Metodología de *Booch*

Booch fue el creador del diseño orientado a objetos, el cual continúa siendo muy importante en el desarrollo del método. *Booch* modeló el diseño orientado a objetos basándose en "un enfoque lógico, en el que se define las clases, los objetos y sus relaciones, y un enfoque físico en la que se define una arquitectura de módulos y procesos".[3] Las técnicas de diagramación proporcionadas por Booch para documentar el enfoque físico son dos:

[1] PINCIROLI, Fernando. Algunas consideraciones sobre "Procesos de Desarrollo de Software. Conferencia UNAB, 2004. Diap 7.
[2] Lbid., Diap 8.
[3] UNAM, Metodologías de Ingeniería de Software, Departamento de Auditoria Informática, Subdirección de Sistemas DCAA, 1997. Pág.29.

- **Diagramas de módulo.** Estos diagramas se utilizan para mostrar como se asignan las clases y los objetos en los módulos, como por ejemplo los programas, paquetes y actividades del diseño físico. El término módulo en el método de Booch se utiliza cuando se va a describir algún componente del diseño.

- **Diagramas de procesos.** Los diagramas de procesos son los que muestran como se asignan los módulos para los procesadores de hardware.

3.7.2 *Hood* (*Hierarchical Object-Oriented-Design*)

El diseño jerárquico orientado a objetos (*HOOD*) pertenece a un grupo de métodos también orientados a objetos los cuales "tratan de integrar la orientación a objetos con los métodos de diseño estructurado"[1]. El método *HOOD* tiene como diferencia principal con los métodos estructurados que sus componentes del software toman sus características basándose en cosas del mundo real, en vez de las funciones que el sistema tiene que realizar.

La metodología *HOOD* no tiene un método de análisis orientado a objetos. El modelo lógico normalmente es construido utilizando el análisis estructurado; Cuando se transforma del modelo lógico al modelo físico se torna complicado y se hace difícil construir un diseño que concuerde con el análisis.

3.7.3 Metodología de *Coad* y *Yourdon*

Coad y *Yourdon* han dado una dirección completa al análisis y diseño orientado a objetos. "Un diseño orientado a objetos es construido a partir de cuatro componentes"[2]:

- Componente del ámbito del problema.
- Componente de interacción humana.

[1] Lbid., Pág.29.
[2] UNAM, Metodologías de Ingeniería de Software, Departamento de Auditoria Informática, Subdirección de Sistemas DCAA, 1997., Pág.30.

- Componente de administración de tareas.
- Componente de administrador de datos.

Cada uno de estos componentes está compuesto por clases y objetos. Los cuatro componentes juntos forman el modelo físico del sistema. Cabe señalar que todos los diseños de *Coad* y *Yourdon* orientado a objetos tienen la misma estructura en el nivel más alto. Una debilidad del método de *Coad* y *Yourdon* consiste en que tiene una notación compleja, la cual es difícil para utilizarla sin tener herramientas de soporte.

3.7.4 *OMT* (*Object Modelling Technique*)

Rumbaugh creó La técnica de modelación de objetos (*Object Modeling Technique OMT*) la cual contiene dos actividades de diseño:

- Diseño del sistema.
- Diseño del objeto.

El diseño del sistema debe ejecutarse durante la fase de AD (Análisis y Diseño), mientras que el diseño del objeto deberá ejecutarse durante la fase de DD (Diseño y Desarrollo).

"Los pasos convencionales del diseño del sistema son"[1]:

- Organizar el sistema en subsistemas y ordenarlos en capas y divisiones.
- Identificar la concurrencia inherente en el problema.
- Asignar subsistemas a los procesadores.
- Definir la estrategia de implementación del administrador de datos.

[1] UNAM, Metodologías de Ingeniería de Software, Departamento de Auditoria Informática, Subdirección de Sistemas DCAA, 1997. Pág.31.

- Identificar las fuentes globales y definir el mecanismo para controlar el acceso a ellos.
- Elegir un enfoque para la implementación de control del software.
- Considerar las condiciones de los límites.
- Establecer cambios fuera de las prioridades.

El enfoque *OMT* para el diseño de sistemas contiene varias ideas de diseño que son aplicadas de manera genérica.

3.7.5 Metodología de *Shlaer* y *Mellor*

Shlaer y *Mellor* ofrecen un Lenguaje de Diseño Orientado a Objetos (OODLE), el cual es derivado de la notación del modelo de *Booch* y del modelo de *Buhr*. En este modelo existen cuatro tipos de diagramas:

- Diagrama de Clases.
- Gráfica de la estructura de Clase.
- Diagrama de Dependencias.
- Diagrama de Herencia.

"Existe un diagrama de clase para cada clase, el diagrama de clase define las operaciones y atributos de la clase. La gráfica de la estructura de clase define la estructura del módulo de la clase, el control y el flujo de datos entre los módulos de las clases; existe una gráfica de la estructura de la clase para cada clase. Los diagramas de Dependencia muestran las dependencias entre clases. Los diagramas de herencia muestran la herencia de relaciones entre clases"[1].

El diseño de *Shlaer* y *Mellor* es el más complejo de los métodos orientados a objetos.

[1] UNAM, Metodologías de Ingeniería de Software, Departamento de Auditoria Informática, Subdirección de Sistemas DCAA, 1997. Pág.32.

3.7.6 *UML* (*Unified Modeling Language*)

El Lenguaje de Modelamiento Unificado (*UML - Unified Modeling Language*) es un lenguaje de entorno gráfico el cual se utiliza para observar, especificar, construir y documentar todas las partes que componen el desarrollo de software. *UML* es una metodología para realizar el modelamiento de objetos como las funciones de sistema que son conceptuales, así como también cosas concretas como por ejemplo escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reutilizables. El objetivo principal de *UML* es proporcionar un material de apoyo que permita definir diagramas y poder entender el modelamiento de los diagramas ya existentes.

"*UML* es solo un lenguaje y por tanto es tan sólo una parte de un método de desarrollo de software. *UML* es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo incremental" [1].

UML puede utilizarse en sistemas que contengan mucho software, como por ejemplo: sistemas de información de empresas, servicios financieros, transporte, telecomunicaciones, defensa o industria aeroespacial, comercio, etc.

UML esta comprendido por tres bloques básicos de construcción los cuales son necesarios conocer para lograr comprender el lenguaje, una vez se logra comprender estos bloques básicos de construcción y como se pueden combinar entre ellos, se esta en capacidad de leer y crear modelos básicos; en el momento que se logre un dominio más completo de el lenguaje, se pueden crear modelos utilizando características mas avanzadas. Los bloques básicos de construcción de *UML* son: Elementos, Relaciones, Diagramas.

[1] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. El Lenguaje Unificado de Modelado. Addison Wesley, 1999. Pág.11.

Elementos. Estos bloques básicos son orientados a objetos y usados para crear modelos bien formados. En *UML* existen cuatro clases de elementos.

- **Elementos Estructurales**
- Clases
- Interfaces
- Colaboraciones
- Casos de Uso
- Clases Activas
- Componentes
- Nodos

- **Elementos de Comportamiento**
- Mensajes
- Estados

- **Elementos de Agrupación**
- Paquetes

- **Elementos de Anotación**
- Notas

Relaciones. Es la unión de los elementos entre sí.

- Dependencias
- Asociación
- Generalización
- Realización

Diagramas. Es una representación grafica para tener diferentes perspectivas durante la visualización del sistema.

- Diagramas de Clase
- Diagramas de Objetos
- Diagramas de Casos de Uso
- Diagramas de Secuencia
- Diagramas de Colaboración
- Diagramas de Estados
- Diagramas de Actividades
- Diagramas de Componentes
- Diagramas de Despliegue

3.8 INGENIERÍA DE SISTEMAS DE SOFTWARE

La ingeniería de sistemas de software se define por *Blanchard* "como la aplicación efectiva de esfuerzos científicos y de ingeniería para transformar una necesidad operativa en una configuración definida de un sistema mediante el proceso iterativo de análisis de requisitos"[1], se caracteriza por tener:

- Una estructura de arriba hacia abajo con la cual se logra ver el sistema como un todo.
- Un enfoque del ciclo de vida el cual considera todas las fases del proceso desde el diseño hasta que es retirado el sistema.
- Una orientación en equipo, que incluye las disciplinas necesarias para diseñar oportuna y repetitivamente; Asegurando que los objetivos del diseño se logren de manera eficaz y efectiva por medio de la integración necesaria.

[1] LEÓN SERRANO, Gonzalo. Ingeniería de Sistemas de Software. Isdefe - c/ Edison, 1998.Pág.31.

La ingeniería de sistemas de software es un tipo de especialización de la ingeniería de sistemas la cual cada vez es más importante y esta conectada a la ingeniería de los sistemas de software actual. Según *Pressman* la ingeniería de sistemas de software es "el establecimiento y uso de principios de ingeniería robustos, orientados a obtener software económico que sea fiable y funcione de manera eficiente sobre máquinas reales"[1].

3.9 MODELOS DE PROCESO DEL SOFTWARE

En el momento de solucionar problemas reales en una empresa para el diseño de algún tipo de software, un ingeniero del software o un equipo de ingenieros deben tener una habilidad de desarrollo que acompañe al *proceso* que es lo que mantiene unido a las capas de tecnología y permite un desarrollo racional y oportuno de la ingeniería de sistemas, *métodos* que es cómo se construye técnicamente el software, y capas de *herramientas* que proporciona un enfoque automático ó semi-automático para el proceso y para los métodos.

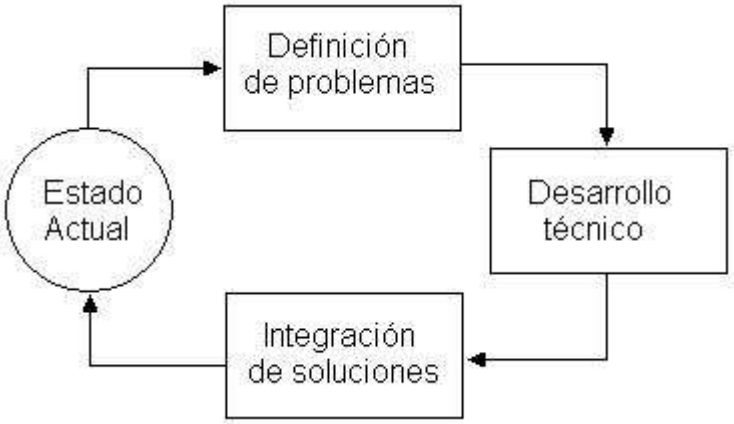
Para seleccionar el modelo de proceso de ingeniería de software debe tener ciertos aspectos como lo son la naturaleza del proyecto que significa qué utilidad va a tener el software, los métodos qué son los pasos a seguir durante el desarrollo de del software, las herramientas que sirven para el desarrollo de la metodología a utilizar, los controles y entregas.

“Todo el desarrollo del software se puede caracterizar como bucle de resolución de problemas en el que se encuentran cuatro etapas distintas: *Status quo* (Estado actual), definición del problema, desarrollo técnico e integración de soluciones.”[2]

Figura 4. Etapas del Desarrollo del Software

[1] LEÓN SERRANO, Gonzalo. Ingeniería de Sistemas de Software. Isdefe - c/ Edison, 1998.Pág.32.

[2] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág. 19.



PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág. 19.

3.9.1 Modelo lineal secuencial

Este llamado también “ciclo de vida básico o modelo en cascada, en este modelo sugiere un enfoque sistemático, secuencial para el desarrollo del software que comienza con en un nivel de sistemas y progresa con el análisis, diseño codificación, pruebas y mantenimiento”[1].

- **Análisis.** Se intensifica el proceso de reunión de requisitos y se centra especialmente en el software, el ingeniero del software debe tener en cuenta la información del sistema, su función, rendimiento y comportamiento.

- **Diseño.** Esta actividad es en realidad un proceso de varios pasos que se reúne en cuatro atributos distintos de programa: estructura de datos, arquitectura de software, representaciones de interfaz y detalle procedimental ó algoritmo.

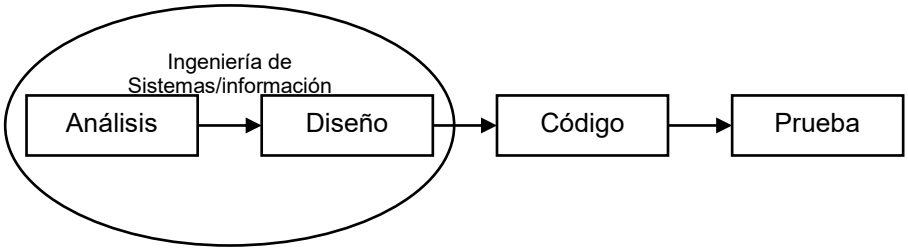
- **Generación de Código.** En esta actividad se plasma todo el diseño a un lenguaje que sea entendible por una máquina.

[1] Lbid.,Pág.20.

- **Pruebas.** después de haber hecho el código completo, se inicia las pruebas del programa; estas pruebas son procesos lógicos internos del software para la detección de errores.

- **Mantenimiento.** El software puede sufrir cambios después de ser entregado al cliente porque se pueden encontrar errores ó por cambios externos.

Figura 5. Modelo Lineal secuencial

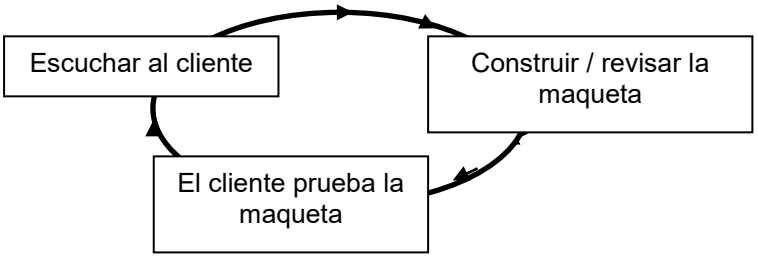


PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág. 20.

3.9.2 modelo de construcción de prototipos

Frecuentemente los clientes definen un cierto número de objetivos globales, pero no especifican los requerimientos detallados de entrada de proceso ó de salida, esto hace que este modelo no tenga una orientación bastante efectiva; este paradigma comienza con la recolección de requisitos, en común acuerdo el desarrollador y el cliente encuentran y definen los objetivos globales para el software, además identifican los requisitos conocidos y las áreas del esquema en donde es obligatorio ser más claro. En este modelo se desarrolla un prototipo en el que ocurren iteraciones cuando se pone a punto para satisfacer las necesidades del cliente. Desarrollar un prototipo es muy útil cuando el cliente tiene una necesidad o esta desorientado sobre los detalles.

Figura 6. Modelo de Construcción de Prototipos



PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.21

3.9.3 Modelo DRA

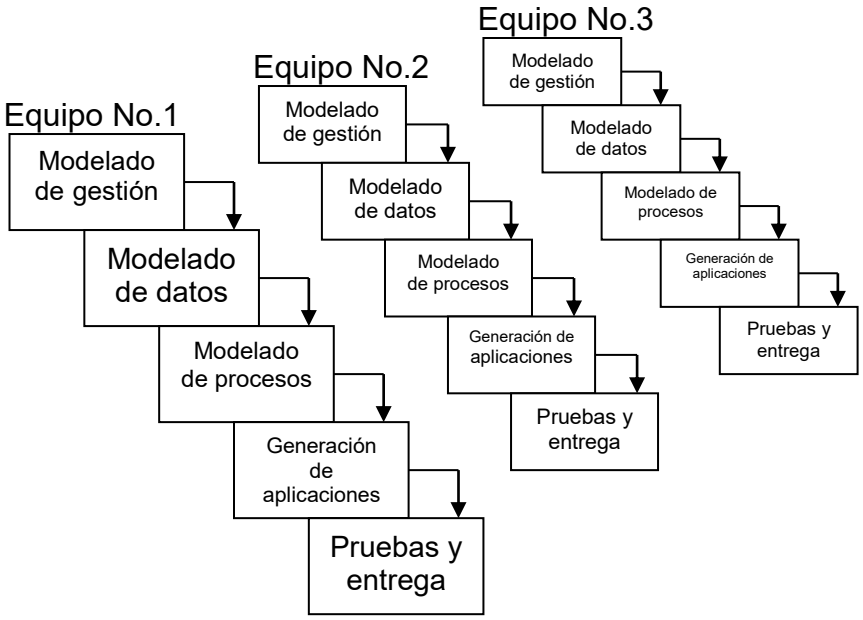
"El desarrollo Rápido de Aplicaciones (*DRA*) es un modelo de proceso del desarrollo de software lineal secuencial extremadamente corto que enfatiza un ciclo de desarrollo extremadamente corto"[1]. El *DRA* es un modelo basado en componentes, si en realidad se entienden los requerimientos del sistema y se definen cuales son los límites de este, permite a los desarrolladores construir sistema completamente funcional en cortos periodos de tiempo; "el enfoque *DRA* comprende las siguientes fases"[2]:

- **Modelado de Gestión.** En esta fase se deben tener en cuenta algunas preguntas: Que información conduce el proceso de gestión, que información se genera, quien la genera, a donde va la información y quien la procesa.
- **Modelado de Datos.** Se definen los atributos de cada uno de los objetos y las relaciones entre estos.
- **Modelado del Proceso.** Los objetos de datos definidos como parte de la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión.

[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág .22.
[2] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág.22.

- **Generación de Aplicaciones.** En esta fase se debe usar obligatoriamente técnicas de cuarta generación que hacen que se facilite el desarrollo del software.
- **Pruebas y Entrega.** Durante todo el proceso se enfatiza en la reutilización, esto permite que ya se hayan comprobado con anterioridad muchos de los componentes y ayuda a que se reduzca el tiempo de pruebas.

Figura 7. Modelo *DRA*



PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.22.

3.9.6 Modelos evolutivos de proceso de software

"El software al igual que todos los sistemas complejos, evoluciona con el tiempo. Los requisitos de gestión y de productos a menudo cambian conforme a que el desarrollo proceda haciendo que el camino que lleva al producto final no sea real"[1]. Los modelos evolutivos de proceso de software surgen comúnmente porque los desarrolladores de software no alcanzan a terminar los productos completamente en las fechas estipuladas por causa del mercadeo, por este motivo

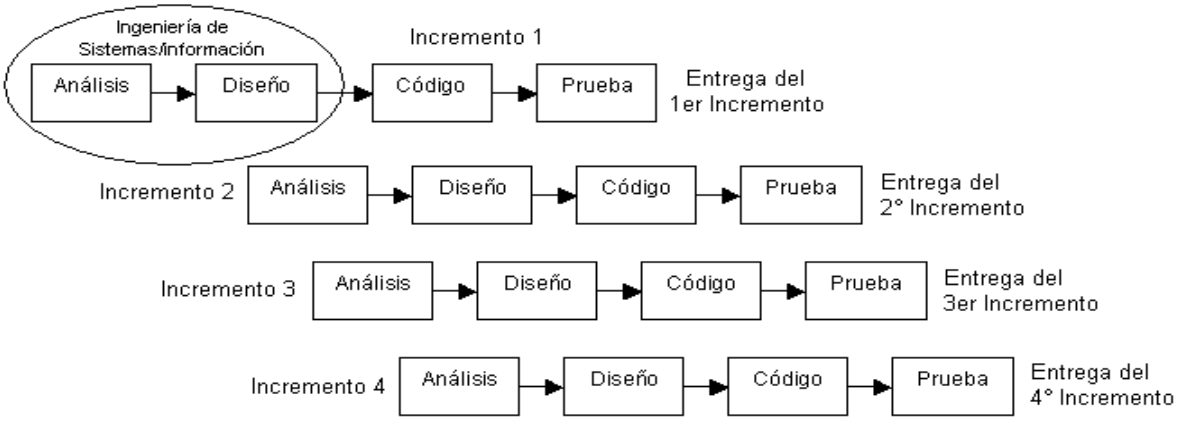
[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.23.

se hacen versiones limitadas para poder suplir la presión competitiva y de gestión. Algunos modelos evolutivos de proceso de software son:

3.9.5 El modelo incremental

Este modelo "combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos"[1]. El modelo incremental aplica secuencias lineales de forma escalonada mientras progresa en el tiempo. Cada secuencia lineal produce un incremento del software. En el momento de utilizar el modelo incremental la primera entrega es un producto vital porque contiene los requisitos básicos, aunque no están plasmados los requisitos suplementarios, esta entrega es una versión incompleta del producto final pero a medida que va avanzando en el tiempo se va mejorando el producto. Este modelo es especialmente usado cuando el personal no se encuentre disponible para una completa implementación en el plazo máximo en que se haya establecido para la entrega del proyecto.

Figura 8. Modelo Incremental



[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002. Pág. 23.

PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.24.

3.9.6 El modelo espiral

"Este modelo de proceso de software evolutivo que conjuga la naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial"[1]. En el modelo espiral siempre desarrolla en una serie de versiones incrementales, al principio del desarrollo de este modelo se hacen diseños plasmados en papel ó prototipos del sistema, mientras que en al final del desarrollo de este modelo se realizan versiones más completas que hacen que sistema sea más robusto. Normalmente en este modelo se puede ver las siguientes actividades:

- **Comunicación con el Cliente.** En esta actividad es donde el desarrollador interactúa con el cliente para la recolección de requisitos entre otras cosas.

- **Planificación.** En esta actividad se define los recursos a utilizar, el tiempo que se va a gastar, y además información pertinente al proyecto.

- **Análisis de Riesgos.** Esta actividad es donde se evalúa los riesgos técnicos del proyecto y los de gestión también.

- **Ingeniería.** Esta actividad es donde se construyen diversas representaciones de la aplicación a realizar.

- **Construcción y Acción.** esta es la actividad donde se construye, se prueba, se instala y se proporciona soporte al usuario de la aplicación como por ejemplo el manual de usuario.

[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.24.

- **Evaluación del Cliente.** En esta actividad es donde se analiza las reacciones del cliente con respecto a la aplicación que se realizo teniendo en cuenta que el cliente es el que evalúa lo que se realizó.

Figura 9. Modelo Espiral

PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.24.

3.9.7 El modelo espiral *WinWin*

"El objetivo de este es mostrar los requisitos del cliente. En un contexto ideal el desarrollador simplemente pregunta al cliente lo que necesita y el cliente proporciona detalles suficientes para continuar, en pocas palabras el desarrollador y el cliente entran en un proceso de negociación donde el cliente puede ser preguntado para sopesar la funcionalidad, rendimiento, y otros productos o características del sistema frente al coste y al tiempo"[1].

[1] PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.26.

Esto quiere decir que tanto el desarrollador como el cliente ganan siempre porque el primero consigue el presupuesto más fácilmente y puede lograr una fecha de entrega del software razonable, mientras que el segundo gana porque obtiene un producto que logra satisfacer mayoría de las necesidades.

Figura 10. Modelo en Espiral *WinWin*

PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.Pág.26.

3.10 *RUP* (PROCESO UNIFICADO DE RATIONAL).

"El Proceso Unificado de Rational es un proceso de ingeniería de software; proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo"[1]. La finalidad del *RUP* es la de cerciorarse de que la producción del software sea de alta calidad y que se pueda ajustar a todas las necesidades de los usuarios finales teniendo en cuenta una serie de costos y un tiempo estimado que sea predecible.

[1] MARTÍNEZ, Alejandro. MARTÍNEZ, Raúl. Guía a Rational Unified Process. Escuela Politécnica Superior de Albacete – Universidad de Castilla la Mancha. 2002. Pág.1.

En pocas palabras el *RUP* es una metodología para el desarrollo de software la cual agrupa todos los aspectos que se deben tener en cuenta a través de todo el proceso del ciclo de vida del software, teniendo como objetivo facilitar la construcción de proyectos de software de gran, mediana y pequeña magnitud. La metodología del *RUP* se caracteriza por ciertos aspectos:

- **Guiado y/o manejado por casos de uso.** Los casos de uso son muy útiles durante todo el proceso de desarrollo del software puesto que ayuda a comprender todas las actividades que se van a desarrollar en las fases de diseño, implementación y pruebas del sistema.

- **Centrado en arquitectura.** Los componentes más importantes del sistema están contenidos dentro de la arquitectura la cual depende de las plataformas de software, sistemas operativos, protocolos, manejadores de bases de datos, etc. Esto significa que lo que se esta desarrollando quede muy bien especificado para tener una idea clara de lo que se esta desarrollando.

- **Iterativo e incremental.** El *RUP* se encuentra divido en ciclos, en cada uno de estos se realizan una serie de actividades haciéndolos ver como si fueran proyectos más pequeños, en los cuales se desarrollan múltiples iteraciones de un mismo proceso para mejorarlo.

3.10.1 Fases

El *RUP* se divide en cuatro fases: Inicio, Elaboración, Construcción y Transición.

3.10.1.1 Inicio. Se trata de explorar el problema para decidir si va a continuar ó se va dejar. Los objetivos de esta fase son:

- Establecer el ámbito del proyecto y sus límites.

- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Mostrar al menos una arquitectura candidata para los escenarios principales.
- Estimar el costo en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

Los productos de la fase de inicio deben ser:

- Requerimientos.
- Análisis.
- Casos de Uso.
- Diagrama de Clases.
- Diagramas de Estado.

3.10.1.2 Elaboración. El propósito de la fase de elaboración es analizar el dominio del problema, establecer los cimientos de la arquitectura, desarrollar el plan del proyecto y eliminar los mayores riesgos.

Cuando termina esta fase se llega al punto de no retorno del proyecto: a partir de ese momento pasamos de las relativamente ligeras y de poco riesgo dos primeras fases, a afrontar la fase de construcción, costosa y arriesgada. Es por esto que la fase de elaboración es de gran importancia. Los objetivos de esta fase son:

- Definir, validar y cimentar la arquitectura.
- Completar la visión.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Debe incluir los costos si procede.
- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Al terminar deben obtenerse los siguientes productos:

- Diagramas de Interacción.
- Diagramas de Colaboración.
- Diagramas de Secuencia.
- Diagramas de Actividad.
- Diagramas del Detalle de Clases.

3.10.1.3 Construcción. La finalidad principal de esta fase es alcanzar la capacidad operacional del producto de forma incremental a través de las sucesivas iteraciones. Durante esta fase todas los componentes, características y requisitos, que no lo hayan sido hecho hasta ahora, han de ser implementados, integrados y probados, obteniéndose una versión del producto que se pueda poner en manos de los usuarios (una versión beta). Los objetivos de esta fase son:

- Minimizar los costes de desarrollo mediante la optimización de recursos y evitando el tener que rehacer un trabajo o incluso desecharlo.
- Conseguir una calidad adecuada tan rápido como sea práctico.
- Conseguir versiones funcionales (alfa, beta, y otras versiones de prueba) tan rápido como sea práctico.

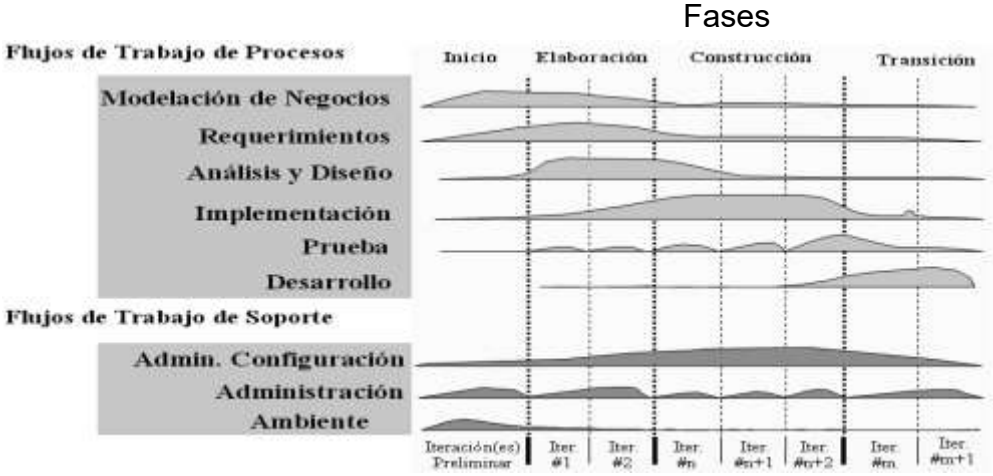
Los productos de la fase de construcción deben ser:

- Modelos Completos (Casos de Uso, Análisis, Diseño, Despliegue e Implementación).
- Diagramas de Componentes.
- Diagramas de Despliegue.
- Arquitectura íntegra (mantenida y mínimamente actualizada).
- Manual Inicial de Usuario (con suficiente detalle).

- Prototipo Operacional – beta.
- Caso del Negocio Actualizado.

3.10.1.4 Transición. La finalidad de la fase de transición es poner el producto en manos de los usuarios finales, para lo que se requerirá desarrollar nuevas versiones actualizadas del producto, completar la documentación, entrenar al usuario en el manejo del producto, y en general tareas relacionadas con el ajuste, configuración, instalación y el uso del producto.

Figura 11. Proceso Unificado de Rational



BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. El Lenguaje Unificado de Modelado. Addison Wesley, 1999. Pág.401.

3.11 CIBERNÉTICA

El Pensamiento Sistémico, se origina de la preocupación de los norteamericanos al finalizar la Segunda Guerra Mundial, su preocupación era defenderse de los bombardeos alemanes. En el *MIT* (Instituto Tecnológico de Massachussets), se nombro a *Norbert Wiener*, matemático de este instituto, quien logro solucionar técnicamente este problema, todo a partir de una máquina de combate

semiautomática, que era capaz de ajustarse, con base en la información recibida vía radar, de esta forma detectaba el vuelo de los aviones, para dirigir proyectiles capaces de interceptar los vuelos alemanes, todo esto usaba una nueva teoría matemática que permitía predecir eventos con cierta precisión, utilizando información estadística incompleta. No obstante era necesario un mecanismo en movimiento variable, y así surgió el concepto de realimentación ya que este instrumento se ajustaba corrigiendo automáticamente el funcionamiento, logrando orientarse hacia un blanco en movimiento.

La cibernética surgió como la ciencia destinada a establecer las relaciones entre las diversas ciencias y se le conoce como la ciencia del control y de la comunicación, que hace un cambio total en la manera de observar e intervenir un sistema, llenando los espacios vacíos no investigados por ninguna ciencia, y que permite que cada una de las ciencias utilice los conocimientos generados por las demás para su desarrollo. Según *Wiener* la Cibernética es *"*La ciencia del control y de la comunicación en el animal y la máquina" que deja ver hipótesis nuevas sobre la naturaleza de los sistemas abiertos y sobre la estructura de nuevos conceptos como los de realimentación, homeóstasis y caja negra, desarrollados con base en analogías biológicas. La cibernética también conlleva a otros campos como supervivencia, adaptación, desarrollo, crecimiento, flexibilidad y estabilidad.

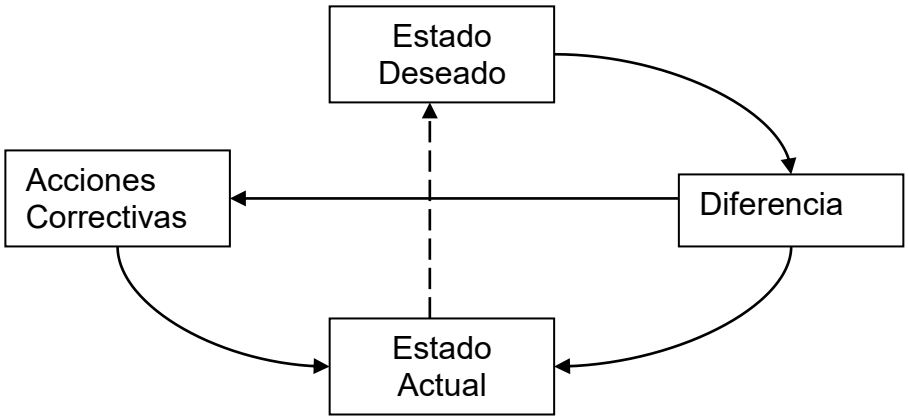
3.12 CIBERNÉTICA ORGANIZACIONAL

Stafford Beer, estableció una nueva disciplina llamada Cibernética Organizacional la cual se preocupa por aclarar el enfoque de la *viabilidad* como un objetivo principal de todas las organizaciones que se están desarrollando en un entorno rápido y en continuo cambio, esto se origina por la mayoría de los mercados internacionales y entornos tecnológicos, entornos de conocimiento, etc., Los cuales están en un acelerado y constante cambio. Dentro de este marco las organizaciones viables son todas aquellas que logran adaptarse de forma rápida a cualquier eventualidad o cambio dentro de ellas. Según "*Espejo* y *Schwaninger* la

Cibernética Organizacional es: La ciencia de la organización efectiva"[1], que se consolida a través del tiempo.

La Cibernética Organizacional ha proporcionado una herramienta para mejorar el manejo de la información y la capacidad organizacional, esta herramienta es el Modelo del Sistema Viable (*MSV*).

Figura 12. Esquema de la Cibernética Organizacional



ACEVEDO MAZA, Juan Francisco. Estudiante ingeniería de sistemas UNAB.

3.13 MODELO DEL SISTEMA VIABLE (MSV)

El *MSV* es una de las herramientas principales que proporciona a la Cibernética la capacidad de mejorar la organización en la administración moderna, permitiendo que esta sea analizada en forma dinámica y sin rigidez en donde el manejo de la información además permite identificar cuatro procesos básicos: la autorregulación, autoestructuración, consonancia y coherencia; para que estos procesos sean posibles la estructura organizacional debe irse adaptando continuamente de esta manera la organización tendrá flexibilidad suficiente para

[1] BEER, Stafford. Decision and Control. Nueva York: Wiley, 1966. Pág. 289.

involucrar todas las áreas de complejidad que se consideran relevantes para su desarrollo.

3.13.1 Procesos Básicos del *MSV*

Autorregulación. La Autorregulación logra que las distintas partes de una organización desarrollen en forma natural los flujos de información mas importantes para el control de los procesos y la supervivencia; además esta relacionado con el comportamiento del sistema, basándose en un conjunto de referencias que permiten adaptabilidad a su medio ambiente; Por eso es importante conocer el componente vital del sistema y reconocer todas las fronteras de este, para que se adapte más fácilmente al medio.

Autoestructuración. Se relaciona con la estructura y los cambios que se vayan necesitando, basándose en las referencias que se necesitan para poder mantener la capacidad organizacional; también hace referencia a la capacidad que posee una organización para lograr adaptarse al medio, en pocas palabras obtener la flexibilidad de esta para adaptarse a los cambios.

Consonancia. Se basa en la periodicidad que hay entre los diferentes procesos que se desarrollan dentro de la organización, aquí es muy importante considerar cual es el periodo de las actividades que se realizan, ya que así se permite establecer los diferenciales procedímentales, y también ver los desfases que puedan presentarse entre estos.

Coherencia. Este proceso básico se encamina a que la Misión y la Visión de la organización sean orientadas hacia un mismo camino, en una organización no se puede tener una Misión y una Visión sin que estas estén acompañadas de unas acciones oportunas para poder conseguir las metas propuestas.

El *MSV* ofrece un lenguaje para estudiar la viabilidad de las organizaciones y un método para analizarlas, se puede ayudar a los responsables de una organización

a dialogar de forma más efectiva y a actuar de manera más coherente. La manera como opera el *MSV* en una organización, es analizándola como un todo, integrando sus diferentes unidades en una acción realimentándola continuamente, velando por la supervivencia interna y externa de la organización. Debido a que la organización es analizada como un sistema cibernético que procesa información a través de circuitos cerrados de realimentación, cada nivel maneja un tipo particular de información, controla de una u otra forma la actividad de sus niveles inferiores y a la vez esta controlado por el nivel superior.

El *MSV* permite observar una organización desde la perspectiva de las relaciones entre los diversos grupos e interlocutores internos y externos que vendrían a ser los clientes, proveedores, interventores, etc.

Para que las organizaciones sean adaptativas, deben desarrollar esquemas de interacción social que permitan balancear y suavizar la acción de los diferentes grupos que estimulan un aprendizaje permanente de estos esquemas y de la empresa. El *MSV*, además sirve para distinguir entre los niveles operacionales básicos de cualquier organización social y una nueva organización de sistemas reconocibles y categorizados en “sistemas o subsistemas”, también conocidos como funciones administrativas, estas funciones permiten entender y conocer en detalle como opera la organización y en especial identificar quienes son responsables de cada función dentro de la misma; estas funciones son: Implementación, Coordinación, Monitoreo y Control, Inteligencia y Política.

La cibernética organizacional en las diferentes organizaciones ya sean de carácter público ó privado es de gran importancia, Se propone la Cibernética Organizacional como: “Un concepto de organización social al Modelo del Sistema Viable, y sugiere algunas reflexiones que este modelo plantea como conceptos de

cambios y aprendizaje"[1] o como "Adoptar el enfoque cibernético involucra un cambio radical en la manera de ver las organizaciones"[2].

Finalmente la cibernética organizacional sugiere que la efectividad de una organización, tiene relación íntima con sus habilidades para desarrollar el potencial de cada uno de sus miembros, mientras que al mismo tiempo conserva su identidad como organización. Resalta el peligro de desarrollar organizaciones extremadamente centralizadas o descentralizadas y sugiere criterios para diseñar organizaciones balanceadas.

3.14 INDICADORES DE GESTIÓN

Los indicadores de gestión permiten observar a una organización todas las variables fundamentales sobre las que requiere garantizar su comportamiento, visualizando su influencia sobre la organización y los cambios internos y externos.

Para identificar los indicadores que son las variables que filtran información para evaluar una situación se utiliza el *MSV* (Modelo del Sistema Viable); que permite el tratamiento de indicadores que se deben tener en cuenta en la identificación de las variables críticas y el desarrollo de un conjunto de mediciones para ellas. Al realizar la valoración y el seguimiento se permite identificar los problemas organizacionales, y además sugerir las recomendaciones para mejorarla. "Un indicador consta de un valor normativo (capacidad o nivel de meta satisfactorio), valor estratégico (potencialidad o nivel de meta sobresaliente), resultado táctico (actualidad o valor del indicador)"[3].

[1] ESPINOSA, Ángela María. Una visión de las organizaciones Sociales. Bogotá, 1999.Pág.3.
[2] VELANDIA, Néstor Orlando. Guía Teórica y práctica de la Cibernética Organizacional. Bogotá. 1998. Pág.29.

[3] MORALES MONTEJO, Clemencia. Seminario Indicadores de Gestión. Universidad de los Andes. Facultad de Ingeniería Industrial. Bogotá. 1994.

El éxito de la definición de los indicadores, consiste en detectar únicamente los que son primordiales y seleccionar los que son apropiados para una óptima definición de costo-beneficio. Se pueden definir diferentes clases de indicadores tales como:

3.14.1 Indicadores atómicos

Son los se determinan para un sistema en tiempo real, las medidas se toman todo el tiempo y las excepciones se reportan cada vez que ocurre algo.

3.14.2 Indicadores moleculares

Este se maneja cuando hay más de un nivel de recursión, se hace más difícil las transformaciones que van de muchas a una, debido a la complejidad de los niveles.

3.14.3 Indicadores de eficiencia

Detectan los diferentes ciclos de procesos internos de la empresa (adaptadores organizacionales).

3.14.4 Indicadores de eficacia

Se relaciona con procesos interconectados con el medio ambiente a un nivel de recursión (gerencia medio).

3.14.5 Indicadores de estabilidad

Es identificar con claridad lo que se hizo hoy, lo real, se compara con lo que ha podido hacer, teniendo en cuenta el nivel actual de recursos y los obstáculos operacionales como si todo hubiera sido organizado en forma óptima. El indicador es una razón a la que se le asignan eventos.

El control se ejerce de acuerdo a indicadores que son una medida de lo que la organización hace en un determinado momento. La descomposición de objetivos

en metas y de actividades tiene implicación en el proceso de definición de indicadores. Cuando estos están fuera de control debe ser posible detectar inestabilidad.

Las actividades primarias son las que identifican la organización e implementan las tareas implícitas en la identidad organizacional o sea los productos que se ofrecen en el mercado y de los cuales depende su viabilidad. Entonces es necesario tener una metodología que suministre la información sobre indicadores básicos que identifiquen donde hacer la intervención en forma oportuna. Solo de esta manera la empresa se irá evaluando constantemente con respecto a los indicadores que se establecen y que se supone son suficientes para garantizar que esta logre los resultados que se desean.

Hay necesidad de estudiar no solo los indicadores internos sino externos situados fuera de los límites de la organización y que influyen en aspectos estructurales y de comportamiento.

Los indicadores deben diseñarse para las actividades organizacionales a diferentes niveles de agregación. Debe ser posible identificar un conjunto de indicadores los cuales deben proveer una medida de acerca del comportamiento de la organización; dichas medidas deben poder ayudar a los administradores a tomar acciones que ayuden a mejorar el grado de comportamiento de las situaciones.
En la medición de los indicadores es importante tener medidas de su comportamiento en términos de actualidad, capacidad y potencialidad.

Actualidad. Es un valor fluctuante basado en la medición acerca de lo que se hace en el presente con los recursos y restricciones existentes.

Capacidad. Es lo máximo que podría hacerse con los recursos existentes y bajo las restricciones presentes, es un valor fijo. Es importante hacer explícitos los recursos y restricciones que se relacionan con la capacidad máxima.

Potencialidad. Es lo máximo que se puede obtener si se desarrollan los recursos y se remueven los cuellos de botella para mejorar la capacidad.

Los sistemas basados en indicadores sirven para la toma de decisiones oportunas, mantienen un flujo permanente de información sobre éstos que permiten reflejar la salud de la empresa, proporcionan al gerente la posibilidad de mantener el control para un desempeño exitoso.

3.15 ÍNDICES

Cuando las tres mediciones de los indicadores están hechas es posible tener índices. Estos ayudan a observar el comportamiento de la organización en el corto, mediano, largo plazo y a visualizar qué sucede bajo ciertas circunstancias si se mejora el comportamiento de los indicadores en el tiempo.

3.15.1 Índice de logro o productividad

Es el resultado entre actualidad y capacidad máxima. Mide los posibles desarrollos organizacionales considerando recursos que están presentes pero inactivos.

3.15.2 Índice de latencia

Es el resultado entre capacidad y potencialidad, mide que tan lejos puede llegar el sistema en uno, dos ó tres años. Aquello que se esta presente pero no se ha alcanzado.

3.15.3 Índice de comportamiento

Es el resultado entre actualidad y potencialidad. Su significado se define de acuerdo con el presente y el futuro. Produce un balance entre los alcances en el corto tiempo y la necesidad de desarrollar recursos para que la productividad se mantenga o incremente en el futuro.

4 DISEÑO DEL SISTEMA Y DEL SISTEMA COMPUTARIZADO

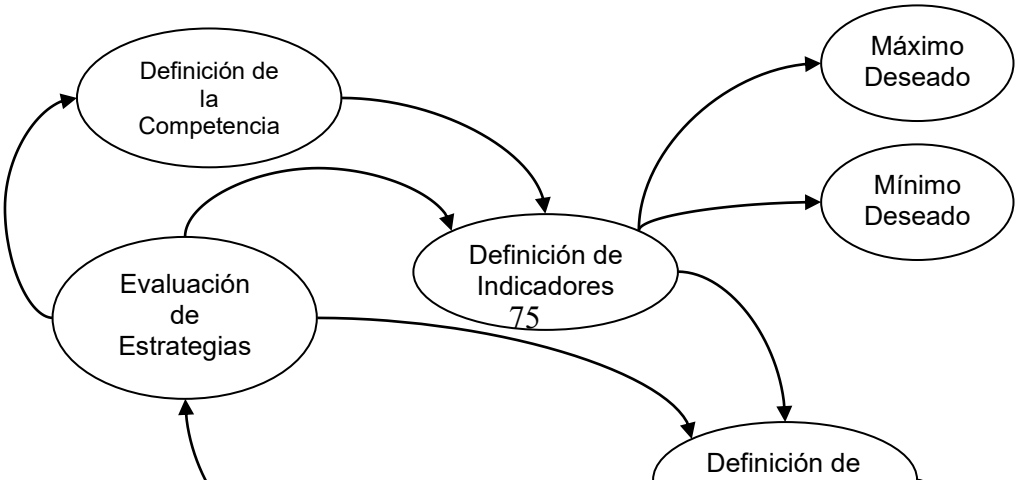
Para realizar el diseño del sistema propuesto se tuvieron en cuenta aspectos tales como el ciclo de gestión de competencias propuesto, la determinación de la metodología de ingeniería de software, los indicadores de evaluación (internos-

externos), también la selección y la evaluación de la plataforma tecnológica que contiene la elección del sistema operativo, la plataforma tecnológica de desarrollo, bases de datos, herramienta de diseño, y herramienta de programación.

4.1 CICLO DE GESTIÓN DE COMPETENCIAS PROPUESTO

El ciclo de gestión de competencias propuesto comprende la definición de competencias de donde se desprende los indicadores que contienen estrategias que son acciones para lograr metas, a los indicadores se les asignan valores deseados (máximo deseado, mínimo deseado), se definen las estrategias, que son acciones para lograr las metas propuestas; los indicadores se evalúan por medio de una serie de pruebas que se le hacen a las estudiantes con preguntas tipo ECAES y dependiendo de el valor obtenido en las pruebas se realiza una comparación, se reflexiona sobre ello para poder realizar evaluaciones de estrategias y así llevar acabo acciones correctivas ya sea sobre las competencias, los indicadores, las estrategias, las preguntas y los valores deseados.

Figura 13. Ciclo de gestión de competencias propuesto



GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas UNAB.

4.2 DETERMINACIÓN DE METODOLOGÍA DE INGENIERÍA DE SOFTWARE

Se eligió UML como metodología de ingeniería de software para el desarrollo del diseño del sistema, porque maneja un entorno gráfico que ayuda al diseñador de software a tener una visión más completa del modelo a desarrollar, además UML tiene herramientas que ayudan en las diferentes fases del desarrollo del software. Aunque UML es independiente del proceso, para poder aprovechar esta herramienta al máximo se debe utilizar una metodología de desarrollo de software que sea dirigido por casos de uso, que sea centrado en la arquitectura y que sea iterativo incremental como lo es el Proceso Unificado de Rational (RUP).

4.2.1 Diagramas de casos de uso[1]

Hay varios motivos por los cuales los casos de uso son buenos, se han hecho populares y se han ido adoptando universalmente, existen varias razones

[1] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar.El Proceso Unificado de Desarrollo de Software. Addison Wesley, 2000. Pág.138.

fundamentales para utilizar diagramas de Casos de Uso, estas son las más importantes:

- Proporcionan un medio sistemático e intuitivo de capturar los requisitos funcionales centrándose en el valor añadido para el usuario.

- Dirigen todo el proceso de desarrollo debido a que la mayoría de las actividades como el análisis, diseño y prueba se llevan a cabo partiendo de los casos de uso.

Los Casos de Uso se componen de Actores y Casos de Uso, que sirven para, delimitar el sistema de su entorno, esbozar quién y qué (Actores) interactúan con el sistema, y que funcionalidad (Casos de Uso) se espera del sistema, Las descripciones detalladas de las funcionalidades del sistema.

La actividad de encontrar los actores y casos de uso es tal vez la actividad más decisiva para poder obtener adecuadamente los requisitos, y es responsabilidad del analista de sistemas, pero el analista de sistemas no puede hacer este trabajo solo, necesita de la ayuda de los clientes, los usuarios, y también de otros analistas; con una breve descripción general o una especificación detallada de lo que se necesita se puede hallar los actores y casos de uso.

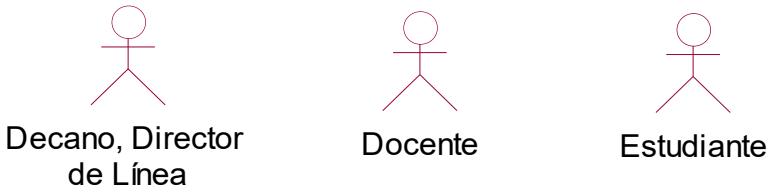
Esta actividad consta de 4 pasos[1]:

- Encontrar Actores.
- Encontrar Casos de Uso.
- Describir brevemente cada Caso de Uso.
- Describir el modelo de Casos de Uso completo.

[1] Lbid., Pág.139.

○ **Actores.** Para encontrar los actores el analista del sistema, con los requisitos dados y junto con el cliente identifican los usuarios e intenta organizarlos por categorías representados por actores. En el sistema realizado se encontraron 3 tipos de actores que son Decano - Director de Línea; Docente; Estudiante.

Figura 14. Actores del Sistema

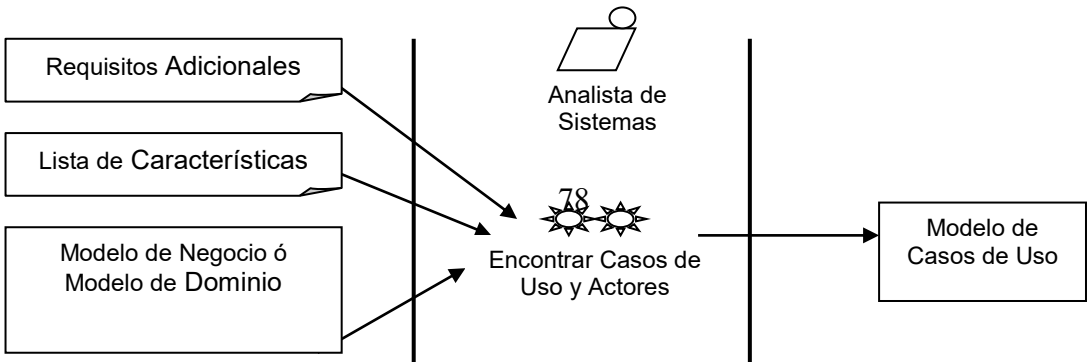


GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas UNAB.

Se deben identificar cuáles son los actores que representan sistemas externos y los actores para la operación del sistema, el analista de sistemas asigna un nombre a los actores y describe brevemente los papeles de cada actor y para qué lo utiliza en el sistema; encontrar nombres relevantes para los actores es importante para comunicar la semántica deseada.

○ **Casos de uso.** El analista de sistemas puede encontrar los casos de uso a través de talleres con los usuarios y clientes, el analista va repasando con cada uno de los actores proponiendo cada uno de los casos de uso relacionados para dicho actor; se elige un nombre para cada caso de uso de forma que representa una secuencia de acciones concreta que añade valor a un actor.

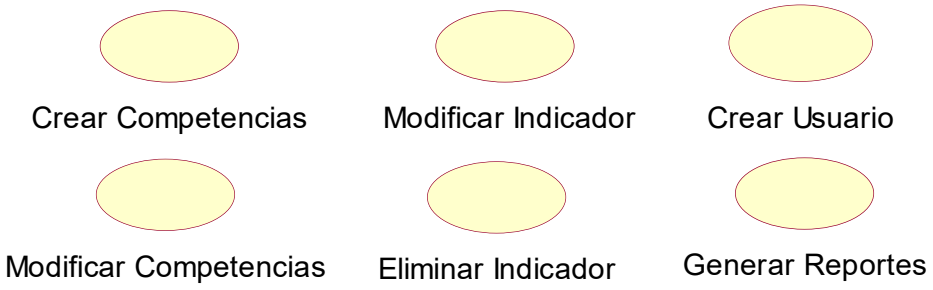
Figura 15. Entradas y resultados para identificar actores y casos de uso



BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar.El Proceso Unificado de Desarrollo de Software. Addison Wesley, 2000. Pág.138.

Estos son algunos de los casos de uso que se encontraron en el sistema; para ver los diagramas de casos de uso ir al **anexo B**.

Figura 16. Casos de Uso del Sistema



ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

4.2.2 Diagramas de secuencia[1]

Un diagrama de secuencia es un diagrama de interacción que resalta la ordenación temporal de los mensajes. Un diagrama de secuencia presenta un conjunto de objetos y los mensajes enviados y recibidos por ellos, en si los

[1] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar.El Lenguaje Unificado de Modelado. Addison Wesley, 1999. Pág.84.

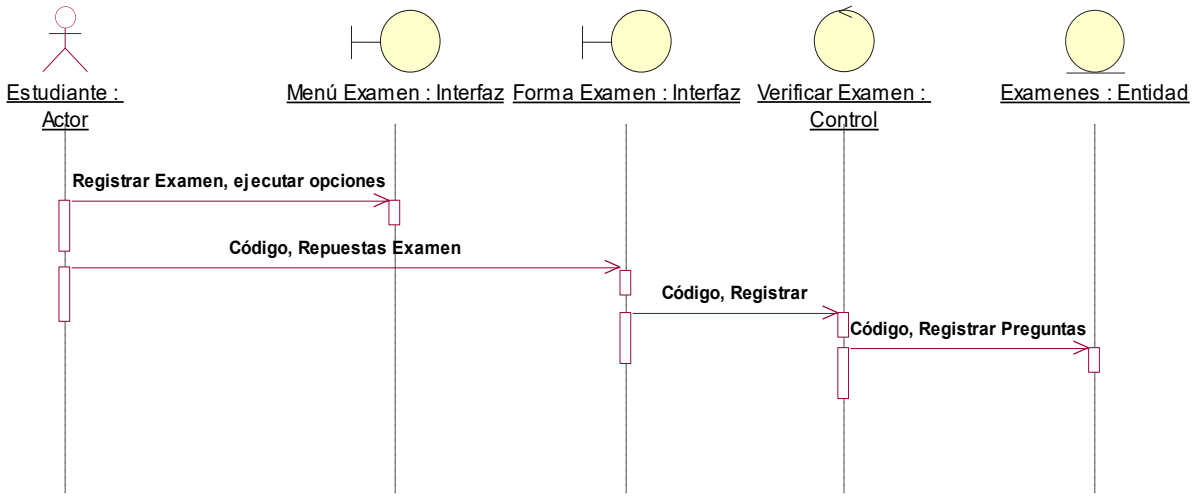
diagramas de secuencia se utilizan para describir la vista dinámica del sistema. Un diagrama de secuencia se compone de actores, interfaces, controles y entidades.

Los diagramas de secuencia tienen dos características que los distinguen de los diagramas de colaboración.

- La línea de vida: la línea de vida de un objeto es la línea discontinua vertical que representa la existencia de un objeto a lo largo de un periodo de tiempo.

- El foco de control: es un rectángulo delgado y estrecho que representa el período de tiempo durante el cual un objeto ejecuta una acción.

Figura 17. Diagrama de Secuencia



ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

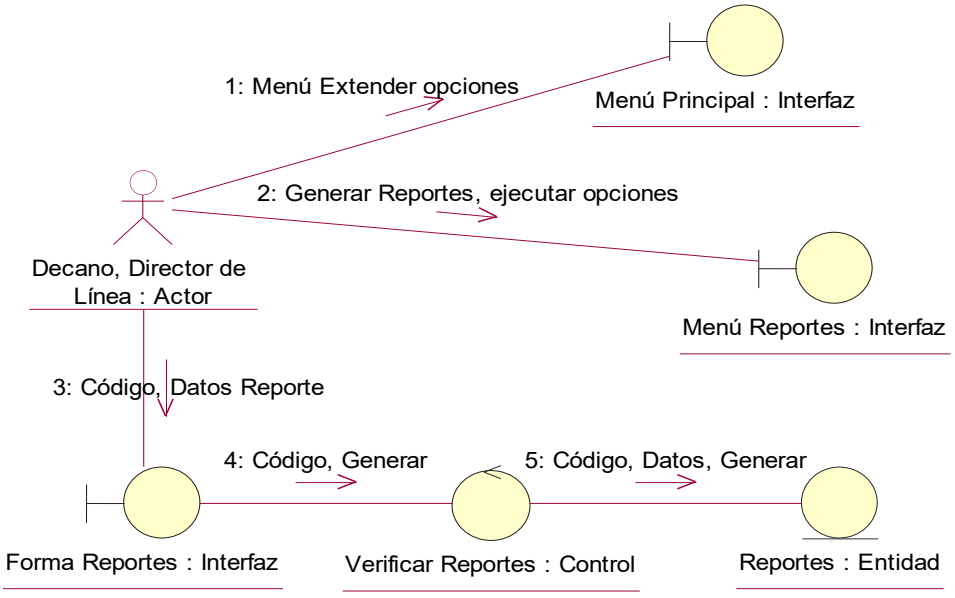
Para ver los diagramas de secuencia ir al **anexo C.**

4.2.3 Diagramas de colaboración[1]

Un diagrama de colaboración es un diagrama de interacción que resalta la organización estructural de los objetos que envían y reciben mensajes, los diagramas de colaboración tienen dos características que los distinguen de los diagramas de secuencia.

- El camino para indicar como se enlaza un objeto a otro.
- El número de secuencia para indicar la ordenación temporal de un mensaje, se precede de un número, que se incrementa secuencialmente por cada nuevo mensaje en el flujo de control.

Figura 18. Diagrama de Colaboración



ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

Para ver los diagramas de colaboración ir al **anexo D.**

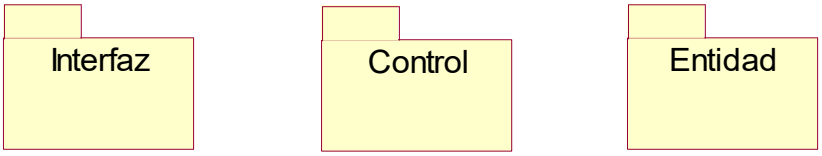
[1] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar.El Lenguaje Unificado de Modelado. Addison Wesley, 1999. Pág.84.

4.2.4 Diagramas de paquetes[1]

Un paquete es un mecanismo de propósito general para organizar elementos en grupos, gráficamente un paquete se representa como una carpeta, cada paquete tiene un nombre que lo distingue de los otros paquetes. Un paquete se dibuja normalmente mostrando sólo su nombre.

Estos son los paquetes que hay en el sistema; para ver los diagramas de paquetes ir al **anexo E.**

Figura 19. Diagrama de Paquetes



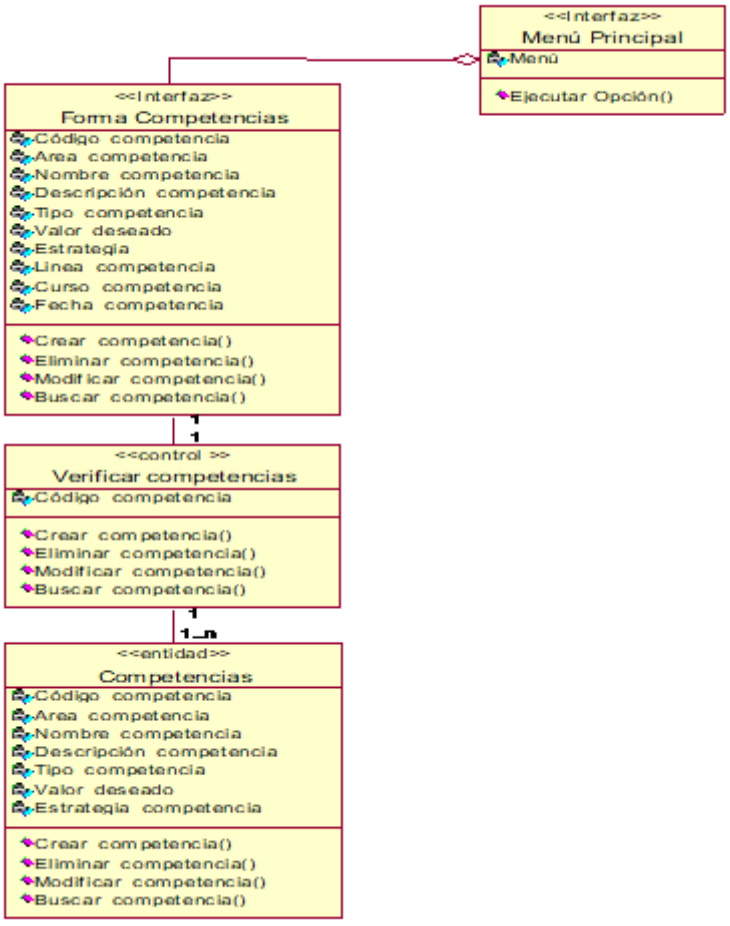
GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas.

4.2.5 Diagramas de clases[2]

Un diagrama de clases presenta un conjunto de clases, interfaces y colaboraciones y las relaciones entre ellas. Un diagrama de clases son los diagramas más comunes en el modelado de sistemas orientados a objetos. Los diagramas de clases se utilizan para describir la vista de diseño estática de un sistema. Los diagramas de clases contienen normalmente: Clases, Interfaces, Controles, Entidades y Relaciones de dependencia, generalización y asociación.

Figura 20. Diagrama de Clases

[1] BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar.El Lenguaje Unificado de Modelado. Addison Wesley, 1999. Pág.148.
[2] Lbid. Pág.94.



ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

Para ver el diagrama de clases completo ir al **anexo F.**

4.3 INDICADORES DE EVALUACIÓN DEL SISTEMA.

4.3.1 indicadores externos

Nombre: Rendimiento del sistema.

Definición*:* Este indicador aparece para saber como es el funcionamiento del sistema de gestión de competencias, para esto es necesario tener ciertas características mínimas para que dicho sistema sea operable.

Forma de evaluar: Porcentaje – %.

Periodicidad: Diaria.

Valor Mínimo: 51%

Valor Máximo: 100%

Valor Deseado: 78%

Nombre: El sistema es multi-usuario.

Definición: Este indicador aparece para saber si el sistema soporta cierta cantidad de usuarios cuando se ejecuta el software en un determinado tiempo.

Forma de evaluar: Si – No.

Periodicidad: Diaria.

Valor Mínimo: 40

Valor Máximo: 1000

Valor Deseado: 300

Nombre: Tipos de usuario.

Definición**:** Este indicador sirve para poder medir si en el sistema existe diferenciación (privilegios) entre los tipos de usuario (Decano, Director de Línea, Docente, Alumno).

Forma de evaluar: Si – No.

Periodicidad: Única.

Valor Mínimo: 1

Valor Máximo: 3

Valor Deseado: 2

Nombre: Tiempo de respuesta de solicitud de reportes.

Definición*:* Este indicador aparece por la necesidad de saber si al momento de pedir un reporte de cualquier tipo el tiempo de respuesta es aceptable (en segundos).

Forma de evaluar: Tiempo.

Periodicidad: Diaria.

Valor Mínimo: 3 Segundos.

Valor Máximo: 20 Segundos.

Valor Deseado: 7 Segundos.

Nombre: Acceso seguro.

Definición*:* Este indicador aparece por la necesidad de que el sistema sea seguro en cuánto a la autenticación de usuarios potenciales, manejo de contraseñas y que sea segura la transmisión de los datos de la misma.

Forma de evaluar: Si – No.

Periodicidad: Diaria.

Valor Mínimo: 0 Accesos ilegales.

Valor Máximo: 0 Accesos ilegales.

Valor Deseado: 0 Accesos ilegales.

Nombre: Facilidad de uso.

Definición: Este indicador sirve para poder medir si el sistema es fácil de operar y que sea de fácil entendimiento.

Forma de evaluar: Alto – Medio – Bajo.

Periodicidad: Semanal.

Valor Mínimo: Medio.

Valor Máximo: Alto.

Valor Deseado: Medio.

Nombre: Crecimiento del sistema.

***Definición**:* Este indicador aparece para poder saber si el sistema puede ampliarse a otras facultades ó a toda la universidad sin la necesidad de hacer cambios drásticos.

Forma de evaluar: Si – No.

Periodicidad: Semestral.

Valor Mínimo: 1 Facultad.

Valor Máximo: Todas las Facultades.

Valor Deseado: 1 Facultad.

Nombre: Cambios del sistema.

Definición: Este indicador nace para poder saber si el sistema soporta modificaciones de cualquier tipo en el sistema.

Forma de evaluar: Si – No.

Periodicidad: Semestral.

Valor Mínimo: Ningún cambio.

Valor Máximo: Varios Cambios.

Valor Deseado: Pocos Cambios.

4.3.2 Indicadores Internos

Nombre: Gestiona competencias.

Definición: Este indicador es para medir si el sistema permite evaluar las competencias por medio de los exámenes, y si el sistema permite hacer cambios en la evaluación de competencias para un mejor desempeño.

Forma de evaluar: Si – No.

Periodicidad: Diaria.

Valor Mínimo:

Valor Máximo:

Valor Deseado:

Nombre: Administrar competencias

Definición: Este indicador se mide si el sistema permite crear, modificar y eliminar competencias.
Forma de evaluar: Si – No.
Periodicidad: Diaria.
Valor Mínimo: 1 competencia.
Valor Máximo: 600 competencias.
Valor Deseado: 70%.

Nombre: Administrar indicadores
Definición: Este indicador sirve para medir si el sistema permite administrar los indicadores relacionados con las competencias.
Forma de evaluar: Si – No.
Periodicidad: Diaria.
Valor Mínimo: 1 indicador.
Valor Máximo: 1000 indicadores
Valor Deseado: 70%

Nombre: Administrar preguntas
Definición: Este indicador mide si el sistema permite administrar las preguntas asociadas a los indicadores que a su vez están relacionados con las competencias.
Forma de evaluar: Si – No.
Periodicidad: Diaria.
Valor Mínimo: 50 preguntas
Valor Máximo: 3000 preguntas
Valor Deseado: 60%

Nombre: Generación de reportes.

***Definición**:* Este indicador aparece pues una de las necesidades básicas del sistema de gestión de competencias es la generación de reportes el cual permita sacar datos estadísticos, que sirvan para el apoyo a la toma de decisiones.
Forma de evaluar: Si – No.
Periodicidad: Cada vez que se solicite un reporte.
Valor Mínimo: 1 Reporte.
Valor Máximo: 5 Reportes.
Valor Deseado: 3 Reportes.

Nombre: Realiza consultas.
Definición: Este indicador sirve para saber si el sistema permite realizar consultas a los administradores del sistema (Decano, Administradores de Línea) y a los docentes.
Forma de evaluar: Si – No.
Periodicidad: Semanal.
Valor Mínimo: 1 consulta.
Valor máximo: 5 Consultas.
Valor Deseado: 3 Consultas.

Nombre: Genera exámenes.
Definición: Este indicador permite saber si el administrador del sistema puede genera los exámenes solicitados por el equipo docente.
Forma de evaluar: Si – No.
Periodicidad: Cada vez que se solicite un parcial.
Valor Mínimo: 1 Examen.
Valor Máximo: Todos los exámenes solicitados por los docentes.
Valor Deseado: 5 exámenes.

Nombre: Muestra gráficas estadísticas.

Definición: Este indicador sirve para medir si el sistema es capaz de mostrar gráficas estadísticas en base a los datos arrojados por los estudiantes al realizar los exámenes.

Forma de evaluar: Si – No.

Periodicidad: Cada vez que se desee ver una gráfica estadística.

Valor Mínimo: 1 Gráfica.

Valor Máximo: Todas las gráficas solicitadas.

Valor Deseado: 5 Gráficas.

4.4 SELECCIÓN Y EVALUACIÓN DE PLATAFORMA TECNOLÓGICA

4.4.1 Sistemas operativos

Para realizar una comparación entre sistemas operativos se debe llevar a cabo un proceso delicado y ser totalmente imparcial; abarcando los requerimientos que la mayoría de los usuarios buscan en una plataforma. “Lo primero que es necesario preguntarse antes de empezar es: ¿cuales son las bases para la comparación?, la respuesta a esta pregunta ha sido dividida en dos aspectos: administrativo y técnico”[1].

4.4.1.1 Comparación a nivel administrativo

- **Esquema de licenciamiento.** El esquema de licenciamiento es el acuerdo que se firma entre las dos partes: desarrollador y usuario del software, que por lo general el usuario descuida limitándose a aceptar el acuerdo sin ni siquiera leerlo brevemente. Se encuentra relacionado con el precio ya que generalmente lo que se compra son licencias, Existen diversos esquemas de licenciamiento que varían entre cada compañía productora.

- ***FreeBSD*.** Este sistema operativo es libre, su kernel y gran cantidad de sus aplicaciones está licenciado bajo la licencia *BSD*, utiliza algunas de las

[1] PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - www.canapro.org.co, 2004.

herramientas generadas por el proyecto GNU, por lo cual algunas porciones de *FreeBSD* están licenciadas por la *GPL (General Public License)*. En cuanto al precio, *FreeBSD* se puede conseguir de forma gratuita descargándolo desde Internet.

- ***GNU*/Linux.** El sistema operativo *GNU*/Linux se encuentra liberado y protegido a su vez por *GPL* (*General Public License*) la cual ofrece las siguientes libertades básicas: Libertad para ejecutar el programa, con cualquier propósito, para modificar el programa y adaptarlo a sus necesidades, para redistribuir copias, para distribuir versiones modificadas del programa, de tal manera que la comunidad pueda beneficiarse con sus mejoras.

En cuanto al precio, *GNU*/Linux se encuentra disponible en Internet sin costo alguno, sin embargo incluso los propios fabricantes de las diferentes distribuciones o cualquier persona o empresa puede cobrar por el sistema.

- **Mac *OS* X.** El licenciamiento en el Mac *OS* X incluye un esquema tanto libre como propietario, este sistema tiene su base fundamental (*core*) en el sistema libre Darwin, el cual es liberado bajo la licencia *APSL* (*Apple Public Source License*), también tiene componentes propietarios, como por ejemplo la interfaz gráfica *Aqua* la cual fue introducida con la aparición de los iMac.

- ***NetBSD*.** Este sistema operativo es una variante de *BSD* y se rige por la licencia *BSD* esto implica que se garantice su libre redistribución para cualquier fin.

- ***OpenBSD*.** Este sistema operativo es otra variante *BSD*, al igual que los otros es software libre y se rige por la licencia *BSD*, aunque sujeto a los siguientes lineamientos: la redistribución debe mantener cualquier nota de Copyright del

original preservando así la propiedad intelectual, el software regido por esta licencia se entrega sin ningún tipo de garantía.

- **Windows 98.** El esquema de licenciamiento ofrece varias alternativas desde la licencia *OEM* para los computadores con este sistema preinstalado, hasta licencias de tipo *campus agrement* cuando se requiere licenciar un número significativo de máquinas.

- **Windows 2000.** Su esquema de licenciamiento es similar al de Win98, aunque para esta versión es aún un poco más complicado, ya que no sólo se debe tener en cuenta el número de máquinas a licenciar, sino también el número de procesadores en cada máquina. Windows 2000 tiene 3 versiones, cada una de estas versiones posee características que la hacen más adecuada a determinadas necesidades.

Tabla 1. Versiones de Windows 2000

Versión	***Descripción***	***Precio Unitario***
Professional	Es la versión para equipos de escritorio, provee únicamente funcionalidades básicas.	USD$ 319
Server	Provee funcionalidades de servidor de archivos e impresión, así como capacidades de servidor Web.	USD$ 999
Advanced Server	Además de las capacidades de Win2000 Server provee servicios de alta disponibilidad.	USD$ 3999

MICROSOFT, Versiones de Windows 2000, Citado en Marzo de 2004. Dirección Web: <www.microsoft.com/productos/win2000.html>

- **Windows XP.** Tiene el mismo licenciamiento de Windows 2000, Windows XP viene en dos versiones y obviamente con dos precios diferentes; *Home Edition*, que es una versión orientada al sector familiar, su precio es de USD$ 199, y

Professional, el cual incluye mayores capacidades de seguridad y está orientado al sector empresarial, su precio es de USD$ 299.

o **Estabilidad y desempeño.** Existen varias pruebas para medir el rendimiento de un sistema, por lo general se desempeñan en condiciones generadas, es decir se realizan en laboratorios y con máquinas especialmente puestas a punto para la comparación.

"Estabilidad hace referencia a las capacidades de la máquina para soportar la carga sin tener alteraciones que impliquen el bloqueo o reinicio de sistema"[1], lo cual se divide en la perdida de tiempo o en posibles perdidas de la información. El desempeño es la relación entre trabajo y tiempo, es decir la administración de los recursos del sistema de forma optima, logrando obtener una ejecución rápida de las aplicaciones pero sin afectar la estabilidad.

- **FreeBSD.** Según las estadísticas el mercado de servidores web *FreeBSD* es uno de los sistemas operativos más estables ya que, "los 50 sitios web que más tiempo han durado sin tener que reiniciarse están repartidos casi equitativamente entre máquinas con sistema operativo *FreeBSD* y *OpenBSD*"[2].

- **GNU/Linux.** Su rendimiento varía entre las distribuciones, pero también depende en gran medida del hardware sobre el que se está ejecutando, *GNU*/Linux en un sistema operativo bastante confiable tanto en rendimiento como en estabilidad. *GNU*/Linux es altamente estable, la mayoría de bloqueos ocurren en la maquina, más a causa de algún tipo de experimento que por inestabilidad del sistema.

- **Mac OS X.** "La estabilidad en los sistemas Macintosh ha sido altamente calificada a través de su historia, ahora teniendo como base otro gran sistema

[1] PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - http://www.canapro.org.co, 2004.
[2] CONSULTORA INDEPENDIENTE NETCRAFT - http://www.netcraft.com, 2004

operativo en términos de estabilidad, Apple ha logrado una muy buena combinación y este nuevo producto se considera ahora suficiente competencia incluso a sistemas como *GNU*/Linux o *FreeBSD* en términos de confiabilidad"[1].

- ***NetBSD*.** Se caracteriza por su limpieza en la codificación, a su vez es muy estable y tiene buen desempeño.

- ***OpenBSD*.** El principal propósito de *OpenBSD* es la seguridad, tiene la estabilidad de *FreeBSD*. se caracterizan por su altísimo tiempo sin necesidad de mantenimiento.

- **Windows 98.** La estabilidad es precisamente una de las grandes debilidades de este sistema.

- **Windows 2000**. Este es uno de los mejores sistemas Windows, integra el superior rendimiento y estabilidad de NT, su estabilidad especialmente en las versiones Server y Advanced Server puede llegar a compararse con la de algunos UNIX, hecho que le ha permitido incluso ganar una porción del mercado de los servidores en Internet.

- **Windows XP.** En sus primeras versiones tuvo varias deficiencias en cuanto a estabilidad, sin embargo en la actualidad ha logrado mejorar ampliamente en este aspecto aunque sin lograr superar a Win 2000.

- **Facilidad de uso.**

- ***FreeBSD, NetBSD, OpenBSD, GNU*/Linux**. En estos sistemas la interfaz primaria es la línea o intérprete de comandos, la cual les provee gran flexibilidad,

[1] PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - http://www.canapro.org.co, 2004.

pero esta puede llegar a ser complicada para algunos usuarios, en especial si están acostumbrados a trabajar exclusivamente con interfaces gráficas. sin embargo esto no es ningún impedimento para que los **BSD* y *GNU*/Linux puedan tener interfaces gráficas bastante amigables y que le permitan al usuario realizar algunas de las operaciones a las que tiene acceso por el interprete de comandos.

- **Soporte.** "Soporte es la ayuda que puede prestar una compañía o persona en caso de que existan fallos o incluso otro tipo de problemas"[1].

- ***FreeBSD, NetBSD, OpenBSD*.** En el mundo existen cientos de empresas que se dedican a este trabajo, en los sitios Web de cada uno de estos sistemas los desarrolladores recomiendan algunas de ellas. En Internet se encuentra gran documentación y tutoriales.

- ***GNU*/Linux.** En *GNU*/Linux el soporte se da de la misma forma que en los sistemas *BSD* y la mayoría de distribuciones que son desarrolladas por una empresa, poseen la opción de adquirirlas con soporte técnico.

- **Mac OS X.** Apple sólo ofrece soporte a través de Internet a sus usuarios registrados, aunque también mantiene gran cantidad de documentación para que pueda ser accedida libremente.

- **Windows 98, 2000, XP.** En estas plataformas existe muchísimas empresas que brindan soporte de forma independiente, también existen algunas que son contratadas o certificadas directamente por el fabricante, además esta empresa tiene representación en varios países del mundo.

4.4.1.2 Comparación a nivel técnico. Con la comparación a nivel técnico se toman algunos detalles que son muy importantes para el administrador de los recursos

[1] STALLING, William, Sistemas Operativos. Prentice Hall. 1997. Pág.60.

informáticos, los cuales permiten identificar cual ó cuales son los sistemas operativos que mejor se adaptan a nuestras necesidades.

- **Compatibilidad con otras plataformas.** "Cuando se habla de compatibilidad con otras plataformas se deben tener en cuenta diferentes conceptos"[1]: *Soporte a Sistemas de Archivos Diferentes al Nativo*: Es la capacidad que tiene una plataforma de leer y/o escribir en sistemas de archivos de otras plataformas diferentes; *Ejecución de Aplicaciones Compiladas para otras Plataformas*: Es cuando los sistemas operativos son capaces de ejecutar aplicaciones que fueron originalmente diseñadas para trabajar en una plataforma diferente, la mayoría de las veces esto se lleva a cabo por intermedio de otra aplicación que actúa como una capa que permite emular las llamadas al sistema y algunos otros procesos del sistema operativo para que el programa crea que se está comunicando con la plataforma para la cual fue compilado; *Compartir Recursos en Red Con Otras Plataformas*: Cuando el sistema operativo es capaz de comunicarse con sistemas diferentes en otras máquinas, y obviamente brindarles la posibilidad de utilizar sus recursos.

- ***FreeBSD, NetBSD, OpenBSD*.** Estos sistema operativos poseen soporte para lectura y escritura de los siguientes sistemas de archivos: *FAT16, FAT32, ext2, ext3 y NTFS* para sólo lectura. Poseen compatibilidad binaria para ejecutar aplicaciones de GNU/Linux, **BSD* y *SCO*; también es posible ejecutar aplicaciones DOS y Windows. Para poder compartir recursos con otros UNIX, se utiliza el protocolo *NFS* (*Network File System*), mientras que para compartir con Windows el protocolo *SMB* (*Samba*).

- ***GNU*/Linux[2]**. Posee soporte para los siguientes tipos de sistemas de archivos: *JFS* (*Journaled File System*), *XFS, Minix FS, FAT 16, FAT 32, NTFS,*

[1] PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - www.canapro.org.co, 2004.
[2] CARRETO PEREZ, Jesús. Sistemas Operativos una Visión Aplicada. Mc GrawHill, 2001. Pág.613.

ADFS, BeFS (*File System BeOS*), *BFS* (*Unixware Boot Filesystem*), *EFS, HPFS*. Puede ejecutar algunas aplicaciones creadas para UNIX y aplicaciones diseñadas para sistemas Windows. Tiene compatibilidad en red con otras plataformas por medio de los protocolos *NFS, SMB* (*Samba*), *NCP* (*NetWare Core Protocol*).

- **Mac OS X**. Tiene la posibilidad de soportar binarios de UNIX, en especial los del tipo *BSD* y Linux, pero como además tiene componentes Mac puede soportar también aplicaciones hechas para versiones anteriores de ese sistema, Puede ejecutar aplicaciones Windows el por medio de el Virtual PC.

- **Windows 98.** Soporta aplicaciones para DOS y Windows, se tienen la posibilidad de ejecutar binarios para Mac utilizando software de terceros. Soporta únicamente sistemas de archivos *FAT 16* y *FAT 32*. Comparte archivos e impresoras con otros Windows utilizando de forma nativa el protocolo *SMB* (*NetBIOS*), puede trabajar en redes Novell Netware.

- **Windows 2000 y XP.** Solo soporta sistemas de archivos *FAT 32* y *NTFS*. Compartir archivos por *SMB*, aunque es posible compartir por medio de *NFS*, al igual que Win98 también se puede compartir con *Novell Netware*.

- **Portabilidad.** "Cuando en sistemas operativos se habla de portabilidad esto hace referencia a la posibilidad de utilizarlos en diferentes tipos de procesadores incluso si tienen arquitecturas o diseños diferentes"[1].

- ***FreeBSD, OpenBSD, GNU*/Linux.** Estos sistemas operativos se pueden ejecutar en las plataformas Intel 386, 486, Pentium, P II, PIII, PIV, y en un buen número más de plataformas como por ejemplo Intel ia64, Sparc, UltraSparc, PPC (Macintosh), Alpha y Alpha64.

---

[1] PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - www.canapro.org.co, 2004.

- **Mac*OS* X.** Este sistema operativo está diseñado para la arquitectura típica del Mac, es decir los procesadores Power Mac.

- ***NetBSD*.** Este sistema operativo se puede ejecutare en más de 50 diferentes procesadores que van desde los grandes servidores Sparc, Alpha, hasta pasar por los de escritorio Intel, *PPC*, e incluso en dispositivos.

- **Windows.** Los sistemas Windows únicamente se ejecutan sobre arquitecturas Intel, sin embargo existe una versión modificada denominada Windows CE, orientada al mercado móvil.

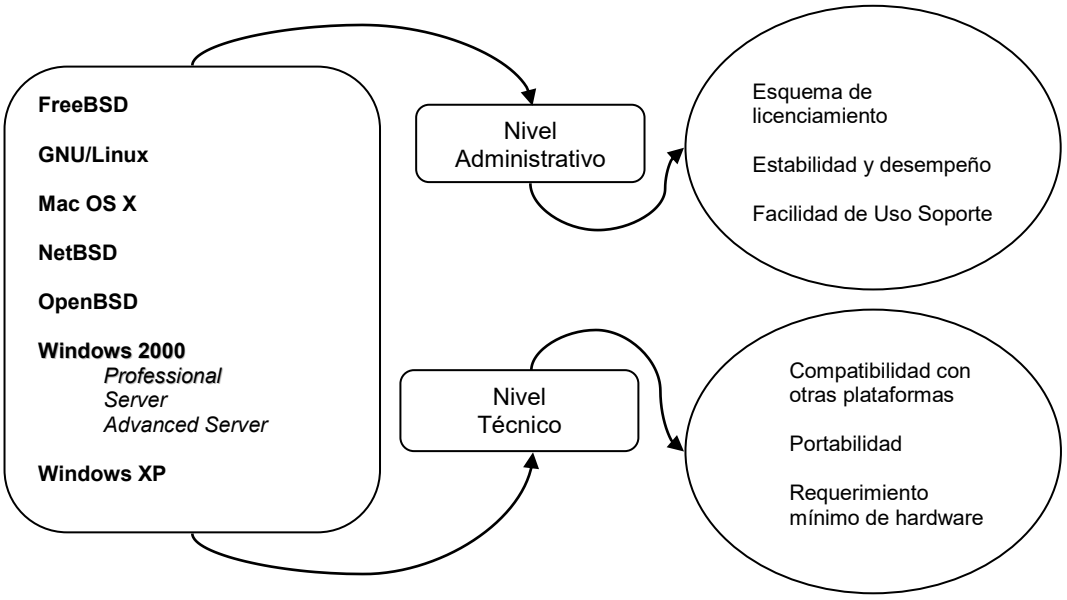
- **Requerimientos mínimos de hardware.** Estos son los principales requerimientos mínimos de hardware para cada uno de los sistemas operativos anteriores, todos sobre la plataforma x86, excepto en el caso del Mac *OS* X, donde se muestra sobre el procesador Power PC.

Tabla 2. Requerimientos Hardware Para Sistemas Operativos

S.O.	Procesador Mínimo	Mínimo De Ram	Espacio Mín. En Disco Duro
FREEBSD	386	16 Mb para instalación, se reduce hasta 4 Mb.	10Mb a 50Mb
GNU/LINUX	386	4 Mb	100 Mb
MAC OS X	PowerPC G3	128 Mb	2 Gb
NETBSD	386	4 Mb	50 Mb
OPENBSD	386	8 Mb	50 Mb
WINDOWS 98	486	16 Mb	165 Mb
WIN 2000 PROFESSIONAL	Pentium 133 Mhz	64 Mb	650 Mb
WIN 2000 SERVER / ADVANCED S.	Pentium 133 Mhz	128 Mb	1 Gb
WINDOWS XP HOME E. / PROF.	Pentium 233 Mhz	128 Mb	1.5 Gb

PAC HÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - www.canapro.org.co/~wilfred_com/s.o/sistemas_operativos/sistemas_operativos-node17.html, 2004.

Figura 21. Criterios de evaluación de sistemas operativos.



GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas UNAB.

4.4.2 Plataformas tecnológicas de desarrollo

Las plataformas tecnológicas son fundamentales en la definición de los sistemas de información. En la actualidad, existen dos plataformas estándares: *J2EE* y Microsoft .*NET*.

Los directores de los departamentos son los que deben elegir una plataforma tecnológica que sea eficiente y que proporcione a las aplicaciones una independencia respecto a las evoluciones técnicas futuras. Antes de escoger una plataforma es muy importante tener en cuenta cuales son los objetivos y las necesidades de la organización así como también los límites presupuestarios, de plazos y de riesgos aceptados por la misma.

En el momento de decidir cual plataforma se implementara es necesario tener en cuenta las plataformas hardware y sistemas operativos que soportarán las aplicaciones. La elección también de ser determinada por el conocimiento y la cultura existente en la organización, como por ejemplo, conocimiento sobre Java, uso de productos Microsoft, etc. Además, influirá también en la decisión la necesidad de integrar con las aplicaciones ya existentes en la organización, así como la complejidad de la aplicación a construir y el tamaño del proyecto.

- **Java (*J2EE*).** Esta plataforma es un estándar para el desarrollo de aplicaciones empresariales multicapa, en base al cual se pueden adquirir productos que proporcionan servicios añadidos, desarrollados por diversos proveedores, tales como *BEA WebLogic, IBM WebSphere, Oracle9iAS, Sun One*, etc. Estos productos utilizan una especificación de la máquina virtual (*VM*) de Java, la cual garantiza que cualquier programa de Java se pueda ejecutar en una plataforma que tenga la *VM* implementada.

J2EE se orienta al uso de un único lenguaje que soporta, que es Java y a la integración, que consiste en acceder a la plataforma con otros lenguajes, esto se logra a través de tecnología de interfaz *JNI* (*Java Native Interface*), o por medio de la interoperabilidad que ofrecer *CORBA*.

Con un uso estricto de J2EE es posible construir aplicaciones por medio de un IDE de un proveedor y su puesta en producción en un servidor de aplicaciones licenciado por otro proveedor diferente.

- **Microsoft .NET.** "Es un entorno estructurado alrededor de .*NET* Framework y de los servidores .Net. El .*NET* Framework integra la máquina virtual *CLR* (*Common Language Runtime*) disponible únicamente en entorno Windows"[1]. Por tanto su objetivo principal no es la interoperabilidad a través de plataformas heterogéneas. .NET permite el uso de los diferentes lenguajes incluidos en Visual Studio .Net (Visual Basic.NET, C++.NET y C# que es equivalente a Java con la excepción de portabilidad). Están también disponibles compiladores para *Cobol, Visual Perl, Visual Phyton y Delphi o Eiffel*.

"Visual Studio .Net Dispone de un *IDE* (*Entorno Integrado de Desarrollo*), además es estable, maduro, potente y permite la creación de aplicaciones y su rápida depuración"[2]. De esta forma, se permite la creación de interfaces para clientes Win32 (*WinForms*) o con interface Web (*Web Forms*) a través de páginas ASP.NET.

Microsoft .*NET* va más allá de soportar estos lenguajes, también ofrece plena interoperabilidad entre ellos, por lo que es posible construir un componente en un lenguaje, introducirlo en una aplicación escrita en otro distinto e incluso heredarlo

[1] PLATT, David S. Introducing Microsoft .Net. Microsoft Press, 2001. Pág. 16.
[2] Lbid. Pág.. 23

y añadir nuevas características en un tercero. Respecto a esta capacidad de Microsoft *.NET* de trabajar con varios lenguajes existe cosas a favor y en contra:

- La capacidad de una migración más sencilla para antiguos programadores, reduciendo el tiempo de formación. Además, trabajar con un lenguaje conocido proporciona gran productividad individual.

- La existencia de varios lenguajes de programación en una única empresa acarrea efectos negativos.

- La complejidad de mantenimiento aumenta. Si una aplicación está realizada en varios lenguajes se necesitan expertos en varios lenguajes, aumentando los costes considerablemente.

- La productividad del grupo decrece. No se pueden comunicar fácilmente los conocimientos de unos a otros.

- Transferencia de conocimientos. Si se deja un proyecto, es necesario conocer el mismo lenguaje para continuar con el desarrollo.

Es importante que una empresa dedicada al desarrollo de software sigua un criterio homogéneo y realice todos sus desarrollos utilizando un único lenguaje, ya sea Java, C# o cualquier otro, independientemente de la plataforma utilizada.

4.4.2.1 Comparación de plataformas tecnológicas de desarrollo. **"***J2EE* y *.NET* tienen aspectos muy parecidos; los principales son"[1]:

[1] TORRES, Luis; Comparativa de Plataformas Tecnológicas de Desarrollo, Citado en Junio de 2004 Dirección Web:<www.ciberteca.net/articulos/programacion/net/ lenguajes .asp>

- Un lenguaje de programación unido a una librería de clases, que compila a un código intermedio independiente de la máquina (*IL*) en Microsoft *.NET* y (*Bytecodes*) en Java.
- El código se ejecuta en una máquina virtual, que proporciona el entorno de ejecución el cual transformará el lenguaje intermedio a código propio de la máquina en la que se corre la aplicación, *CLR* (*Common Language Runtime*) en Microsoft *.NET* y *JRE* (*Java Runtime Environment*) en *J2EE*.

- Las dos plataformas disponen de multitud de servicios para facilitar el trabajo del programador, como por ejemplo, el acceso a bases de datos, los servicios de directorio, las transacciones distribuidas, etc.

- Durante el proceso de construcción, comparten arquitectura multicapa, integran componentes funcionales interoperables y autónomos, generan aplicaciones web basadas en clientes ligeros y se caracterizan por una alta sensibilización en la reutilización de componentes para mejorar la mantenibilidad del software y acortar el tiempo de construcción.

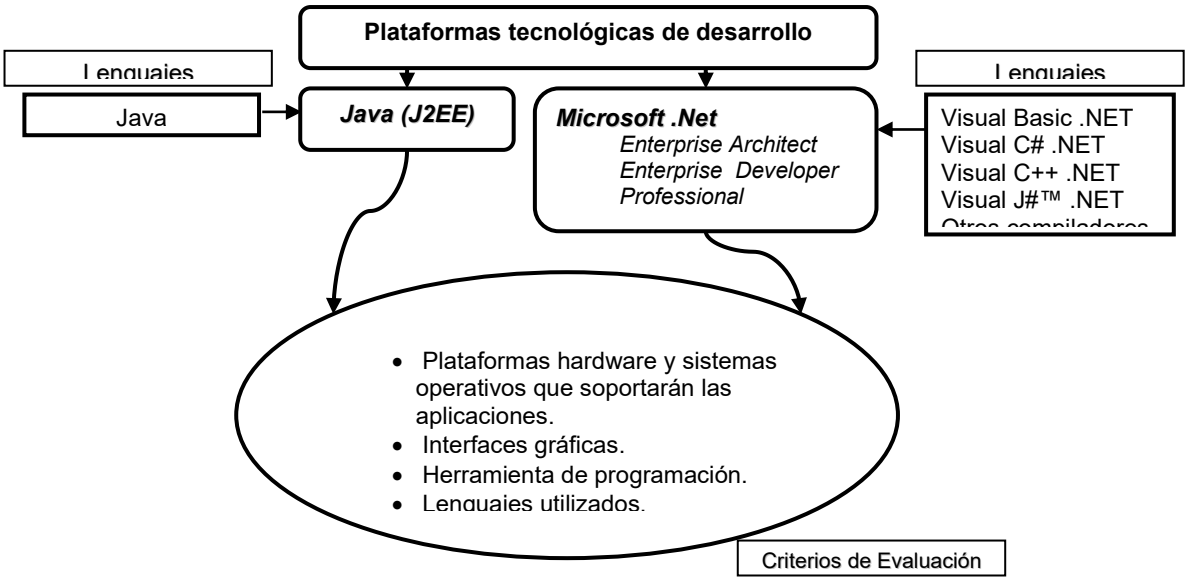
- Ambas plataformas contienen múltiples posibilidades arquitectónicas, por lo que es conveniente disponer de una amplia experiencia y conocimiento para tomar las decisiones adecuadas en la construcción de estas aplicaciones; También es importante utilizar ciclos de desarrollo iterativo donde la concepción y diseño de la aplicación se realicen mediante métodos como *UML* (*Unified Modeling Language*) y metodologías específicas que aseguren la concepción e integración de componentes.

Tabla 3. Comparativa Plataformas tecnológicas de desarrollo

Característica	***.NET***	***J2EE***
Tipo de tecnología	Producto	Estándar
Empresas que lo ofrecen	Microsoft	Más de 30
Librerías de desarrollo	.NET Framework SDK	Java core API
Intérprete	CLR	JRE
Páginas dinámicas	ASP.NET	Servlets, JSP
Componentes	.NET Managed Components	EJB
Acceso a bases de datos	ADO.NET	JDBC, SQL/J
Servicios Web	SOAP, WDSL, UDDI	SOAP, WDSL, UDDI
Interfaces gráficas	Win Forms y Web Forms	Java Swing
Herramienta de programación	Visual Studio .NET	Dependiente del fabricante
Transacciones distribuidas	MS-DTC	JTS
Servicios de directorios	ADSI	JNDI
Lenguajes utilizados	C#, Visual Basic, C++, otros	Java
Lenguaje intermedio	IL	Bytecodes

TORRES, Luis; Comparativa de Plataformas Tecnológicas de Desarrollo, Citado en Junio de 2004 Dirección Web:<www.ciberteca.net/articulos/programacion/net/ lenguajes .asp>

Figura 22. Criterios de evaluación de la plataforma tecnológica de desarrollo



ACEVEDO MAZA, Juan Francisco. Estudiante de Ingeniería de Sistemas UNAB.

4.4.3 Bases de datos

"Una base de datos es un sistema formado por un conjunto de datos organizados de forma que se controla el almacenamiento de datos y son independientes de los programas que los usan"[1]. El acceso a los datos se hace de diversas formas y debe cumplir los siguientes requisitos:

- Los usuarios pueden acceder simultáneamente y cada uno tendrá acceso a una información particular.
- Se controlará el acceso de todos los usuarios asegurando la confidencialidad y la seguridad.
- Los datos se almacenan sin redundancia, excepto excepcionalidades.
- Se podrán usar distintas formas de acceso, asegurándose la flexibilidad en las búsquedas.
- Tienen mecanismos concretos de recuperación de información en caso de fallo de la computadora.
- El cambio del soporte físico no repercutirá en los programas, ni en la base de datos.
- Se podrán modificar los datos y las relaciones entre ellos sin afectar a los programas.
- Incorporarán una interfaz de usuario que permita usarla de forma cómoda y sencilla.

Se dispone de varias alternativas, por lo cual se deben conocer las peculiaridades que cada uno de estos sistemas tiene y que lo convierten en el más adecuado según los intereses del programador.

- **Bases de datos Access**. Para que la aplicación sobre base de datos Access no tenga problemas, es recomendable que cumpla estas condiciones:

[1] DATE, C.J, Introducción a las Bases de Datos. Addison Wesley Iberoamericano, 2001. Pág. 5

- El volumen de datos a manejar sea pequeño.
- El número de usuarios simultáneos no sea muy alto.

Cuando por alguno de los motivos anteriores, la aplicación no es consistente, o no va a poder cumplir alguno de los mismos, se recomienda el uso de un sistema de base de datos más robusto, como *SQL Server*.

o **Bases de datos *MS SQL Server***. *SQL Server* es un sistema de bases de datos completo y potente, el cual es ideal para los programadores especializados en productos Microsoft: *ASP*, Visual Basic, etc. Además, es un sistema de base de datos perfectamente adecuado para aplicaciones críticas y con cualquier grado de complejidad. Ofrece otras características avanzadas orientadas a mantener la integridad de la base de datos, como son los triggers.

o **Bases de datos *MYSQL***. *MySQL* tiene como principales características su velocidad y su precio reducido. Es un servidor de bases de datos muy utilizado en aplicaciones *PHP* o Perl en servidores Linux. En general, si no necesita características como transacciones, procedimientos almacenados, *triggers* o sentencias *SQL* complejas, *MySQL* cumplirá la misma función que otras bases de datos más potentes, pero de forma más rápida y con un coste menor. Sirve ara aplicaciones Windows, las limitaciones de *MySQL* son dadas por sus carencias respecto de los otros sistemas de bases de datos y por el grado de criticidad de la aplicación, *MySQL* no es adecuada para aplicaciones críticas. No tiene manejo de triggers por lo que no se pueden establecer reglas de integridad y consistencia a nivel de servidor.

o **Bases de datos *PostgreSQL*.** *PostgreSQL* es un servidor de bases de datos de código abierto más potente que existe y es una alternativa a *MySQL* cuando se necesitan características avanzadas como transacciones, procedimientos almacenados, *triggers*, vistas, etc. Este servidor de bases de datos es utilizado por

los programadores de *servlets* de Java y, en general, por todos aquellos que realizan aplicaciones cliente servidor complejas en Linux/Unix. Para aplicaciones Windows, *PostgreSQL* tiene similares prestaciones a las de SQL Server. La mayor limitación de *PostgreSQL* viene dada por su velocidad ya que es un sistema de bases de datos lento.

Tabla 4. Comparativa de Bases de datos

Criterios	**Bases de datos**			
	Access	**SQL Server**	**MySQL**	**PostgreSQL**
Plataforma	Windows	Windows	Windows / Linux	Windows / Linux
Velocidad	Bajo	Alto	Alto	Bajo
Volumen Datos	Bajo	Alto	Alto	Alto
Integridad	Bajo	Alto	Bajo	Alto
Potencia	Bajo	Alto	Alto	Alto
Coste/MB	Alto	Bajo	Alto	Alto

ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

4.4.4 Herramientas de diseño orientado a objetos

Las herramientas *Case* están diseñadas buscando un aumento en la productividad, apoyando el desarrollo de sistemas de información y logrando una comunicación más efectiva con los usuarios, integrando el trabajo que se realiza en el sistema, desde el principio hasta el fin del ciclo de vida; además ayudan a identificar y articular todos los requerimientos de los usuarios de manera concisa y apuntando al objetivo de lograr sistemas que satisfagan las necesidades planteadas durante las etapas de análisis y diseño; obteniendo sistemas que requieran menos mantenimiento y generen mayor calidad y productividad.

Tabla 5. Herramientas de diseño orientado a objetos

	ObjectTeam	Paradigm Plus	Racional Rose	Select Enterprise 5.1	System Architect 4.0
Contacto	www.cayennesoft.com	www.platinum.com	www.rational.com	www.selectst.com	www.popkin.com
Metodología	OMT + Use Cases + Event trace, Object Model habilitado + UML,Q4/97	OMT + Use Cases, Booch Rumbaugh, UML	Booch, OMT, Especializada en UML.	UML integrado a LogicWorks ERWIN, CORBA	UML, Booch 94, OMT, Gane/Sarson, Schaler/Mellor
Plantillas de Clases	Capaz de hacer ingeniería hacia delante y hacia atrás usando plantillas de clases.	n/e	Extiende OMT básico representando las plantillas elegantemente	Usa OMT en combinación con Jacobson Use Cases. Soporte de Paquetes y combinación de diagramas.	n/e
Ingeniería a la Inversa	Mas fácil que Rose, menos opciones GUI, mas personalizable. Solo genera diagramas de herencia del proceso de ingeniería de reverso.	Los dibujos están desordenados.	Fácil, muy sofisticado y personalizable.	Integración Bidireccional	Ingeniería de reverso a través de SQL.
Interface de Usuario	Edita clases en diagramas. Buen browser de clases personalizable.	Difícil de usar, Atributos pobres de Browser.	Usa paradigma de interface de documentos múltiple, Edita clases vía dialogo, fácil de hacer diagramas buenos en forma equitativa, buen browser de clases.	Integra a Microsoft Word para la documentación del diseño, permite explorar diagramas secuénciales de manera interactiva.	Integrado a la generación de código, las pantallas se cargan a los proyectos.
Desempeño	Buen desempeño bajo UNIX.	Lento	Sin fallas en el desempeño, muy rápido, lento en ingeniería de reverso.	Buen desempeño y rapidez	Lento con varios problemas.
Generación de código.	Buena	n/e	Buena	Buena	n/e

GÓMEZ DEL MORAL, Diego. CONSUEGRA, Martín. Estudio De Diversas Herramientas UML Para El Desarrollo De Software.

- **Rational Rose.** Es una herramienta *CASE* que soporta de forma completa la especificación de *UML*; Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo completo que representa el dominio del problema y el sistema de software.

Rational Rose utiliza un proceso de desarrollo iterativo controlado, donde el desarrollo se lleva a cabo en una secuencia de iteraciones. También permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

Rational Rose puede generar código en distintos lenguajes de programación a partir de un diseño en *UML*. Además proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

4.4.5 Herramienta de programación

Visual Studio .NET. "Microsoft Visual Studio .NET, es una herramienta práctica, que los desarrolladores pueden usar para: crear una nueva generación de Internet, crear aplicaciones poderosas de manera rápida, efectiva y para expandirse a cualquier plataforma o dispositivo"[1].

[1] FOXALL, James. Practical Standars For Microsoft Visual Basic .Net. Microsoft Press, 2001. Pág. 12.

Visual Studio *.NET* tiene un ambiente de programación único creado exclusivamente para servicios Web *XML*, los servicios Web *XML* permiten que los desarrolladores ensamblen aplicaciones con código existente y/o nuevo, independientemente de la plataforma, lenguaje de programación o de el modelo de objetos, permitiendo que las aplicaciones se comuniquen y compartan datos a través de Internet; esta disponible en 3 Ediciones[1].

- ***Enterprise Architect***: permite aprovechar la herramienta de programación líder de la industria y crear una guía de arquitectura para los equipos de programación.

- ***Enterprise Developer***: proporciona una plataforma de programación empresarial en equipo eficaz para generar con rapidez aplicaciones y servicios Web *XML*.

- ***Professional:*** permite generar con rapidez aplicaciones y servicios Web *XML* de próxima generación, pensados para cualquier dispositivo compatible con Internet y para integrarse con otros lenguajes de programación y sistemas operativos.

Las tres ediciones de Visual Studio.*NET* contienen los siguientes productos:

[1] MICROSOFT, Herramientas de Desarrollo, Citado en Junio de 2004. Dirección Web: <www.microsoft.com/latam/vstudio/producto/resumen.asp>

Tabla 6. Lenguajes de Programación Visual Studio *.NET*

Producto	Característica
Visual Basic .NET	Visual Basic .NET permite una programación completamente orientada a objetos con herencia de implementación, manejo estructurado de excepciones y subprocesamiento libre.
Visual C# .NET	Visual C# .NET, un lenguaje de programación orientado a componentes que es inmediatamente familiar a los desarrolladores de C++ y Java, es ideal para crear soluciones con acceso a datos, marcos de trabajo para negocios y software comercial.
Visual C++ .NET	Visual C++ .NET permite que los desarrolladores creen aplicaciones administradas y no-administradas, usando el .NET Framework, Active Template Library (*ATL*) Server y Microsoft Foundation Classes (*MFC*).
Microsoft Visual J#™ .NET	Visual J# .NET es una herramienta de desarrollo para programadores del lenguaje Java que quieren crear aplicaciones y servicios en el .NET Framework.

MICROSOFT, Herramientas de Desarrollo, Citado en Junio de 2004. Dirección Web: <www.microsoft.com/latam/vstudio/producto/resumen.asp>

- **Visual Basic .NET**. Es una herramienta productiva para la creación de aplicaciones que se ejecutan en el sistema operativo Microsoft Windows®. Con Visual Basic *.NET*, los programadores pueden seguir aprovechando sus conocimientos y capacidades para crear la próxima generación de aplicaciones y servicios Web *XML*. Visual Basic.

- **Características de Visual Basic .NET**.

- Permite Crear Aplicaciones Interactivas Para Windows: Los programadores pueden crear aplicaciones para Windows por medio de Windows Forms, aprovechando todas las características de la interfaz de usuario del sistema operativo Windows. Visual Basic *.NET* incluye todas las herramientas de programación rápida de aplicaciones como la creación, arrastrando y colocando, además de aplicaciones para Windows que aprovechan totalmente las bases de datos y los servicios Web *XML*.

- Permite Crear Aplicaciones y Servicios Web *XML*: Con el Diseñador de Web Forms y el Diseñador *XML*, se pueden utilizar las características de Microsoft IntelliSense® y la capacidad para completar etiquetas; o bien, poder elegir el editor *WYSIWYG* en donde lo que ve es lo que se imprime para crear aplicaciones Web interactivas arrastrando y colocando elementos.

- Permite Crear Aplicaciones Web Móviles: Visual Basic *.NET* ofrece características de Internet móvil que permiten a los programadores crear una interfaz Web móvil única, compatible con una amplia gama de dispositivo como *WML* 1.1 para teléfonos móviles *WAP*, *HTML* compacto (*cHTML*) para teléfonos i-mode y HTML para Pocket PC, dispositivos de mano y localizadores (*pagers*). Los controles móviles de servidor generan de manera inteligente la reproducción y paginación apropiada para cada dispositivo Web, proporcionando una experiencia completa y coherente al usuario a la vez que mantiene la flexibilidad del programador.

- Centro de Control Más Avanzado Para los Programadores: La página de inicio de Visual Basic *.NET* es un portal para programadores que permite tener acceso con un solo clic a información acerca de los proyectos usados recientemente, las preferencias personales, las actualizaciones de productos y la comunidad *MSDN*

Online. El Cuadro de herramientas ampliado muestra una vista dinámica de los componentes del proyecto, como los controles de Windows Forms y Web Forms, los elementos *HTML*, los objetos y los miniprogramas.

- Plantillas y Asistentes que Permiten Ahorrar Tiempo: La plantilla de servicios Web *XML* crea e implementa automáticamente los diversos componentes de un servicio Web. El Asistente para la instalación permite distribuir las aplicaciones *.NET* de forma sencilla.

- Ayuda Disponible al Instante: La Ayuda dinámica proporciona acceso con un solo clic a la ayuda pertinente, independientemente de la tarea que se esté realizando. *MSDN* Online Access proporciona vínculos directos a ejemplos, grupos de noticias, actualizaciones y descargas de Visual Basic *.NET* en el entorno de desarrollo integrado (*IDE*).

- Lenguaje Sencillo y Popular: Es un lenguaje de programación fácil de leer y de escribir. La compilación en segundo plano proporciona información al instante y señala los errores con un subrayado ondulante.

- Funciones de Programación Ampliadas: La implementación lado a lado acaba con los conflictos entre versiones y la herencia permite reutilizar el código de cualquier lenguaje basado en *.NET*. El Control de excepciones estructurado proporciona un código de control de errores más elegante y fácil de mantener.

- El lenguaje de Visual Basic *.NET* se ha actualizado, simplificado y modernizado, ahora tiene acceso a un conjunto de herramientas mucho más completas y eficaces que en las versiones anteriores de Visual Basic. Para satisfacer la fuerte demanda de los clientes. Visual Basic *.NET* ofrece un amplio conjunto de nuevas características, como capacidades de diseño completamente orientado a objetos, subprocesamiento libre y acceso directo a Microsoft *.NET*

Framework. Asimismo, el lenguaje de Visual Basic se ha optimizado, eliminando palabras clave obsoletas que se habían heredado, mejorando la seguridad de tipos y revelando las construcciones de bajo nivel.

- Se integra plenamente con otros lenguajes de Microsoft Visual Studio® .*NET*.

Se pueden programar componentes de aplicaciones en diferentes lenguajes de programación, las clases se pueden heredar de clases escritas en otros lenguajes utilizando la herencia entre lenguajes. Con el depurador unificado, ahora se pueden depurar aplicaciones en varios lenguajes, independientemente de si se ejecutan localmente o en equipos remotos, .*NET* Framework proporciona un amplio conjunto de interfaces de programación de aplicaciones (*API*) para el sistema operativo Microsoft Windows e Internet.

Tabla 7. Requerimientos mínimos del sistema para Visual Basic .*NET*

Procesador	Computadora personal (PC) con un procesador Pentium II, 450 (MHz).
Sistema Operativo	Microsoft Windows® XP Professional Microsoft Windows 2000 Professional Microsoft Windows 2000 Server Microsoft Windows NT® 4.0 Workstation Microsoft Windows NT 4.0 Server
Memoria	**Microsoft Windows XP Professional** 160 MB de RAM; se recomiendan 192 MB **Windows 2000 Professional** 96 MB de RAM; se recomiendan 128 MB **Windows 2000 Server** 192 MB de RAM; se recomiendan 256 MB
Disco Duro	500 MB en el disco de sistema, 2.0 Giga bites (GB) en el disco a instalarse
video	Monitor Súper VGA (800 x 600) o superior a 256 colores

MICROSOFT, Herramientas de Desarrollo, Citado en Junio de 2004. Dirección Web: www.microsoft.com/latam/vbasic/producto/requerimientos.asp

Después de haber estudiado y evaluado todos los aspectos técnicos con respecto a la plataforma tecnológica se llegó a la conclusión de que para realizar el sistema de la manera ideal se debe utilizar el siguiente esquema de desarrollo:

Como sistema operativo se eligió Windows 2000 Server ya que este es uno de los mejores sistemas Windows, tiene buen rendimiento y estabilidad, provee funcionalidades de servidor de archivos y capacidades de servidor web.

Aunque *J2EE* y *.NET* tienen aspectos muy parecidos, se selecciono como plataforma tecnológica de desarrollo a *.NET* por que no se cierra a un solo lenguaje como lo hace *J2EE*, permitiendo el uso de los diferentes lenguajes incluidos en Visual Studio .Net como por ejemplo Visual Basic.*NET*, *C++*.*NET*, *C#* que es equivalente a Java y *J#* para programadores java que quieren crear aplicaciones en .*NET*.

SQL Server es un sistema de bases de datos que se incluyó en el esquema ideal para la implementación del sistema, ya que es muy completo y potente, además porque es ideal para los productos Microsoft, Visual Basic, *ASP*, etc. Es un sistema de bases de datos superior en todo sentido en comparación con los demás como podemos ver en la Tabla 4. (Pág. 85).

Se utilizó *Rational Rose*, ya que posee buenas características como se muestra en el cuadro comparativo de herramientas *CASE* (Página 88), además porque fue desarrollada por los creadores de *UML* (*Booch, Rumbaugh y Jacobson*) siendo esta una herramienta especializada en esta metodología sobre la cual se modelo el sistema, además cubre todo el ciclo de vida de el proyecto; permitiendo que los

usuarios tracen y modifiquen diagramas fácilmente, reduciendo el tiempo que se gasta normalmente en el trazo manual de los diagramas.

Tabla 8. Plataforma Tecnológica Ideal para la implementación del sistema.

Sistema operativo	Windows 2000 Server.
Plataforma Tecnológica de Desarrollo	.NET
Base de Datos	SQL Server.
Herramienta de Diseño Orientado a Objetos	Rational Rose.
Herramienta de Programación	Visual Studio .NET Developer

GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas UNAB.

Debido a algunas restricciones tales como el periodo académico, fechas de entregas de documentación proyecto, entrega de prototipo, adquisición de software, entre otros, se decidió utilizar el siguiente esquema de trabajo, que hace posible una rápida implementación del modulo del sistema.

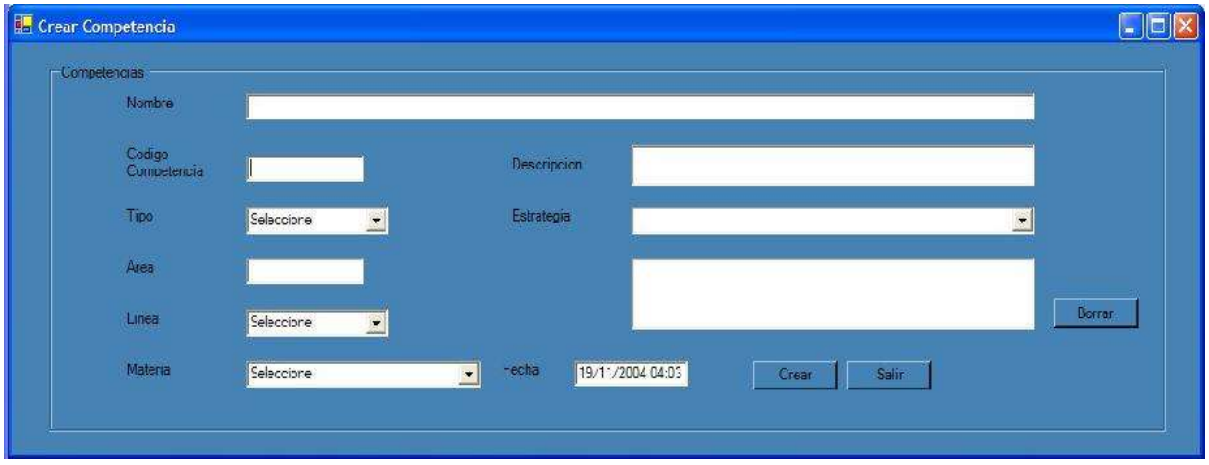
Tabla 9. Plataforma Tecnológica más viable para la implementación del modulo del sistema.

Sistema operativo	Windows 2000, XP.
Plataforma Tecnológica de Desarrollo	.NET
Base de Datos	Microsoft Access.
Herramienta de Diseño Orientado a Objetos	Rational Rose.
Herramienta de Programación	Visual Studio .NET Professional

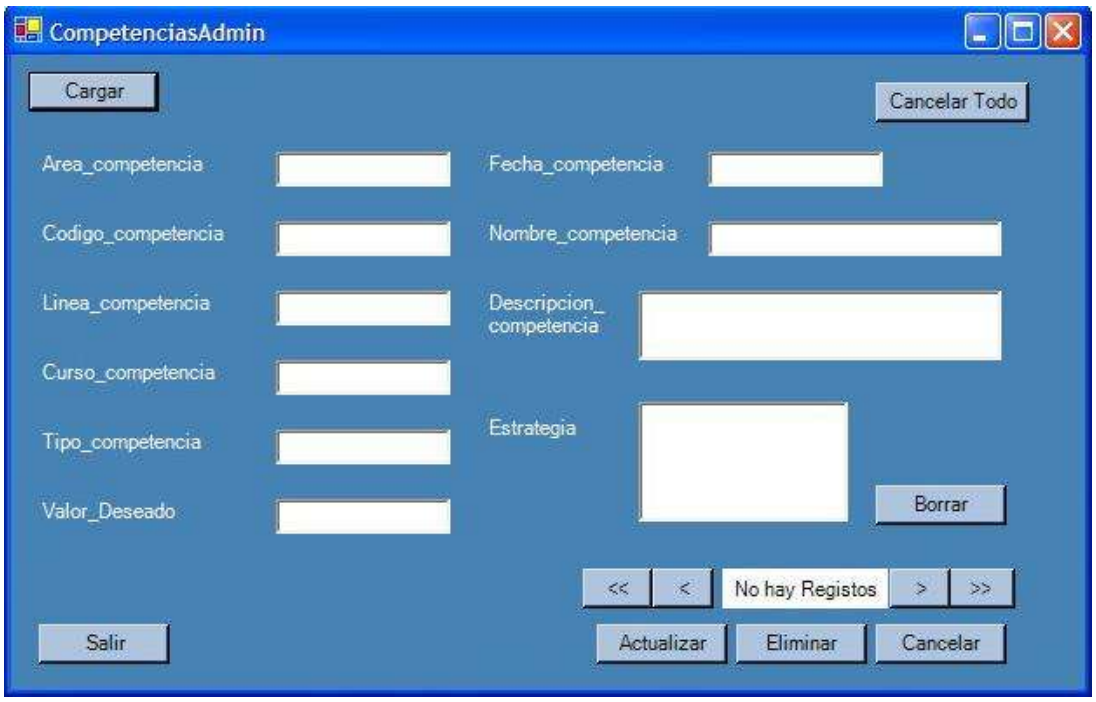
ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

5. IMPLEMENTACIÓN DE UN MODULO DEL SISTEMA

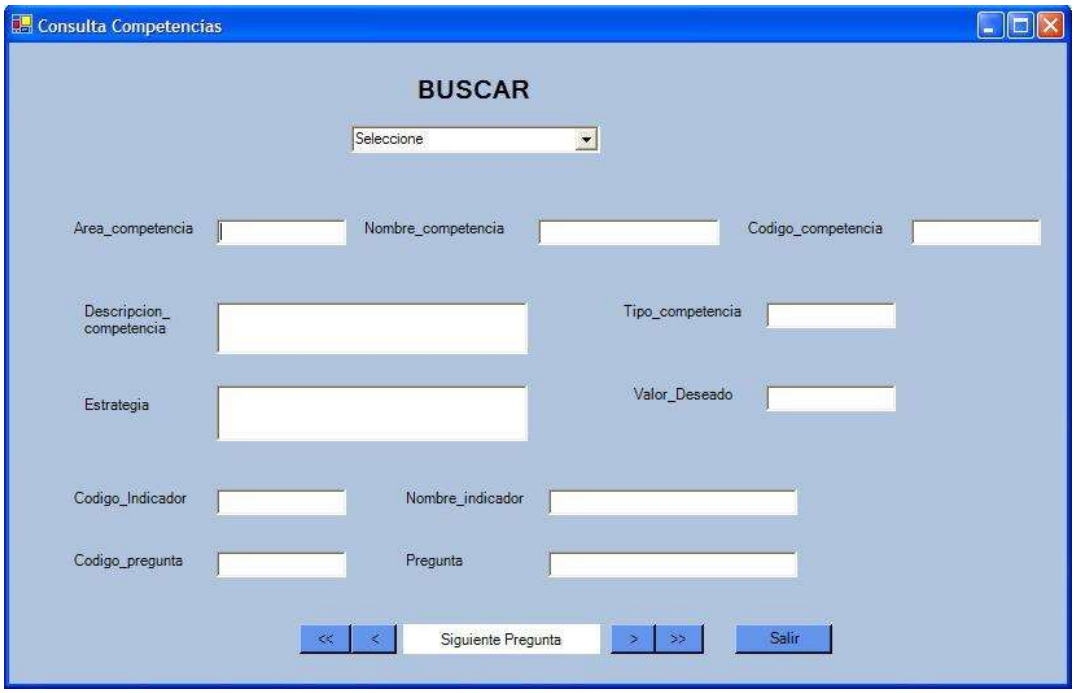
En esta ventana, se realiza el acceso de los usuarios y del administrador del sistema; dependiendo del tipo de usuario se activan los campos y las ventanas que podrá utilizar.



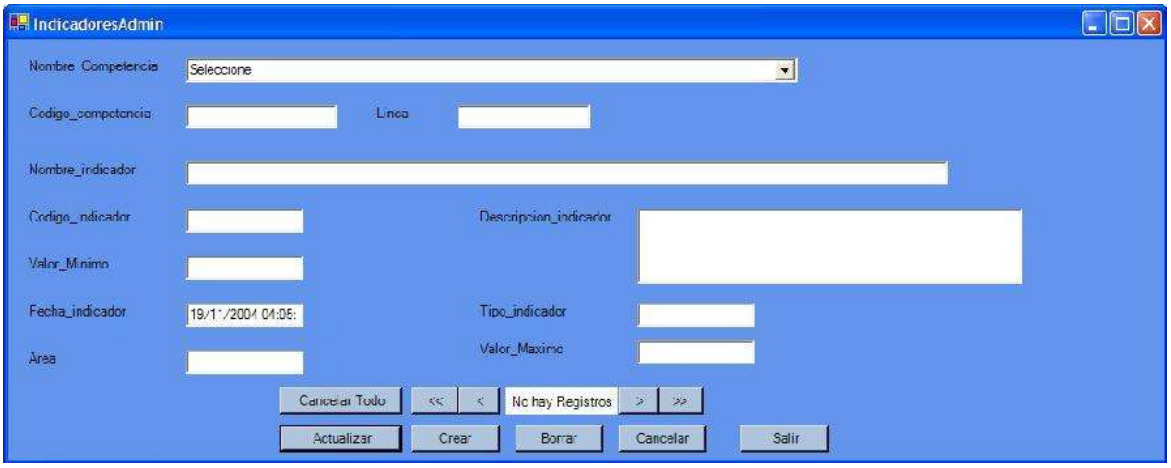
Esta ventana solo le compete al decano y los directores de línea, aquí se crean las competencias, con su respectiva descripción, área a la que pertenece, línea, tipo y materia.



Al igual que la ventana anterior, esta solo le compete al decano y los directores de línea, aquí se actualizan, las competencias.

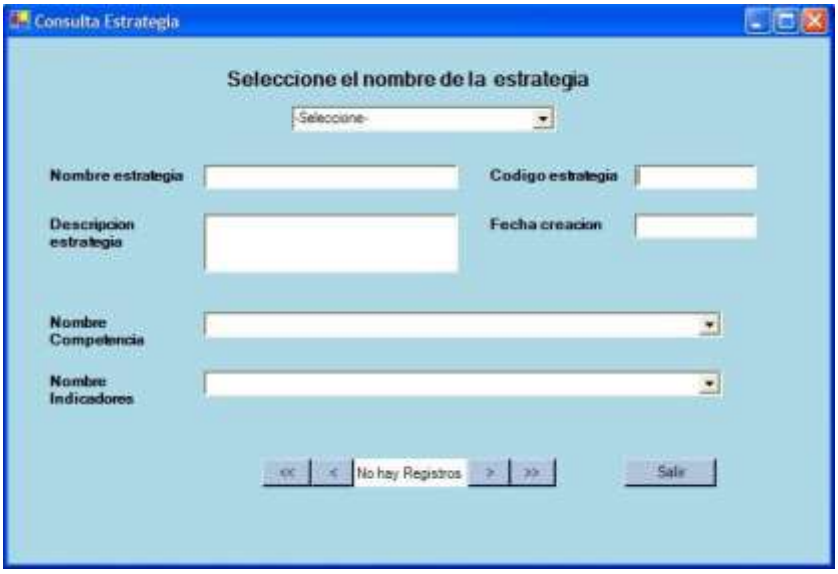


En esta ventana se realizan las consultas acerca de las competencias, y se muestran todos los campos relacionados con dicha competencia.

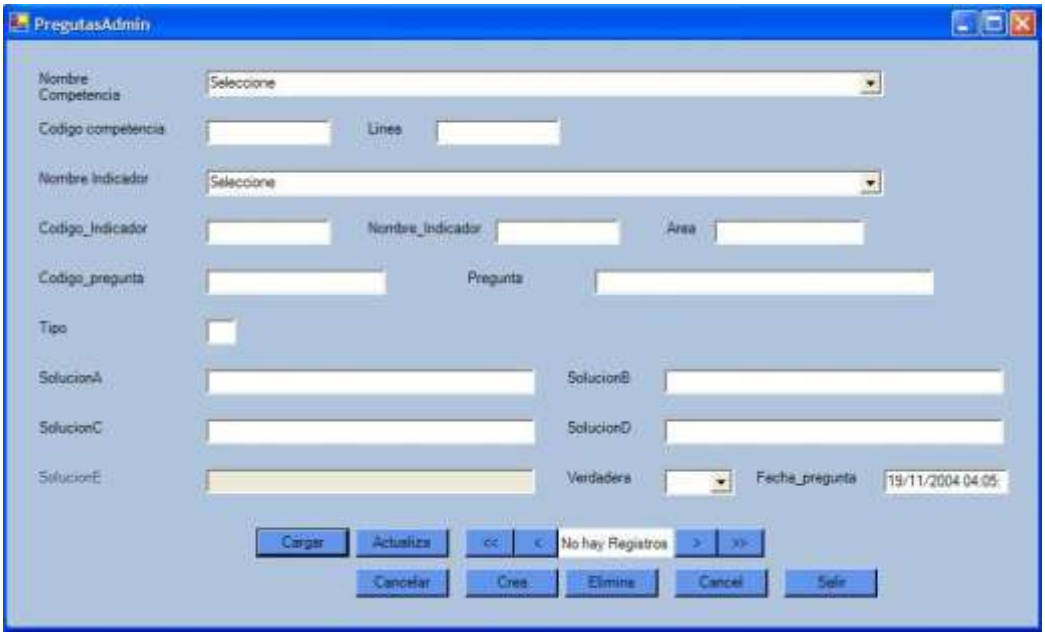


En esta ventana se pueden crear, actualizar o eliminar indicadores.

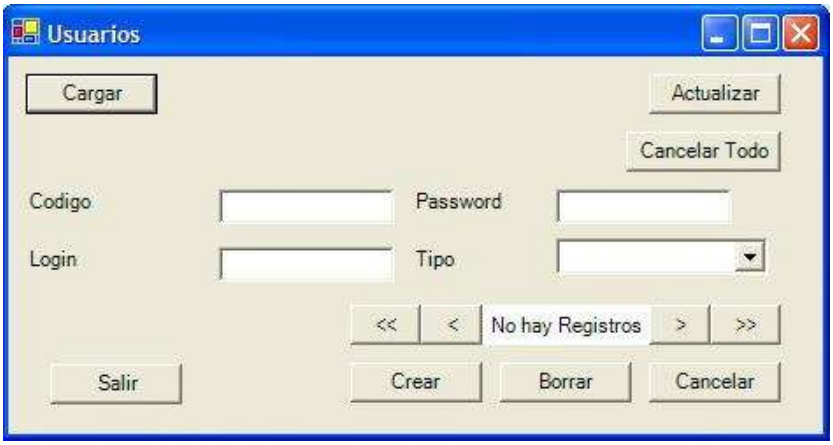
En esta ventana se pueden crear o actualizar las estrategias que se utilizaran en el sistema.



En esta ventana se realiza la consulta de las estrategias y se muestra la competencia y el indicador con los que esta relacionada.



En esta ventana se realiza la creación, actualización o eliminación de preguntas, y se relaciona con una competencia y sus respectivos indicadores.



Aquí se crean los tipos de usuarios, dándole sus respectivos privilegios en el sistema, los usuarios de este sistema son Decano/director de línea, Docentes, Estudiantes.

6. EVALUACIÓN DEL SISTEMA

Indicadores Externos.

Tabla 10. Evaluación del sistema (Indicadores Externos)

Nombre	**Evaluación**
Rendimiento del sistema	No Aplica
El sistema es multi-usuario	No
Tipos de usuario	Si
Tiempo de respuesta en solicitud de reportes	No Aplica
Acceso seguro	Si
Facilidad de uso	Alto
Crecimiento del sistema	Si
Cambios del sistema	Si

GUTIÉRREZ TORRES, Cristian Javier. Estudiante Ingeniería de Sistemas UNAB.

Indicadores Internos.

Tabla 11. Evaluación del sistema (Indicadores Internos)

Nombre	**Evaluación**
Gestiona competencias	Si
Administrar competencias	Si
Administrar indicadores	Si
Administrar preguntas	Si

Generación de reportes	No
Realiza consultas	Si
Genera exámenes	No
Muestra gráficas estadísticas	Si

ACEVEDO MAZA, Juan Francisco. Estudiante Ingeniería de Sistemas UNAB.

7. CONCLUSIONES

Este proyecto es una experiencia integral de la ingeniería de sistemas, porque cuando existe un problema de ingeniería se debe plantear una metodología de trabajo para asegurar el cumplimiento de los objetivos propuestos brindando una solución funcional viable, en donde se tienen una serie de pasos que comprenden el análisis y la definición del problema permitiendo conceptualizar e identificar plenamente el problema, el estudio de alternativas de solución para lograr buenos resultados, el diseño de una posible solución, la implementación del diseño y una evaluación de este.

Dentro de la metodología de trabajo se planteó un estudio de la plataforma tecnológica, ya que para el desarrollo del proyecto no se debe utilizar una herramienta simplemente porque el desarrollador la sepa usar, o porque sea la que él le guste; la herramienta que se debe utilizar es la que se adapte al problema, de lo contrario se estaría adaptando el problema a la herramienta haciendo que el proceso de desarrollo ingenieril no sea el mejor.

El uso de herramientas de modelado para el desarrollo del proyecto, cómo Racional Rose, tiene una serie de ventajas para los desarrolladores y para el usuario final, como lo es el diagrama de casos de uso, el diagrama de secuencia, el diagrama de colaboración, y el diagrama de paquetes que permite observar, especificar, construir y documentar todas las partes que componen el desarrollo de software, dejando como resultado una especificación formal del proceso llevado a cabo en el desarrollo de la herramienta.

En el desarrollo de proyectos ingenieriles, los indicadores de evaluación son muy importantes puesto que ayudan a medir que tanto la solución propuesta soluciona ó no el problema.

Algunos de los indicadores empleados para la evaluación del sistema no pueden ser aplicados a dicha evaluación, ya que requieren de algunas características, uso de equipos especiales, o bien, que el software este terminado por completo.

La cibernética organizacional, como corriente del pensamiento sistémico permite asumir la organización desde diferentes puntos de vista; al aplicarla en la solución del problema, la cibernética organizacional proporciona un ciclo de mejoramiento sobre la calidad académica de la facultad de ingeniería de sistemas.

Este proyecto fue muy enriquecedor en muchos aspectos tanto en el profesional como en el personal; ya que en el ámbito profesional permitió fortalecer todos aquellos conocimientos teóricos adquiridos durante todo el proceso de formación en la universidad y en el ámbito personal permitió adquirir experiencia en el desarrollo de proyectos en otras áreas y especialmente trabajar en grupo.

RECOMENDACIONES

Este proyecto es una primera versión con relación a la gestión de competencias, por tal motivo se sugiere que sea retomado este proyecto.

Debe realizarse a la par con el desarrollo del sistema un proceso de cambio cultural en la organización, facultad de ingeniería de sistemas FIS, que haga sostenible una gestión integral de las competencias en el proceso de formación de los estudiantes, como indicador de la madurez y experiencia adquirida por la organización. Este cambio cultural en la organización debe estar relacionado con:

- El establecimiento de competencias y sub-competencias:
  - Competencias Generales.
  - Competencias por nivel.
  - Competencias por área.
  - Competencias por curso.

- El Establecimiento de Indicadores:
  - Grupos de indicadores para cada competencia.
  - Valoración porcentual de indicadores.
  - Grados de los indicadores (Si/No, Alto/Medio/Bajo).

BIBLIOGRAFÍA

ACOFI. Exámenes De La Calidad De La Educación Superior ECAES.2003.

BEER, Stafford. Decision and Control. Nueva York: Wiley, 1966.

BLANCHARD, Benjamín S. Ingeniería de Sistemas. Isdefe - c/ Edison, 1996.

BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. El Lenguaje Unificado de Modelado. Addison Wesley, 1999.

BOOCH, Grady, RUMBAUGH, James, JACOBSON, Ivar. El Proceso Unificado de Desarrollo de Software. Addison Wesley, 2000.

CARRETERO PÉREZ, Jesús. Sistemas Operativos una Visión Aplicada. Mc GrawHill, 2001.

Convenio ACOFI - ICFES. Especificaciones De Los Exámenes De Estado De Calidad De La Educación Superior En Ingeniería De Sistemas / Informática. 2003.

DATE, C.J, Introducción a las Bases de Datos. Addison Wesley Iberoamericano, 1986.

ESPINOSA, Ángela María. Una visión de las organizaciones Sociales. Bogotá, 1999.

FOXALL, James. Practical Standars For Microsoft Visual Basic .Net. Microsoft Press, 2001

Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Informe Nacional de Resultados.2003.

Instituto Colombiano Para El Fomento De La Educación Superior - ICFES. Transformación Del ICFES.2003.

LEÓN SERRANO, Gonzalo. Ingeniería de Sistemas de Software. Isdefe - c/ Edison, 1998.

MARTÍNEZ, Alejandro. MARTÍNEZ, Raúl. Guía a Rational Unified Process. Escuela Politécnica Superior de Albacete - Universidad de Castilla la Mancha. 2002.

MORALES MONTEJO, Clemencia. Seminario Indicadores de Gestión. Universidad de los Andes. Facultad de Ingeniería Industrial. Bogotá. 1994.

PACHÓN LÓPEZ, Wilfredo L. Comparativa de Sistemas Operativos - www.canapro.org.co/~wilfred_com/s.o/sistemas_operativos/ sistemas_operativos-node17.html, 2004

PINCIROLI, Fernando. Algunas consideraciones sobre “Procesos de Desarrollo de Software. Conferencia UNAB, 2004.

PRESSMAN, Roger S. Ingeniería de Software: Un enfoque Práctico. Mc GrawHill, 2002.

ROA VALERO, Alberto. Hacia un modelo de aseguramiento de la Calidad en la Educación Superior en Colombia: estándares básicos y acreditación de excelencia, Miembro del Concejo Nacional de Acreditación. 2003.

STALLING, William, Sistemas Operativos. Prentice Hall. 1997.

UNAM, Metodologías de Ingeniería de Software, Departamento de Auditoria Informática, Subdirección de Sistemas DCAA, 1997.

VELANDIA, Néstor Orlando. Guía Teórica y práctica de la Cibernética Organizacional. Bogotá. 1998.

Referencias Web

COMFAMA (Caja de Compensación Familiar de Antioquia), Proyecto Empresarial de Competencias, Citado en Marzo de 2004. Dirección Web: <www.comfama.com.co/contenidos/servicios/Educaci%C3%B3n/Competencias%20laborales/competenciaslaborales.asp>

ICFES (Instituto Colombiano para el Fomento de la Educación Superior), ECAES, Citado en Marzo de 2004. <www.presidencia.gov.co/cne/2003/agosto/28/15282003.htm>

MICROSOFT, Versiones de Windows 2000, Citado en Marzo de 2004. Dirección Web: <www.microsoft.com/productos/win2000.html>

MICROSOFT, Herramientas de Desarrollo, Citado en Junio de 2004. Dirección Web: <www.microsoft.com/latam/vstudio/producto/resumen.asp>

MICROSOFT, Herramientas de Desarrollo, Citado en Junio de 2004. Dirección Web: <www.microsoft.com/latam/vbasic/producto/requerimientos.asp>

TORRES, Luis; Comparativa de Plataformas Tecnológicas de Desarrollo, Citado en Junio de 2004 Dirección Web:<www.ciberteca.net/articulos/programacion/net/ lenguajes .asp>