UNIREMINGTON®
CORPORACIÓN UNIVERSITARIA REMINGTON
RES. 2661 MEN JUNIO 21 DE 1996

INGENIERÍA DE SOFTWARE I

INGENIERÍA DE SISTEMAS

FACULTAD DE CIENCIAS BÁSICAS E INGENIERÍA

Vicerrectoría de Educación a Distancia y virtual

2016

El módulo de estudio de la asignatura INGENIERÍA DE SOFTWARE I es propiedad de la Corporación Universitaria Remington.
Las imágenes fueron tomadas de diferentes fuentes que se relacionan en los derechos de autor y las citas en la bibliografía.
El contenido del módulo está protegido por las leyes de derechos de autor que rigen al país.
Este material tiene fines educativos y no puede usarse con propósitos económicos o comerciales.

AUTOR

Piedad María Metaute Paniagua
Ingeniera de Sistemas, Especialista en Finanzas, Magister en Educación y Desarrollo Humano, diplomados en Competencias Pedagógicas, Diseño Curricular, seminarios sobre Formación en procesos y Técnicas de Investigación, sobre Negociación Electrónica, sobre Evaluación por Competencias pruebas ECAES, ICFES, SABER. Con amplia experiencia docente en los diferentes ciclos de educación desde Básica Secundaria, Técnica, Tecnológica, Profesional, Especialización en temáticas relacionadas con los sistemas, la computación y la informática, Asesora y Jurado de proyectos de grado, Coordinación de semilleros de investigación, Investigadora; así como experiencia en empresas del sector productivo en lo referente a la construcción de productos de software.
Piedad.metaute@uniremington.edu.co - pmetaute@gmail.com

Nota: el autor certificó (de manera verbal o escrita) No haber incurrido en fraude científico, plagio o vicios de autoría; en caso contrario eximió de toda responsabilidad a la Corporación Universitaria Remington, y se declaró como el único responsable.

RESPONSABLES

Jorge Mauricio Sepúlveda Castaño
Decano de la Facultad de Ciencias Básicas e Ingeniería
jsepulveda@uniremington.edu.co

Eduardo Alfredo Castillo Builes
Vicerrector modalidad distancia y virtual
ecastillo@uniremington.edu.co

Francisco Javier Álvarez Gómez
Coordinador CUR-Virtual
falvarez@uniremington.edu.co

GRUPO DE APOYO

Personal de la Unidad CUR-Virtual

EDICIÓN Y MONTAJE

Primera versión..

Segunda versión.

Tercera versión.

Cuarta versión 2016

Derechos Reservados

Esta obra es publicada bajo la licencia Creative Commons. Reconocimiento-No Comercial-Compartir Igual 2.5 Colombia.

CONTENIDO

Pág.

1 MAPA DE LA ASIGNATURA

INGENIERÍA DE SOFTWARE I

PROPÓSITO GENERAL DEL MÓDULO

La Ingeniería de Software I, se orienta hacia la adquisición de los fundamentos, métodos y herramientas, donde el estudiante propone alternativas de solución a necesidades informacionales del contexto, que deban ser resueltas a través de productos de software, donde el objeto principal radica en la especificación del requisito de software, cumpliendo con características de calidad, bajo los lineamientos de la Ingeniería de Requisitos, propuestas desde el SWEBOK (Cuerpo de Ingeniería del Conocimiento).

OBJETIVO GENERAL

Especificar los requisitos de software, a través de situaciones problemáticas del contexto que deban solucionarse por medio de un producto de software, utilizando herramientas, métodos y las buenas prácticas que suministra el SWEBOK (Cuerpo del conocimiento de la Ingeniería de Software).

OBJETIVOS ESPECÍFICOS

- Identificar el contexto sobre el cual se apoya la Ingeniería de Software en relación a métodos, metodologías y herramientas que permitan el manejo adecuado de los procesos de Ingeniería de Software.
- Aplicar procesos de elicitación, utilizando herramientas adecuadas para la extracción y tratamiento de los requerimientos de clientes y/o usuarios.
- Especificar los requisitos de software, aplicando validación, documentación y herramientas de Ingeniería de Software, garantizando así calidad en los requisitos a desarrollar en el producto informático.
- Identificar los componentes de un proyecto basado en metodología ABP (Aprendizaje Basado en Problemas) y los principios de la investigación formativa, como estrategia pedagógica para la especificación de los requisitos de software, como base fundamental en la construcción de un producto informático de calidad

UNIDAD 1

(Principios, métodos y metodologías en la Ingeniería de Software)

UNIDAD 2

(Elicitación de Requerimientos)

UNIDAD 3

(Especificación de requisitos de software)

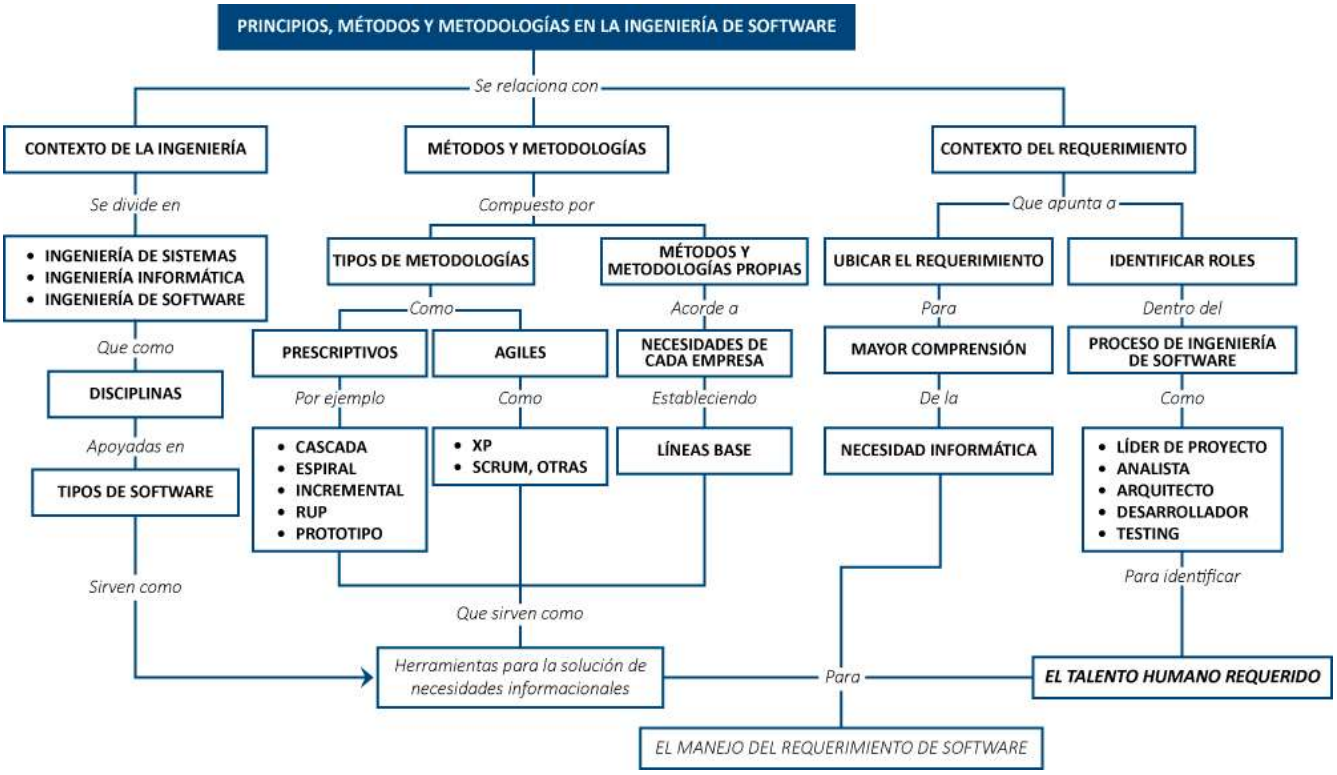
UNIDAD 4

(Proyecto de Ingeniería de Software (Aprendizaje Basado en Problemas e Investigación Formativa)

2 UNIDAD 1: PRINCIPIOS, MÉTODOS Y METODOLOGÍAS EN LA INGENIERÍA DE SOFTWARE

Las herramientas TIC (Tecnologías de Información y Comunicación) se han convertido en recursos indispensables para el apoyo a cualquier tarea que se realiza dentro de ésta sociedad considerada, la sociedad de la información y el conocimiento, son utilizadas actualmente para la automatización de procesos, la comunicación de las personas desde cualquier parte del mundo, tareas personales, tareas profesionales, incremento de ganancias, aumento de ingresos, optimización de tiempos y recursos, diversión, apoyo a procesos educativos, entre otros, transversalizando así las culturas, los pueblos y en conjunto la dinámica social. Ver video "Un día de vidrio" https://www.youtube.com/watch?v=Usj5MBGkhKY

Mapa conceptual 1 (Principios, métodos y metodologías en la Ingeniería de Software)



2.1 TEMA 1: RELACIÓN: INGENIERÍA DE SISTEMAS, INGENIERÍA INFORMÁTICA, INGENIERÍA DE SOFTWARE Y CONTEXTUALIZACIÓN DE LA INDUSTRIA DEL SOFTWARE

Para el futuro Ingeniero de Sistemas, es importante contextualizarse en aspectos como la disciplina que fundamentará su perfil ocupacional y profesional, por lo tanto es necesario que se involucre en temas como la ingeniería de sistemas, ingeniería informática y especialmente sobre la Ingeniería de Software, como disciplina que requiere ser fortalecida dentro de los procesos de construcción del software; industria que viene creciendo rápidamente, requiriendo personal cualificado que aporte en innovación y creatividad para cubrir la demanda de productos de software con calidad.

2.1.1 UN POCO DE HISTORIA SOBRE LA EVOLUCIÓN DE LOS SISTEMAS DE CÓMPUTO

Los avances de la tecnología, especialmente las telecomunicaciones, mediadas por la evolución de internet, han facilitado la intercomunicación con el mundo, dentro de una realidad llamada globalización. Si nos detenemos un poco en la historia, el ser humano, a través de su creatividad e innovación ha buscado la forma de mejorar procesos utilizando la automatización de sus tareas, buscando con ello agilizar el desarrollo de sus actividades, que en definitiva se orientan hacia la minimización de tiempos, recursos y esfuerzos, facilitando la realización de éstas.

Es así que ha buscado insumos o materia prima de la misma naturaleza, los ha transformado, evolucionado, creando, otros insumos y productos, a los cuales les ha buscado aplicación. Dentro del contexto actual, los datos, la información, su análisis y uso se constituyen, en el elemento más importante para tomar cualquier tipo de decisión en la sociedad. Ver imagen 1, donde se ilustra el proceso evolutivo de la humanidad, a través de los avances tecnológicos.

Imagen 1Evolución de la tecnología a través de la historia

Fuente: http://staticdn.lovities.com/img/post/1649458010/la-evolucion-de-la-tecnologia-9414.jpg

Los avances tecnológicos en relación a los equipos de cómputo que la sociedad disfruta actualmente, obedecen a esfuerzos de muchísimos años de historia que han marcado hitos importantes en cada cultura, en cada época, en cada generación que se ha recibido aportes de las anteriores y que de igual forma sigue generando nuevos conocimientos, acorde a las necesidades propias o ajenas, consolidando así realidades, dentro de una sociedad que sigue aportando a la consecusión de nuevos logros en materia tecnológica, como se puede observar en la imagen 2, donde los inicios del computador datan desde el origen del ábaco y la invención de una serie de máquinas que han servido para la realización de operaciones rudimentarias y básicas, evolucionando con el tiempo, hasta lograr complementar dichas máquinas que aún siguen sufriendo cambios importantes en nuestros días.

Imagen 2Evolución del computador en la línea del tiempo

Fuente: Fuente: https://lh3.googleusercontent.com/LAOS_abAkY00AID6KZS9xll7KtE1CjyZtAluAJ3SxC7oMCA-gUS-A8qRsvVJtmGwzKt_Tw=s159

En los años 40, a partir de la consolidación del primer computador electro_mecánico, el (ENIAC), en materia de computación se generan procesos de evolución por generaciones no mayor a 10 años entre ellas, siempre orientadas al mejoramiento de los equipos, minimizando tamaños y optimizando procesos, velocidad, nivel de procesamiento, eficacia, eficiencia, donde a partir de la quinta genración (años 90), los cambios se suscita tan rápidamente, que la clasificación de generaciones por décadas se pierde, ya que la tecnología evoluciona cada día con resultados visibles para los usuarios finales, donde las máquinas se ubican en uso y en desuso en muy corto tiempo. La imagen 3 muestra las principales características que representaron cada una de las generaciones.

Imagen 3Evolución del computador por generaciones

Fuente: http://image.slidesharecdn.com/evoluciondelcomputador-141021113539-conversion-gate02/95/evolucion-del-computador-1-638.jpg?cb=1413891559

A partir de los años 90, la industria de la computación, diversifica sus productos de hardware, creando nuevas especificaciones que se orientan a la optimización de los equipos de cómputo, mejorando resolución de pantallas, de cámaras, potencia, velocidad y precisión en procesadores, equipos más versátiles y livianos, presentación, variedad en marcas, facilidad para conectividad, accesibilidad en costos, entre otras características que puestas a disposición de todo tipo de personas (adultos, jóvenes y niños), se convierten en herramientas cotidianas de uso.

Pero cuando se hace alusión al computador, no se puede pensar en que las máquinas por sí solas, pueden ser utilizadas para realizar funciones específicas, lo que lleva a poner especial atención en el software, entendido como instrucciones que son utilizadas para realizar funciones deseadas, definido también como estructuras de datos que permiten la manipulación de la información o programas que se realizan con fines específicos como por ejemplo la liquidación de una nómina, la gestión de una facturación, inventarios, matrículas, transacciones bancarias, herramientas ofimáticas, procesamiento de datos, gestión gerencial para las empresas, juegos, control de artefactos, apoyo a procesos médicos, aplicaciones para el agro, entre un sinnúmero de fines que aún ni nos imaginamos en los cuales el software tendría gran funcionalidad. Por lo tanto, se hace indispensable integrar e hardware (máquinas o equipos físicos), con el software (programas), ya que no se puede hablar de un computador sin programas o un programa sin máquina que permita ejecutar la funcionalidad para lo cual fue creado.

Los sistemas informáticos, siguen presentando cambios acelerados, lo que fácilmente se puede percibir, en tamaño, presentación, forma, capacidad, velocidad, seguridad, facilidad de uso, funcionalidad, practicidad, convirtiéndose en un mercado competitivo para la industria del hardware y del software. Ver imagen 4.

Imagen 4Evolución de las especificaciones del hardware

Fuente: http://cdn3.computerhoy.com/sites/computerhoy.com/files/editores/user-17809/cuadro_especificaciones.jpg

Al igual que las máquinas, el software también ha evolucionado en el tiempo

Dividiéndose en cuatro eras:

- **Primera era (1946-1965):** No existía planeación para el desarrollo de software, al igual que métodos o metodologías para la construcción de software, por lo tanto, se trabajaba en base a prueba y error, disponiéndose de pocos recursos que permitían crear software, siendo en el año 1957 cuando nace el lenguaje Fortran en su primera versión, seguido por el lenguaje Cobol en el año 1960.
- **Segunda era (1965-1972):** Se busca simplificar el código y se trabajan bajo la teoría de la multiprogramación, sistemas en tiempo real que para el apoyo a la toma de decisiones, convirtiéndose el software en un producto, donde se generó una crisis debido a la demanda del producto, complejidad y desafíos, naciendo en 1968 los principios de la ingeniería del software como una necesidad que permitiese dar solución a la crisis del software debido a los escasos recursos con los que se contaba en dicha época.
- **Tercera era (1972-1985):** Aparece el concepto de los sistemas distribuidos (se accede a la información a través de las redes de computadores), donde aparecen redes de área local y global, de igual forma la aparición de lenguaje de programación Basic en 1975, ofrecería otras herramientas para la programación a bajo nivel.

- **Cuarta era (1985-1995):** Se evoluciona en cuestión de redes de información, tecnologías orientadas a objetos, redes neuronales, sistemas expertos, software de inteligencia artificial y la aparición de lenguajes orientados a objetos como Java en 1990.
- **Quinta era (2000…):** Se amplía el concepto aplicado de la Programación Orientada a Objetos (el código de programación se elabora para ser reutilizado), aumentando la presencia de aplicaciones web y el incremento de métodos, herramientas, tecnologías de punta (nueva), que buscan mayor cubrimiento a las necesidades informacionales de la sociedad, donde el cambio de versiones de los programas se realiza de forma acelerada, debido a la competitividad existente por el crecimiento de la industria del software.

La siguiente imagen representa la evolución en la forma como se ha construido software y la forma como se viene haciendo actualmente, donde hasta antes de iniciar el año 2000, los programas se construían utilizando programación estructurada, realizándose reproceso, ya que el código no se podía reutilizar eficientemente, lo cual generaba alto consumo de recursos. A partir del 2000, la forma como se programa el software cambia hacia la programación orientada a objetos, donde la forma como se codifica se hace más dinámica al poder reutilizar código y componentes dentro del mismo programa u otros, es así como los estilos de programación siguen evolucionando, buscando mayor eficiencia en el proceso y en el producto.

Imagen 5 Evolución de la programación

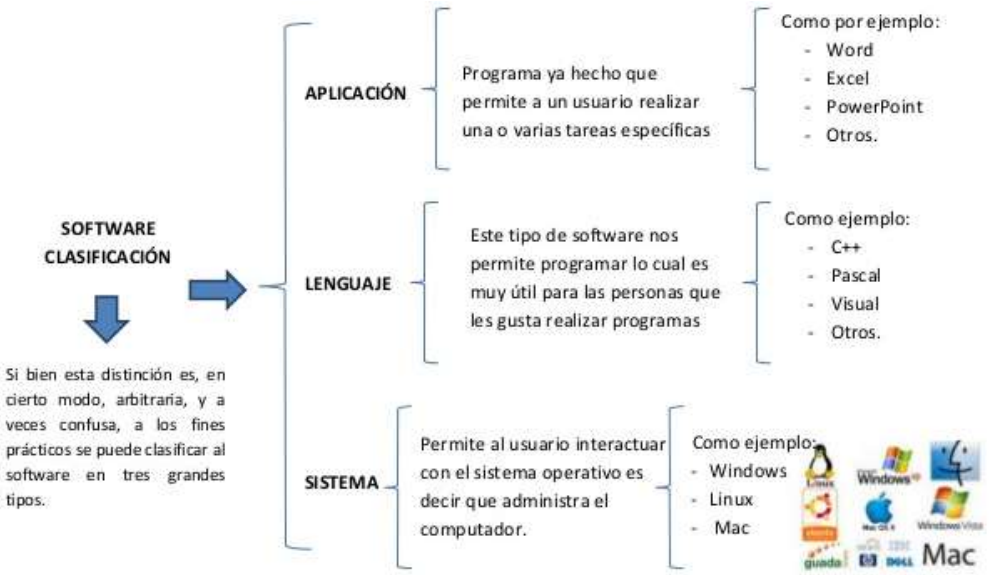
Fuente: http://img.photobucket.com/albums/v727/Maky_chan/ohsioh/WS1.jpg

2.1.2 TIPOS DE SOFTWARE

Es importante tener clara la clasificación del software, ya que normalmente se requieren varios de ellos para la realización de tareas. Entre la clasificación de software, se tienen cuatro grandes grupos como son:

- **Los sistemas operativos:** Siendo éstos el software principal que debe tener un equipo que trabaje bajo los principios de un computador (computador de escritorio, portátil, celular, robot, equipo para juegos, entre otros), ya que es el puente, interprete, administrador de los recursos de hardware y los programas que se instalen, teniendo en cuenta que sin éste, dichos equipos no funcionan. Entre los principales sistemas operativos tenemos (Windows, Linux, Mac, Android, entre otros).
- **Los lenguajes de programación:** Corresponde a aquellos programas que son utilizados por los expertos desarrolladores de software, para crear otros programas, acorde a los requerimientos de clientes o usuarios o acorde a oportunidades de negocio, que requieran ser automatizados. Entre los lenguajes de programación más utilizados en la actualidad tenemos (Java, .Net, PHP, entre otros), donde la utilización del lenguaje, depende del tipo de proyecto, de la empresa productora de software o del cliente.
- **Los Sistemas de Gestión de Bases de Datos:** Se relaciona con software creado para la Gestión y administración de los datos, permitiendo entre otras cosas, guardar, consultar, modificar, eliminar datos e información, donde la información se ha convertido en el activo más importante de cualquier empresa, institución, organización, corporación, e incluso a nivel personal y profesional, siendo los motores de bases de datos el soporte fundamental para la administración de dichas bodegas de datos. Entre los Sistemas de Gestión de Bases de Datos, más utilizados actualmente tenemos (SQL Server, MySQL, Oracle, Informix, entre otros).
- **EL software de aplicación**: (Todo tipo de software diferente a los anteriores como por ejemplo, software para oficina, contable, gestión empresarial, graficadores, juegos, control de máquinas, entre otros) que apoyen la gestión de procesos personales, académicos, empresariales, entre otros uso y campos de aplicación. En la imagen 6, se puede observar la clasificación del software.

Imagen 6 Clasificación del software



Fuente: http://image.slidesharecdn.com/sistemaoperativolibreypropietario-131009083228-phpapp01/95/sistema-operativo-libre-y-propietario-2-638.jpg?cb=1381307575

Los que, integrados a las redes de datos y demás dispositivos de comunicaciones, conforman la arquitectura informática que está disponible para todo tipo de clientes o usuarios que utilizan los recursos computacionales de acuerdo a su necesidad. En la imagen 7, se puede observar una arquitectura muy completa de recursos tanto de hardware como de software, así como de redes de comunicación.

Imagen 7 Arquitectura Informática

Fuente: https://cabenavides.files.wordpress.com/2014/10/arquitectura-desarrollo-de-software.gif

2.1.3 RELACIÓN DE LA INGENIERÍA DE SISTEMAS, INGENIERÍA INFORMÁTICA E INGENIERÍA DE SOFTWARE.

Es importante conocer las diferencias entre la Ingeniería de Sistemas, la Ingeniería Informática y la Ingeniería de Software, donde según Sommerville, (2005), la Ingeniería de Sistemas "es la actividad de especificar, diseñar, implementar, validar, utilizar y mantener los sistemas socio-técnicos. Los ingenieros de sistemas no sólo tratan con el software, sino también con el hardware y las interacciones del sistema con los usuarios y su entorno. Deben pensar en los servicios que el sistema proporciona, las restricciones sobre las que el sistema se debe construir y funcionar y las formas en las que el sistema es usado para cumplir con su propósito".

En el caso de Pressman (2010), define la ingeniería de sistemas como la disciplina que se centra en diversos elementos, analizando, diseñando y organizando esos elementos en un sistema que puede ser un producto, un servicio o una tecnología para la transformación de información o control de información.

En relación a la Ingeniería Informática es la rama de la ingeniería que aplica los fundamentos de la ciencia de la computación, la Ingeniería electrónica y la ingeniería de software, para el desarrollo de soluciones integrales de cómputo y comunicaciones, capaces de procesar información de manera automática, siendo ésta mas orientada a los sistemas de software y la electrónica.
En el caso de la ingeniería de software, según Sommerville, (2005), "es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software". La ingeniería del software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento de éste después de que se utiliza.

La diferencia entre Ingeniería de Sistemas e ingeniería de software, radica en que la Ingeniería de Sistemas aborda todos los aspectos del desarrollo de sistemas informáticos, incluyendo hardware, software e ingeniería de procesos, mientras que la ingeniería de software es parte de ese proceso, dedicándose solamente a los procesos relacionados con la construcción de software, siendo ésta última una derivación de la ingeniería de sistemas.

Imagen 8 Evolución tecnológica

Fuente: http://victorronco.com/wp-content/uploads/2014/01/wasanga.com_.jpg

Por lo tanto nos concentraremos en la ingeniería de software como disciplina que se encarga de la construcción de productos de software, teniendo en cuenta varias fases que van desde que el cliente solicita un requerimiento, hasta que dicho requerimiento convertido en software sale del mercado o es descontinuado.

INGENIERIA DE SISTEMAS Enlace

REDIS Antioquia - Promoción Ingeniería de Sistemas Enlace

2.2 TEMA 2: MÉTODOS Y METODOLOGÍAS APLICADAS EN EL PROCESO DE INGENIERÍA DE SOFTWARE

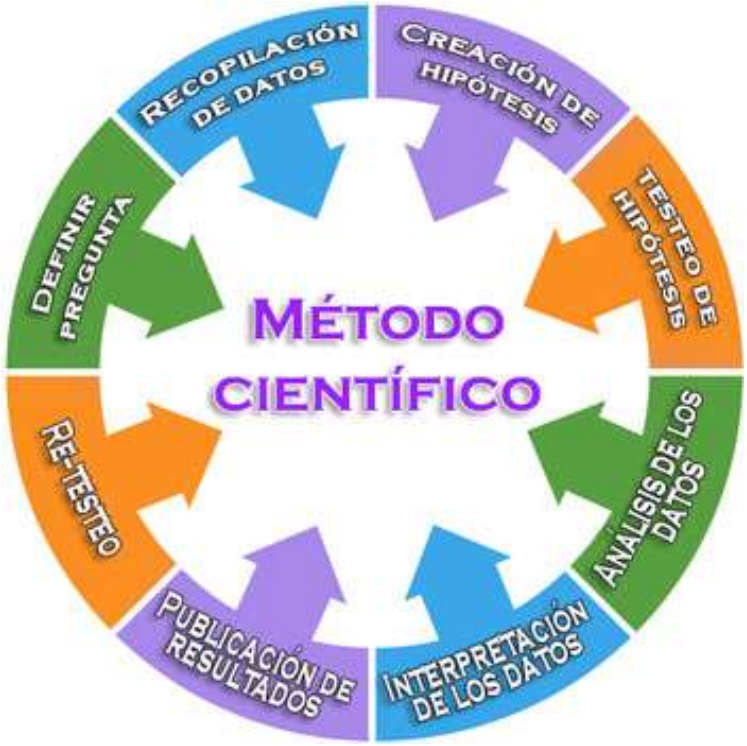
La Ingeniería de Software, se apoya en diferentes herramientas que permiten abordar sus procesos, durante el ciclo de vida que va desde las necesidades del cliente, hasta que el producto es entregado a satisfacción al cliente y/o usuario, e incluso en etapas posteriores a dicha entrega, relacionadas con mantenimiento, procesos de Gestión de Configuración de Software, por lo tanto se requieren Métodos y Metodologías como por ejemplo Metodologías prescriptivas, ágiles, propias, entre otras que son utilizadas de acuerdo a las necesidades de las empresas desarrolladoras de software.

2.2.1 EL MÉTODO Y LA METODOLOGÍA

En relación a ambos conceptos, vale la pena realizar aclaración entre el significado que tiene cada uno de ellos:

Método: es un concepto que es definido como el modo, manera o forma de realizar un proceso o actividad de forma secuencial o sistemática, el cual posee una estructura organizada que posee metas u objetivos claros. El método se fundamenta en el **QUÉ** se debe hacer. Un ejemplo claro se puede visualizar en el conocido método científico, que parte de un problema a estudiar, fundamentado en una pregunta, seguida de la recolección de evidencia y datos, que conlleva a la formulación de la hipótesis, revisión documental, análisis de datos y documentos, con posterior interpretación de datos, de los cuales se obtienen unos resultados que se comparten o publican, buscando con ello realizar aportes científicos: Ver imagen 9.

Imagen 9 Componentes del método científico



Fuente: http://queesel.info/wp-content/uploads/2015/07/metodo-cientifico-.jpg

- **Metodología**: se entiende como parte del proceso que permite sistematizar los métodos y las técnicas necesarias para llevarla a cabo el objetivo o meta propuesta. La metodología hace relación a una serie de pasos o procedimiento orientados a **CÓMO** se aplicará el método, como por ejemplo la identificación de necesidades, planificación, calidad, análisis, diseño, codificación, pruebas y mantenimiento de productos de software, cuyos pasos pueden variar acorde al fin u objetivo que se desee alcanzar, explicando detalladamente la forma **CÓMO** se desarrollará.

Imagen 10 Metodología

Fuente: http://www.neurorehabilitacion.es/images/cd_metodologia.jpg

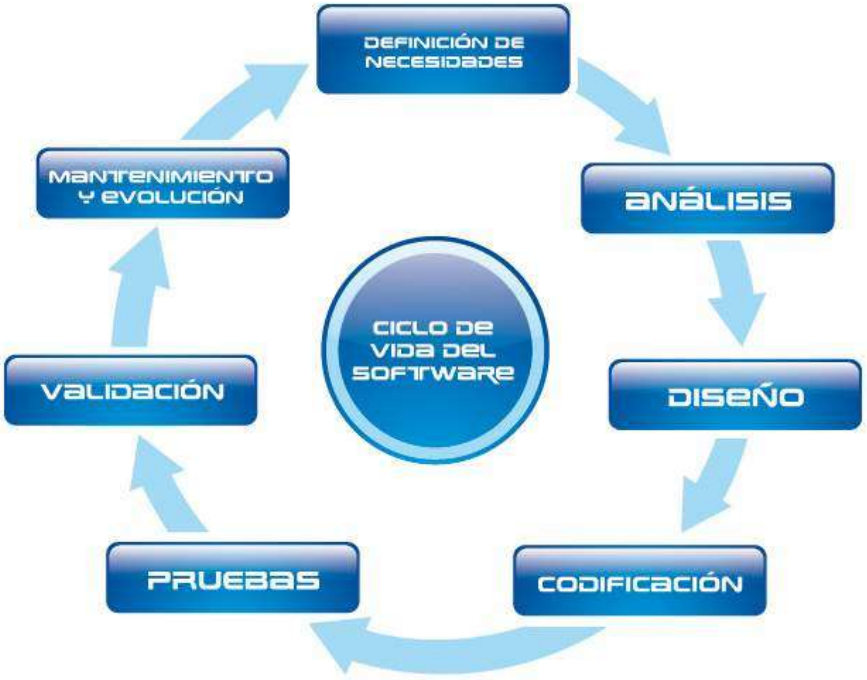
En relación a ambos conceptos, se tomarán uno como complemento del otro, donde el método ofrece lineamientos generales de QUÉ es lo que se debe hacer y la metodología se apoya en dichos lineamientos y detalla la forma de CÓMO se debe aplicar.

2.2.2 TIPOS DE MÉTODOS O METODOLOGÍA PARA EL DESARROLLO DE SOFTWARE

Antes de abordar los métodos o metodologías para la construcción de un producto de software, es importante aclarar que es el Ciclo de vida del software, se relaciona directamente con el proceso que se utiliza para construir, entregar y hacer evolucionar el software, desde que el cliente o usuario realiza un requerimiento hasta que el producto es entregado y retirado del mercado, es así que para su construcción se deban definir fases que permite realizar los respectivos controles, buscando con ello que cada uno de los procesos tengan calidad y ello se vea reflejado en el producto final que se pone en manos del cliente o usuario.

Según la ISO 12207-1, el ciclo de vida de un software se define como "Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto de software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso". Ver imagen 11, donde se muestra el ciclo de vida en mención que parte de una necesidad informática que presenta un cliente o usuario, la cual es estudiada por el experto informático, donde de la mano con el cliente se evoluciona el proceso pasando por varias etapas como análisis, diseños, codificación, pruebas, validaciones, evolución, donde no necesariamente deberá hacerse en ese orden.

Imagen 11 Ciclo de vida del software



Fuente: http://www.ecured.cu/images/5/5e/Cicloevida1.jpg

Por lo tanto, la Ingeniería de Software, a través de sus herramientas busca aplicar las mejores prácticas (experiencias propias o ajenas que hayan arrojado resultados positivos) para la construcción de productos de calidad y utiliza métodos o metodologías que le permitan optimizar los procesos. En relación a métodos o metodologías para desarrollo de software, algunos autores no hacen diferencia entre la una y la otra, son asumidas como herramientas que permiten dar orientaciones claras sobre el tratamiento de cada una de las etapas para la construcción del software. Analizar la imagen 12.

Imagen 12 Sobre métodos y metodologías de software

¿QUÉ ES? Un conjunto distinto de actividades, acciones, tareas, fundamentos y productos de trabajo que se requieren para desarrollar software de alta calidad Proporcionan una guía útil para el trabajo de la ingeniería del software	**¿QUIÉN LO HACE?** Los ingenieros de software y sus gerentes adaptan un modelo prescriptivo de proceso a sus necesidades y después lo siguen La gente que ha solicitado el software participa durante la ejecución del modelo de software
¿POR QUÉ ES IMPORTANTE? Proporciona estabilidad, control y organización a una actividad que si no controla puede volverse caótica.	**¿CUÁLES SON LOS PASOS?** El proceso conduce a un equipo de software a través de un conjunto de actividades del marco de trabajo que se organizan en un flujo de proceso, el cual puede ser lineal, incremental o evolutiva
¿CUÁL ES EL PRODUCTO OBTENIDO? Desde el punto de vista de un ingeniero de software: programas, documentos y datos que se producen como consecuencia de las actividades y tareas que define el proceso	**¿CÓMO PUEDO ESTAR SEGURO DE QUE LO HE HECHO CORRECTAMENTE?** Los mejores indicadores de la eficacia del proceso que se utiliza son la calidad, el tiempo de entrega y la viabilidad a largo plazo del producto que se construye

Los modelos o metodologías para la construcción del software, se clasifican en metodologías prescriptivas y metodologías ágiles.

2.2.3 METODOLOGÍAS PRESCRIPTIVAS

Los modelos o metodologías prescriptivos de software fueron creados para dar un aporte importante a la "crisis del software". Ver video (https://www.youtube.com/watch?v=miGf6iZ0GZY), es así como se definen un conjunto de tareas o actividades, que buscan dar herramientas para ordenar y ejercer control sobre los pasos para la construcción de software, lineamientos que son requeridos para desarrollar productos de calidad y llevar control, estabilidad y organización a una actividad que requiere especial cuidado.

El modelo o metodología prescriptiva, se creó con el fin de organizar las tareas o actividades para ser aplicadas por la ingeniería de software, en las etapas del ciclo de vida del software, es de aclarar que la aplicación de

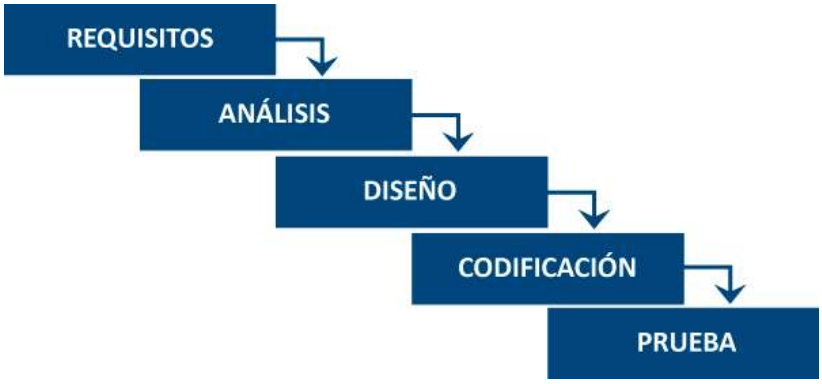
procesos prescriptivos, no se debe asumir como algo estático, cerrado, debe considerarse como herramientas que pueden adaptarse a diferentes necesidades, procesos y proyectos. Entre los modelos o metodologías Prescriptivos más utilizados tenemos:

- **Metodología Cascada**

La metodología cascada, es considerada la primera metodología utilizada para la construcción del software, de igual forma la base para la creación de las demás metodologías.

Modelo en Cascada Enlace

Imagen 13 Metodología cascada



Fuente:https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcRdngZDtD5D3dNn0e2FyFN3rCboOYVfFBlxW18jCI57d_dkKzxWxg

Algunas ventajas y desventajas

- **Ventajas**

✓ La cantidad de recursos necesarios para implementar este modelo es mínima.

✓ La documentación se produce en cada etapa del desarrollo.

✓ Las pruebas se realizan al final del proceso.

- **Desventajas**

✓ No se puede volver atrás.

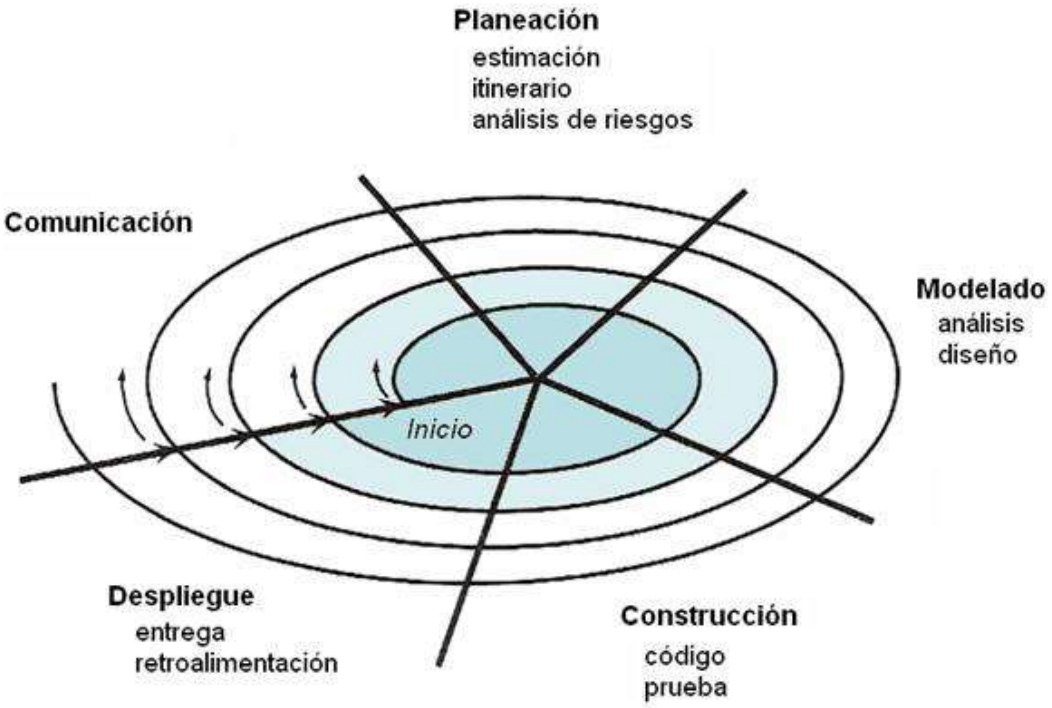
✓ Cualquier cambio genera confusión y caos.

✓ Hasta el final del proceso, no se sabe si lo que se diseñó, obedece a la necesidad del cliente.

Metodología Espiral

Metodología propuesta inicialmente por Boehm en 1976, es un modelo evolutivo que se apoya en prototipos, fue el primero modelo que manejó las temáticas de riesgos como todo aquello que puede afectar el normal desarrollo de un proyecto. Ver video: https://www.youtube.com/watch?v=Wsvr-aB0tPQ

Imagen 14 Metodología espiral



Fuente: https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcQIZ0r8d2xNo_5u8DUX8qDfFLU4V1nxMCj9KYOHQseQBauTvpBSPg

Algunas ventajas y desventajas

- **Ventajas**

✓ El desarrollo repetido o continuo, ayuda en la gestión de riesgo.

✓ La adaptabilidad en el diseño del modelo de espiral en la ingeniería de software se adapta a cualquier número de cambios, que pueden ocurrir durante cualquier fase del proyecto.

✓ La construcción de prototipo se realiza en pequeños fragmentos o trozos

✓ El cliente puede obtener el control sobre la administración del nuevo sistema.

- **Desventajas**

✓ Los modelos en espiral funcionan mejor para los grandes proyectos.

✓ Requiere mucha planificación.

Metodología Incremental

Propuesto por Harlan Mills en el año 1980. Posee un enfoque incremental que busca adquirir experiencia con el sistema, donde en cada incremento que se realice, entrega de un producto completamente operacional. Ver video: https://www.youtube.com/watch?v=H2Rg5l48K1A

Imagen 15 Metodología incremental

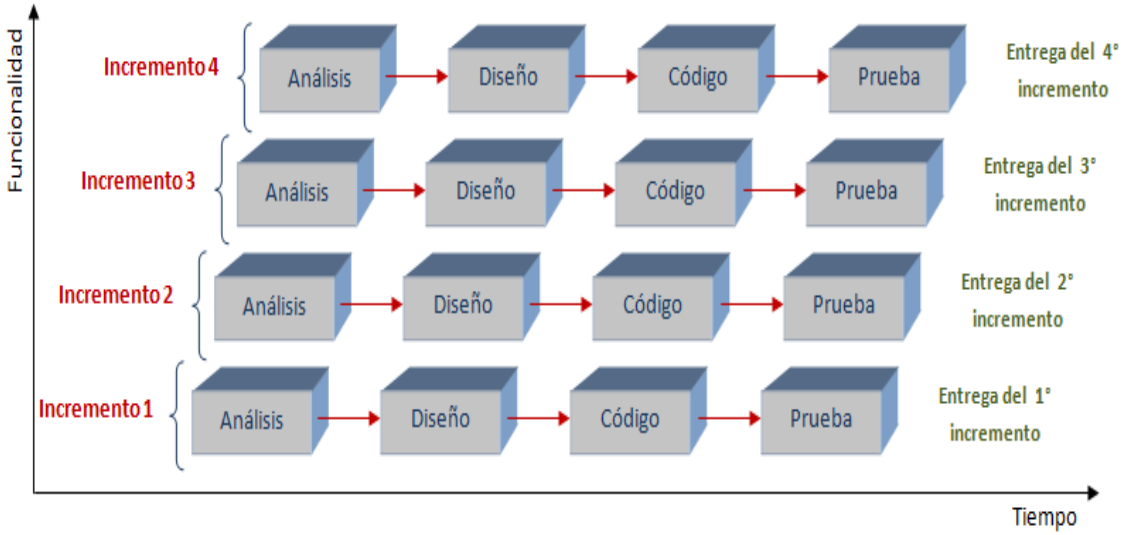


Figura 1: El Modelo Incremental

Fuente: https://procesosoftware.wikispaces.com/Modelo+Incremental

Algunas ventajas y desventajas

- **Ventajas**

✓ Se reduce el tiempo de desarrollo inicial, ya que se implementa la funcionalidad parcial.

✓ Impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del software.

✓ Fortalece el modelo en Cascada realimentándolo, reduciendo sus desventajas solo al ámbito de cada incremento.

✓ Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos.

- **Desventajas**

✓ No es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido y/o de alto índice de riesgos.

✓ Requiere de mucha planeación, tanto administrativa como técnica.

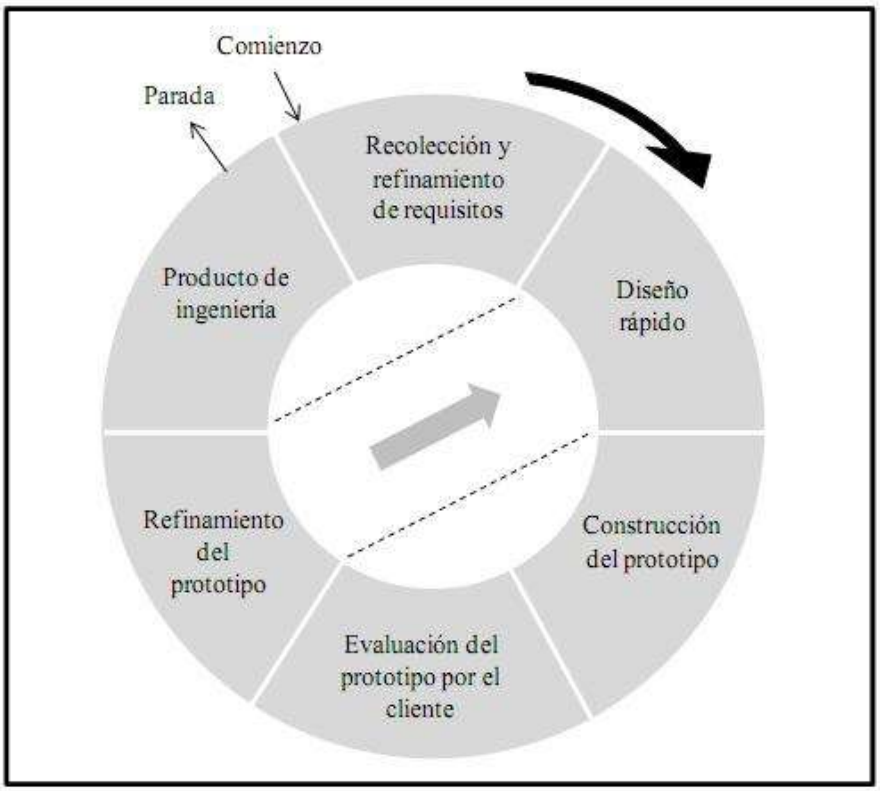
✓ Requiere de metas claras para conocer el estado del proyecto.

Metodología Prototipo

El modelo de prototipos permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario poseen el mismo objetivo sobre la funcionalidad del software. Ver Video:

Modelo por Prototipos Enlace

Imagen 16 Metodología Prototipo



Fuente: https://sisteminformacii.wikispaces.com/METODOLOG%C3%8DA+DE+ROGER+PRESSMAN

Algunas ventajas y desventajas

- **Ventajas**

✓ Permite el desarrollo de un sistema a partir de requisitos poco claros o cambiantes.

✓ Son más fáciles de abordar con los usuarios finales.

✓ El usuario participa más activamente en la construcción del producto de software.

✓ Se reduce el riesgo o la incertidumbre del software.

✓ Su uso redunda en una mayor satisfacción del usuario con el producto final, ya que ha participado activamente de su diseño

✓ Proporcionan al usuario un mayor conocimiento del sistema con una curva menor de capacitación.

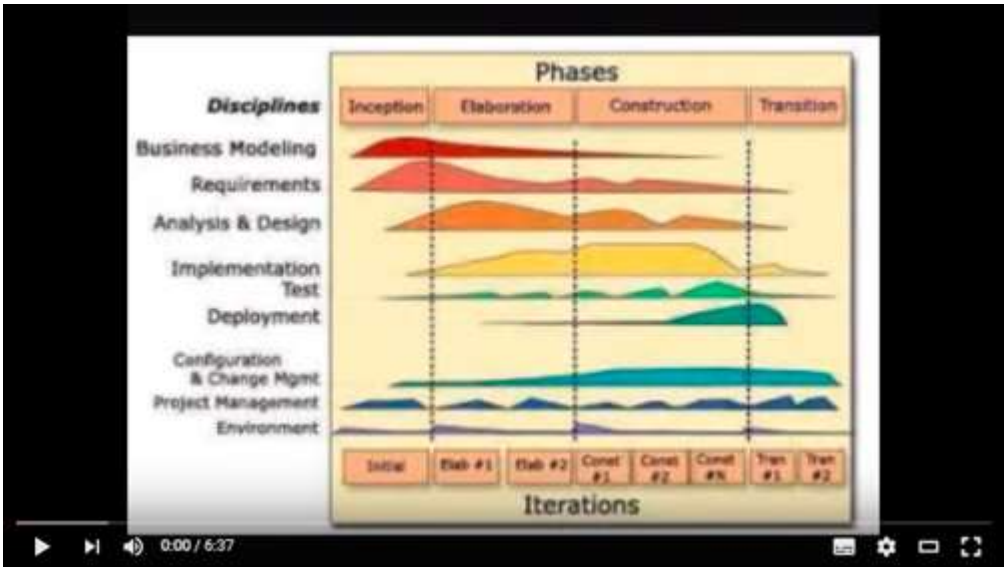
✓ Permite a todos los involucrados entender bien y mejor el problema antes de la implementación final.

- **Desventajas**

- ✓ El usuario quiere empezar a trabajar desde el primer momento con el prototipo para solucionar su problema particular cuando el prototipo es solo un modelo de lo que será el producto.
- ✓ Los prototipos pueden generar otro tipo de problema si su presentación y discusión con los usuarios no es controlada
- ✓ Requiere participación activa del usuario.
- ✓ Requiere de experiencia en los diseñadores del software.

Metodología RUP

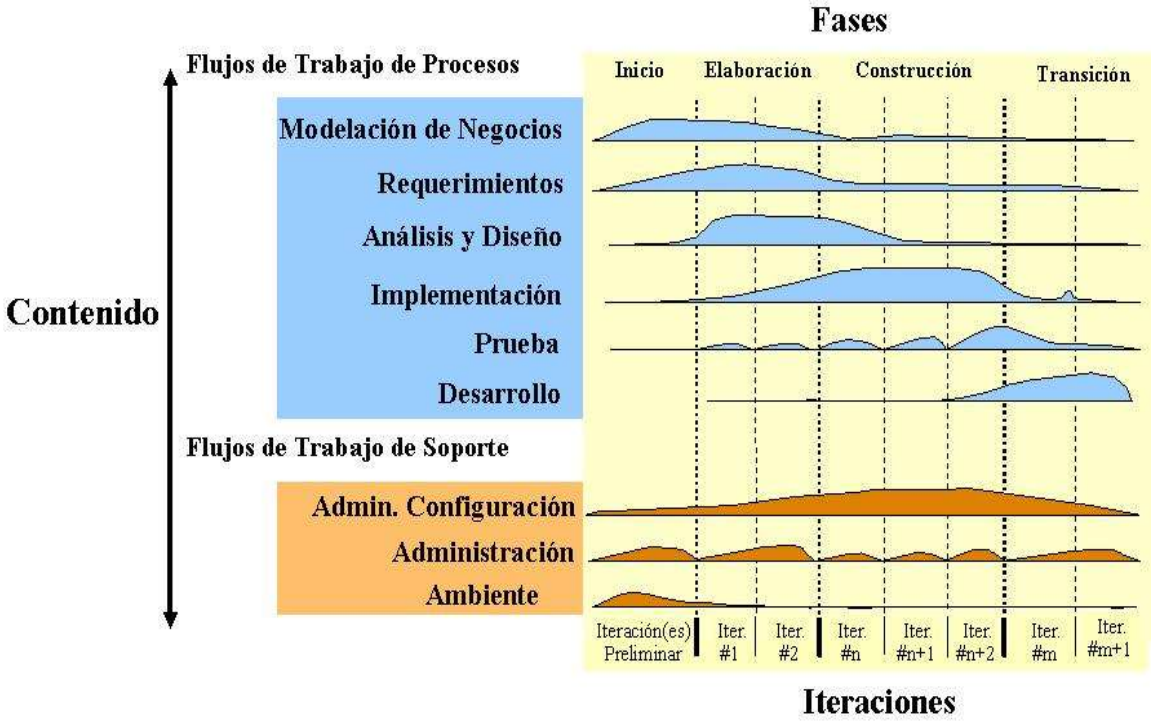
Es una metodología propuesta por Rational (IBM) se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado UML, especial para ser aplicado a la programación orientada a objetos (POO).



Rup mini Enlace

Proceso Unificado de desarrollo del Software Enlace

Imagen 17 Metodología RUP



Fuente: http://metodologiadesoftware.blogspot.com.co/2012/11/fases-del-modelo-rup_27.html

Algunas ventajas y desventajas

- **Ventajas**

- ✓ RUP ha madurado con el tiempo el uso del UML
- ✓ Se trabaja basado en estándares internacionales
- ✓ Adaptable a la organización.
- ✓ Utiliza buenas herramientas para la construcción e implementación.
- ✓ Define actividades, roles y responsabilidades desde jefes de proyectos hasta los analistas y desde desarrolladores y equipos de prueba.
- ✓ Tiene excelente planeación del proyecto y el proceso de construcción.

- **Desventajas**

- ✓ Características avanzadas la sintaxis de modelación requiere de notaciones que no poseen los desarrolladores promedio.
- ✓ Es un modelo costos (requiere personal capacitado, excelentes herramientas).
- ✓ Metodología rígida, no pensada para el bienestar de los especialistas informáticos, trabajo bajo presión.

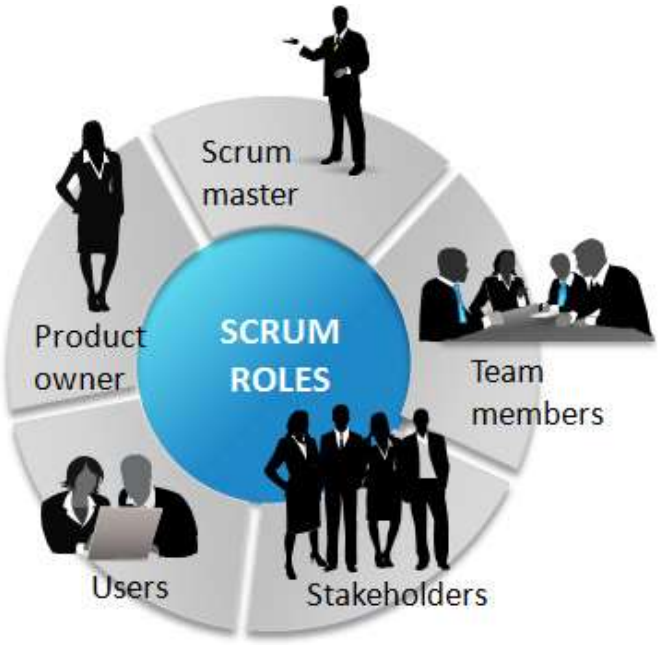
2.2.4 METODOLOGÍAS ÁGILES

Las Metodologías Ágiles o "ligeras" constituyen un nuevo enfoque en el desarrollo de software, la toma de decisiones se hace a corto plazo, de forma colaborativa, su orientación es a equipos de desarrollo pequeños, su flexibilidad ante los cambios y su ideología de colaboración, son algunas de sus fortalezas. Algunas metodologías Ágiles:

Scrum

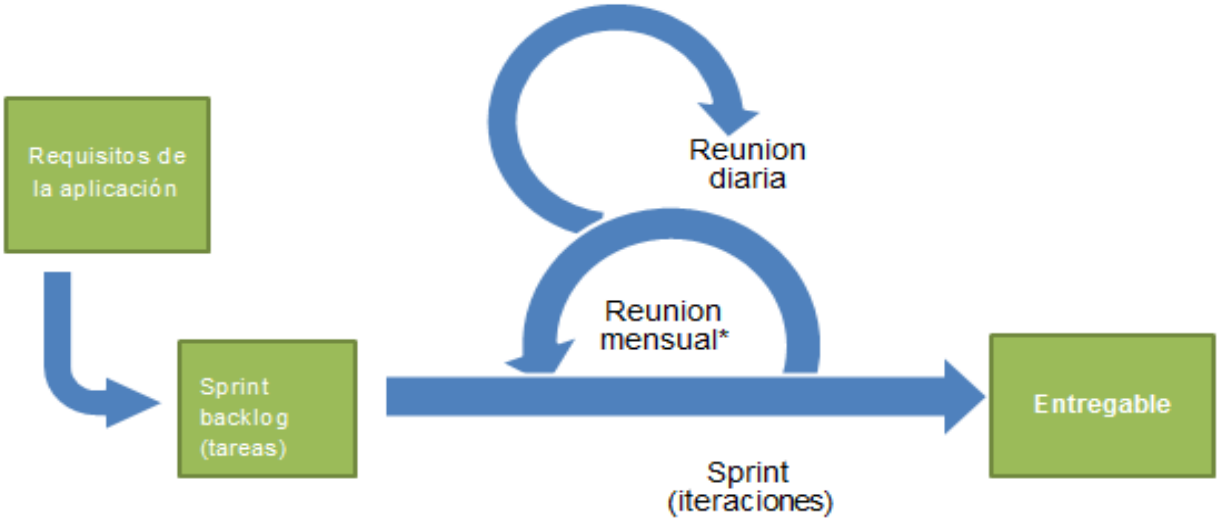
Es un modelo adaptado para desarrollo de software por Ken Schwaber en 1995, donde presentó "Scrum Development Process", como conjunto de prácticas y roles (Ver imagen 18), y que puede tomarse como punto de partida para definir el proceso de desarrollo (Ver imagen 19) que se ejecutará durante un proyecto. Ver Video: https://www.youtube.com/watch?v=PlLHc60egiQ

Imagen 18 Metodología Scrum



Fuente: http://www.cprime.com/static/ScrumRoles.jpg

Imagen 19 Sprint de Scrum



Fuente: https://es.wikipedia.org/wiki/Scrum#/media/File:Scrumm.PNG

Algunas ventajas y desventajas

- **Ventajas**

✓ Entrega de un producto funcional al finalizar cada Sprint.

✓ Posibilidad de ajustar la funcionalidad en base a la necesidad de negocio del cliente.

✓ Visualización del proyecto día a día.

✓ Alcance acotado y viable.

✓ Equipos integrados y comprometidos con el proyecto, toda vez que ellos definieron el alcance y se auto administran.

- **Desventajas**

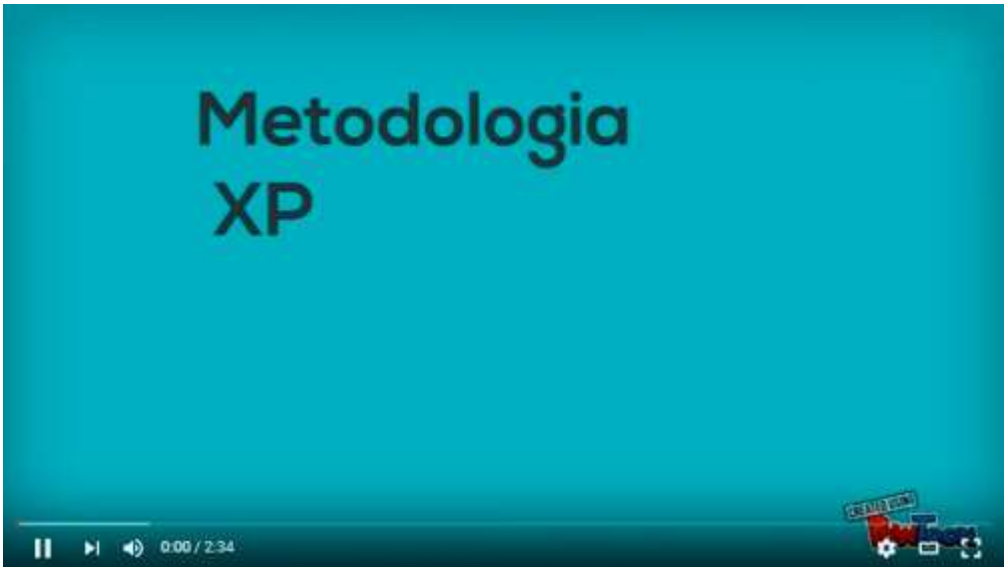
✓ No genera toda la evidencia o documentación de otras metodologías.

✓ No es apto para todos los proyectos.

✓ Algunas veces se hace necesario complementarlo con otras metodologías.

eXtreme Programming (XP)

Metodología formulada por Kent Beck, en 1999 que busca potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Ver video:



Metodologia xp Enlace

Imagen 20 eXtreme Programming (XP)

Fuente: http://es.slideshare.net/coesiconsultoria/4-desarrollo-gil-del-software

Algunas ventajas y desventajas

- **Ventajas**

✓ Facilidad de adaptarse tanto al desarrollo de sistemas pequeños como grandes.

✓ Optimiza el tiempo de desarrollo, permite realizar el desarrollo en parejas para complementar el conocimiento, el código es sencillo y entendible.

✓ Trabaja con poca documentación que se necesita para elaborar el desarrollo del sistema.

✓ Trabaja de la mano con el cliente.

Desventajas

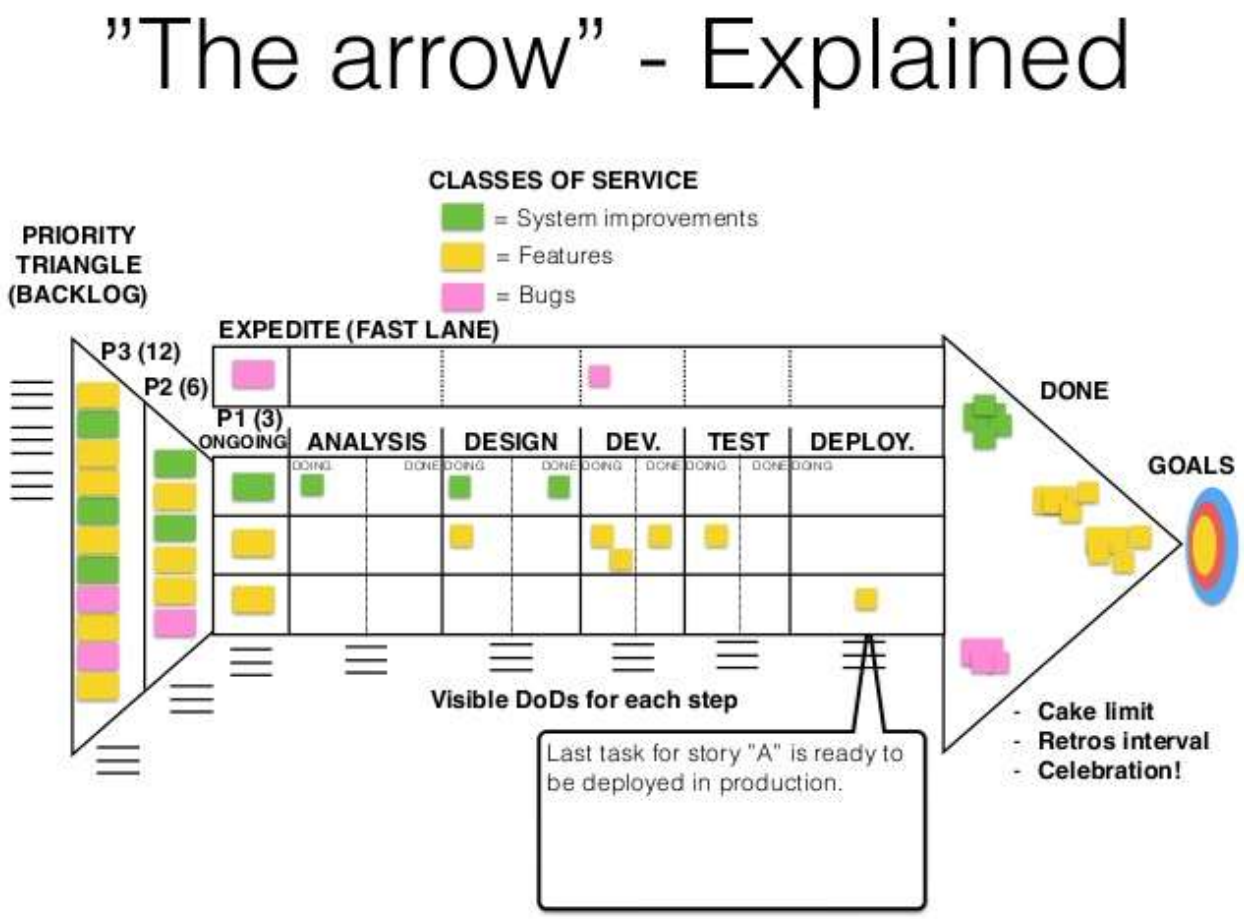
✓ El costo y tiempo no es definido desde el inicio del proceso, pues el sistema va creciendo con cada entrega que se le realiza al cliente.

- ✓ Se requiere de la presencia constante del cliente lo cual resulta difícil de lograr.
- ✓ Requiere mucho trabajo en equipo, a veces difícil de lograr.

Kanban

Es una metodología, especial para la gestión de tareas, buscando la priorización de las actividades, así como su seguimiento, tanto de las actividades como al equipo de trabajo, no es una metodología exclusiva para procesos de software, sino aplicable a cualquier tipo de proyecto. Ver video: https://www.youtube.com/watch?v=I-H-WXAX_oM

Imagen 21 Metodología Kanban



Fuente: http://image.slidesharecdn.com/thearrow-advancedkanbanboard-150621164413-lva1-app6891/95/the-arrow-advanced-kanban-board-4-638.jpg?cb=1434905565

Algunas ventajas y desventajas

- **Ventajas**

✓ Disminuir o eliminar los procesos acumulados.

✓ Cumplir los tiempos de entrega demandados por el cliente.

✓ Mejorar la calidad del producto por una mejor detección de los defectos del mismo.

✓ Evitar el manejo excesivo de materiales.

✓ Facilitar el control de la producción.

✓ Obtener un sistema de producción flexible según la demanda.

- **Desventajas**

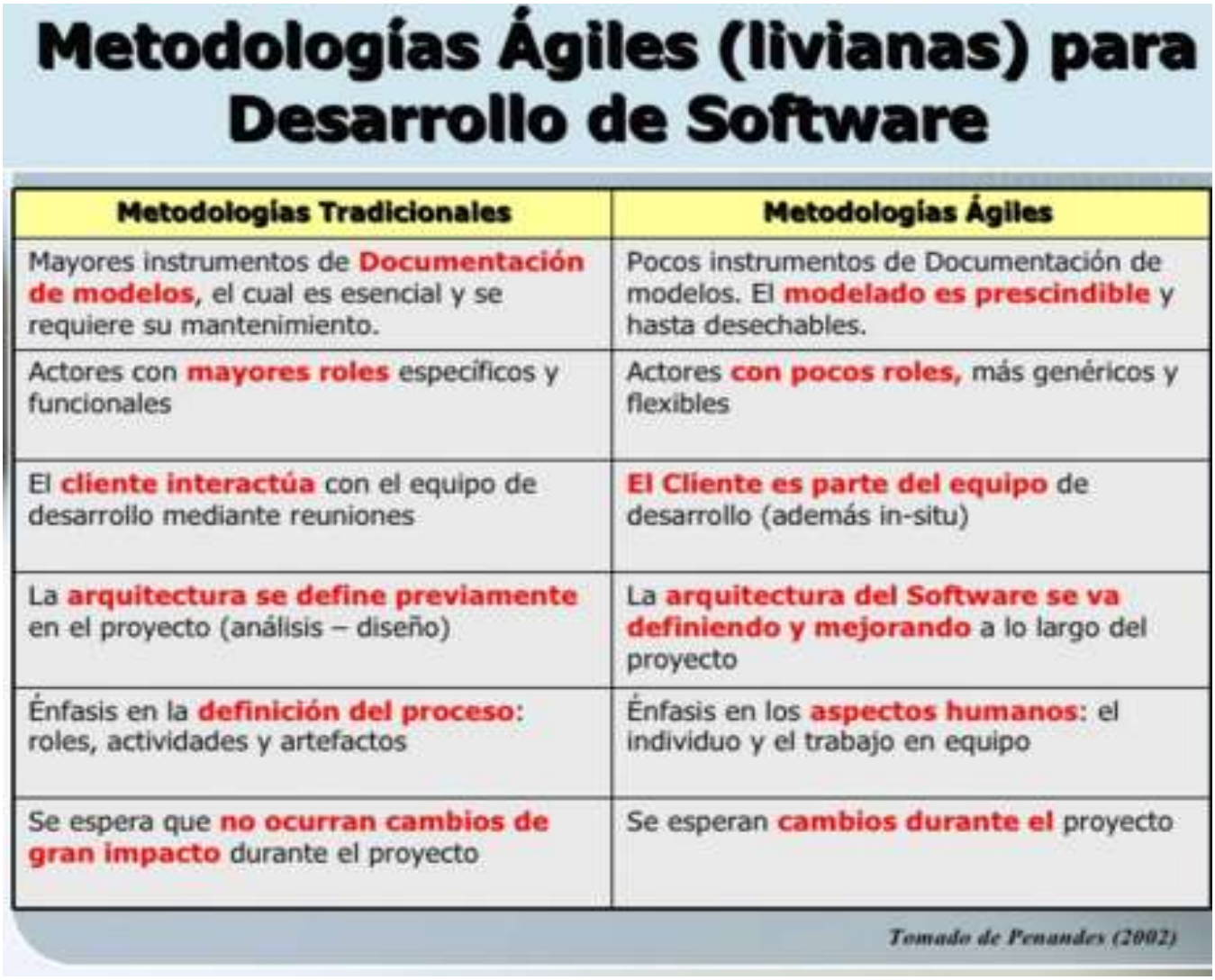
✓ No maneja previsión al riesgo, en caso de cambios profundos

✓ Es difícil de imponerles este método a los proveedores.

✓ No es muy aplicado en culturas occidentales.

Existen muchas empresas de desarrollo de software que no siguen ninguna metodología prescriptiva o ágil, de la forma como lo propone cada autor, sino que toman elementos de varias de ellas y conforman sus propias metodologías, acorde a su necesidad, a la necesidad del cliente y/o al tipo de proyecto. Ver imagen 22 sobre comparación de beneficios entre metodologías prescriptivas o tradicionales y metodologías ágiles.

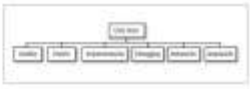
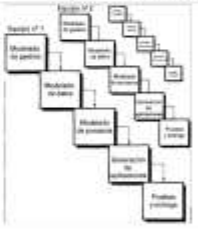
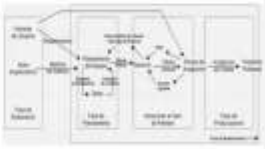
Imagen 22 Comparativo entre metodologías tradicionales y ágiles



Fuente:http://image.slidesharecdn.com/metologaagilesdesarrollosoftware-090711112252-phpapp02/95/metologa-agiles-desarrollo-software-xp-3-728.jpg?cb=1247311607

La siguiente imagen complementa la temática abordada en relación a diferentes metodologías para desarrollo de software, es importante anotar que las metodologías mencionadas arriba no son las únicas. Y que a medida que madura la ingeniería de software, irán apareciendo otras que lo único que buscan es ofrecer herramientas que faciliten la construcción de dicho producto. Ver imagen 23.

Imagen 23 Resumen sobre algunas metodología sobre desarrollo de software

MODELOS CICLO DE VIDA	CARACTERITICAS	VENTAJAS	DESVENTAJAS	GRAFICA
LINEAL	Este es el más básico de todos los modelos, y sirve como bloque de construcción para los demás modelos de ciclo de vida. La visión de este modelo para el desarrollo de software es muy simple: dice que el desarrollo de software puede ser a través de una secuencia simple de fases.	• Es sencillo. • Fácil de utilizar. • Planificación sencilla.	•Un error puede no encontrarse hasta la fase de validación lo cual puede ser muy costoso. •Los cambios introducidos durante el desarrollo pueden confundir al equipo profesional en las etapas tempranas del proyecto. •Hasta que el proyecto no este avanzado no se ven los resultados.	
CASCADA	Es uno de los primeros modelos de desarrollo de software que considera las diferentes actividades como fases separadas. Inicialmente propuesto por Royce en 1970. Al iniciar na nueva actividad debe esperarse a la finalización de la actividad anterior.	• Es un modelo sencillo y disciplinado. • Es fácil aprender a utilizarlo y comprender su funcionamiento. • Está dirigido por los tipos de documentos y resultados que deben obtenerse al final de cada etapa. • Ha sido muy usado y, por tanto, está ampliamente contrastado.	•Es necesario tener todos los requisitos al principio. •Si se han cometido errores en una fase es difícil volver atrás. •No se obtiene el producto hasta el final y esto causa: Impaciencia en el cliente, si hubo algún error en el análisis no se descubre hasta la finalización del producto, y es mas lento que los demas modelos.	
SASHIMI	El ciclo de vida tipo Sashimi podría ser considerado como una variación del ciclo de vida en cascada, en el cual las diferentes etapas pueden ser solapadas, permitiendo así aumentar la eficiencia mediante la retroalimentación entre las etapas. El nombre Sashimi deriva de una comida oriental que viene representado en rodajas de pescado crudo en Japón.	•No necesita tanta documentación debido a la continuidad de las fases. •Permite devolverse. •Permite aumentar la eficiencia debido a la retroalimentación de capas.	•Existe la dificultad de identificar el inicio y el fin de cada etapa. •Al hacer las cosas en paralelo si hay problemas de comunicación pueden surgir inconsistencias. •Mas difícil de controlar el progreso del proyecto debido a que los finales de cada fase ya no son un punto de referencia claro.	
ESPIRAL	En el modelo Espiral el software se construye en una serie de versiones incrementales. En las primeras iteraciones la versión incremental podría ser un modelo en papel o bien un prototipo. En las últimas iteraciones se producen versiones cada vez más completas del sistema diseñado.	•No necesita una definición completa de los requisitos para empezar a funcionar. •El riesgo en general es menor, porque si todo se hace mal, solo se ha perdido el tiempo y recursos invertidos en una iteración (las anteriores iteraciones están bien). •El riesgo de sufrir retrasos es menor, ya que al identificar los problemas en etapas tempranas hay tiempo de subsanarlos. •Puede adaptarse y aplicarse a lo largo de la vida del software. •Reduce los riesgos antes de que se conviertan en problemas. •Controla muy bien los riesgos y mientras mas iteraciones se realicen, menos riesgos habrá. •Monitorea y controla los riesgos continuamente.	• Necesita de la participación continua por parte del cliente. • Puede resultar difícil convencer a algunos clientes de que el enfoque evolutivo es controlable. • Solo resulta aplicable para proyectos de gran tamaño. • Supone una carga de trabajo adicional, no presente en otros ciclos de vida. • Requiere una considerable habilidad para la evaluación y resolución del riesgo, y se basa en esta habilidad para el éxito. • Si un riesgo importante no es descubierto y gestionado, indudablemente surgirán problemas. • Es bastante complicado de realizar y su complejidad puede incrementarse hasta hacerlo impracticable.	
DRA (Desarrollo rápido de aplicaciones)	Es un modelo de proceso del desarrollo de software lineal secuencial que enfatiza un ciclo de desarrollo extremadamente corto.	•Los entregables pueden ser fácilmente trasladados a otra plataforma. •Visibilidad temprana. •Menor Codificación manual. •Mayor involucramiento de los usuarios. •Ciclos de desarrollo mas pequeños. •Interfaz gráfica estándar.	•Los entregables pueden ser fácilmente trasladados a otra plataforma. •Visibilidad temprana. •Menor Codificación manual. •Mayor involucramiento de los usuarios. •Ciclos de desarrollo mas pequeños. •Interfaz gráfica estándar.	
XP (XTREME PROGRAMING)	La metodología XTREME PROGRAMING pretende que el desarrollo de un proyecto de software sea un desarrollo ágil, disciplinado y aporte soluciones sencillas, basado en las relaciones interpersonales y la velocidad de reacción.	• Se trabaja estrechamente con el cliente. • Usertórias (Historias de usuario), historias escritas por el cliente que describen los escenarios sobre el funcionamiento del software. • Define un estándar en el tipo de codificación. • Los programadores trabajan en parejas. • Las pruebas o testing se realizan durante cada interacción. • El código puede ser manipulado por todo el equipo. • Cada dos semanas se hace entrega al cliente de una versión. • Programación organizada. • Menor taza de errores. • Satisfacción del programador.	• Es recomendable emplearlo solo en proyectos a corto plazo. • Alias comisiones en caso de fallar.	
RUP (PROCESO UNIFICADO DE DESARROLLO)	Es un proceso de ingeniería de software que suministra un enfoque para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta y de mayor calidad para satisfacer las necesidades de los usuarios que tienen un cumplimiento al final dentro de un límite de tiempo y presupuesto previsible. El RUP mejora la productividad del equipo ya que permite que cada miembro del grupo sin importar su responsabilidad específica pueda acceder a la misma base de datos incluyendo sus conocimientos. Esto hace que todos compartan el mismo lenguaje, la misma visión y el mismo proceso acerca de como desarrollar un software.	• Promueve la reusabilidad. • Reduce la complejidad del mantenimiento (extensibilidad y facilidad de cambios). •Riqueza semántica. •Disminuye la brecha semántica entre la visión interna y la visión externa del sistema. •Facilita la construcción de prototipos. •Reutilización. El diseñador piensa en términos del comportamiento de objetos y no en detalles de bajo nivel. •Confiabilidad, Integridad y Estabilidad. Mantenimiento más sencillo. Modificaciones locales.	• Por el grado de complejidad puede no resultar muy adecuado. • El RUP es generalmente mal aplicado en el estilo cascada. • Requiere conocimientos del proceso y de UML.	

Fuente: elaboración propia.

2.2.5 LAS BUENAS PRÁCTICAS EN LA INGENIERÍA DEL SOFTWARE

El concepto de "Las buenas prácticas", se debe a que la Ingeniería del software es una disciplina relativamente nueva que ha aprendido de sus experiencias o de las experiencias de otras empresas que se dedican al desarrollo de software y que a través de la historia han buscado mejorar sus procesos o prácticas en la construcción del software, buscando la satisfacción del cliente o usuario, lo que les ha obligado a buscar estrategias, metodologías prescriptivas, ágiles o conformar sus propias metodologías, ya que en cuestión de construcción de software, no existe la última palabra de cómo se debe construir. Existe normas internacionales que hacen alusión al software y da parámetros, sobre QUÉ se debe buscar, construir, qué resultados se deben esperar, a lo que cada empresa debe adaptar sus metodologías buscando el CÓMO cumplir con dichas normatividades y necesidades de software. Ver video Metodología ágiles: adaptando la Ingeniería del software a los negocios,

Javier Garzas - Metodologías ágiles: adaptando la ingeniería del software a los negocios Enlace

2.3 TEMA 3: CONTEXTO DE LOS REQUERIMIENTOS Y ACTORES DEL PROCESO DE INGENIERÍA DE SOFTWARE

Para el Ingeniero de Sistemas es importante el abordaje de los requerimientos o necesidades del cliente o usuario, ya que se constituyen en la etapa más importante del proceso de Ingeniería de Software, en relación a que las principales fallas que se presentan en el proceso de construcción y en el producto terminado, está relacionado con el deficiente manejo de los requerimientos, lo que puede llevar a reproceso, que repercuten directamente en sobre costos, consumo de recursos excesivos, incumplimiento de tiempos y baja funcionalidad del producto, entre otros.

2.3.1 ROLES EN RELACIÓN AL SOFTWARE

Es importante aclarar que existen dos tipos de usuarios, usuarios informáticos y no informáticos, donde los usuarios informáticos son aquellos que se relacionan directamente con la construcción del producto de software desempeñando roles de expertos en el ciclo de vida del software, especialmente en actividades relacionadas con su construcción o mantenimiento; en el caso de los usuarios no informáticos, éstos tienen que ver con aquellas personas que utilizan el producto construido para agilizar procesos a nivel laboral, académico, diversión, entre otros.

Roles de los especialistas en desarrollo de software	
Líder de proyecto: Persona que tiene a su cargo la planeación, ejecución, evaluación y control del proyecto de software y es el responsable de que el software cumpla con las necesidades del cliente y los estándares de calidad.	*Imagen 24 Líder de proyectos*  *Fuente:http://www.projectopen.es/wp-content/uploads/2013/04/Gestion-Proyectos.jpg*
Analista: Persona que se encarga de tomar los requerimientos del cliente y convertirlos en requisitos de software, buscando que sean lo suficientemente claros con el fin de evitar fallas de funcionalidad en el producto final.	*Imagen 25 Analista*  *Fuente:http://www.dablones.org/wp-content/uploads/2014/06/service_pianificare_pop.jpg*

Diseñador: Persona encargada de diseñar las arquitecturas del como vistas de usuario, interfaz gráfica, estructuración del código, estructuración de almacenamiento de datos, entre otros diseños que se requieran.

Imagen 26 Diseñadora

Fuente:http://www.dotuk.net/wp-content/themes/dotuk/timthumb.php?src=http://www.dotuk.net/wp-content/uploads/2012/12/web-developer-colour.png&h=342&w=424&zc=1

Desarrollador: Persona encargada de la codificación del programa, apoyado en los lenguajes de programación y demás herramientas como modeladores, bases de datos, así como los requisitos de software, cuyo resultado final será un producto funcional acorde a las necesidades del cliente.

Imagen 27 Desarrollador

Fuente:http://www.iprojectpartners.com.bo/_/rsrc/1421436002341/software-develop.jpg?height=213&width=320

Documentador: Persona encargada de documentar, realizando manuales técnicos (cómo se construye el software), manuales de usuario (cómo funciona cada componente funcional del producto terminado) y manual de implementación, donde aparecen las indicaciones que debe tener presente el usuario final para utilizar el producto como por ejemplo tipo de equipo que requiere para que el software funcione, paso a paso de instalación entre otras.

Imagen 28 Documentador

Fuente:http://4.bp.blogspot.com/-hWfhbYRnxqU/UxETYmULYeI/AAAAAAAAAOY/gjct-YEB2RI/s1600/front1.jpg

Testing: Persona encargada de realizar pruebas al producto, verificando su funcionalidad, acorde a las solicitudes del cliente, así como las características de calidad como por ejemplo seguridad, usabilidad, funcionalidad, confiabilidad, eficiencia, mantenibilidad, portabilidad.

Entre otros roles que de acuerdo al tipo de empresa y proyecto se puedan asignar.

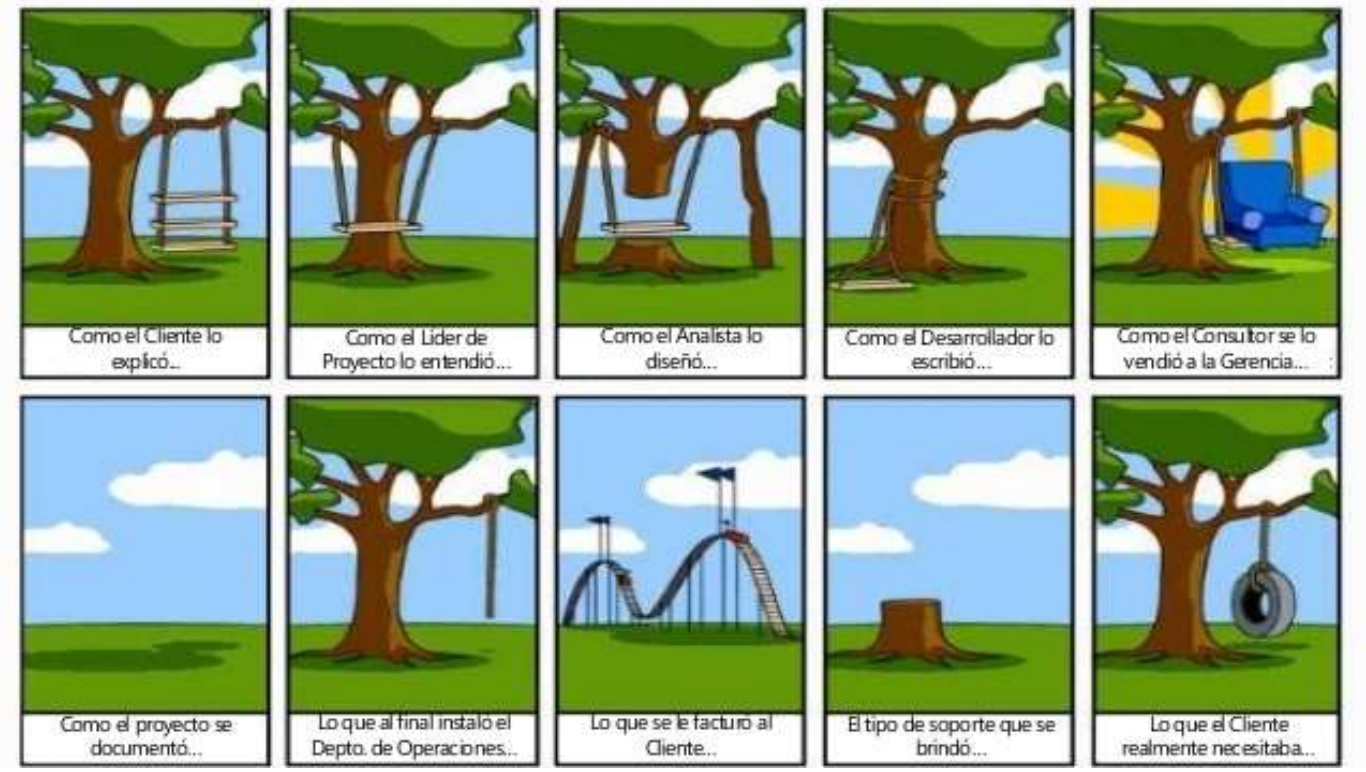
Imagen 29 Testing

Fuente:http://www.liderdeproyecto.com/articulos/imagenes/software-testing.jpg

2.3.2 LOS REQUERIMIENTOS DE SOFTWARE

La construcción de un software, nace de una necesidad de un cliente o usuario o de una oportunidad de automatizar un proceso o también como iniciativa de una persona o empresa en proponer un nuevo producto informático. Es así como los requerimientos, se constituyen en la etapa más importante del proceso de ingeniería del software, ya que de un buen requerimiento se desarrolla un producto de calidad y de un mal requerimiento se incrementa, durante todo el proceso el problema, generando pérdida de tiempo y alto consumo de recursos. Observar imagen 30.

Imagen 30 Fallas en el proceso de Ingeniería de Software



Fuente: http://image.slidesharecdn.com/desarrollodesoftwaremoderno-130218013248-phpapp02/95/desarrollo-de-software-moderno-6-638.jpg?cb=1361645598

En las anteriores imágenes, se puede apreciar el ciclo de vida de desarrollo de software, así como las fallas que se han presentado y que aún se siguen presentando en el proceso de construcción del software. Se realiza una breve explicación de cada una de ellas iniciando el recorrido de izquierda a derecha:

- **Primera imagen:** el cliente realiza un requerimiento de software, acorde a las necesidades de automatizar procesos, explicándole al líder del proyecto de software lo que requiere que se le construya.
- **Segunda imagen:** El líder del proyecto a escuchar al cliente, entiende que debe construir lo que muestra la segunda diapositiva, existiendo un desfase, entre lo que pidió el cliente y lo que entendió el líder del proyecto de software.
- **Tercera imagen:** en la etapa del análisis y diseño del software, al evolucionar el proceso de construcción del software, se entiende que lo que se debe elaborar es lo que muestra la tercera imagen, al no existir claridad suficiente.
- **Cuarta imagen:** cuando se pasa a la etapa de codificación, el resultado se encuentra suficientemente alterado en relación a la solicitud del cliente de la primera imagen.

- **Quinta imagen:** los asesores, entienden que lo que se debe construir debe tener determinadas características, tergiversando aún más el proceso.
- **Sexta imagen:** la documentación de cómo se está realizando el proceso es inexistente, lo que dificulta saber en definitiva de qué forma se está construyendo el software, lo cual se agrava cuando la persona encargada de determinada fase de construcción del software, debe abandonar el proyecto, ya que quien retome el proceso no tiene documentación para orientarse.
- **Séptima imagen:** La instalación del software en la máquina del usuario final, no obedece a la necesidad real, ya que durante las diferentes etapas para la construcción del software, se trabajó con datos erróneos, generando un programa no funcional para la necesidad del cliente.
- **Octava imagen:** debido a todas las fallas presentadas, reproceso, pérdida de tiempo, subutilización de recursos (personas, equipos de cómputo, horas extras, servicios públicos, incapacidades debido a stress laboral, entre otros), el costo de producción sube por lo tanto el costo de producto aumenta, generando inconformidad en el cliente debido al incremento del precio de venta en la facturación.
- **Novena imagen:** el soporte que se da al cliente después de implementado el software, es ineficiente debido a las fallas presentadas desde el inicio de la construcción, quedando el cliente con un grado alto de desmotivación frente a la ineficiencia del producto.
- **Décima imagen:** en la última imagen se muestra que ni siquiera el cliente tenía claro lo que realmente necesitaba y que desde el inicio del proceso se presentaron fallas gravísimas que dan como resultado la construcción de un producto de software para necesidades equivocadas.

Lo anterior muestra que, para construir un producto de software de calidad, es necesario tener claridad sobre cada una de las etapas de la ingeniería del software, donde una de las etapas más importantes es la que corresponde a la ingeniería de requerimientos, los cuales deben tener la suficiente validación, antes de avanzar a otras etapas, ya que en caso contrario se generan reproceso, dando como resultado productos de software ineficientes y de baja calidad.

2.4 TALLER DE ENTRENAMIENTO UNIDAD 1

Nombre del taller: Taller 1	Modalidad de trabajo: Aprendizaje basado en problemas

Actividad previa: Estudio de la Unidad 1 del módulo

Describa la actividad:

De acuerdo a la situación problemática que se presenta:

1. Elaborar un diagrama donde se plasme el macro sistema de la empresa, explicando los elementos que conforman el sistema de información (Entrada, proceso, Salida, Realimentación, Ambiente).
2. Elaborar un diagrama donde se muestre cada uno de los subsistemas de la empresa, explicando detalladamente a su vez los elementos que lo conforman (Entrada, proceso, Salida, Realimentación, Ambiente).
3. Según la anterior información. Redactar el problema (no dar la solución, sólo hablar del problema).
4. Para mañana a la 08:00am, usted tiene entrevista con el gerente, con el fin de hablar al respecto, por lo tanto debe llevar la entrevista preparada. Elaborar el formato de la entrevista con las preguntas que hará, en este primer encuentro.

Situación problemática

Un centro de manufactura. Empezó a funcionar hace 4 años. Es muy organizado, ya que tiene organigrama (departamentos), estatutos, reglamento, etc.

El departamento de Presupuestos ha afirmado que la Empresa no ha tenido muchas ganancias, pero tampoco ha generado pérdidas. El departamento de producción tiene problemas con sus empleados, ya que entre estos se ha presentado altercados de índole personal y la situación en la planta se ha tornado inmanejable. El departamento de Nómina no ha entregado a tiempo sus pagos, ya que los cheques, nunca están a tiempo y esto ha generado un poco de inconformismo con los empleados en general, el departamento de Ventas tiene muchos pedidos, pero como producción ha tenido problemas con los proveedores, por el no pago de sus facturas, no ha podido cumplirle con los pedidos, debido a esto han quedado mal con los clientes.

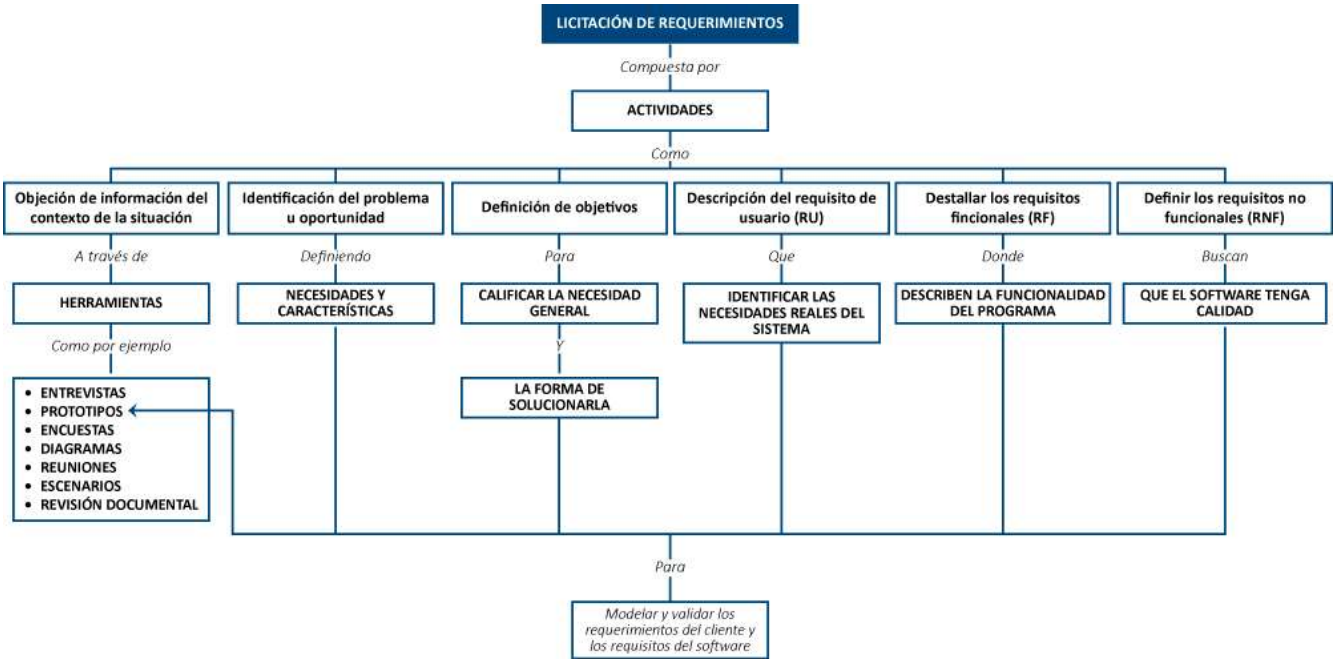
El Gerente está muy confundido, con la situación actual de su Empresa y requiere de sus servicios ya que necesita según él de un sistema informático bien estructurado que le ayude a solucionar el caos que está generando el no pago a tiempo.

Nota la empresa cuenta con una red distribuida por todos los departamentos, excelente Hardware. La información normalmente la manejan en la hoja electrónica Excel, donde se procesa la nómina, se llevan los Costos y Presupuestos, Pedidos, Control de Ventas, Cartera entre otros.

3 UNIDAD 2: ELICITACIÓN DE REQUERIMIENTOS

En relación a la elicitación, ésta consiste en pasar información de una persona a otra o de un medio a otro con el fin de buscar su transformación, es importante aclarar que dicho término es utilizado en el proceso de ingeniería de software como un proceso indispensable que tiene como fin clarificar los requerimientos del cliente, con el fin de convertirlos en requisitos de software.

Para gestionar el proceso de elicitación de requerimientos, es necesario tener conocimiento del entorno en el cual se mueve la necesidad de automatizar los procesos a través de medios informáticos, donde para ello es necesario utilizar una serie de herramientas, que permiten clarificar tal necesidad, con el fin de identificar el problema y detallarlo de forma adecuada, para que los objetivos del proyecto se orienten hacia metas reales, que den cumplimiento a los requisitos del usuario (lo que el usuario necesita), requisitos funcionales del producto de software (lo que el software ejecutará), así como los requisitos no funcionales (lo que debe llevar sin que el usuario lo pida), relacionados con factores de calidad que debe cumplir dicho producto, donde todo lo anterior debe ser validado, antes de pasar a la fase de diseño y construcción del software.



3.1 TEMA 1: DISEÑO DE HERRAMIENTAS PARA LA ELICITACIÓN E IDENTIFICACIÓN DEL REQUERIMIENTO

Para el manejo de los requerimientos es importante que el Ingeniero de Sistemas, esté en capacidad de diseñar, apropiarse y utilizar las herramientas que requiere para la toma adecuada de dichos requerimientos, entre las que podemos mencionar (encuesta, entrevista, revisión documental, tormenta de ideas, casos de uso, entre otras), que son elegidas de acuerdo a variables como por ejemplo el tipo de cliente, el tipo de software requerido, entre otros.

3.1.1 ORIGEN DE LOS REQUERIMIENTOS

El origen de los requerimientos, surge de diferentes fuentes y necesidades que buscan satisfacer al cliente o usuario. Un requerimiento de software, normalmente se da por la necesidad de automatizar uno o varios procesos, ya sea porque se viene realizando de forma manual o porque se desea mejorar la forma como se lleva actualmente.

El software ha tomado fuerza, en cualquier ámbito como por ejemplo el académico, profesional o personal y requerido en todos los ambientes como el económico, político, social, cultural, entre otros, donde en definitiva busca agilizar tareas, dinamizar la forma como se vienen realizando las cosas, todo ello debido a la misma evolución de la sociedad, donde se requiere información en tiempo real, pero que dicha información, posea validez para la toma de decisiones, siendo éste producto la herramienta requerida para ello.

3.1.2 HERRAMIENTAS PARA LA ELICITACIÓN

La elicitación de los requerimientos, es una de las actividades más importantes de la Ingeniería de Requisitos de software y comprende la identificación del contexto donde se mueve el problema o situación a solucionar a través de medios informáticos, así como el origen de la necesidad, permitiendo detectar claramente el problema que se desea resolver. Según Manies, Nikual (2011), “La Elicitación de Requisitos –ER– es la piedra angular en el desarrollo de proyectos software y tiene un impacto muy alto en el diseño y en las demás fases del ciclo de vida del producto. Si se realiza apropiadamente, puede ayudar a reducir los cambios y las correcciones en los requisitos. Además, la calidad de la elicitación determina la exactitud de la retroalimentación al cliente acerca de la integridad y validez de los requisitos”.

El siguiente video proporciona aportes importantes sobre la e licitación de requerimientos de software: Ver link:

Elicitación - Ingeniería de Requerimientos Enlace

Entre algunas de las herramientas para abordar la elicitación de requerimientos tenemos:

- **Entrevista:** conversación que se tiene entre el entrevistador y el entrevistado, la cual se basa en una serie de preguntas que son orientadas por el entrevistador y sobre las que la persona entrevistada da su respuesta o su opinión. Para que en una entrevista se tenga éxito, es importante que el entrevistador realice una serie de actividades previas a la entrevista, como por ejemplo indagar por su cuenta aspectos relevantes sobre el contexto del tema a tratar, elaboración de un cuestionario de máximo 15 preguntas, donde se abarquen los principales temas, acorde al objetivo de la entrevista, llegar puntual a la entrevista y tener información sobre el entrevistado (a).
- **Encuesta:** está relacionada con una estructura que posee preguntas que tienen como fin indagar sobre temas que los encuestados conocen y que son de interés para quien dirige dicha encuesta. Una encuesta debe poseer preguntas claras y concretas, de tal forma que sean fácilmente resueltas y posteriormente tabuladas y analizadas, para la obtención de información útil.
- **Tormenta de ideas (Brainstorming)**: Este es un modelo que se usa para generar ideas, cuyo origen se relaciona con un grupo de personas que se encuentren familiarizados con el tema de interés, la intención en su aplicación es la de generar la máxima cantidad posible de requerimientos para el sistema, no hay

que detenerse en pensar si la idea es o no del todo utilizable, la intención de este ejercicio es generar, en una primera instancia, muchas ideas, que serán depuradas de acuerdo a la necesidad informática, basada en lineamientos de la ingeniería de software.

Los participantes son personas que son afectadas por el resultado del sistema y normalmente cumplen con los siguientes interrogantes:

- ¿Quiénes usarán el sistema?
- ¿Quiénes conocen el proceso que gestiona el sistema?
- ¿Quiénes invierten en el sistema?
- ¿A quiénes afecta el sistema?
- ¿Quiénes mantienen el sistema?

- **Revisión documental (Arqueología de documentos),** consiste en la revisión de la documentación utilizada por la empresa; por ejemplo, boletas, facturas, remisiones, recibos de caja, formatos de nómina, etc. Esta herramienta permite obtener información de la forma como se llevan actualmente los procesos e información. Con el fin de validar dichos documentos, es aconsejable realizarse preguntas como por ejemplo: ¿Cuál es el propósito de este documento? ¿Quién lo usa? ¿Por qué? ¿Para qué? ¿Cuáles son las tareas que realizan con este documento? ¿Se puede encontrar una relación entre los documentos? ¿Cuál es el proceso que realiza la conexión? ¿Cuál es el documento que da más problemas a los usuarios? Lo anterior permite obtener información real y útil para comprender los requerimientos del cliente y da una idea mucho más clara de las necesidades que se desean solucionar a través de un producto de software.
- **Prototipo:** Es una herramienta muy práctica para dar mayor claridad sobre los requerimientos, especialmente cuando el cliente no está seguro, en relación a lo que desea, por lo que el experto informático proporciona simulaciones del posible producto (pantallazos, informes, menú), que luego son presentados al usuario final, permitiendo conseguir una importante retroalimentación, sobre el requerimiento. Es importante aclararle al usuario final que dicho prototipo no corresponde al software definitivo.
- **Diagramas:** para comprender los requerimientos aparecen varios diagramas como por ejemplo el diagrama de actividad, diagrama de casos de uso, diagrama espina de pescado, matriz DOFA, los cuales son utilizados para comprender el entorno en el cual se encuentra el negocio, describiendo su funcionamiento interno y su relación con el ambiente, así como la interacción entre procesos y actividades. Con estas herramientas se pueden analizar los flujos de información que intervienen en los

distintos procesos, buscando la forma en la cual el sistema puede ayudar a obtener mayor valor para la actividad o conjunto de actividades estudiadas.

- **Videos:** a la hora de comprender un requerimiento del cliente, es importante servirse de varias herramientas, entre las que se tienen los videos, ya que existen procesos difíciles de entender, por sencillos que sean, teniendo en cuenta que el informático no maneja todas las áreas del conocimiento. Es así como el video permite ver y analizar en detalle ese proceso la cantidad de veces que sea necesario, ya que se captura el paso a paso del trabajo, lo que evita que el experto en informática imponga sus expectativas y preferencias.
- **Observación:** La observación directa es importante y permite evitar malos entendidos a la hora de levantar los requerimientos, vamos a suponer que debes explicar por teléfono (sin cámara), la forma como se debe amarrar unos zapatos, utilizando los cordones. Ésta explicación puede tardar mucho tiempo y los resultados pueden ser insatisfactorios. En el caso que el aprendiz estuviese observando el proceso de cómo realizarlo, sería mucho más eficiente el aprendizaje, ahorrándose tiempo en dicha actividad.

  Por lo tanto, es importante que la persona encargada de levantar los requerimientos del software observe directamente la dinámica del sistema, la evolución de los procesos, la cultura de la organización, buscando analizar el problema para dar soluciones de software eficientes.

De igual forma es importante aclarar que existen muchas más herramientas para la elicitación de los requerimientos, y que éstas dependen del tipo de cliente, tipo de necesidad, tipo de empresa, la forma de realizar el trabajo de la empresa desarrolladora y que se pueden utilizar varias herramientas para el entendimiento del requerimiento del cliente, donde lo importante es suministrar información clara, coherente y útil para la construcción de un software que satisfaga las necesidades del cliente. Leer el artículo "La elicitación de requisitos en el contexto de un proyecto software" http://web.usbmed.edu.co/usbmed/fing/v2n2/v2n2a4.pdf.

Al elegir la(s) herramienta(s), es necesario preparar la forma como se van a utilizar, evitando improvisaciones y resultados inesperados que pueden ser contraproducentes.

3.1.3 SITUACIÓN PROBLEMÁTICA COMO ESTRATEGIA DIDÁCTICA Y PEDAGÓGICA PARA APLICAR EL PROCESO DE INGENIERÍA DE REQUISITOS DE SOFTWARE

En el desarrollo del módulo, se trabajará con una situación problemática, sobre la cual se aplicará el proceso de ingeniería de software, especialmente el relacionado con la ingeniería de requisitos, buscando con ello combinar elementos teóricos y prácticos, que permitan al estudiante plasmar en dicha situación herramientas de Ingeniería del software, siendo lo anterior una aproximación real entre la academia y el sector productivo.

Situación problemática:

En la facultad de Ciencias Básicas e Ingeniería de la Corporación Universitaria Remington, se tiene dificultad con la gestión de los proyectos de grado, ya que se presenta inconsistencias en la información, existen reproceso, pérdida de información, ineficiencia a la hora de dar respuestas sobre el estado de los proyectos, pérdida de tiempo. La forma como se viene manejando es a través de la hoja electrónica Excel complementado con planillas manuales, es importante tener presente que los proyectos de grado, pueden ser tipo Monografías, Investigación, Software entre otros, el equipo para desarrollar el trabajo puede ser máximo de tres estudiantes, cuyo equipo puede estar conformado por estudiantes de diferentes carreras. Además, se maneja la información de los Asesores de dichos proyectos, así como el tema o los temas sobre los cuales trata el proyecto (Facturación, El Negocio Virtual, Negocios del siglo XXII, Literatura Colombiana, Ecología, entre otros), se debe tener en cuenta que cada estudiante solo debe de figurar en un proyecto de grado y estar matriculado en una sola carrera. Debe registrarse la fecha, en la cual se le dio o dará asesoría al proyecto. En cualquier momento se requieren informes y consultas para el decano, asesores, estudiantes, secretaria de la facultad y no siempre se tiene la información a la mano, generándose inconformidad en muchas ocasiones.

Entre algunas de las solicitudes que se requieren en relación a los proyectos de grado son:

- Nombre de los proyectos con sus Integrantes.
- Proyecto con sus respectivos asesores, y sus fechas de asesoría, tanto para los aprobados, como para los que se encuentran en proceso.
- Consulta por parte de los estudiantes para visualizar los proyectos por tema, por tipo de proyecto, para que sirva como referencia a proyectos posteriores.
- Consulta sobre las fechas de asesorías de los proyectos

- Consultar sobre proyectos aprobados, antes de realizar el proceso de graduación, entre otras.

Fuente: elaboración propia.

Siguiendo con la situación anterior, usted debe realizar una entrevista al decano(a) de la facultad de Ciencias Básicas e Ingeniería, mañana a las 08:00am, para lo cual debe prepararse.

3.1.4 ALGUNAS ESTRATEGIAS PARA PLANEAR UNA ENTREVISTA

- **Investigación previa:** Investigar todo lo que más se pueda sobre dicha empresa, ya sea por personas que tengan relación directa con ella, por páginas web, por revistas, entre otras de manera que se tenga suficiente información como por ejemplo reseña histórica, misión, visión, objetivos, organigrama y demás información que se pueda recopilar.
- **Diseño previo:** Preparar un diseño o formato de entrevista, lo cual dará seguridad, confianza y asegurará información puntual sobre los temas a tratar, además servirá de guía para orientar la conversación, aprovechando al máximo el tiempo destinado para ésta.
- **Puntualidad:** Presentarse a la entrevista en la hora pactada para ello, ni antes, ni después.
- **Pedir autorización para grabar:** Si se desea grabar la entrevista, se debe solicitar la autorización al entrevistado(a)
- **Excelente presentación personal:** Es importante aclarar que la presentación personal, da seguridad para éste tipo de actividades.

3.1.5 LINEAMIENTOS SUGERIDOS PARA EL DISEÑO DEL FORMATO DE LA ENTREVISTA:

Para el diseño de una entrevista, es necesario tener claro el objetivo que se quiere alcanzar con ésta, de igual forma el tiempo establecido para realizarla, pero especialmente un derrotero que permita guiar la conversación con el entrevistado, de ésta forma se evita tener tiempos muertos, donde ambos se queden en silencio por no tener unos temas puntuales para abordar, es así como el tiempo que se tenga destinado para el desarrollo de la entrevista, debe ser aprovechado al máximo. Ver formato propuesto para entrevista.

FORMATO DE ENTREVISTA	
Objetivo: Identificar los requerimientos del cliente, en relación a la gestión de los proyectos de grado para la facultad de Ciencia Básicas e Ingeniería de la Corporación Universitaria Remington.	
Fecha:	Hora Inicial: _____Hora final:_____
Entrevistado(a)_____	Cargo:_____
Entrevistador(a)_____	Cargo:_____
Temáticas a abordar: 1. Historia de la empresa (Esto con el fin de contextualizar la empresa) 2. Organigrama de la empresa (para visualizar su organización, áreas, sedes) 3. Infraestructura tecnológica (equipos de cómputo, redes, software) 4. Requerimientos de software (procesos a sistematizar, tipo de producto requerido, éste tema es el principal.) 5. Personal que puede apoyar el proceso (personal relacionado con los procesos y que utilizará el software) 6. Cualificación del personal en herramientas informáticas (para planear posibles capacitaciones) 7. Tiempo en el cual requiere el software (para planear el proceso de construcción del software) 8. Presupuesto para la construcción del software (para analizar costos y presupuesto para el proyecto) 9. Otras preguntas que surjan (las de arriba son parámetros generales) 10. Observaciones:_____ _____ _____ _____.	

Fuente: Elaboración propia.

3.1.6 IDENTIFICACIÓN DEL REQUERIMIENTO

Para realizar la identificación del requerimiento, se debe tener contacto directo con el personal relacionado con el proceso a sistematizar durante toda la construcción del software, no sólo en la etapa inicial del requerimiento, ello garantiza la detección de errores en etapas tempranas, ya que, si se detecta un error en etapas posteriores, resultaría su corrección mucho más costosa que si se detecta en la etapa donde está sucediendo.

La etapa de identificación, también se conoce como la etapa donde se realiza la extracción del requerimiento, la cual consiste en descubrir el problema que el sistema debe resolver, los diferentes servicios que el sistema de prestar, las restricciones, la claridad sobre las reglas del negocio, se descubren las limitaciones técnicas o tecnológicas para cumplir con algunos requerimientos, etc. Pero, en definitiva, descubrir los requerimientos del sistema no sólo implica preguntar a las personas qué quieren: es un proceso delicado que involucra comprender el domino de aplicación, es decir, obtener un conocimiento del área general de aplicación del sistema; comprender el problema en sí, lo que implica que se debe extender y especializar el conocimiento sobre el dominio general para que se aplique al cliente en particular; comprender el negocio, por tanto, se debe entender en profundidad cómo es que este sistema interactuará afectará a las partes del negocio que estarán involucradas y cómo puede contribuir a lograr las metas de la empresa; finalmente, comprender las necesidades y restricciones de los usuarios del sistema, en particular, se deben entender los procesos de trabajo que se supone que el sistema apoyará y el rol de cualquier otro sistema que actualmente se involucre en dichos procesos.

Es importante, entonces, que la extracción sea efectiva, ya que la aceptación del sistema dependerá de la satisfacción de las necesidades del cliente y de la funcionalidad y la calidad en la automatización del trabajo.

Pasos :

- **Aplicar la o las herramientas elegidas para la extracción del requerimiento** (el cliente y/o usuario, debe estar completamente involucrado(a) en el proceso).
- **Identificar el o los problemas a solucionar**: Es importante detectar el problema que se está generando actualmente, antes de ofrecer cualquier solución.
  - **Situación problemática, sobre el caso de estudio**

    En relación a la situación problemática definida, se puede deducir que se presentan los siguientes problemas en el manejo de proyectos de grado, en la facultad de Ciencias Básicas e Ingeniería de la Corporación Universitaria Remington.

    Así:

    - Dificultad con la gestión de los proyectos de grado.
    - Inconsistencias en la información.
    - Reproceso.
    - Pérdida de información.
    - Ineficiencia a la hora de dar respuestas sobre el estado de los proyectos.
    - Pérdida de tiempo.
    - Inconformidad de los usuarios.

Listado de necesidades y características Caso de estudio

En lo que tiene que ver con las necesidades sobre el manejo de información se encuentra que la necesidad hace parte del tipo de información que se requiere automatizar y en relación a las características tiene que ver con los datos que agrupados conforman una información con significado como se muestra en la siguiente tabla 1.

Tabla 1 Necesidades y características de la situación problemática

CASO DE ESTUDIO	
NECESIDADES	**CARACTERÍSTICAS**
Información de las carreras	Código, nombre, duración.
Información de los estudiantes	Identificación, primer nombre, segundo nombre, primer apellido, segundo apellido, dirección, teléfono fijo, teléfono móvil, fecha de nacimiento, correo electrónico, entre otros datos.
Información de los temas que se pueden abordar	Código del tema, descripción del tema.
Información de los tipo de proyectos	Código, nombre.
Información de los proyectos de grado	Código del proyecto, fecha de matrícula, fecha de terminación, nombre, descripción, estado, observaciones.
Información de los asesores	Identificación, primer nombre, segundo nombre, primer apellido, segundo apellido, dirección, teléfono fijo, teléfono móvil, fecha de nacimiento, correo electrónico, estado civil, número de hijos, entre otros datos.
Información sobre las asesorías	Código de la asesoría, fecha, hora inicial, hora final, duración de la asesoría, estado, observaciones.

Fuente: elaboración propia.

Objetivos del proyecto caso de estudio.

Es importante tener claro los objetivos que se quieren lograr, acorde a los requerimientos del cliente y a los requisitos del software, de tal forma que se convierta en un producto funcional para el usuario final. Es recomendable abordar un solo objetivo general y varios específicos.

Siguiendo con la situación abordada, se podrían plantear los siguientes objetivos:

- Objetivo General:

Éste objetivo debe mostrar la solución al problema planteado, donde se muestra claramente lo que el software debe ofrecer. Ejemplo.

- ✓ Construir un software que permita la gestión de los proyectos de grado de los estudiantes de la Corporación Universitaria Remington, especialmente los de la facultad de Ciencias Básicas e Ingeniería, evitando inconsistencias, reproceso, pérdida de tiempo e información, garantizando así calidad en dicho proceso.

- Objetivos específicos: éstos se pueden redactar como objetivos técnicos en los cuales se muestra el cómo se creará el software, teniendo en cuenta los pasos de la ingeniería de software (requerimientos, análisis, diseños, codificación, pruebas, documentación, entre otros). Para otro tipo de proyecto es viable trabajar objetivos funcionales, los cuales se relacionan con la utilidad que prestará el software en relación al proceso automatizado a través del medio informático.

Ejemplo:

- Objetivos Específicos Técnicos:

- ✓ Identificar los requerimientos del cliente, utilizando herramientas como entrevista, encuesta, revisión documental y prototipos.
- ✓ Realizar el análisis de los requerimientos, utilizando diagramas de casos de uso y escenarios con el fin de que se conviertan en requisitos del software.
- ✓ Diseñar la interfaz gráfica del software, así como la base de datos y demás arquitecturas, definiendo así los soportes del software.
- ✓ Codificar el producto, utilizando la programación orientada a objetos y el lenguaje de programación java.
- ✓ Realizar las pruebas unitarias y de integración, asegurando así la funcionalidad del software.
- ✓ Realizar la revisión de calidad del producto, buscando con ello el cumplimiento de los requisitos internos y externos del producto, garantizando así la calidad.

- ✓ Construir la documentación del producto, tanto la del manual técnico, como manual del usuario y de implementación.

Si el producto debe llevar objetivos específicos funcionales, éstos se relacionan con lo que debe cumplir el software, acorde a las solicitudes del cliente y al estudio del experto informático. En éste caso nos concentraremos en los objetivos específicos técnicos que nos dicen cómo se construirá el software, en relación a los específicos funcionales, se abordarán de igual forma en el alcance que se defina para el producto.

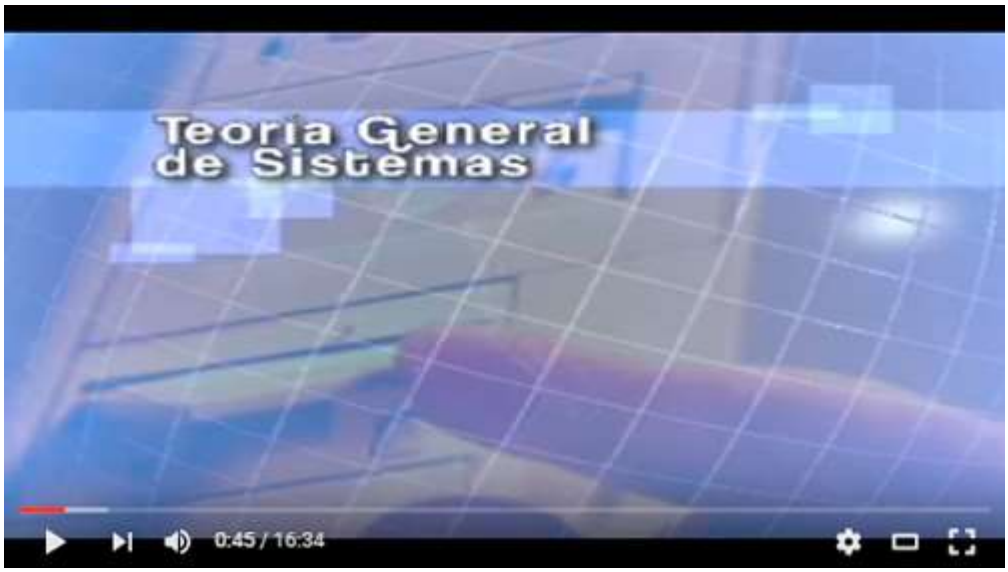
3.2 TEMA 2: TÉCNICAS PARA EL ANÁLISIS, MODELADO Y VALIDACIÓN DE LOS REQUISITOS

El requerimiento que realiza el cliente o usuario en relación a su necesidad específica, debe ser analizado por el Ingeniero de Sistemas, modelado y validado convirtiéndolo en un requisito claro y coherente que permita a ambas partes ponerse de acuerdo, con el fin de minimizar los errores durante todo el proceso de construcción del producto informático.

3.2.1 ANÁLISIS DEL REQUISITO

El requerimiento del cliente o usuario después de extraído, debe ser analizado y convertido en requisito de software, donde cada empresa de desarrollo adapta o crea estrategias para su análisis, lo que permite llevar a un nivel mucho más clara esa necesidad informática. Es así como se proponen herramientas sencillas que permite su análisis, entre las que se muestran las siguientes:

- **Tabla basada en la teoría de sistemas:** el software posee los elementos de un sistema de información como son entradas, procesos, salidas, debe tener procesos de retroalimentación y se mueve dentro de un ambiente que lo rodea. Para ampliar más la teoría general de sistemas ver el video



Teoría General de Sistemas (TGS) Enlace

Tomaremos de dicha teoría los siguientes elementos:

- **Entrada:** significa todo aquello que se requiere para que el sistema informático pueda realizar su proceso central, que en el caso de la gestión del proyecto a nivel de asesorías, las entradas están relacionadas con los datos que se requieren como por ejemplo estudiantes, proyectos, carreras, temas, asesores, entre otros, lo que se necesita para poder prestar las asesorías.
- **Proceso:** consiste en la parte central del software como puede ser la gestión de las asesorías, ya que un proyecto se puede gestionar siempre y cuando se den las asesorías pertinentes, por lo tanto, el proceso del software en mención, está relacionada con las asesorías.
- **Salida:** se refiere a los informes o consultas que arrojará el software como, por ejemplo:
  - ✓ Listado por nombre de los proyectos con sus Integrantes.
  - ✓ Consulta de proyecto con sus respectivos asesores, y sus fechas de asesoría, tanto para los aprobados, como para los que se encuentran en proceso.
  - ✓ Consulta por parte de los estudiantes para visualizar los proyectos por tema, por tipo de proyecto, para que sirva como referencia a proyectos posteriores.
  - ✓ Consulta sobre las fechas de asesorías de los proyectos.
  - ✓ Informe de proyectos aprobados, antes de realizar el proceso de graduación, entre otras

- **Retroalimentación:** son revisiones que se deben hacer en el caso que las salidas no sean las esperadas, por lo tanto, se deberán revisar las entradas y el proceso, esperando que la salida sea la adecuada.

En el caso de estudio y aplicando ésta herramienta, se evolucionan las necesidades de la siguiente forma.

Tabla 2 Análisis de requerimientos utilizando los elementos de un sistema de información

ENTRADAS	PROCESO	SALIDAS
Información de las carreras	Gestión de asesorías	Listado por nombre de los proyectos con sus Integrantes.
Información de los estudiantes		Consulta de proyecto con sus respectivos asesores, y sus fechas de asesoría, tanto para los aprobados, como para los que se encuentran en proceso.
Información de los temas que se pueden abordar		Consulta por parte de los estudiantes para visualizar los proyectos por tema, por tipo de proyecto, para que sirva como referencia a proyectos posteriores.
Información de los tipos de proyectos		Consulta sobre las fechas de asesorías de los proyectos.
Información de los proyectos de grado		Informe de proyectos aprobados, antes de realizar el proceso de graduación
Información de los asesores		

Fuente: elaboración propia.

- **Los actores del sistema:** para entender el propósito del sistema es importante definir los actores que se involucran con éste, especialmente aquellos usuarios no informáticos que manipularán el sistema en su parte operativa o de administración. Un actor puede ser una persona, una máquina, un dispositivo, entre otros. Que de alguna forma interactúe con el sistema. En el ejercicio que se está desarrollando, los

actores serían el decano, la secretaria, el asesor, el estudiante y el administrador del software. Ver tabla 3.

Tabla 3 Actores del sistema

ACTOR	FUNCIÓN
DECANO	Persona encargada de gestionar los procesos más importantes de la facultad de Ciencias Básicas e Ingeniería.
SECRETARIA	Persona encargada de gestionar los procesos relacionados con la gestión de los proyectos de grado de los estudiantes.
ASESOR	Docente encargado de prestar asesoría a los proyectos de grado.
ESTUDIANTE	Estudiantes con proyectos en curso o interesados en realizar proyectos de grado.
ADMINISTRADOR	Será la persona encargada de responder por toda la información del sistema, por lo tanto el administrador tendrá a su cargo todo el control del programa.

Fuente: Elaboración propia.

- **Tabla general para casos de uso:** ésta tabla profundiza los requisitos del software, permitiendo comprender más a fondo las necesidades informacionales, ya que se realiza un análisis más profundo sobre la información que se debe manejar, así como las acciones que se ejecutarán sobre cada gestión, de igual forma determina los actores que se involucran con el sistema, tanto a nivel operativo como administrativo.

  Se puede visualizar, que es posible realizar la asignación de acciones a cada uno de los actores del sistema, información recopilada del cliente, quien es el que decide que actores tendrán determinados roles en el sistema, a ese proceso de asignar actores para cada una de las acciones, se le llama perfiles, tema que será abordado en el siguiente módulo. Analizar tabla 4, donde se puede observar claramente un análisis del requisito mucho más completo, relacionado con el caso de estudio.

Tabla 4 Tabla general para casos de uso

GESTIONES	ACCIONES	ACTORES				
		SECRETARIA	DECANO	ESTUDIANTE	ASESOR	ADMINISTRADOR
Gestión carrera	Crear	X				X
	Consultar	X	X			X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X			X
	Salir	X	X			X
Gestión estudiante	Crear	X				X
	Consultar	X	X		X	X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X		X	X
	Salir	X	X		X	X
Gestión tema	Crear	X				X
	Consultar	X	X		X	X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X		X	X
	Salir	X	X		X	X
Gestión tipo de proyecto	Crear	X				X
	Consultar	X	X	X	X	X
	Modificar	X				X
	Guardar	X				X

	Inhabilitar	X				X
	Cancelar	X	X	X	X	X
	Salir	X	X	X	X	X
Gestión proyecto	Crear	X				X
	Consultar	X	X	X	X	X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X	X	X	X
	Salir	X	X	X	X	X
Gestión asesor	Crear	X				X
	Consultar	X	X			X
	Modificar	X				X
	Guardar	X				X
	Inhabilitar	X				X
	Cancelar	X	X			X
	Salir	X	X			X
Gestión de asesoría	Crear				X	X
	Consultar	X	X	X	X	X
	Modificar				X	X
	Guardar				X	X
	Inhabilitar				X	X
	Cancelar	X	X	X	X	X
	Salir	X	X	X	X	X
Gestión Informes	Listado por nombre de los proyectos con sus Integrantes.	X	X		X	X

	Informe de proyectos aprobados, antes de realizar el proceso de graduación	X	X			X
Gestión consultas	Consulta de proyecto con sus respectivos asesores, y sus fechas de asesoría, tanto para los aprobados, como para los que se encuentran en proceso.	X	X	X	X	X
	Consulta por parte de los estudiantes para visualizar los proyectos por tema, por tipo de proyecto,	X		X		X

	para que sirva como referencia a proyectos posteriores.					
	Consulta sobre las fechas de asesorías de los proyectos.	X	x	X	X	X
Administración Perfil	Crear					X
	Consultar					X
	Modificar					X
	Guardar					X
	Inhabilitar					X
	Cancelar					X
	Salir					X
Administración Usuario	Crear					X
	Consultar					X
	Modificar					X
	Guardar					X
	Inhabilitar					X
	Cancelar					X
	Salir					X

Fuente: elaboración propia.

3.2.2 MODELAMIENTO Y VALIDACIÓN DEL REQUISITO

Para el modelamiento, nos valdremos del UML (Unified Modeling Language o en español, Lenguaje de Modelamiento Unificado), ver video: https://www.youtube.com/watch?v=8Vs7jfHG8Tc.

UML es un lenguaje gráfico que se utiliza para especificar, diseñar, construir y documentar un sistema informático, de igual forma aporta herramientas para colaborar con la codificación del programa, el diseño de bases de datos entre otros aspectos, siendo el lenguaje de modelado más utilizado en la industria del software. Es importante aclarar que el UML, no es un lenguaje de programación. El UML, puede ser aplicado a muchas metodologías de desarrollo de software como por ejemplo RUP, acorde a necesidades, estrategias y herramientas requeridas.

UML cuenta con varios tipos de diagramas, los cuales muestran diferentes aspectos de las entidades representadas, como:

Tipos de diagramas UML

- **Diagramas de estructura:**

  - ✓ Diagrama de clases
  - ✓ Diagrama de componentes
  - ✓ Diagrama de objetos
  - ✓ Diagrama de estructura compuesta
  - ✓ Diagrama de despliegue
  - ✓ Diagrama de paquetes

- **Diagramas de comportamiento**

  - ✓ Diagrama de casos de uso
  - ✓ Diagrama de estados

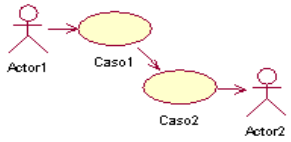
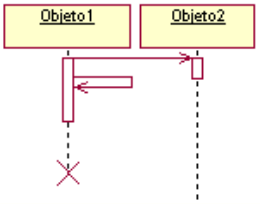
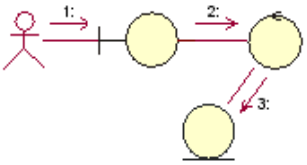
- **Diagramas de Interacción:**

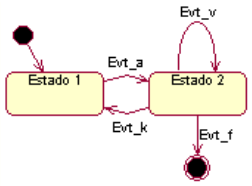
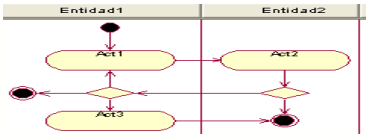
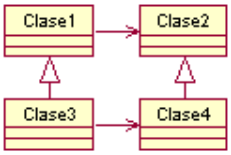
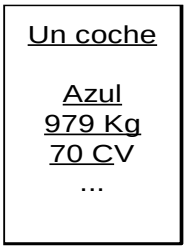
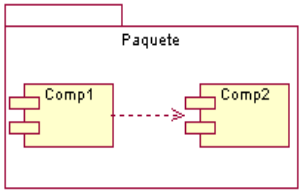
  - ✓ Diagrama de secuencia
  - ✓ Diagrama de comunicación
  - ✓ Diagrama de tiempos (UML 2.0)

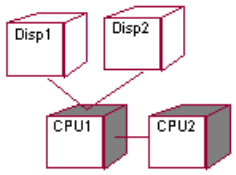
En éste módulo se trabajará sólo con algunos de ellos, los demás se abordarán en los módulos de Ingeniería de Software II y III.

A continuación, se describen algunos diagramas UML. Ver tabla 5

Tabla 5 Algunos diagramas UML

DIAGRAMA	IMAGEN	DESCRIPCIÓN GENERAL
CASOS DE USO		Muestra un conjunto de casos de uso, los actores implicados y sus relaciones. Son diagramas fundamentales en el modelado y organización del sistema.
SECUENCIA		Son diagramas de interacción, muestran un conjunto de objetos y sus relaciones, así como los mensajes que se intercambian entre ellos. Cubren la vista dinámica del sistema. El diagrama de secuencia resalta la ordenación temporal de los mensajes, mientras que el de colaboración resalta la organización estructural de los objetos, ambos siendo equivalentes o isomorfos. En el diagrama de colaboración de la figura de la izquierda, se puede ver que los elementos gráficos no son cajas rectangulares, como cabría esperar, y en su lugar encontramos sus versiones adornadas. Estas versiones tienen como finalidad evidenciar un rol específico del objeto siendo modelado. En la figura encontramos de izquierda a derecha y de arriba abajo un Actor, una Interfaz, un Control (modela un comportamiento) y una Instancia (modela un objeto de dato).
COLABORACIÓN		

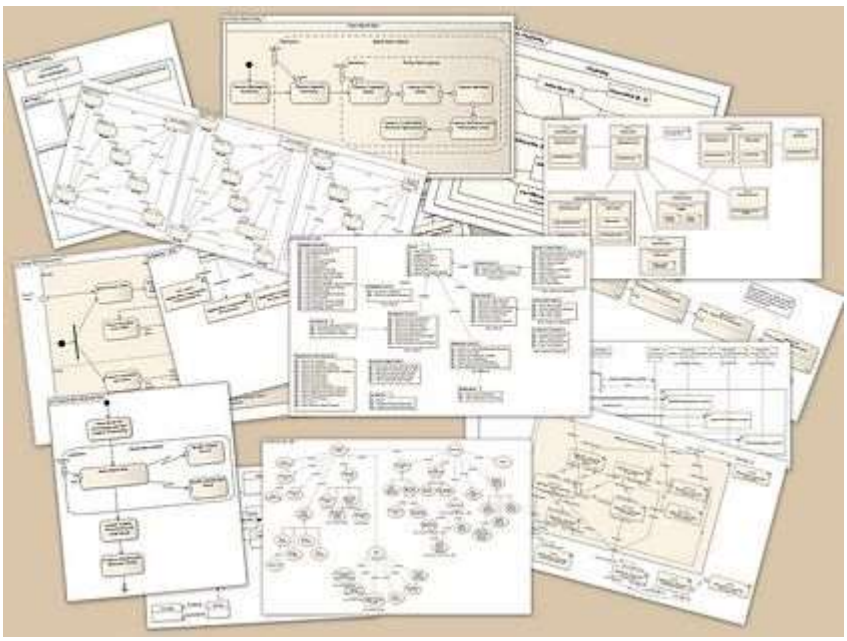
ESTADOS		Muestra una máquina de estados, con sus estados, transiciones, eventos y actividades. Cubren la vista dinámica de un sistema. Modelan comportamientos reactivos en base a eventos.
ACTIVIDADES		Tipo especial de diagrama de estados que muestra el flujo de actividades dentro de un sistema.
CLASES		Muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones, cubriendo la vista de diseño estática del sistema.
OBJETOS		Análogo al diagrama de clases, muestra un conjunto de objetos y sus relaciones, pero a modo de vista instantánea de instancias de una clase en el tiempo.
COMPONENTES		Muestra la organización y dependencias de un conjunto de componentes. Cubren la vista de implementación estática de un sistema. Un componente es un módulo de código, de modo que los diagramas de componentes son los análogos físicos a los diagramas de clases.

DESPLIEGUE		Muestra la configuración del hardware del sistema, los nodos de proceso y los componentes empleados por éstos. Cubren la vista de despliegue estática de una arquitectura.

Fuente: elaboración propia.

Es así como el lenguaje de modelado UML, permite clarificar el proceso de ingeniería de software, buscando que los requerimientos de software se conviertan en especificaciones claras para el sistema, ofreciendo diagramas que se pueden aplicar en cada una de las etapas como son los requerimientos, el análisis, la construcción, la documentación, las pruebas, el despliegue, la gestión de configuración, entre otras, siendo la herramienta elegida por los especialistas informáticos. Ver imagen 31, donde se puede observar algunos diagramas, utilizando el UML.

Imagen 31 Ejemplo de diagramas UML



Fuente: «UML Diagrams» de Kishorekumar 62

Para aplicar a la situación que se viene trabajando y que nos sirve de ejemplo para comprender el modelado de los requisitos de software, utilizaremos el software Enterprise Architect, entendiendo que no es el único que se puede utilizar. Para ello se pueden emplear varios software entre licenciados y libre como por ejemplo: Cacoo,

Argo UML, Edraw, Visio, Enterprese Arquitect en versiones libres o cualquier otro software que permita modelar diagramas tipo UML.

- **Diagramas de Casos de Uso:** Pressman (2002), "Los casos de uso modelan el sistema desde el punto de vista del usuario. Los casos de uso deben cumplir los siguientes objetivos: definir los requisitos funcionales y operativos del sistema (producto), diseñando un escenario de uso acordado por el usuario final, y el equipo de desarrollo; proporcionar una descripción clara y sin ambigüedades de cómo el usuario final interactúa con el sistema y viceversa, y proporcionar una base para la validación de las pruebas", P.367.

Para la aplicación de los casos de uso al ejercicio que se viene construyendo, nos basaremos en la llamada tabla general para casos de uso, la cual posee información clara que se puede evolucionar a través de los diagramas de casos de uso, de igual forma se trabajará con el software Enterprise Architect.

- **Diagrama de casos de uso principal:** contiene las gestiones a nivel general y la relación de asociación con los actores, en algunos diagramas de caso de uso principales, aparece la asociación directa con el actor, pero de igual forma al encerrar en un recuadro los casos de uso y el actor quedar por fuera, significa que de alguna manera dicho actor se relaciona con los casos de uso que contiene el diagrama. Ver imagen 32.

Imagen 32 Diagrama de caso de uso principal (Sistema Proyectos de grado)

- **Diagramas de Casos de Uso extendidos**: hacen alusión al diagrama principal de casos de uso, permitiendo ampliar cada uno de ellos, donde se puede visualizar las acciones que se pueden realizar con el manejo de los datos e información del sistema, donde es importante tener en cuenta que para manipular datos el usuario (actor), debe realizar una serie de acciones llamadas normalmente el CRUD (Create, Read, Update, Delete), las cuales se relacionan a las cuatro operaciones básicas que se realizan con los datos, como son crear o registrar un nuevo dato, realizar consultas o búsquedas de datos e información, modificar, editar o actualizar los datos e información y eliminar o borrar, que actualmente se refiere a inhabilitar, ya que la información no se debe borrar, con el fin de conservar los históricos, en cuyo caso se debe inhabilitar cuando ya no se esté utilizando. Ver tabla 6

Tabla 6 Operaciones Básicas con los datos e información

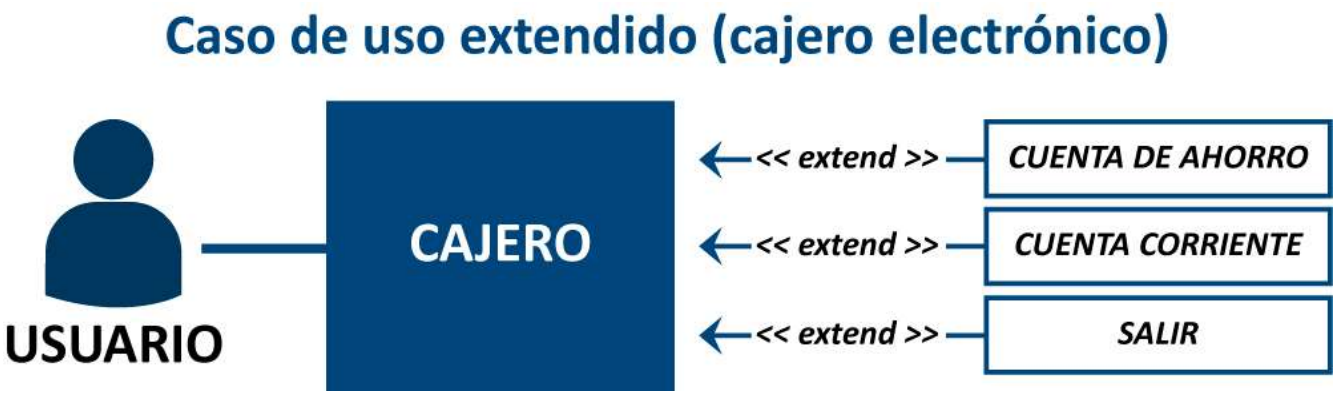
OPERACIONES BÁSICAS CON LOS DATOS E INFORMACIÓN CRUD	
Create	Crear, registrar o nuevo
Read	Consultar, buscar
Update	Modificar, editar, actualizar
Delete	Borrar, eliminar, inhabilitar
Se pueden trabajar con otras operaciones como por ejemplo Pagar, Enviar, Iniciar, entre otras. Esto depende del tipo de proceso que se desea programar y a la respectiva necesidad del cliente. En éste caso nos concentraremos especialmente en las cuatro operaciones básicas que se realizan con los datos.	

- **Diagramas de Casos de Uso Extendidos e Incluidos**

  Los diagramas de caso de uso extendidos, se dividen de acuerdo a su tipo de asociación con otros diagramas de casos de uso, en extend e include.

  ✓ Extend: una asociación tipo extend, significa que la asociación, depende de un caso de uso base y no tiene que pasar por otro caso de uso como condición para que la acción se pueda realizar. Ver imagen 33.

Imagen 33 Caso de uso extendido

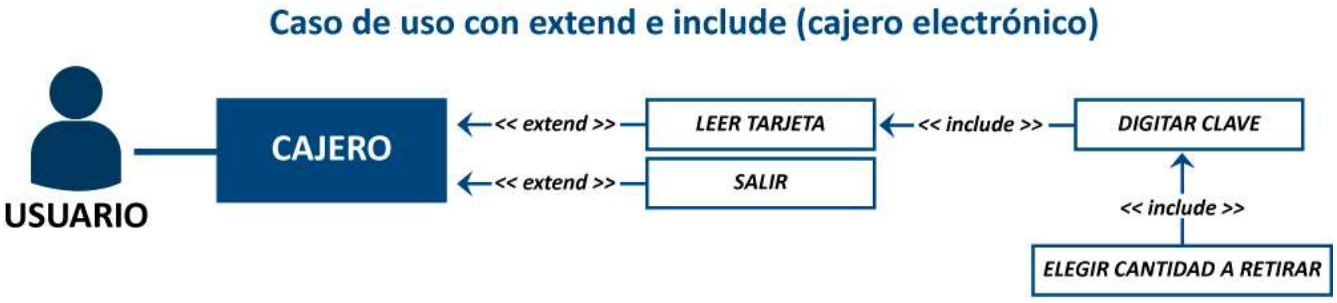


Fuente: Elaboración propia.

Lo que quiere decir que cuenta de ahorro, cuenta corriente y salir son extensiones de cajero e independientes entre sí, por lo tanto, para ejecutar ahorro no hay que pasar por corriente o viceversa, de igual forma en cualquier momento se puede elegir la opción de salir.

- ✓ Include: Significa que, para elegir un caso de uso, obligatoriamente se debe ejecutar el caso de uso que lo incluye, como precondición para que pueda funcionar. Siguiendo con el ejemplo del cajero vamos a suponer que se desea ubicar en la opción ahorro, pero para que esto sea posible se debe digitar la clave de la tarjeta y para digitar la clave, el cajero debe leer la tarjeta, lo cual indica que existen procesos que funcionan, siempre y cuando se hayan ejecutado otros, en ese caso unos procesos requieren que se incluyan otros antes para poder funcionar. Ver imagen 34.

Imagen 34 Caso de uso incluido



- **Diagramas casos de uso extendidos (Sistema proyectos de grado)**

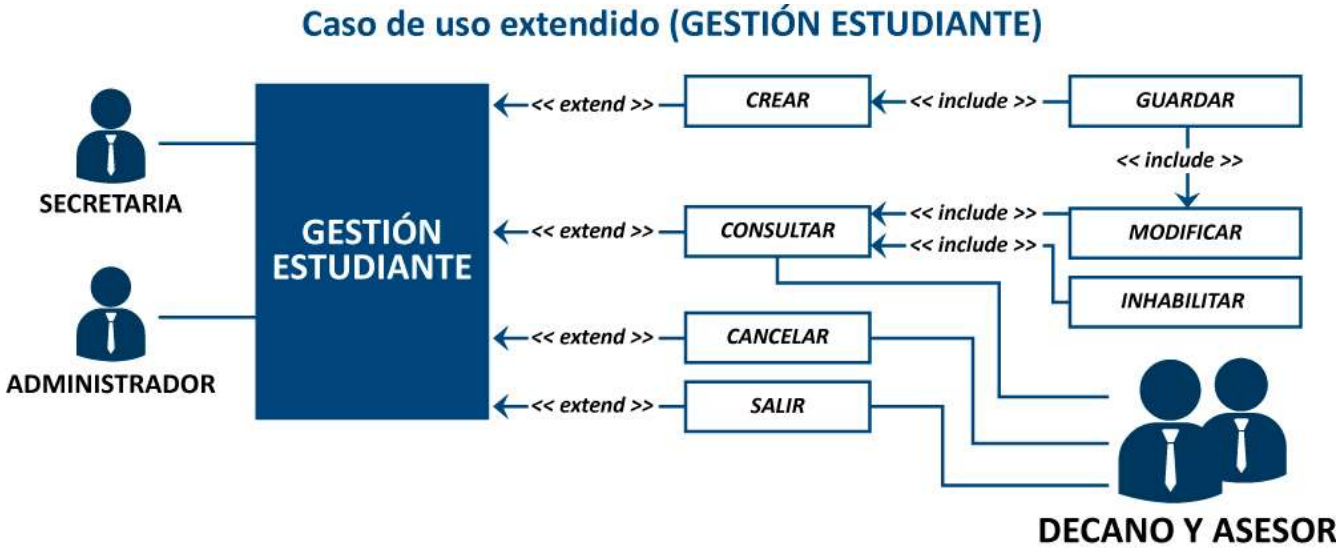
Los siguientes diagramas muestran en detalle el diagrama de casos de uso principal, permitiendo visualizar las acciones que cada actor puede ejecutar en el sistema, cuando éste se encuentre en funcionamiento, lo cual recibe el nombre de perfiles de usuario, ya que no todos los usuarios del sistema podrán manejar toda la información. Dicha asociación se representa con una línea recta, que va desde el actor hasta el caso de uso respectivo.

Es importante tener presente que la tabla general para casos de uso, descrita en páginas anteriores, muestra todas las gestiones, acciones y actores que se retoman nuevamente en los casos de uso y que permiten realizar una revisión más concreta sobre el análisis de los requerimientos del cliente y permite el acercamiento hacia los requisitos que el software debe cumplir. Ver imágenes casos de uso extendidos para cada una de las gestiones.

Imagen 35 Caso de uso extendido Gestión Carrera (Sistema proyectos de grado)

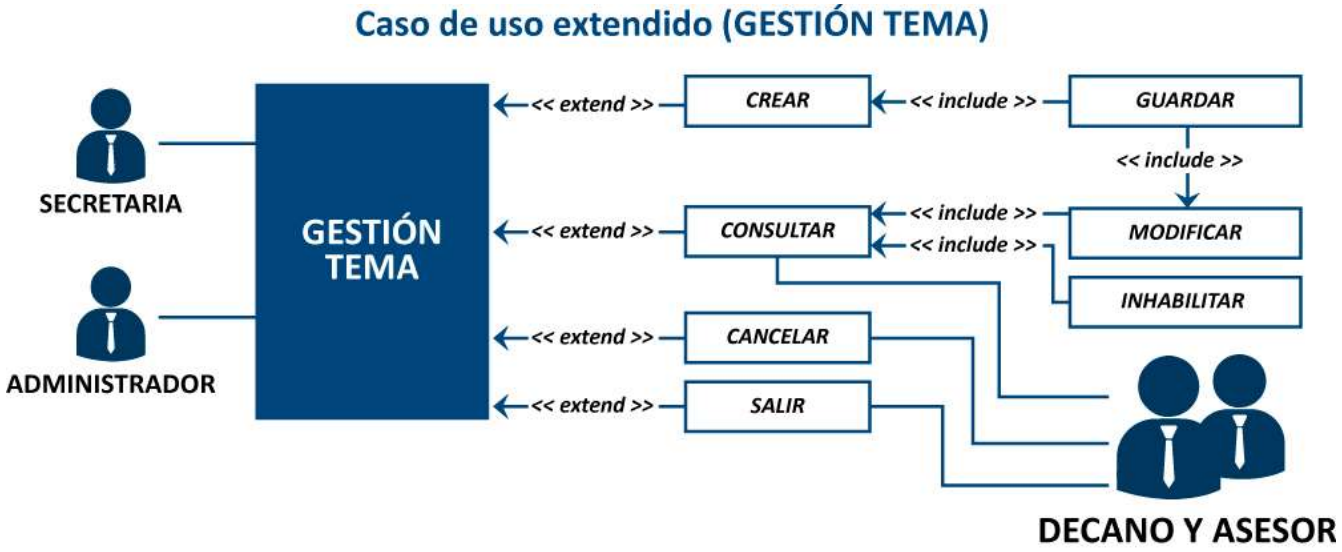
Fuente: elaboración propia.

Imagen 36 Caso de uso extendido Gestión estudiante (Sistema proyectos de grado)



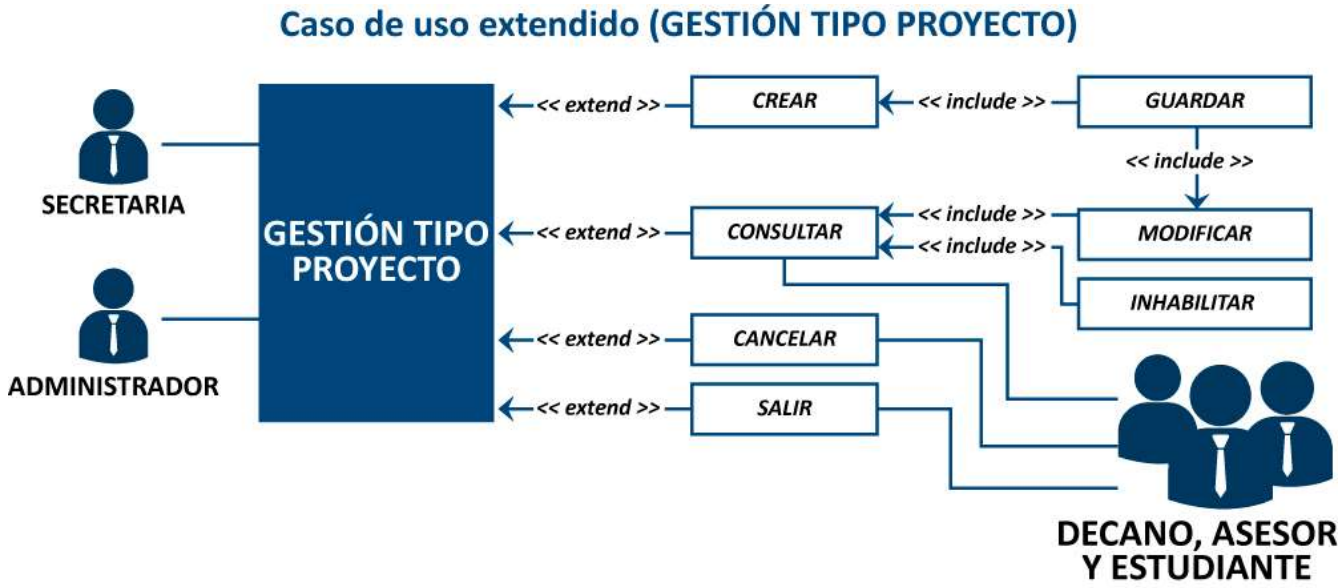
Fuente: elaboración propia.

Imagen 37 Caso de uso extendido Gestión Tema (Sistema proyectos de grado)



Fuente: elaboración propia.

Imagen 38 Caso de uso extendido Gestión Tipo_Proyecto (Sistema proyectos de grado)

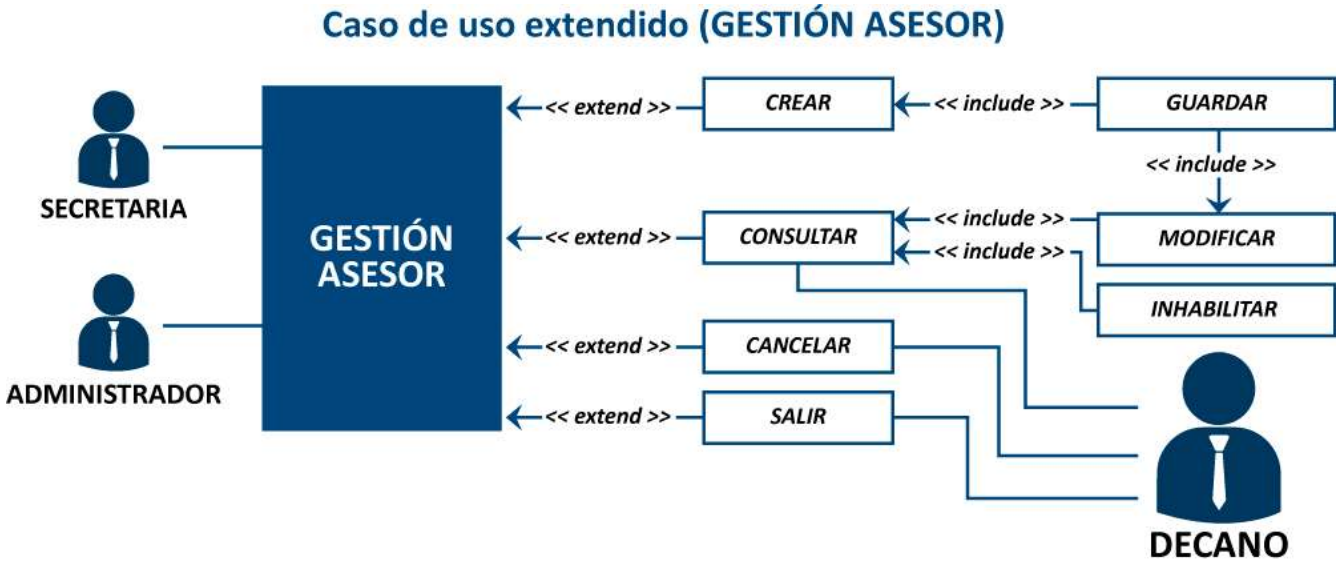


Fuente: elaboración propia.

Imagen 39 . Caso de uso extendido Gestión Proyecto (Sistema proyectos de grado)

Fuente: elaboración propia.

Imagen 40 Caso de uso extendido Gestión Asesor (Sistema proyectos de grado)



Fuente: elaboración propia.

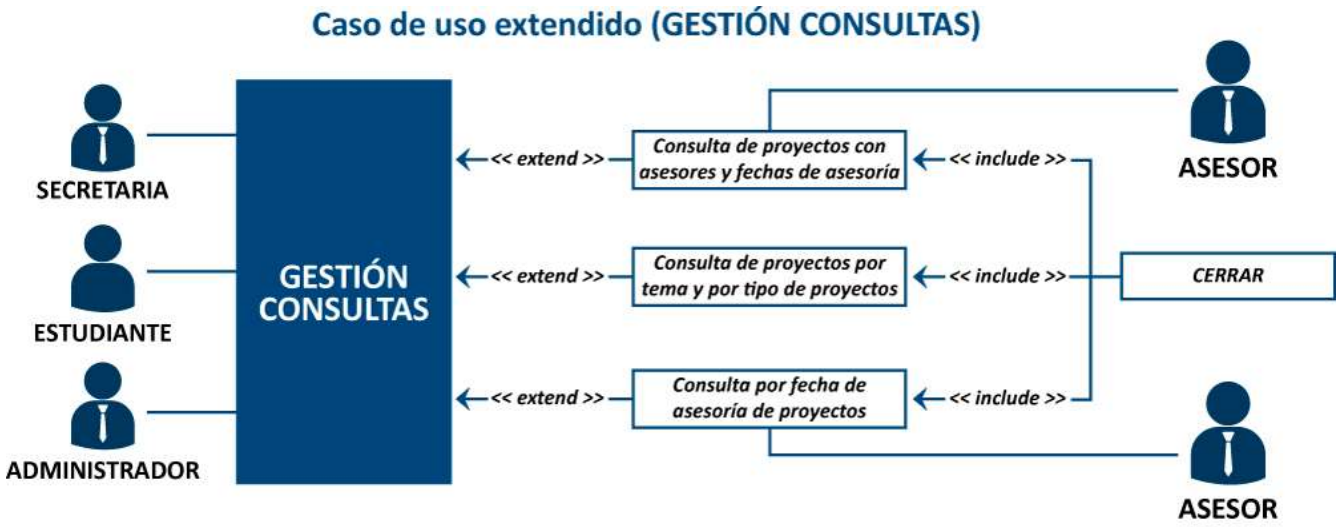
Imagen 41 Caso de uso extendido Gestión Asesoría (Sistema proyectos de grado)

Fuente: elaboración propia.

Imagen 42 Caso de uso extendido Gestión Informes (Sistema proyectos de grado)

Fuente: elaboración propia.

Imagen 43 Caso de uso extendido Gestión Consultas (Sistema proyectos de grado)



Fuente: elaboración propia.

Imagen 44 Caso de uso extendido Gestión Perfil (Sistema proyectos de grado)

Fuente: elaboración propia.

Imagen 45 Caso de uso extendido Gestión Usuario (Sistema proyectos de grado)

Fuente: elaboración propia.

Nota: si nos adelantamos un poco en el proceso de ingeniería de software para el sistema "Proyectos de grado", nos damos cuenta que cada gestión corresponde a un formulario y cada una de los extend e include corresponden a opciones que permiten manipular datos e información (botones de comando) y que de igual forma desde ésta etapa que corresponde a la ingeniería de requisitos, se define los actores que interactuarán con el sistema, por lo tanto se conoce el perfil o rol que tendrá cada usuario.

3.3 TEMA 3: INTRODUCCIÓN A LOS RIESGOS (PROYECTO, PROCESO, PRODUCTO)

Desde etapas tempranas del proceso de Ingeniería de Software, es importante tener presente los tipos de riesgos que se pueden presentar y que atenten contra el proyecto, el proceso y el producto de software, analizando su impacto y previendo las posibles soluciones en caso de que no se pueda evitar dicho suceso.

3.3.1 SOBRE EL RIESGO

En su libro sobre análisis y gestión del riesgo, Robert Charette presenta la siguiente definición de riesgo: "En primer lugar, el riesgo afecta a los futuros acontecimientos. El hoy y el ayer están más allá de lo que nos pueda preocupar, pues ya estamos cosechando lo que sembramos previamente con nuestras acciones del pasado. La pregunta es, podemos por tanto, cambiando nuestras acciones actuales, crear una oportunidad para una situación diferente y, con suerte, mejor para nosotros en el futuro. Esto significa, en segundo lugar, que el riesgo implica cambio, que puede venir dado por cambios de opinión, de acciones, de lugares. En tercer lugar, el riesgo

implica elección y la incertidumbre que entraña la elección. Por tanto, el riesgo, como la muerte, es una de las pocas cosas inevitables de la vida". Ver video sobre la gestión del riesgo, según norma ISO 31000:

ISO 31000-Gestión de Riesgos Enlace

Es así como se debe entender que el riesgo es un acontecimiento, con probabilidad de que pueda existir y más aún en la temática que estamos abordando, ya que en lo relacionado con el software, cada día los especialistas, deben buscar solución a situaciones inesperadas, ya que el software parte de necesidades u oportunidades abstractas, siendo el mismo software un producto intangible y muchas veces impredecible, donde el riesgo implica dos características, donde la primera es la incertidumbre de que pueda o no ocurrir y la segunda la pérdida, donde si éste se convierte en realidad existe la probabilidad de que se presenten consecuencias no deseadas.

Toda empresa desarrolladora de software, debe considerar dos tipos de estrategias para abordar los riesgos, las reactivas y la proactivas, donde las estrategias reactivas se dan, cuando el suceso ha acontecido, lo que lleva a buscar soluciones rápidas para abordar tal situación, como por ejemplo cuando el sistema tiene virus, se busca la forma de eliminarlo, asumiendo las consecuencias que de ello se deriven, en relación a las estrategias proactivas, éstas se orientan hacia la previsión, antes de que el acontecimiento suceda, siguiendo con el ejemplo del virus informático, previamente se ha instalado antivirus que en el caso de que se detecte un virus dentro del sistema, éste actúe con el fin de evitar consecuencias como por ejemplo pérdida de información, alteración de

procesos, daño en dispositivos físicos de cómputo, entre otras dificultades que giran alrededor de dicha información. Por lo tanto, es importante prever los riesgos, antes de que sucedan.

La temática en relación al riesgo es bastante amplia, por lo tanto, se abordará de forma más profunda en el siguiente módulo de Ingeniería de Software II. En éste se hará una breve inducción al respecto.

3.3.2 ALGUNOS TIPOS DE RIESGO EN LA INGENIERÍA DE SOFTWARE

Entre muchos tipos de riesgos, en el proceso de Ingeniería de Software, se hará alusión a algunos que pueden afectar los proyectos, los procesos y el producto de software.

- **Riesgos del proyecto de software**: Se relacionan con la probabilidad de que el plan del proyecto, no se lleve a cabo o sufra alteraciones, acorde a la planificación realizada, siendo posible que los tiempos no se cumplan, que aumente el costo de producción, generando consecuencias no deseadas tanto para el cliente como para quienes están a cargo del proyecto, como por ejemplo fechas de entrega poco realistas, poca experiencia en la construcción de software, mala comunicación con el cliente, disminución de personal a cargo del proyecto, confusión en las necesidades informacionales, entre otras.

- **Riesgos del producto de software:** éstos riesgo parten del interrogante ¿Qué características especiales de este producto pueden estar amenazadas por el plan del proyecto?, entre algunas características se tienen el rendimiento que el producto de software tenga, en relación a las tareas que deba ejecutar de forma eficiente, al costo de producción y precio de venta para el cliente, de igual forma la facilidad de soporte, el cumplimiento de cronogramas acorde a planificación del proyecto, la usabilidad del producto por el cliente o usuario, la seguridad que posee el producto ante accesos no autorizados, la capacidad de protección de información, entre otros aspectos que se deben considerar para evitar que el producto no cumpla con la calidad requerida.

- **Riesgos en el proceso del software:** durante el proceso de construcción del software, es importante tener presente los riesgos que se puedan presentar, sin importar el tipo de metodología a utilizar, ya que de alguna forma se debe tener presente que el proceso tiene implícito unas necesidades que se deben convertir en requisitos del software, de alguna forma deben figurar diseños, codificaciones, se deben

realizar pruebas para asegurar que lo que se está construyendo obedece a la funcionalidad esperada, que pueden aparecer cambios a medida que se avanza en el proceso y que de alguna forma debe quedar la evidencia escrita de la forma como se desarrolló, ya que se debe pensar en que otras personas pueden darle continuidad a los procesos.

Dentro del proceso de construcción, pueden aparecer riesgos como por ejemplo técnicos, donde si lo que se está pidiendo requiere de tecnología nueva, no probada anteriormente, si los desarrolladores, no tienen conocimiento avanzado de los lenguajes o cualquier producto que se requiera, si el cliente no tiene claridad con lo que realmente necesita, si existen dificultades para el trabajo colaborativo con el cliente o dentro del equipo de desarrollo, si la metodología aplicada es la adecuada para el proyecto, si el personal se encuentra motivado, entre otros aspectos que pueden generar dificultades en el correcto desarrollo del producto.

Es así como en cada actividad que se realice, existen riesgos potenciales que pueden traer dificultades al proyecto, producto y proceso, donde su debida previsión, análisis de impacto, así como las medidas preventivas y correctivas que se consideren, podrían minimizar las respectivas dificultades que a diario deben sufrir las empresas que se dedican a la construcción de éste tipo de productos informáticos.

Para complementar, realizar lectura sobre el siguiente documento: Ver link sobre riesgos de software:

http://dis.um.es/~barzana/Informatica/IAGP/IAGP_riesgos.html

3.3.3 EJEMPLO SOBRE ANÁLISIS DE RIESGOS

En la tabla 7 Análisis de riesgos, se puede observar la descripción de algunos riesgos, el tipo de riesgo, la probabilidad de que ocurra y el plan o acción que puede ser predictivo, para evitar que suceda o proactivo cuando ya ha sucedido, lo ideal es que se tenga el plan de acción previsto e implementado, evitando así pérdida de tiempo, de información, de recursos y todo aquello que atente contra el bienestar de los equipos de trabajo.

Tabla 7 Análisis de Riesgos

ANÁLISIS DE RIESGOS			
DESCRIPCIÓN DEL RIESGO	**TIPO DE RIESGO**	**PROBABILIDAD DE QUE OCURRA**	**PLAN DE ACCIÓN**
No contar con un servidor que procese y guarde la información.	Tecnológico	Media	Solicitar adquirir un servidor que garantice el procesamiento y guarde la información.
No tener acceso a una red de internet o intranet que comunique el servidor con las computadoras que utilizan el sistema de información		Media	Solicitar adquirir una red de internet para que garantice el acceso y la conectividad del servidor con las computadoras.
Que el sistema operativo no sea compatible o no funcione correctamente con el sistema de información.		Media	Desarrollar el producto de software teniendo en cuenta la compatibilidad de las diferentes versiones de los sistemas operativos.
Que el explorador de internet no sea compatible o no corra correctamente con el sistema de información.		Media	Desarrollar el producto de software teniendo en cuenta la compatibilidad de las diferentes versiones de los navegadores de internet.
Que el acceso a la red sea limitado		Media	Solicitar ampliar el ancho de banda y velocidad de la red de internet.
Que los requisitos del producto requieran una interfaz de usuario especial		Media	Utilizar herramientas de desarrollo de software de alta calidad que permitan satisfacer las exigencias de los usuarios.
Que el software no interactúe con hardware nuevo o no probado		Media	Elaborar el código del programa de manera clara que pueda ser interpretado, ejecutado o comprendido por cualquier sistema, arquitectura o aplicación.
Que el software a construir no Interactúe con otros productos de software.		Media	
Que el software a construir no Interactúe con un sistema de base de datos.		Media	
Que el cliente no tenga conocimiento o capacitación en el uso de herramientas tecnológicas	Cliente	Baja	Capacitar al cliente en el uso y manejo de las herramientas tecnológicas.

Que el cliente no tenga una idea formal de lo que se requiere		Baja	Reunirse con el cliente y el personal que hará uso del sistema para establecer de manera general qué es lo que se requiere, cuáles son las necesidades y expectativas, a las cuales se le quiere dar solución con el desarrollo del proyecto. Crear un ambiente de confianza y colaboración que logren la participación activa de los actores del sistema para lograr un buen diseño y funcionamiento del software teniendo en cuentas todas especificaciones técnicas que se quiere que realice el sistema.
Que no esté dispuesto a establecer una comunicación fluida con el desarrollador		Baja	
Que el cliente no esté dispuesto a participar en las revisiones		Baja	
Que el cliente no esté seguro de que la funcionalidad pedida sea factible		Baja	
Que no se lleven a cabo regularmente revisiones técnicas formales de las especificaciones de requisitos, diseño y código.	Proceso	Baja	Realizar de manera diaria las revisiones técnicas a las especificaciones de requisitos, diseño y código, a los procedimientos y casos de prueba, errores y recursos empleados, por lo cual se deberán documentar para tener un control y registro para posteriores revisiones, toma de decisiones o cambios en la implementación y desarrollo del producto.
Que no se lleven a cabo regularmente, revisiones técnicas de los procedimientos de prueba y de los casos de prueba.		Baja	
Que no se documenten todos los resultados de las revisiones técnicas, incluyendo los errores encontrados y recursos empleados.		Baja	
Que no exista algún mecanismo para asegurarse de que el trabajo realizado en un proyecto se ajusta a los estándares de ingeniería del software.		Baja	Emplear los estándares de calidad en el desarrollo del software, siguiendo las normas que se crearon para garantizar la buena calidad del producto, que deba garantizar la seguridad, integridad y disponibilidad de la información que procesara y almacenara el sistema informático.
Que no se emplee una gestión de configuración para mantener la consistencia entre los requisitos del sistema/software, diseño, código y casos de prueba.		Alta	
Que no haya algún mecanismo de control de cambios de los requisitos del cliente que impacten en el software.		Alta	

3.4 TALLER DE ENTRENAMIENTO UNIDAD 2

Nombre del taller: Taller 2	Modalidad de trabajo: Aprendizaje basado en problemas.
Actividad previa: Lectura de la unidad 1 y Unidad 2 del módulo	

Describa la actividad:

Teniendo en cuenta la siguiente situación:

1. Identificar el problema
2. Redactar la pregunta problematizadora
3. Redactar el objetivo general
4. Realizar el listado de requerimientos, detallando las especificaciones hasta su mínima expresión. Ejemplo (Información del estudiante: identificación, nombres, apellidos, fecha de nacimiento, dirección, teléfono, entre otros)
5. Construir la tabla general para casos de uso.
6. Diseñar los casos de uso.

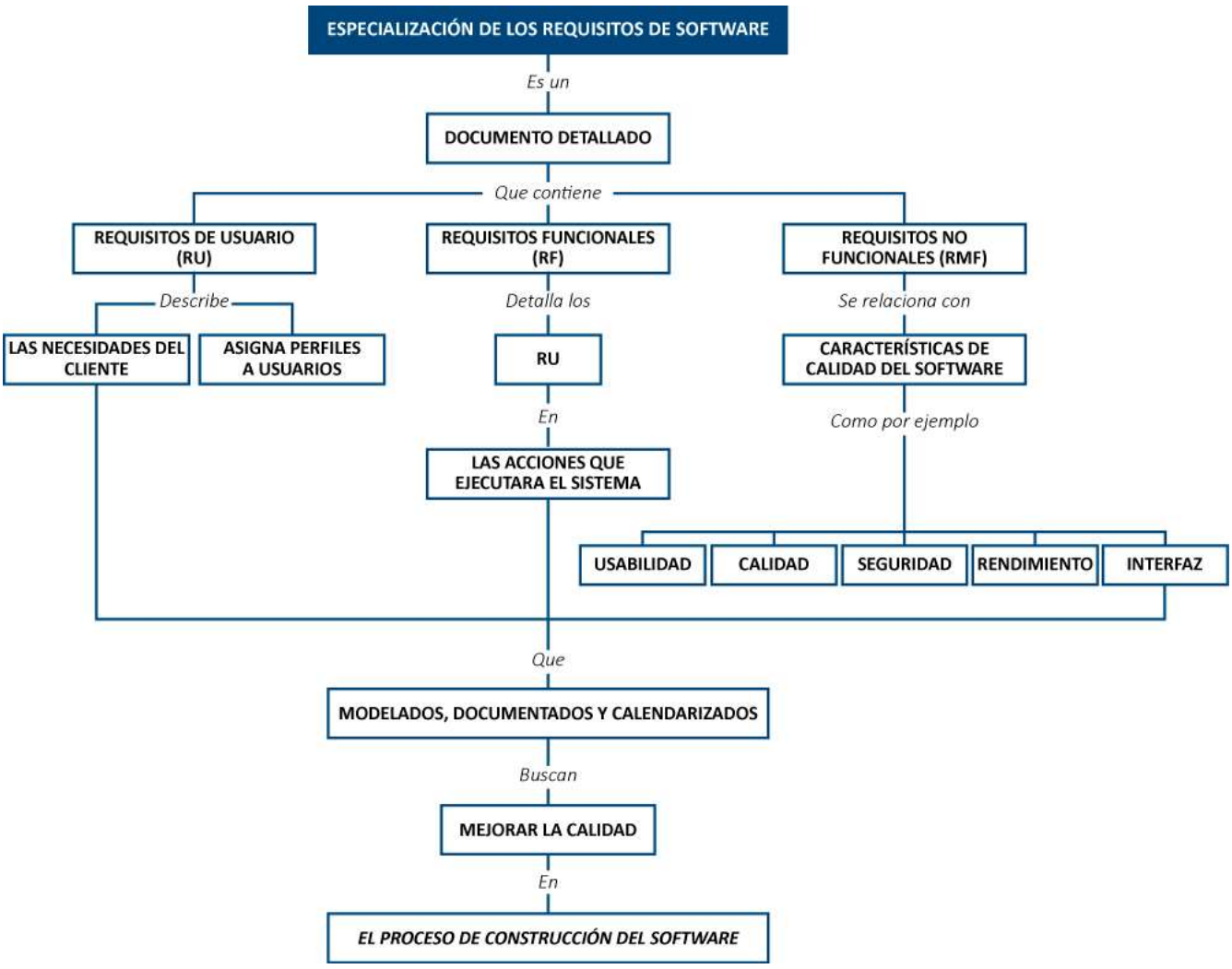
Situación:

Un parqueadero ubicado en el centro de la ciudad, tiene dificultades con el control de ingreso y salida de vehículos, lo que genera pérdida de tiempo y retraso tanto a clientes como a empleados, generando cogestión, stress, desorganización entre otros, es así como se requiere sistematizar su servicio de parqueos. El software debe manejar la información de los vehículos que ingresan al parqueadero, así como el respectivo registro que se genera, el cual debe hacerse al ingreso de este, también debe imprimir el recibo para el pago por horas. El parqueadero también ofrece mensualidades. Dependiendo del tipo de vehículo que pueden ser camiones, buses, busetas, automóviles será el costo de la hora o de la mensualidad. Para el parqueadero también es importante tener la información de los clientes que llevan sus vehículos, así como la información del empleado que registra el servicio y expide el recibo de pago.

Es de anotar que la forma como se está llevando actualmente el manejo del parqueadero es manual, en un cuaderno, donde en ocasiones se pierde la información o no se encuentra en el momento oportuno.

La persona encargada de manejar todo el sistema, será una secretaria que se encuentra ubicada en la cabina de entrada del parqueadero.

4 UNIDAD 3: ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE



Fuente: elaboración propia.

4.1 TEMA 1: REQUISITOS DE USUARIOS, REQUISITOS FUNCIONALES, REQUISITOS NO FUNCIONALES

La especificación está relacionada con la elaboración de documentos que permiten detallar los requisitos del software, los cuales se pueden representar de varias formas de tal forma que se evolucionen los requisitos del software, permitiendo tener un mejor entendimiento de ellos, con orientación hacia la comprensión de los requisitos de usuario (RU), requisitos funcionales (RF), donde se detallan las acciones que el software debe

cumplir y de igual forma los requisitos no funcionales (RNF) que tienen que ver con lo que el software debe cumplir, obedeciendo a requisitos externos que no pide el cliente pero que el software debe tener como por ejemplo la seguridad de la información.

4.1.1 REQUISITOS DE USUARIO (RU)

El Requisito de Usuario (RU), se relaciona directamente con la necesidad del cliente en relación a la automatización del grupo de datos, donde dicha necesidad es traducida al requisito del sistema. El RU describe las diferentes acciones que el usuario puede ejecutar sobre el sistema. Como se puede observar cada herramienta que se aplica, sigue evolucionando y clarificando la necesidad del cliente, de manera que pase hacia otros procesos el requisito concreto. Ver tabla 8, donde se ilustra dichos requisitos.

Tabla 8 Requisitos de Usuario (RU)

REQUISITOS DE USUARIO (RU)			
ID REQUISITO	**NOMBRE DEL REQUISITO**	**DESCRIPCIÓN DEL REQUISITO**	**USUARIO**
RU-001	Gestión Carrera	El sistema permitirá la gestión de la información de las carreras para Crear, Modificar, Inhabilitar, Consultar, Cancelar y Salir	Secretaria, Administrador, Decano
RU-002	Gestión Estudiante	El sistema permitirá la gestión de la información de los Estudiantes para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir.	Secretaria, Administrador, Decano, Asesor.
RU-003	Gestión Tema	El sistema permitirá la gestión de la información de los Temas para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir.	Secretaria, Administrador, Decano, Asesor
RU-004	Gestión Tipo_Proyecto	El sistema permitirá la gestión de la información de los Tipos de Proyecto para Crear, Modificar, Inhabilitar, Cancelar, Consultar, Salir	Secretaria, Administrador, Decano, Asesor, Estudiante

RU-005	Gestión Proyecto	El sistema permitirá la gestión de la información de los proyectos para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir	Secretaria, Administrador, Decano, Asesor, Estudiante
RU-006	Gestión Asesor	El sistema permitirá la gestión de la información del Asesor para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir	Secretaria, Administrador, Decano
RU-007	Gestión Asesoría	El sistema permitirá la gestión de la información de la Asesoría para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir	Asesor, Administrador, Decano, Secretaria, Estudiante
RU-008	Gestión Informes	El sistema permitirá la Ejecución de los informes (Proyectos y sus integrantes y Proyectos aprobados)	Secretaria, Decano, Administrador, Asesor
RU-009	Gestión Consultas	El sistema permitirá la ejecución de las consultas (proyectos con asesores y fechas de asesoría, proyectos por temas y por tipo de proyectos, fecha de asesoría de proyectos)	Secretaria, Estudiante, Administrador, Asesor
RU-010	Gestión Perfil	El sistema permitirá la gestión de la información del perfil para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir	Administrador
RU-011	Gestión Usuario	El sistema permitirá la gestión de la información de los usuarios para Crear, Modificar, Inhabilitar, Consultar, Cancelar, Salir	Administrador

Fuente: Elaboración propia.

4.1.2 REQUISITOS FUNCIONALES (RF)

Un RF, permite detallar el RU en relación a su descripción, lo que equivale a ampliar la información de los diferentes diagramas de casos de uso extendidos, relacionando cada usuario (actor) que ejecutará dichas acciones en el sistema. Un RF, posee un número consecutivo que lo identifica y está asociado a un RU.

El requisito funcional (RF), describe claramente los servicios que el sistema debe proporcionar al usuario final, ya que describe claramente cada una de las acciones que el actor ejecuta en el sistema, es así como se visualiza el rol que cada uno de ellos ejercerá cuando el software se encuentre implementado en la máquina del cliente o usuario.

A continuación, se detallan los RU de las Gestiones Carrera, Asesoría e Informes. Es de anotar que se deben describir todas las gestiones. Ver tabla 9 de Requisitos Funcionales.

Tabla 9 Requisitos Funcionales (RF)

REQUISITOS FUNCIONALES (RF)				
ID REQUISITO	**NOMBRE DEL REQUISITO**	**DESCRIPCIÓN DEL REQUISITO**	**USUARIO**	**ID REQUISITO DE USUARIO**
RF-001	Crear Carrera	Permite registrar la información de las carreras con los siguientes datos: Código, nombre, duración y estado de la carrera. A través de opción guardar.	Secretaria, Administrador	***RU-001***
RF-002	Consultar Carrera	Permite consultar la información de las carreras por los campos código y nombre.	Secretaria, Administrador, Decano	***RU-001***
RF-003	Modificar Carrera	Permite modificar la información de las carreras a excepción del código, utilizando la opción guardar.	Secretaria, Administrador	***RU-001***
RF-004	Inhabilitar Carrera	Permite habilitar o inhabilitar una carrera, con previa confirmación de realizar el proceso.	Secretaria, Administrador	***RU-001***

RF-005	Cancelar Carrera	Permite cancelar el proceso que se esté realizando y regresar al estado inicial.	Secretaria, Administrador, Decano	***RU-001***
RF-006	Salir Carrera	Permite cerrar el formulario	Secretaria, Administrador, Decano	***RU-001***
RF-007	Crear Asesoría	Permite registrar la información de las Asesoría con los siguientes datos Código de la asesoría, fecha, hora inicial, hora final, duración de la asesoría, estado, observaciones. A través de opción guardar.	Asesor, Administrador	***RU-007***
RF-008	Consultar Asesoría	Permite consultar la información de las Asesorías.	Asesor, Administrador, Decano, Secretaria, Estudiante	***RU-007***
RF-009	Modificar Asesoría	Permite modificar la información de las Asesorías, a excepción del código	Asesor, Administrador	***RU-007***
RF-010	Inhabilitar Asesoría	Permite habilitar o inhabilitar una asesoría, con el fin de realizar un borrado lógico que posteriormente se pueda recuperar.	Asesor, Administrador	***RU-007***
RF-011	Cancelar Asesoría	Permite cancelar el proceso que se esté realizando y regresar al estado inicial	Asesor, Administrador, Decano, Secretaria, Estudiante	***RU-007***

RF-012	Salir Asesoría	Permite cerrar el formulario	Asesor, Administrador, Decano, Secretaria, Estudiante	***RU-007***

Fuente: Elaboración propia

En relación a los informes y consultas, éstos no poseen el CRUD, se detalla cada informe y cada consulta que el sistema deba ejecutar.

Tabla 10 Requisito Funcional (Informes)

REQUISITOS FUNCIONALES (RF) INFORMES				
ID REQUISITO	**NOMBRE DEL REQUISITO**	**DESCRIPCIÓN DEL REQUISITO**	**USUARIO**	**ID REQUISITO DE USUARIO**
RF-013	Proyectos y sus integrantes	El sistema mostrará el informe proyectos y sus integrantes, permitiendo visualizar la información por código del proyecto, nombre, identificación del integrante, nombres, apellidos.	Secretaria, Decano, Administrador, Asesor	***RU-008***
RF-014	Proyectos aprobados	El sistema mostrará el informe de los proyectos cuyo estado esté aprobado, permitiendo visualizar los campos (código del proyecto, nombre, estado del proyecto, datos de los integrantes.	Secretaria, Decano, Administrador	***RU-008***

Fuente: Elaboración propia.

4.1.3 REQUISITOS NO FUNCIONALES (RNF)

El requisito no funcional, está relacionado con todo aquello que no solicita directamente el cliente, en relación a lo que debe llevar o cumplir el sistema. Los RNF, tienen que ver con características externas de calidad como por ejemplo seguridad, facilidad de uso, interfaz práctica, velocidad en procesamiento,

amigabilidad, es todo aquello que el sistema debe llevar, buscando con ello la satisfacción de dicho cliente o usuario. En las siguientes tablas se pueden observar algunos RNF.

Tabla 11 Facilidad de uso ("usability")

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	Capacitación, antes de intensificar el uso del sistema los usuarios deben conocer su modo de uso.
RNF-002	Actualizar la información por ingreso o retiro de algún Funcionario.
RNF-003	Diseño adecuado a las necesidades del usuario, para que la aplicación sea intuitiva y sencilla de usar cumpliendo con los siguientes parámetros: Tendrá una interfaz atrayente: formación de los elementos acorde al diseño. La carga de información deberá ser rápida.

Fuente: Elaboración Propia.

Tabla 12 Confiabilidad

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	El sistema debe estar disponible las 24 horas del día
RNF-002	Debe asegurar la permanente actualización de la base de datos, cuando se registre la información.

Fuente: Elaboración Propia.

Tabla 13 Ambiente de trabajo "Performance"

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	Tiempo de respuesta: se espera minimizar el tiempo a un promedio de 20 segundos, con el fin de que no se haga muy pesada la interacción con la Base de Datos manejada vía web.
RNF-002	Asignar suficiente espacio a la base de datos para soportar las grandes cantidades de información suministradas.
RNF-003	Configuración adecuada del equipo, para soportar la correcta instalación de la aplicación.

Fuente: Elaboración Propia.

Tabla 14 Restricciones de diseño

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	El lenguaje de programación del sistema se espera implementar en Java.
RFN-002	Se requiere de licenciamiento para el desarrollo del software.

Fuente: Elaboración Propia.

Tabla 15 Seguridad

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	Encriptación de las claves.
RNF-002	Realizar copia de seguridad, automático cada 6 horas, en servidor ubicado en otro espacio fuera de la empresa.
RNF-003	Los usuarios deberán estar registrados, bajo la modalidad del perfil del sistema.

Fuente: Elaboración Propia.

Tabla 16 Documentación de usuario y sistemas de ayuda

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	Capacitación a los usuarios del sistema, con el fin de lograr un buen manejo del mismo.
RNF-002	Manuales de usuario.

Fuente: Elaboración Propia.

Tabla 17 Interfaz de Usuario

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	Será manejado a través de un computador de escritorio.
RNF-002	Los colores de la aplicación serán gris claro, azul claro y blanco.
RNF-003	El texto será manejado en color negro, la fuente de la letra será Arial tamaño 12.

Fuente: Elaboración Propia.

Tabla 18 Interfaces de comunicación

ID. REQUISITO	DESCRIPCIÓN DEL REQUISITO
RNF-001	El acceso al software será vía web, debe asegurarse la estabilidad y seguridad de la conexión.
RNF-002	Comunicación con la interfaz de usuario.

Fuente: Elaboración Propia.

Los **requisitos no funcionales**, descritos en las anteriores tablas, no equivalen a todos los que se pueden tener en cuenta, **cada software es diferente**, para contextos y necesidades diferentes. Lo que se busca con los RNF, es **garantizar** que el software cumpla con estándares de calidad.

4.1.4 ESCENARIOS DE LOS CASOS DE USO

Los escenarios de los casos de uso permiten evolucionar el diagrama de casos de uso y los Requisitos Funcionales (RF), dando claridad sobre las acciones que realiza el actor y las que realiza el sistema, buscando con ello que la especificación del requisito sea clara, concreta y que se relacione directamente la funcionalidad del respectivo requisito.

A continuación se muestran algunas de las plantillas de casos de uso, donde se pueden visualizar los respectivos escenarios que permiten la evolución del diagrama de casos y los RF, donde aparece el caso de uso, la función de los actores relacionados con el caso de uso, las precondiciones o sea el estado en que debe estar el sistemas para cumplir con su objetivo, la intervención del actor con el sistema, la respuesta del sistema ante determinado estímulo del actor y el resultado del proceso, el cual se puede visualizar en la pos condición. Es importante aclarar que el formato que se presenta en las siguientes plantillas, puede sufrir modificaciones, acorde a las necesidades informacionales.

En el flujo alternativo, se puede visualizar, la acción que realizan el actor y las respuestas que da el sistema, en caso que el flujo normal falle por algún motivo. Es importante aclarar que la descripción de los procesos se puede ampliar e incluso si es necesario, es viable utilizar seudocódigo, ya que existen procesos que lo requieren. En relación a herramientas que sirven para apoyar la modelación del software, existen algunos programas que permiten generar los escenarios, dentro del mismo caso de uso.

La especificación del requisito, debe conservar los mismos nombres, es decir, si en el caso de uso se hace alusión a la extensión (Crear), ésta debe seguir conservando el mismo nombre, y no cambiarse por (Registrar, nuevo, etc.), ya que ello se relacionará más adelante con el diseño, programación y funcionalidad del producto de software resultante.

Escenarios de los Casos de Uso para el sistema Proyectos de grado

Se hará alusión al escenario Gestión Carrera, en el cual se representan sus cuatro procesos como son: Crear una carrera, Consultar, Modificar e Inhabilitar. Es importante tener presente que se deberá construir todos los escenarios que se requieran. Ver tablas, donde se describe cada escenario de la gestión carrera.

Tabla 19 Escenario Crear Carrera (Sistema proyectos de grado)

NOMBRE	CREAR CARRERA	
Descripción	Permite registrar las carreras con los siguientes datos: Código, nombre, duración y estado de la carrera.	
Actor	Secretaria, Administrador, Decano.	
Precondiciones	El usuario debe estar logueado en el sistema	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de Maestros / Gestión Carrera.	2. El sistema despliega la interfaz con las opciones: Código, nombre, duración y estado de la carrera; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón Crear.	4. El sistema activa todos los campos y envía el cursor a Código y activa los botones cancelar y guardar.

	5. El usuario ingresa el código da enter.	El sistema verifica si el código existe.
	5. El usuario ingresa los demás datos de la carrera y va dando enter, para llenarlos todos.	6. El sistema valida los campos requeridos.
	7. El actor da clic en guardar.	8. El sistema almacena y arroja un mensaje de transacción exitosa.
	9. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
		10. El sistema confirma que hay inconsistencia en los datos y arroja el mensaje de alerta.
	11. El usuario da click en el botón cancelar	12. El sistema ubica el cursor en el código
	13. El usuario ingresa los datos correctamente y da clic en guardar.	14. El sistema verifica los datos nuevamente.
		15. El sistema almacena y arroja un mensaje de transacción exitosa.
	16. Clic en Salir	
Post-Condiciones	Existe una nueva carrera Registrada.	
Requisito Funcional	RF-001	

Fuente: Elaboración propia

Tabla 20 Escenario Consultar Carrera (Sistema proyectos de grado)

NOMBRE	CONSULTAR CARRERA	
Descripción	Permite consultar la información de las carreras.	
Actor	Secretaria, Administrador, Decano	
Precondiciones	El usuario debe estar logueado en el sistema.	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros/ Gestión Carrera	2. El sistema despliega la interfaz con las opciones Código, nombre, duración y estado de la carrera; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos Código, nombre, de la carrera para que el usuario realice la búsqueda por cualquiera de estos campos. Activa los botones modificar, inhabilitar y cancelar
	5. El usuario digita los datos para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una carrera existente.
		7. El sistema muestra los datos y todos los campos en estado inactivo.
	8. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
		9. El sistema confirma que los datos ingresados no son de una carrera existente y arroja el mensaje de alerta.
	10. El usuario ingresa los datos correctamente y da enter.	11. El sistema valida los datos nuevamente.

		12. El sistema muestra los datos y todos los campos en estado inactivo.
	13. Clic en salir	
Post-condiciones	Se consultó una carrera existente.	
Requisito funcional	Rf-002	

Fuente: Elaboración propia

Tabla 21 Escenario Modificar Carrera (Sistema proyectos de grado)

NOMBRE	MODIFICAR CARRERA	
Descripción	Permite modificar la información de las carreras	
Actor	Secretaria, administrador	
Precondiciones	El usuario debe estar logueado en el sistema y haber dado clic en el botón consultar.	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros/ Gestión Carrera.	2. El sistema despliega la interfaz con las opciones Código, nombre, duración y estado de la carrera; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos Código, nombre para que el usuario realice la búsqueda por cualquiera de estos campos y activa los botones modificar, inhabilitar, cancelar.
	5. El usuario digita los valores para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una carrera existente.
		7. El sistema muestra los datos y todos los campos en estado inactivo.

	8. El usuario da clic en el botón modificar.	9. El sistema activa los campos nombre, duración y estado de la carrera, envía el cursor al campo nombre y activa el botón guardar.
	10. El usuario modifica los datos y va dando enter.	11. El sistema valida los datos.
	12. El usuario da clic en el botón guardar.	13. El sistema almacena y arroja un mensaje de transacción exitosa.
	14. Clic en salir.	
FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
		15. El sistema confirma que hay inconsistencia en los datos y arroja el mensaje de alerta.
	16. El usuario ingresa los datos correctamente y da clic en guardar.	17. El sistema confirma los datos nuevamente.
		18. El sistema almacena y arroja un mensaje de transacción exitosa.
	19. Clic en salir	
Post-condiciones	Se modificó una carrera.	
Requisito funcional	RF-003	

Fuente: Elaboración propia

Tabla 22 Escenario Inhabilitar Carrera (Sistema proyectos de grado)

NOMBRE	INHABILITAR CARRERA	
Descripción	Permite activar o desactivar una carrera.	
Actor	Secretaria, Administrador	
Precondiciones	El usuario debe estar logueado en el sistema y con los permisos necesarios y haber dado clic en el botón consultar.	
FLUJO BÁSICO		
Pasos	Actor	Sistema
	1. El usuario ingresa al menú de maestros / Gestión carrera.	2. El sistema despliega la interfaz con las opciones Código, nombre, duración y estado de la carrera; deshabilitados, con los botones crear, consultar y salir activos y los demás inactivos.
	3. El usuario da clic en botón consultar.	4. El sistema activa los campos código, nombre para que el usuario realice la búsqueda por cualquiera de estos campos. Activa los botones modificar, inhabilitar, cancelar.
	5. El usuario digita los valores para la búsqueda y da enter.	6. El sistema valida que los datos ingresados sean de una carrera existente.
		7. El sistema muestra los datos y todos los campos en estado inactivo.
	8. El usuario da clic en el botón inhabilitar.	9. El sistema muestra advertencia y valida si está seguro de inhabilitar la carrera, dando opciones si/no
	10. El usuario selecciona la opción sí.	11. El sistema cambia el estado de la carrera y arroja un mensaje de transacción exitosa.
	12. Clic en salir.	

FLUJO ALTERNATIVO		
Pasos	Actor	Sistema
	13. El usuario selecciona la opción no.	14. El sistema muestra los datos y todos los campos en estado inactivo. El sistema inactiva los botones modificar e inhabilitar
	15. Clic en salir	
Post-condiciones	Se cambió el estado de una carrera.	
Requisito funcional	RF-004	

Fuente: elaboración propia.

De igual forma se documenta para cada caso de uso, los respectivos escenarios, utilizando las plantillas en mención u otras que permitan describir el proceso.

4.2 TEMA 2: UTILIZACIÓN DE HERRAMIENTAS PARA MODELADO Y DOCUMENTACIÓN DE LA ESPECIFICACIÓN

El modelado de sistemas software es una técnica que ayuda al experto informático a "visualizar" el sistema a construir, permitiendo verificar el proceso, validarlo y aplicar correctivos a tiempo que en definitiva busca la disminución de recursos para su construcción, así como la calidad del proceso y del producto final.

Para los procesos que maneja la Ingeniería de Software, es importante la utilización de diferentes herramientas que apoyan el trabajo, permitiendo agilizarlo, complementarlo y automatizarlo, es el caso del Lenguaje Modelador UML, que a través de herramientas de software, facilitan entre otros aspectos:

- Administrar Requisitos
- Modelar y analizar los procesos de negocios
- Construir diseño y modelos de comportamientos
- Generar e importar código fuente en una variedad de lenguajes
- Generar e importar esquema de base de datos
- Generar e importar código basado en diferentes lenguajes de programación
- Crear modelos de componentes y de despliegue
- Rastrear cambios

- Administrar pruebas
- Confirmar la trazabilidad desde los requisitos a través y hasta el despliegue
- Documentar su desarrollo de software
- Comunicar y desarrollar proyectos de ingeniería de software basados en el equipo
- Modelado/ingeniería rápida de su desarrollo de software

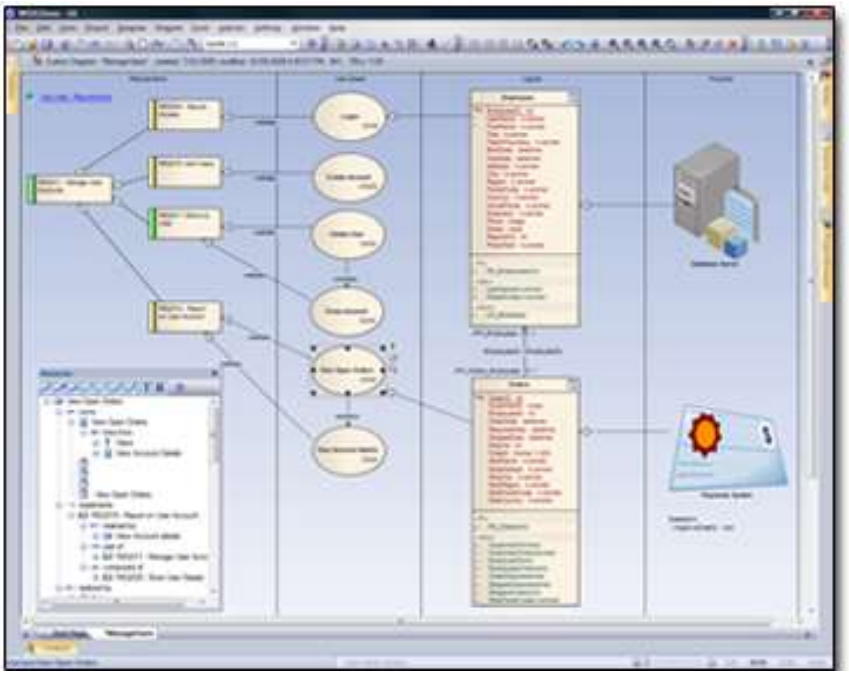
Por lo tanto la construcción de diferentes diagramas, entre ellos: diagramas de actividad, casos de uso, secuencia, clases, despliegue, estado, colaboración, entre otros , de igual forma como esquemas, tablas, escenarios, diseños, permiten clarificar cada etapa del ciclo de vida del software.

4.2.1 ALGUNAS HERRAMIENTAS PARA EL MODELADO

Existe gran variedad de herramientas, entre algunas de ellas tenemos:

- **Enterprise Architect:** una herramienta de modelado y diseño visual, orientada a la construcción de productos de software, modelado de negocio, industria, donde en lo que tiene que ver con software, es aplicado a todo el ciclo de vida (gestión de requisitos, arquitectura de software, programación, pruebas, mantenimiento, gestión del cambio, gestión de proyectos, implementación o despliegue en las máquinas del cliente.

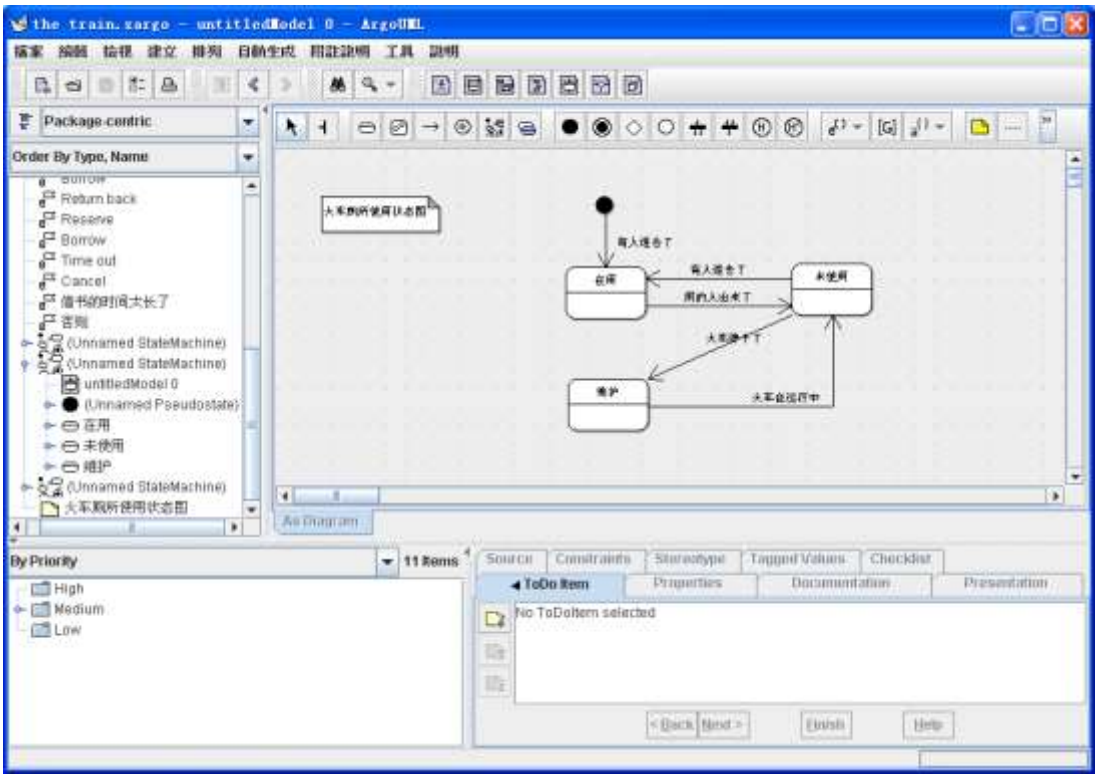
Imagen 46 Pantalla Enterprise Architect



Fuente: http://www.sparxsystems.com.au/products/ea/

ArgoUML: Es una aplicación de diagramado de UML, cuyos diagramas se interrelacionan con los lenguajes de programación: Java, PHP, Python, C++ y Csharp (C#). Dicha herramienta maneja los diagramas de clases, estados, casos de uso, actividad, colaboración, desarrollo, secuencia.

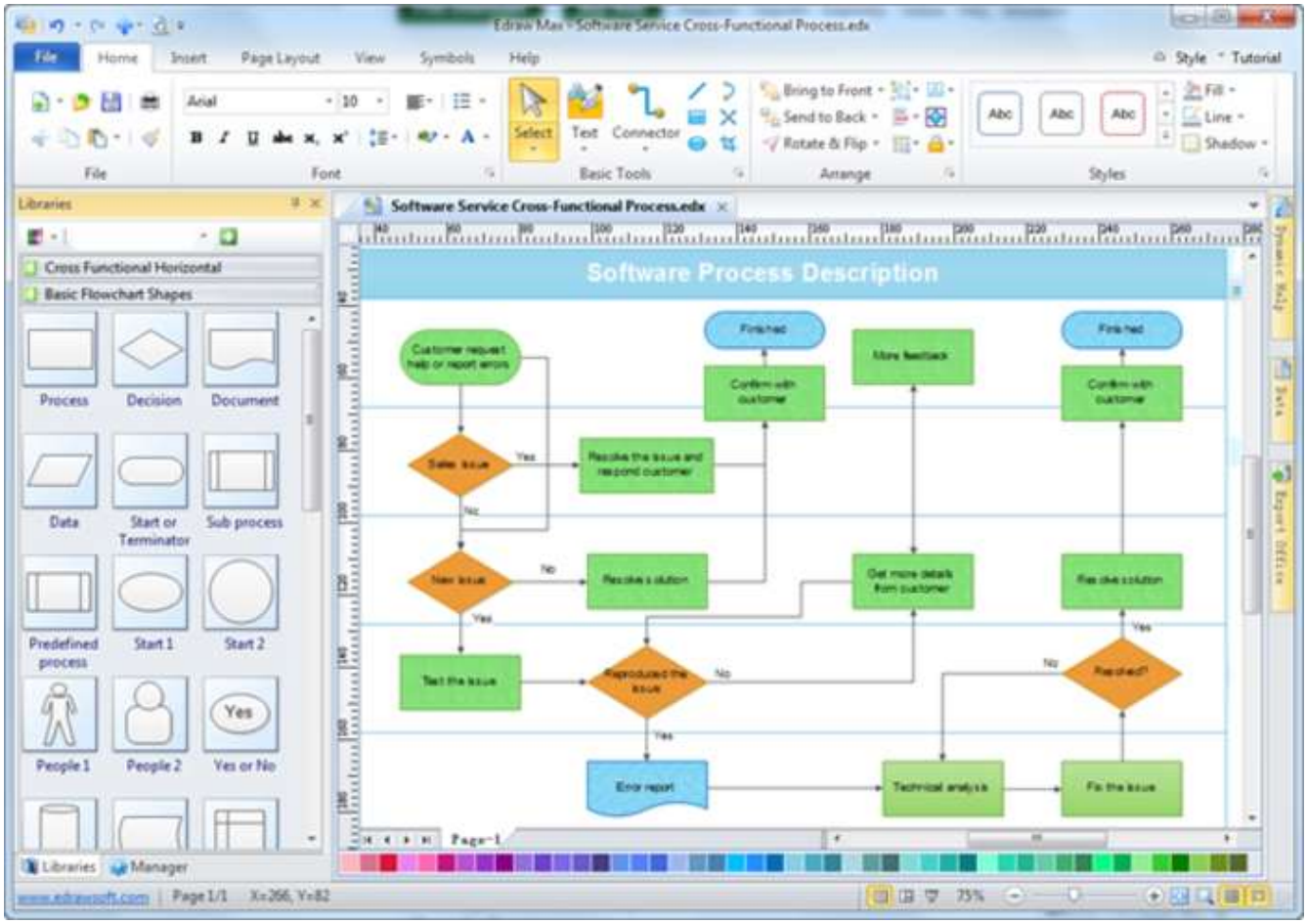
Imagen 47 ArgoUML



Fuente: https://upload.wikimedia.org/wikipedia/commons/6/64/ArgoUML.png

Edraw: Es una herramienta que permite a los programadores e ingenieros crear diagramas UML, flujos de trabajo, estructuras de programas, diagramas de diseño web, diagramas de ingeniería eléctrica y diagramas de base de datos.

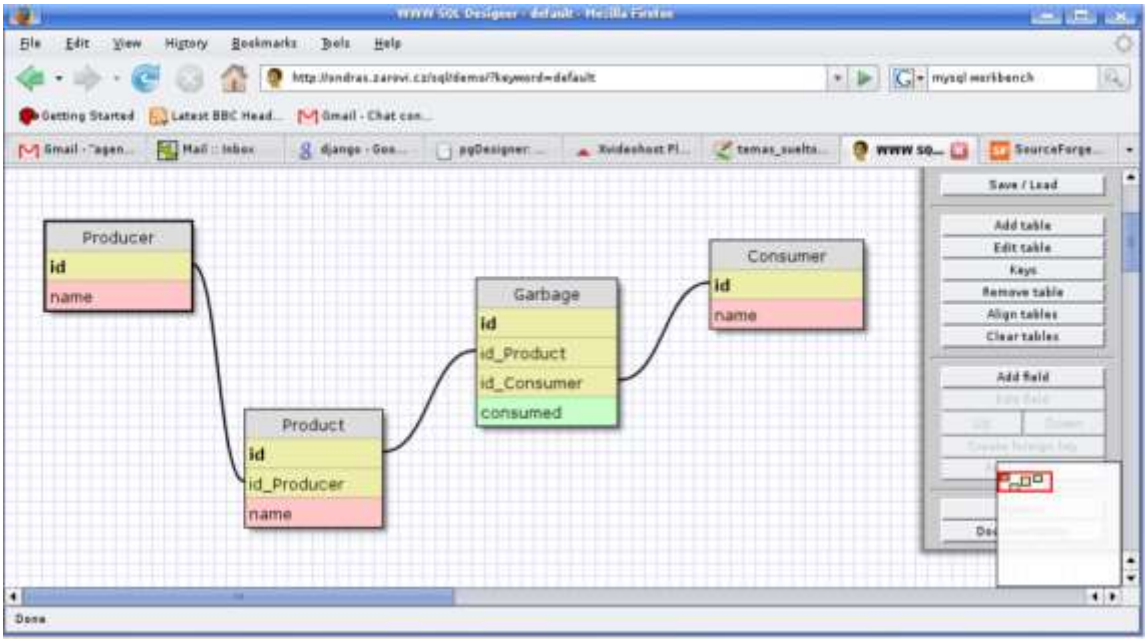
Imagen 48 Edraw



Fuente: http://screenshots.en.sftcdn.net/en/scrn/85000/85569/scr_1385131088-700x493.png

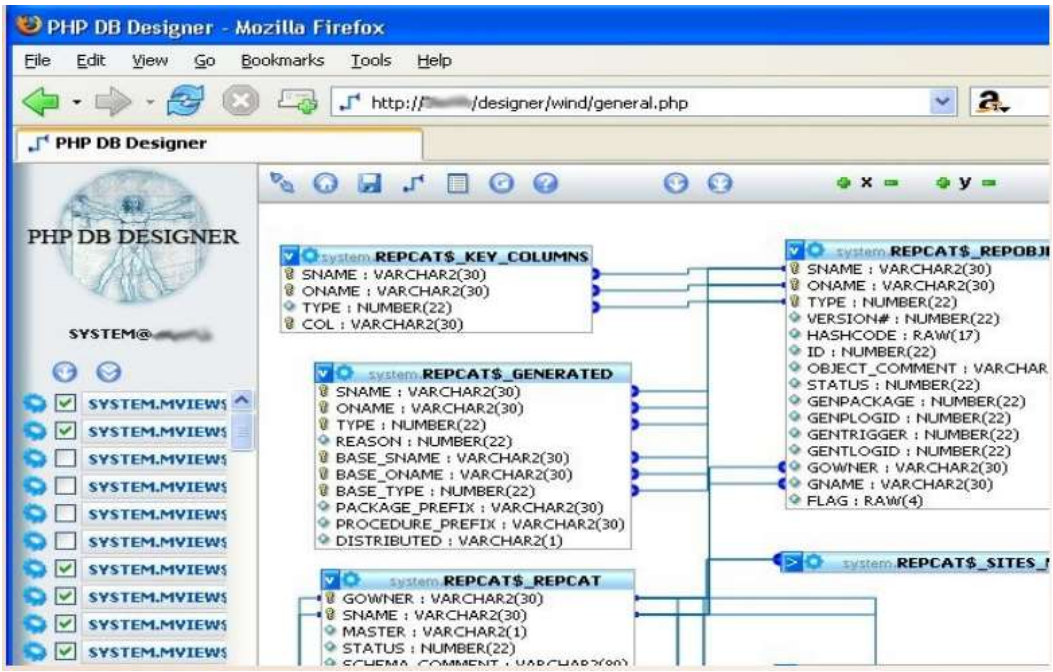
- **Herramienta utilizada para modelar base de datos:** permiten crear su estructura, con el fin de analizarla y validarla, antes de ubicarlas dentro del software como soporte para almacenar datos, y permitir sus operaciones básicas como son consulta, modificación, eliminación de información. Algunos modeladores para base de datos son: Sql Designer, PHP DB Designer, entre otros.

Imagen 49 Sql Designer



Fuente: http://ondras.zarovi.cz/sql/

Imagen 50 PHP DB Designer



Fuente: http://sourceforge.net/projects/phpdbdesigner/

- **Otras herramientas**

  ●**Use Case Maker:** Permite documentar los casos de uso, http://use-case-maker.sourceforge.net/index.html

  ●**ObjectBuilder:** permite documentar clases, relaciones, métodos, etc., http://sourceforge.net/projects/objectbuilder/

  ●**BoUml:** herramienta de diseño UML multiplataforma, es bastante completa tiene todos los diagramas UML estándares y genera código, http://bouml.sourceforge.net/

  ●**Gaphor:** posee las mismas características que BoUml, pero con menos diagramas, http://gaphor.devjavu.com

Entre múltiples herramientas que permiten el modelado y la documentación requerida.

4.2.2 MODELADO UTILIZANDO DIAGRAMAS DE SECUENCIA

- **Diagrama de secuencia:** Es uno de los diagramas UML, cuyo objetivo consiste en mostrar las interacciones entre objetos ordenadas en secuencia temporal. Se puede visualizar los objetos que se encuentran en el escenario y la secuencia de mensajes intercambiados entre los objetos para llevar a cabo la funcionalidad descrita por el escenario. El diagrama de secuencia muestra los objetos como líneas de vida de arriba hacia abajo y muestran la forma como se comunican los objetos y qué mensajes disparan esas comunicaciones. Los diagramas de secuencia no están pensados para mostrar lógicas de procedimientos complejos.

- **Elementos de un diagrama de secuencia**
  - ✓ **Líneas de vida:** Están compuestas por el o los actores que participan en el proceso, y las capas que llevará el software como, por ejemplo, si se trabaja a tres capas, éstas serían (vista, controlador y modelo), aunque pueden ser más capas dependiendo del tipo de software. La vista corresponde a la interfaz gráfica del software como por ejemplo menú, submenú, pantallas; en relación al controlador, éste se relaciona con los componentes de programación a nivel general que llevará el sistema y como tercera capa se encuentran los datos o almacenamiento que se relaciona con el proceso activo.
  - ✓ **Mensajes:** El mensaje se mueve entre los diferentes objetos y siempre van de arriba hacia abajo, hasta que se termina la línea de vida, en dicho caso el sistema libera el objeto, con el fin de que se

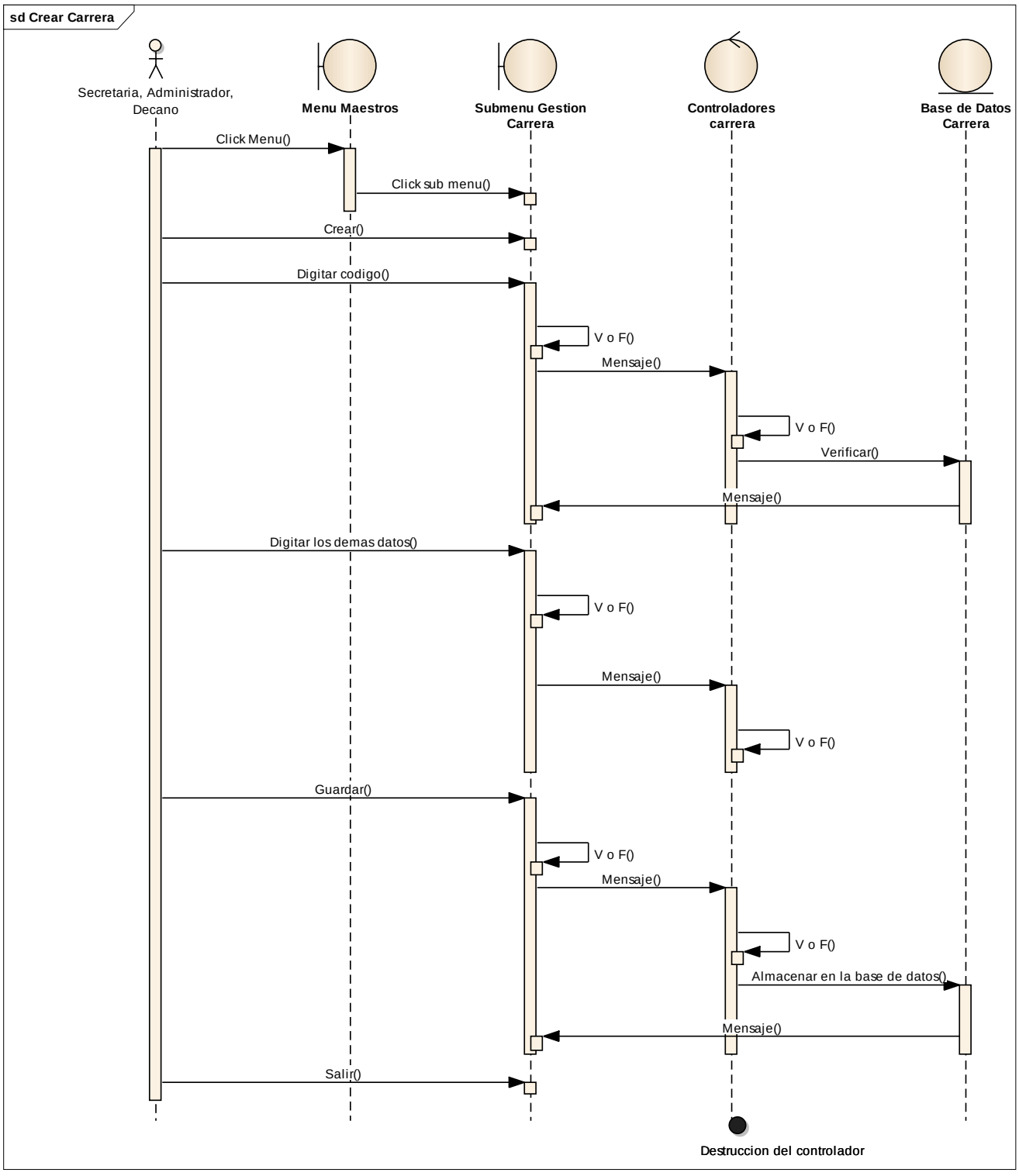
pueda hacer nuevamente uso de él. Para diseñar la liberación o destrucción del objeto, las herramientas modeladoras traer un componente que puede ser un punto o una equis.

El diagrama de secuencia evoluciona el diagrama de casos de uso y se basa en los escenarios de éstos, mostrando de qué forma fluctúa la información en el sistema. Ver video Diagrama de secuencia en link:

Diagrama de Secuencia Enlace

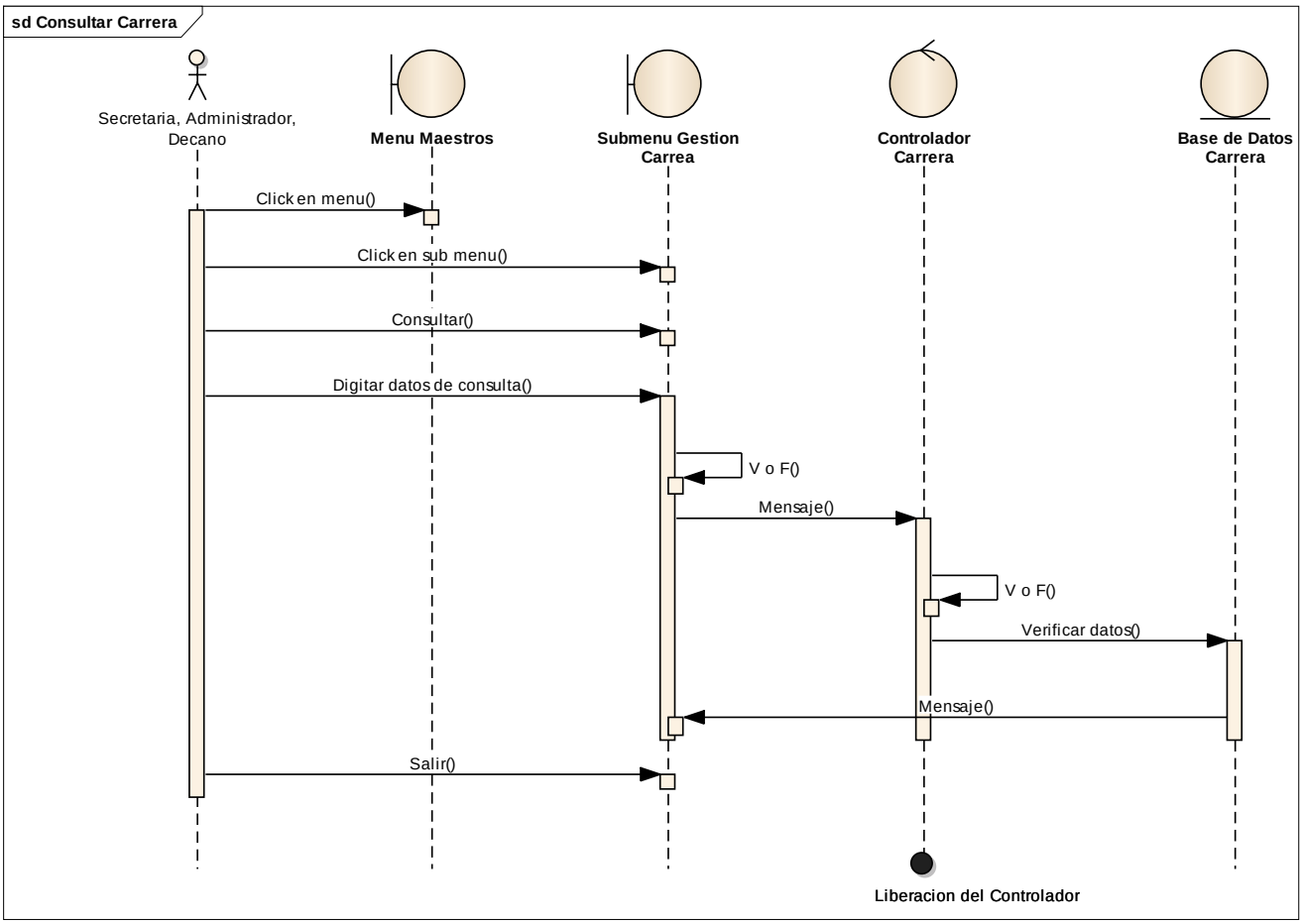
Retomando la situación problemática, utilizada para ejemplificar mejor el proceso de ingeniería de software, se presenta el diagrama de secuencia que permite evolucionar la gestión carrera. Dichos diagramas de secuencia se basan en las plantillas de los escenarios de los casos de uso. Ver imágenes de los diagramas de secuencia.

Imagen 51 Diagrama de secuencia Crear Carrera (Proyectos de grado)



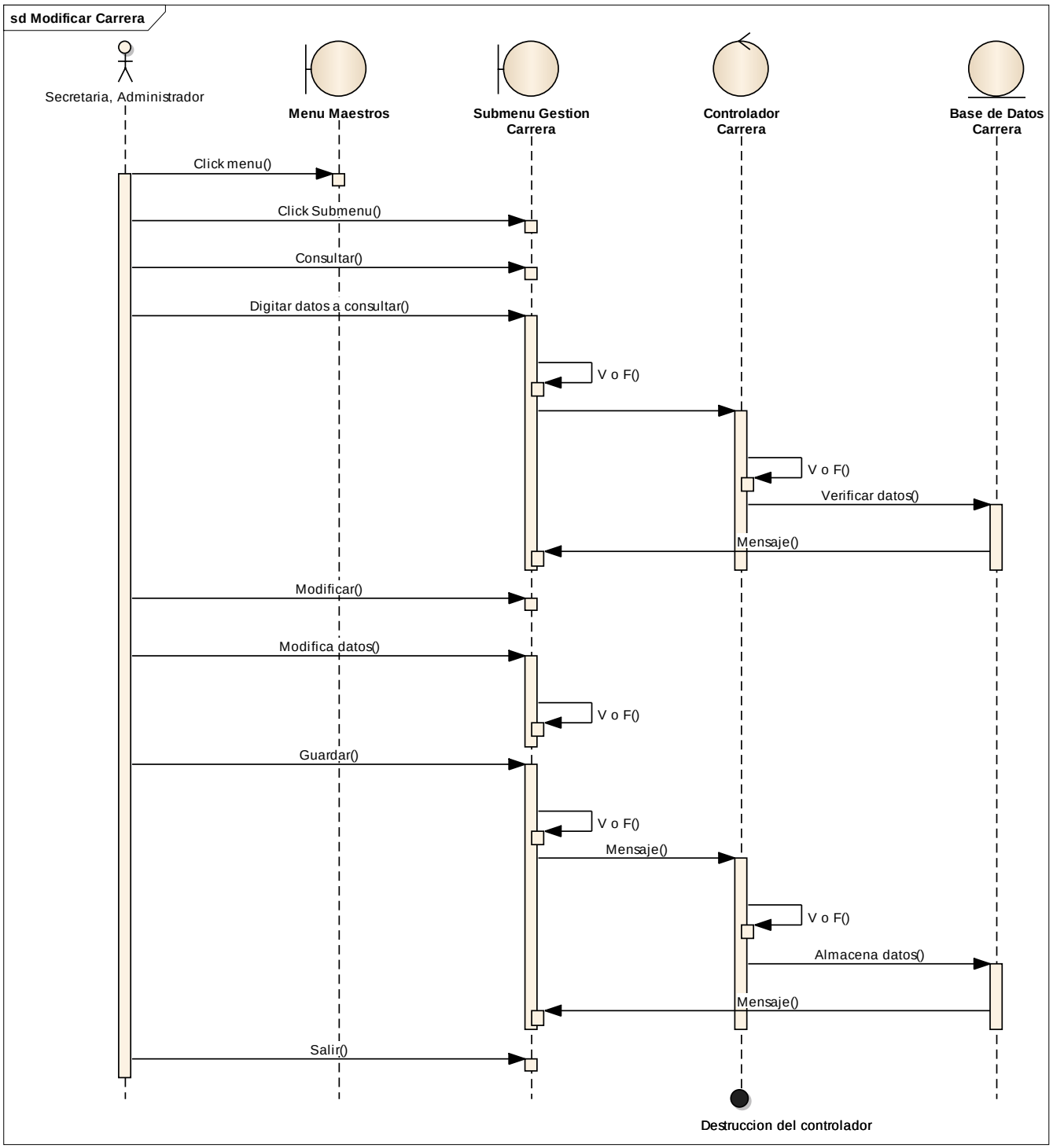
Fuente: Elaboración propia.

Imagen 52 Diagrama de secuencia Consultar Carrera (Proyectos de grado)



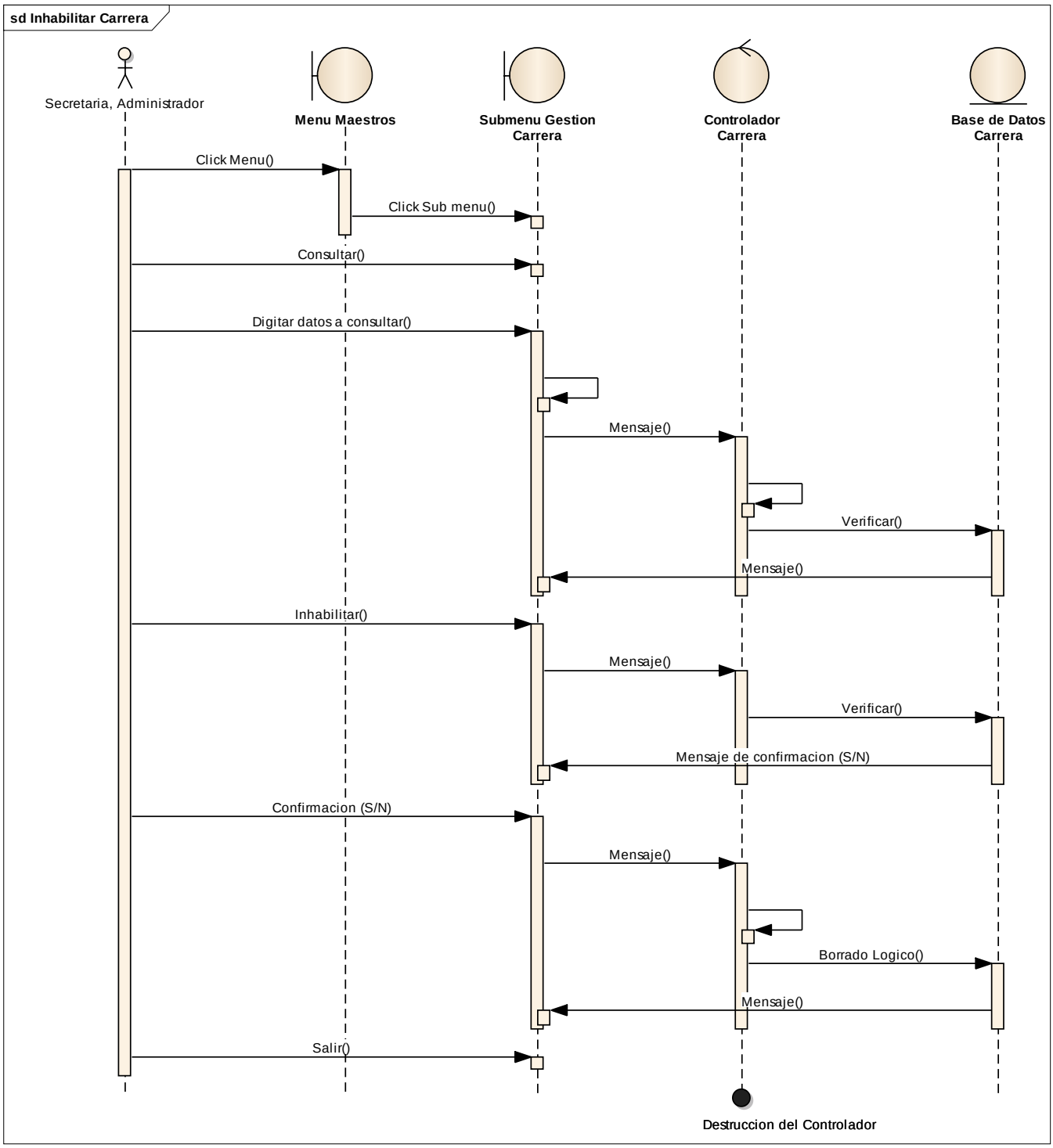
Fuente: Elaboración propia.

Imagen 53 Diagrama de secuencia Modificar Carrera (Proyectos de grado)



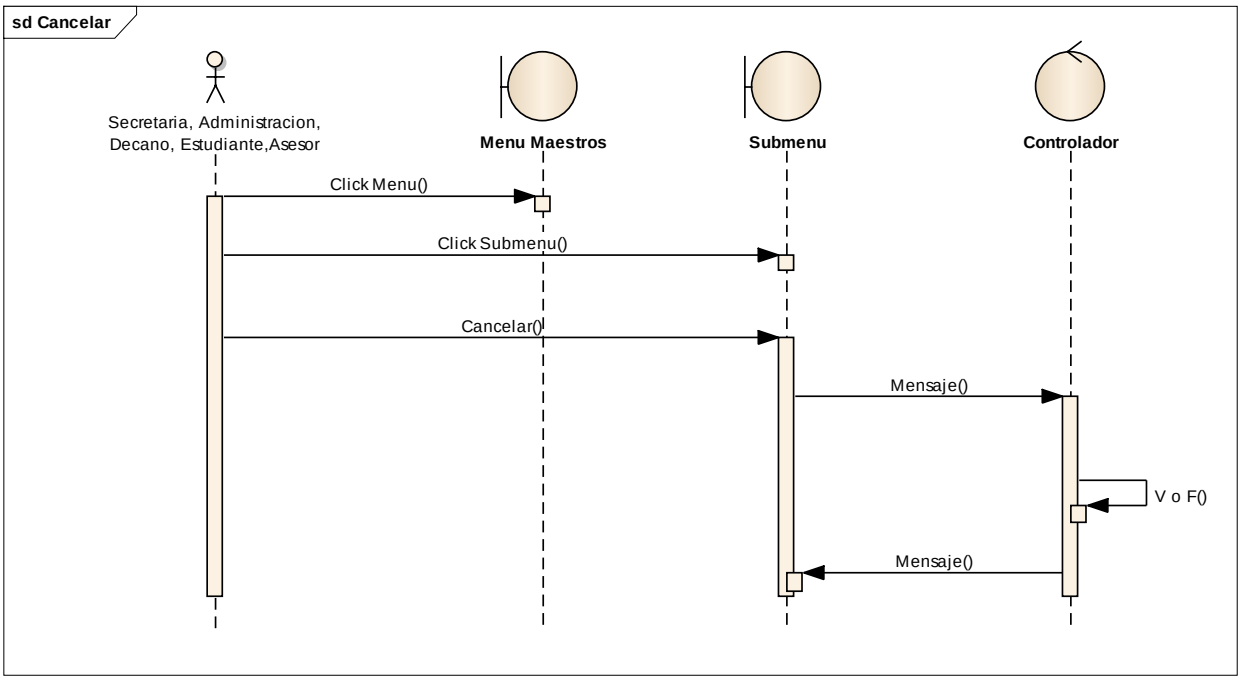
Fuente: Elaboración propia.

Imagen 54 Diagrama de secuencia Inhabilitar Carrera (Proyectos de grado)



Fuente: Elaboración propia.

Imagen 55 Diagrama de secuencia Cancelar Carrera (Proyectos de grado)



Fuente: Elaboración propia.

De igual forma, cada uno de los escenarios, se deben **modelar** y **documenta**r.

4.3 TEMA 3: CALENDARIZACIÓN Y RECURSOS DEL PROYECTO

La planeación de un proyecto, permite realizar seguimiento de las actividades, en relación al tiempo propuesto para ello, además de la utilización de recursos que están representados en talento humano, hardware, software, tiempo, costo, entre otros recursos que permiten la construcción e implementación del producto de software. Ver video Gestión de proyectos al iniciar la planificación de un proyecto de software,

Gestión de proyectos al iniciar la planeación de un proyecto de software Enlace

Es importante realizar la planeación de cada una de las tareas, con el fin de medir tiempos, asignando responsabilidades con la claridad de los resultados que se esperan obtener. Es importante aclarar que no siempre los tiempos establecidos y los recursos presupuestados, se cumplen a cabalidad, ya que en el desarrollo de las actividades se pueden presentar imprevistos que causan retrasos, como por ejemplo el establecimiento de fechas límite irrealizables, los diferentes cambios que se pueden presentar cuando los proyectos están en marcha, las dificultades humanas, la falta de disponibilidad del cliente para apoyar al proyecto, entre otros factores que intervienen con la calendarización propuesta, la cual se involucra a través de un cronograma de actividades.

4.3.1 HERRAMIENTAS PARA CONSTRUIR CRONOGRAMAS

Existen varias estructuras que permiten la planeación de cronogramas, donde uno de los más utilizados corresponde al diagrama de Gantt: conocida como una herramienta que se emplea para planificar y programar tareas a lo largo de un período determinado de tiempo. El diagrama de Gantt permite visualizar las tareas a realizar, realizar seguimiento y control sobre ellas, permitiendo dar un manejo adecuado a la evolución del proyecto. Dicho diagrama, fue propuesto por Henry Laurence Gantt a principios del siglo XX. En dicho diagrama se puede observar claramente los procesos, las actividades, los tiempos asignados, los recursos, responsables, la representación gráfica del proceso, entre otros aspectos que ayudan a monitorizar los procesos programados en dicha calendarización.

4.3.2 SOFTWARE SUGERIDO

Existe en el medio gran variedad de programas que permiten la elaboración de cronogramas para la organización y planeación de procesos, tareas y actividades que van desde programas sencillos como por ejemplo un procesador de texto, una hoja electrónica, hasta software especializado que facilita dicha automatización que van desde versiones libres hasta versiones licenciadas.

Algunos de los programas más utilizados para la construcción de los diagramas de Gantt son:

Imagen 56 Algunas herramientas para construir diagramas de Gantt

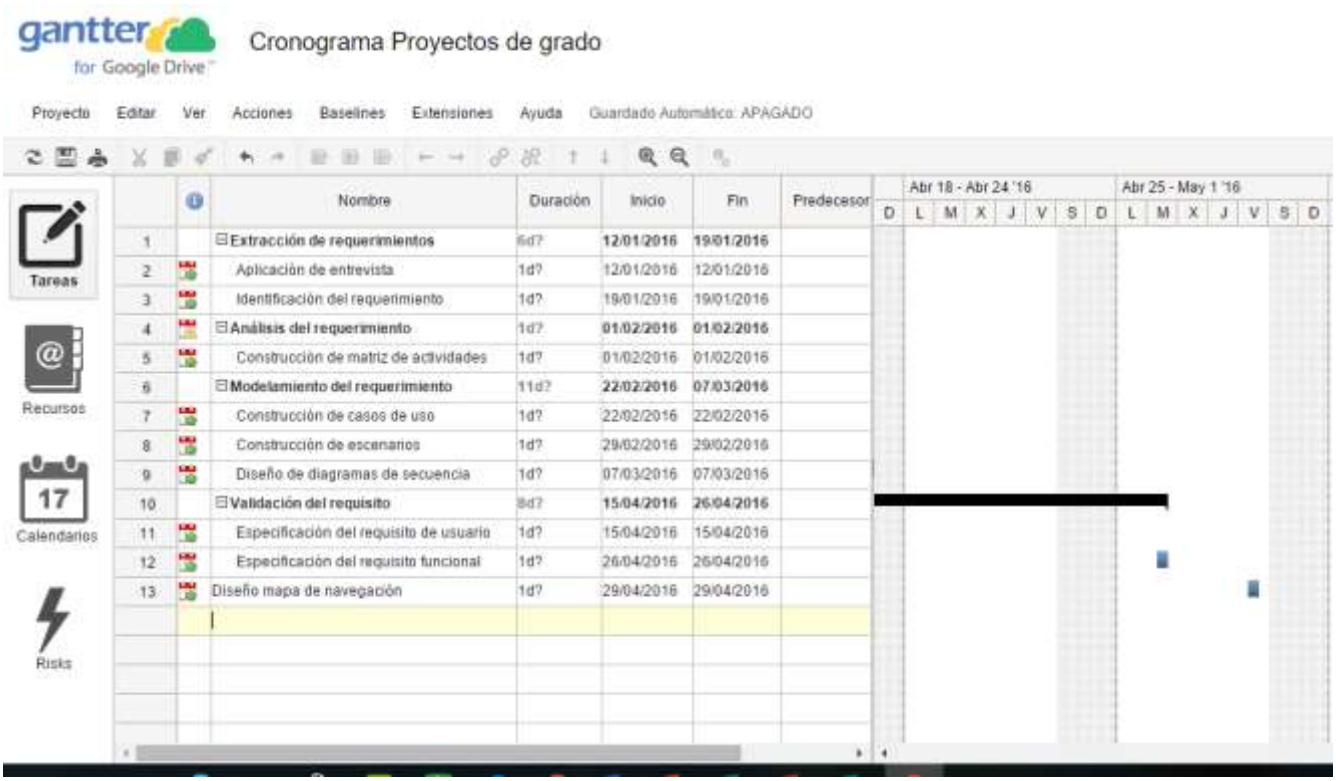
Software	Finalidad	Acceso	Compatibilidad	Coste
Gantt Designer	Diseño	Escritorio	Windows	Gratuito
GanttProject	Gestión	Escritorio	Win, Mac, Linux	Gratuito
GanttPV	Diseño	Escritorio	Win, Mac, Linux	Gratuito
MS Excel	Cálculo	Escritorio/ Web	Windows	De pago
MS Project	Gestión	Escritorio/ Web	Windows	De pago
MS Visio	Diseño	Escritorio/ Web	Windows	De pago
Gantter	Gestión	Web	Online	Gratuito
TaskJuggler	Gestión	Escritorio	Win, Mac, Linux	Gratuito
ProjectLibre	Gestión	Escritorio	Win, Mac, Linux	Gratuito
Smartsheet	Gestión	Web	Win, Mac, Linux	De pago

Fuente: http://www.masquenegocio.com/2015/06/25/herramientas-cronograma-gantt/

4.3.3 CALENDARIZACIÓN APLICADA AL SISTEMA DE EJEMPLO (PROYECTOS DE GRADO)

Sobre la calendarización para el proyecto de estudio, se utiliza la herramienta Gantter, en la cual se puede configurar el proceso a trabajar y las actividades que corresponde a cada proceso, de igual forma permite organizar el tiempo asignado a cada actividad que puede ser por horas, días, semanas, donde cada empresa de desarrollo de software determina su unidad de medida en relación al tiempo. Dichas herramientas permiten configurar los recursos, responsables de las tareas y demás opciones que proporcione el programa, buscando con ello dar información que permita ejercer los respectivos controles sobre el cumplimiento de las actividades planeadas. Ver imagen 57.

Imagen 57 Ejemplo de calendarización (Cronograma Proyectos de grado)



Fuente: elaboración propia.

4.3.4 RECURSOS REQUERIDOS PARA EL PROYECTO

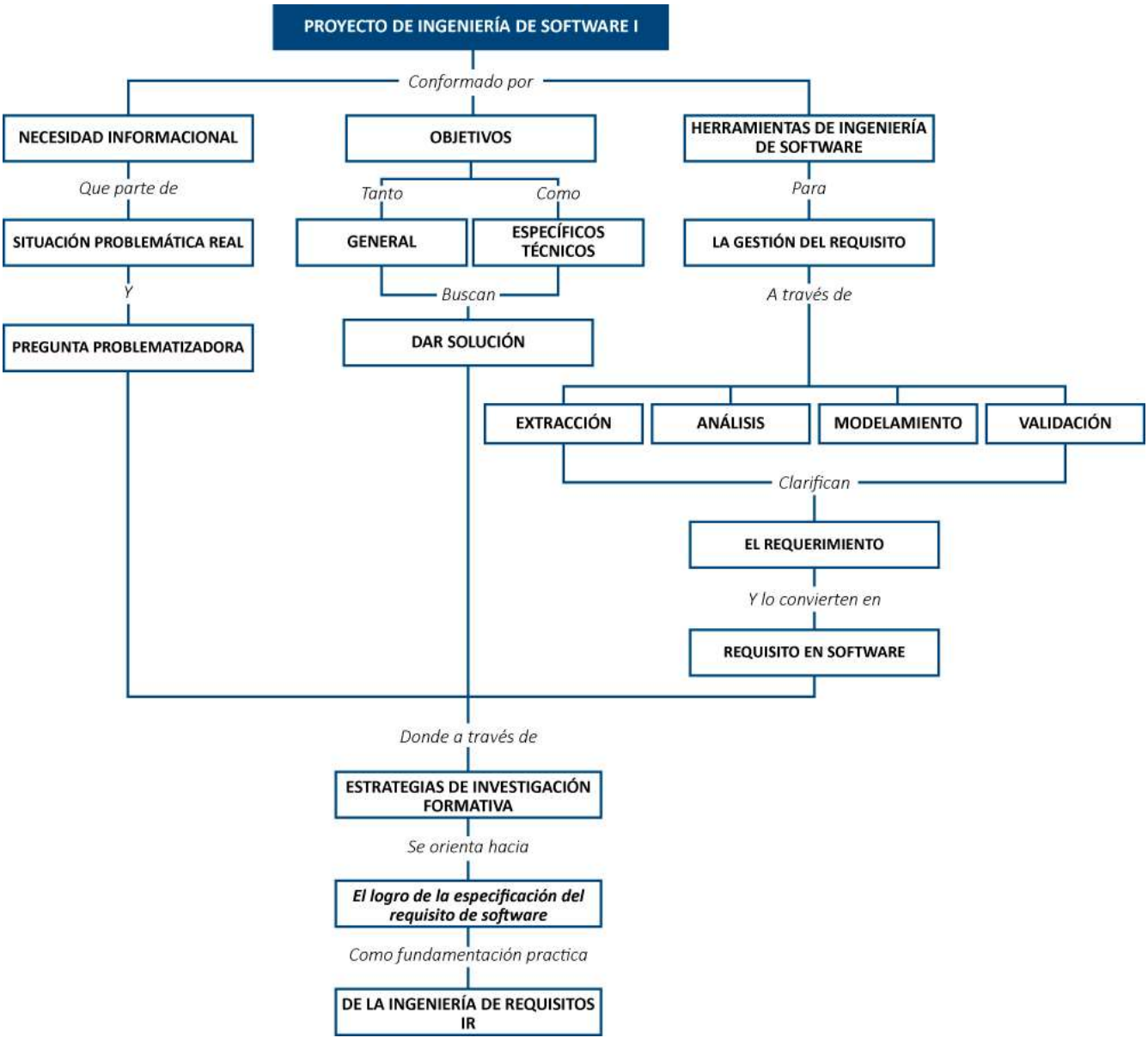
Es importante tener presente que para la realización del proyecto, se requieren recursos que están representados en equipos de cómputo, a nivel de hardware, software, conectividad, servidores, entre otros aspectos técnicos y que de igual forma se hace necesario el talento humano tanto de usuarios informáticos como no informáticos, los que de acuerdo a sus perfiles laborales tienen un compromiso con el desarrollo del sistema para la conformación de los equipos de trabajo, donde en definitiva buscan la construcción de un producto de calidad, donde tanto la parte proveedora del software como la parte del cliente, deben quedar a gusto con los resultados del trabajo.

Para lo anterior es necesario realizar estudio de costos, donde se tenga en cuenta los recursos requeridos para el despliegue del producto de software.

4.4 TALLER DE ENTRENAMIENTO UNIDAD 3

Nombre del taller: Taller 3	Modalidad de trabajo: Aprendizaje basado en problemas
Actividad previa: Realización por parte del estudiante del taller 2, y revisión, retroalimentación por parte del docente.	
Describa la actividad: **Evolucionando la solución del taller 2. Se pide:** 1. Diseñar la tabla de Requisitos de Usuario 2. Diseñar la tabla de Requisitos Funcionales 3. Construir un escenario para cada caso de uso (Requisito Funcional) 4. Diseñar los diagramas de secuencia para cada escenario	

5 UNIDAD 4: PROYECTO DE INGENIERÍA DE SOFTWARE (APRENDIZAJE BASADO EN PROBLEMAS E INVESTIGACIÓN FORMATIVA)



Fuente: elaboración propia

5.1 TEMA 1: GUÍA INGENIERÍA DE SOFTWARE I (ESPECIFICACIÓN DE REQUISITOS DE SOFTWARE)

Dentro de la orientación de proceso pedagógico en relación a la asignatura Ingeniería de Software I, se trabajará con aprendizaje basado en problemas (ABP) e investigación formativa, donde se tomará una necesidad que se tenga en el contexto y que deba ser solucionada a través de un producto de software (con cliente real), el cual se documentará, siguiendo los pasos que ofrece la guía en mención, con el aporte de las diferentes temáticas que ofrece la Ingeniería de Software. Es de anotar que el estudiante trabajará con el mismo proyecto para las tres asignaturas de Ingeniería de Software (I, II, III), terminando con un producto funcional que cumpla con los requerimientos y normas de calidad.

5.1.1 CONDICIONES PARA LA CONSTRUCCIÓN DEL PROYECTO DE INGENIERÍA DE SOFTWARE I

1. El estudiante se basará en documento (guía de Ingeniería de Software I)
2. El proyecto se aplicará a una necesidad real, con cliente real y contará con carta de aceptación debidamente firmada, por parte del cliente (el formato será suministrado por el docente orientador de la asignatura)
3. El estudiante tendrá acompañamiento del docente, para revisión de entregables y asesoría, acorde a planeación del docente.
4. Éste proyecto contará con aportes de investigación formativa, ya que analizará situación problemática, pregunta problematizadora, objetivos técnicos y fundamentación en Ingeniería de Software I.

5.2 TEMA 2: DESARROLLO DE PROYECTO DE SOFTWARE (GUÍA INGENIERÍA DE SOFTWARE I)

El desarrollo de un proyecto con cliente real, permite al estudiante aplicar los conceptos teóricos a una situación práctica, con el fin de dar solución a una necesidad informacional a través de herramientas de Ingeniería de Software, hasta la etapa de la ingeniería de requisitos, convirtiendo la fundamentación teórica del módulo en una solución práctica que tiene como fin acercar al estudiante hacia la contextualización del sector productivo de software. Ver archivo anexo (Guía Ingeniería de Software I).

5.2.1 COMPONENTES DE LA GUÍA, PARA LA CONSTRUCCIÓN DEL PROYECTO

- Portada
- Tabla de contenido generada por el sistema
- Glosario o vocabulario de las palabras relevantes en el documento (mínimo 10)
- Introducción sobre el trabajo
- Justificación del proyecto
- Ubicación de la empresa donde se realizará el proyecto
    - Razón social
    - Reseña histórica
    - Misión y Visión
    - Objeto social
    - Sector al que pertenece
    - Ubicación Geográfica
    - Datos del contacto
    - Tamaño de la empresa
    - Organigrama
    - Carta de aceptación para la realización del proyecto (firmada por el cliente)
- Extracción de requisitos
    - Formato de entrevista
    - Desarrollo de entrevista
    - Análisis de la entrevista
    - Diagrama de actividades
- Definición del problema
    - Antecedentes del problema (paso a paso del sistema actual)
    - Planteamiento del problema (Pregunta problematizadora)
    - Justificación del problema
- Diagrama de carril
- Listado de necesidades y características
- Objetivo general

- Objetivos específicos Técnicos
- Alcance del sistema propuesto en términos de (entradas, procesos y salidas)
- Nombre que se le colocará al sistema de software
- Cronograma de actividades (Calendarización, utilizando diagrama de Gantt)
- Análisis de riesgos
    - Riesgo Tecnológico
    - Riesgo Cliente
    - Riegos Proceso
    - Riesgo Producto.
- Análisis de requisitos
    - Tabla General para casos de uso
    - Requisitos de Usuario
    - Requisitos Funcionales
    - Requisitos no funcionales
- Modelo del Requisito.
    - Descripciones generales de Actores.
    - Especificación del requisito
    - Diagramas del Modelo de Casos de Uso.
    - Documentación o especificación de los Casos de Uso (Escenarios representados en plantillas)
    - Diagrama de secuencia
    - Plantilla para diagrama de secuencia
- Mapa de navegación
- Recursos (hardware, Software, Talento Humano)
- Conclusiones (mínimo tres párrafos)
- Referencias bibliográficas, web grafías.

5.3 TALLER DE ENTRENAMIENTO UNIDAD 4

Nombre del taller: Proyecto	Modalidad de trabajo: Aprendizaje basado en problemas e investigación formativa
Actividad previa: Estudio de unidades, acorde a tiempos para entregables	
Describa la actividad: La actividad, se desarrollará a lo largo de módulo, donde se realizarán asesorías físicas y/o virtuales en relación a los diferentes entregables, los cuales deberán ser suficientemente retroalimentados, tanto por parte del docente como del estudiante quien debe realizar los ajustes pertinentes, hasta que el proyecto cumpla con los objetivos del módulo. El proyecto debe ser individual, evitando dificultades en caso de cambio de jornada, cancelación de materia, entre otras, para evitar que deba iniciar otro proyecto, para las materias ingeniería de software II y III.	

6 PISTAS DE APRENDIZAJE

Tenga presente que: el software es un producto esencial en cualquier actividad y siempre busca simplificar procesos que antes consumían mucho tiempo y recursos.

Recuerde que: a partir del 1990, se da inicio a una nueva era llamada la "Era de la información y el conocimiento", donde hoy, para cualquier empresa, corporación, institución, organización, el activo más importante equivale a la información y para el manejo de ésta el software es indispensable.

Sabía usted que: existen diferentes normas internacionales para garantizar la calidad del software y que una de ellas obedece a La familia de normas ISO/IEC 25000 de 2014, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), la que se orienta especialmente a evaluar la calidad del producto software.

Tenga presente que: existen guías que orientan el proceso de construcción del software como SWEBOK (Cuerpo del conocimiento para la Ingeniería de Software), con el cual se busca realizar un aporte importante al proceso de construcción del software.

Sabía usted que: existen una gran diferencia entre usuario y cliente, ya que el cliente es quien paga por el producto de software y no necesariamente utiliza el software y el usuario, se refiere a quienes usan el software.

Tenga presente que: el requerimiento de software, no es lo mismo que el requisito del software, ya que el requerimiento corresponde a la necesidad que tiene el cliente de automatizar un proceso a través de medios informáticos y el requisito de software corresponde a una etapa posterior que consiste en evolucionar el requerimiento a través de herramientas de ingeniería del software, convirtiéndose en la funcionalidad del sistema.

Recuerde que: el UML, no es un lenguaje de programación, sino un modelador que ayuda a los ingenieros de software a entender cada uno de los procesos del ciclo de vida del software.

Tenga presente que: el software es un producto especial, por lo tanto debe tener altos niveles de calidad, ya que una falla en el sistema, puede significar grandes pérdidas económicas y lo más delicado aún la pérdida de vidas.

Sabía usted que: la tendencia en metodologías de software, está orientada hacia metodologías ágiles, donde el cliente debe pertenecer al equipo de desarrollo, buscando que los resultados funcionales se den rápidamente, pero que no es la metodología definitiva, ya que se siguen realizando cambios en el proceso de construcción del software.

7 GLOSARIO

Actor: Persona, máquina, dispositivo, aplicación y todo aquello que interactúe con el software y que de alguna forma le envíe señales, datos e información, para que éste ejecute algún proceso.

Análisis: Corresponde a una de las fases del ciclo de vida para el desarrollo del software, la cual permite realizar una revisión completa de los requerimientos para pasarlos a la etapa de construcción del software.

Calendarización: Agendamiento de las actividades a realizar en un proyecto, incluyendo tiempos, recursos, responsables.

Ciclo de vida del software: Procesos que se realizan para la construcción del software, donde se divide por etapas su construcción, las que pueden ser requerimientos, análisis, diseño, programación, pruebas, documentación, implementación, gestión de configuración, dependiendo de tipo de metodología.

Cliente: Persona natural o jurídica que paga por un producto o servicio para beneficio propio o de otros.

Diagrama de casos de uso: Representación gráfica correspondiente al UML, que permite modelar procesos donde se relacionan actores con actividades, especialmente utilizado para analizar los requisitos de software.

Diagrama de secuencia: Representación gráfica correspondiente al UML, que permite mostrar la secuencia de mensajes, que se mueven entre objetos del sistema como las vistas de usuario, la programación y los almacenamientos de datos, ampliando los escenarios de los casos de uso.

Elicitación de requerimiento: Término informático, relacionado con el transpaso de información del lado del cliente o usuario al experto informático y viceversa en la etapa inicial del requerimiento de software, buscando clarificar la necesidad.

Hardware: Todo dispositivo o componente físico, relacionado con un equipo o máquina.

Herramientas: Objeto elaborado a fin de facilitar la realización de una tarea mecánica y que permite agilizar los procesos para lo cual fue construida.

Identificación del requerimiento: Parte de la primera fase del requerimiento, relacionada con el estudio del problema para detectar la necesidad informática real, dentro de un contexto determinado.

Ingeniería de sistemas: Disciplina que se encarga de estudiar y comprender los sistemas complejos, integrando varias especialidades como por ejemplo las matemáticas y físicas para desarrollar sistemas que utilicen económicamente los materiales y fuerzas de la naturaleza para el beneficio de la humanidad.

Ingeniería de Software: Área de la ingeniería de sistemas informáticos que estudia los principios, metodologías y técnicas para el desarrollo y mantenimiento de sistemas software.

Las mejores prácticas: Concepto relacionado con una serie de metodologías, sistemas, herramientas, y técnicas aplicadas y probadas con resultados sobresalientes en empresas reconocidas y que pueden ser tomados por otras para buscar la calidad.

Lenguaje de programación: conjunto de símbolos y regla que definen una estructura, así como el significado de sus elementos y expresiones, entendibles tanto para el programador como para la máquina, utilizado por los expertos informáticos para la construcción de software.

Metodología ágil: método de ingeniería de software que basado en desarrollo iterativo e incremental, evoluciona con el tiempo acorde a la necesidad, cuya principal fortaleza obedece a la auto-organización, disciplina y trabajo colaborativo de los equipos de trabajo, con resultados funcionales de software a corto plazo.

Metodología tradicional o prescriptiva: Obedece especialmente a un conjunto de tareas y actividades, centradas en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada en el desarrollo del software.

Modelamiento: estrategia utilizada para clarificar la complejidad de las necesidades informáticas, permitiendo visualizar el sistema a construir, facilitando la comunicación con el cliente y la verificación del modelo para la detección de errores en los procesos de construcción del software.

Programación estructurada: estilo de programación, con orígenes en los años 60, basado en tres estructuras fundamentales como son: la secuencia, la selección (if y switch), así como las iteraciones (bucles for y while), la cual fue desplazada por la programación orientada a objetos.

Programación orientada a objetos: Estilo de programación, con orígenes en los años 90, basado en varias técnicas, incluyendo herencia, cohesión, abstracción, polimorfismo, acoplamiento y encapsulamiento, lo cual permite la reutilización y optimización del código y de componentes ya fabricados.

Requerimiento: Corresponde a las necesidades y deseos pedidos por el cliente y las personas involucradas en el software. Traducido del término inglés (request=requerimiento)

Requisito: Son todas las funcionalidades, características y restricciones que debería tener el software. Traducido del término inglés (requirement=requisito).

Requisito de usuario: Se relaciona con las necesidades que los usuarios expresan verbalmente.

Requisito funcional: Corresponde a la funcionalidad que el sistema debe proporcionar al usuario final.

Requisito no funcional: Son todas aquellas restricciones que afectaran al sistema, buscando que cumpla con características de calidad, como por ejemplo seguridad, agilidad, usabilidad.

Rol: Consiste en el papel que desempeña una persona dentro de un proceso.

Sistema: Conjunto de elementos que ordenadamente y relacionados entre sí cumplen con un objetivo determinado.

Sistemas distribuidos: Conjunto de equipos de cómputo conectados a través de una red de comunicaciones, donde el usuario percibe como si se tratase de un solo sistema.

Situación problemática: Realidad que posee interrogantes sin resolver, tanto de conceptualización representación y aplicación significativa para quien la padece y que requiere ser planteada y resuelta.

Software: Programa compuesto por componentes lógicos necesarios que hacen posible la realización de tareas específicas y que requiere de un equipo físico para su funcionamiento (computador, celular, máquina).

Software libre: Software que puede ser utilizado de forma gratuita y normalmente descargado.

Software licenciado: Persona natural o jurídica que posee los derechos de autor y es libre de venderlo y modificarlo cuando lo desee.

Tecnología de punta: Tecnología avanzada de última generación.

Teoría general de sistemas: Teoría expuesta por Ludwig von Bertalanffy en la década de 1940, que busca relacionar las ciencias naturales y sociales, demostrando que todo lo que nos rodea son sistemas perfectamente integrados.

UML: (Lenguaje Unificado de Modelado) utilizado en la industria del software para visualizar, especificar, construir y documentar un sistema.

Usuario: Se refiere a un conjunto de permisos y de recursos a los cuales tiene acceso una persona, una máquina, un componente u otro sistema/persona que opera directamente el sistema.

Usuario Informático: Experto informática que desempeña roles relacionados con el proceso de ingeniería de software (líder de proyecto, analista, testing, desarrollador, documentador).

Usuario no informático: Persona que opera el sistema en su etapa funcional y que lo utiliza para agilizar procesos.

8 BIBLIOGRAFÍA

8.1.1 FUENTES BIBLIOGRÁFICAS

- G. Kotonya and I.(2000) Sommerville, Requirements Engineering: Processes and Techniques, John Wiley & Sons
- IEEE Computer Society, (2014), SWEBOK (Guide to the Software Engineering Body of Knowledge), Version 3.0. ISBN-10: 0-7695-5166-1
- Jacobson, Ivar; Booch, Grady; Rumbaugh, James (2000) (en Español). El Proceso Unificado de Desarrollo de Software. Pearson Addisson-Wesley.
- Piattini, Mario G. (1996), Análisis y diseño detallado de aplicaciones informáticas de gestión. 1ª ed. RA-MA Editorial, Madrid, 1996
- Pressman, R. (2010). Ingeniería de Software un Enfoque práctico, Séptima Edición. ISBN 978-607-15-0314-5.
- Sommerville, Lan. (2011) Ingeniería de software, novena edición. Pearson, México. ISBN 0137035152 | 9780137035151
- Sommerville, Lan (2005), Ingeniería de software, séptima edición, Pearson Educación, Madrid (España) ISBN: 84-7829-074-5

8.1.2 FUENTES DIGITALES O ELECTRÓNICAS

- IEEE Computer Society, (2004), Software Engineering Body of Knowledge, Consultado el 09 de noviembre de 2015 de: http://www.swebok.org
- IEEE Computer Society, (2014), SWEBOK (Guide to the Software Engineering Body of Knowledge), Version 3.0. ISBN-10: 0-7695-5166-1, consultado el 09 de noviembre de 2015 de: http://www.computer.org/web/swebok/v3-guide
- ISO/IEC 25000, SQuaRE, (2014). Consultado el 09 de noviembre de 2015 de: System and Software Quality Requirements and Evaluation.
- www.uml.org
- Ing. USBMed, Vol. 2, No. 2, Jul-Dic 2011 25 la elicitación de requisitos en el contexto de un proyecto software Markus Manies, Uolevis Nikual Lappeenranta University of Technology, Finland tite-

toimisto@lut.fi (Tipo de artículo: REFLEXIÓN. Recibido el 25/07/2011. Aprobado el 01/11/2011). http://web.usbmed.edu.co/usbmed/fing/v2n2/v2n2a4.pdf

8.1.3 FUENTES BASES DE DATOS ESPECIALIZADAS

- http://scholar.google.es/
- http://dialnet.unirioja.es/
- http://www2.ebsco.com/es-es/Pages/index.aspx
- http://biblioteca.remington.edu.co/es/recursos-electronicos/bibliotecas-virtuales
- http://www.redalyc.org/
- http://biblioteca.remington.edu.co/es/recursos-electronicos/bases-de-datos-libres

8.1.4 VIDEOS

- Un día de vidrio: https://www.youtube.com/watch?v=Usj5MBGkhKY
- Ingeniería de Sistemas Remington: https://www.youtube.com/watch?v=TG5xeZcsiGw
- perfiles de la carrera Ingeniería de Sistemas: https://www.youtube.com/watch?v=RtFFcQqXs6c&feature=youtu.be&app=desktop
- La crisis del software: https://www.youtube.com/watch?v=miGf6iZ0GZY
- Metodología cascada: https://www.youtube.com/watch?v=8cvHnlz17Yc
- Metodología espiral: https://www.youtube.com/watch?v=Wsvr-aB0tPQ
- Metodología Prototipo: https://www.youtube.com/watch?v=gpd0dpCTmFw
- Metodología RUP: https://www.youtube.com/watch?v=tbnU0jZzKTE y https://www.youtube.com/watch?v=M5_C58TWNHU
- Metodología Scrum: https://www.youtube.com/watch?v=PlLHc60egiQ
- Metodología eXtreme Programming (XP): https://www.youtube.com/watch?v=hQaPJN0njpQ
- Metodología Kanban: https://www.youtube.com/watch?v=I-H-WXAX_oM
- Metodología ágiles: adaptando la Ingeniería del software a los negocios, https://www.youtube.com/watch?v=5DvUR1Wdu8s
- Elicitación de requisitos de software: https://www.youtube.com/watch?v=6nDsbZdS-ms
- Teoría general de sistemas: https://www.youtube.com/watch?v=ROdDFC4eUJ8

- UML (Unified Modeling Language o en español, Lenguaje de Modelamiento Unificado): https://www.youtube.com/watch?v=8Vs7jfHG8Tc
- Relaciones entre casos de uso de requerimientos: https://www.youtube.com/watch?v=Bh2Bmq93hCM
- Gestión del riesgo, según norma ISO 31000: https://www.youtube.com/watch?v=t57g5u2eFjk.
- Riesgos de software: http://dis.um.es/~barzana/Informatica/IAGP/IAGP_riesgos.html
- Planificación de un proyecto de software: https://www.youtube.com/watch?v=PTcLnClFku8.
- Diagrama de secuencia: https://www.youtube.com/watch?v=bMtnQI0CMCo

Tabla de imágenes

Lista de tablas

Pág.