SIGCRANE

JORGE ANDRÉS ROA SÁNCHEZ
000293810
JUAN CARLOS MÉNDEZ SALAZAR
000301382
MAURICIO RAMÍREZ GUTIÉRREZ
000293363

CORPORACION UNIVERSITARIA MINUTO DE DIOS
FACULTAD DE INGENIERIA
TECNOLOGIA EN INFORMATICA
SOACHA
2015

SIGCRANE

JORGE ANDRÉS ROA SÁNCHEZ
JUAN CARLOS MÉNDEZ SALAZAR
MAURICIO RAMÍREZ GUTIÉRREZ

SISTEMA DE GESTION WEB PARA AUTOSERVICIO SALAZAR S.A.S

Asesor del Proyecto
Ing. JULIO EDUARDO JEJEN CARRILLO

CORPORACION UNIVERSITARIA MINUTO DE DIOS
FACULTAD DE INGENIERIA
TECNOLOGIA EN INFORMATICA
SOACHA
2015

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Soacha (12 de junio del 2015)

DEDICATORIA

A mi madre Ofir Gutiérrez, mi padre Ferdinel Ramírez, mis hijos Ángelo y Andrés Felipe , por todo el tiempo que juntos hemos tenido que sacrificar.

Mauricio Ramírez

Gracias a mi madre Luz Marina Sánchez, a mi abuela Carmelina Ovalle y a mi pareja Ivonne Pérez por cada palabra de apoyo y fortaleza brindada en el transcurso de tal ardua prueba, la cual gracias a ellas estoy a punto de culminar.

Jorge Andrés Roa

Ante todo a Dios por permitirme realizar una meta más en la vida, a mi madre Emma Julia Salazar, a mi padre José Waldo Méndez, por el apoyo y la confianza, a mi Tía Benilda Dueñas y mi prima Jenny Salazar, por ser los más grandes motivadores e impulsadores para cumplir este proceso y a todos mis compañeros de universidad por haber formado un gran lazo de amistad.

Juan Carlos Méndez Salazar

AGRADECIMIENTOS

A Dios, por habernos dado la fortuna de poder culminar esta etapa de nuestras vidas, a nuestras familias, madres, padres, hermanos, hijos, docentes, compañeros y amigos quienes con su paciencia y sabiduría nos dieron la fortaleza, para poder sortear todas las dificultades tanto académicas como quizás económicas.
En fin, gracias a todos aquellos que de una u otra forma, contribuyeron con su apoyo moral.

.

TABLA DE CONTENIDO

INTRODUCCIÓN

Este trabajo pretende dar a conocer, las técnicas y procedimientos que se han desarrollado como parte de la solución tecnológica, creada para las empresas de transporte y comercialización de servicios de grúa y carga.

Se describen los diferentes modelos y técnicas, necesarios para la creación de Software de gestión web; hecho para las compañías de transporte de grúa. El proyecto tiene como nombre la sigla SICRAME.

Mediante la ejecución de los módulos, necesarios para la operación de las
Compañías de grúas, hemos producido una herramienta tecnológica. La cual permitirá, no solo ganar en tiempos; sino que, además ofrecerá a las empresas una manera eficiente de administrar la información de la operación.

ABSTRACT

This work aims to show the techniques and procedures that have been developed as part of the technological solution, created to transport companies and marketing of cranes and cargo services.

Different models and techniques needed for the creation of web management software are described; made for companies shipping crane. The project is the SICRAME acronym name.

By executing the modules required for operation of the
Towing companies, we have produced a technological tool. Which will allow not only win time; but also it will offer companies an efficient way to manage the operation.

JUSTIFICACIÓN

Existen algunas soluciones que se enfocan en proveer la administración sistematizada de servicios de las empresas, dando como resultado un sistema de gestión local, que permite a las compañías un control especial y eficiente de todos sus movimientos. Es necesario que en un mundo globalizado, la información esté disponible no solo a nivel local; si no que además debe ser accesible desde cualquier lugar, obviamente siguiendo protocolos de seguridad, a fin de ser competitivos e innovadores.

En España por ejemplo, se ha desarrollado un software llamado NOVATRANS, el cual brinda herramientas de administración, para compañías prestadoras de servicios de grúas y cuyo enfoque principalmente gira en torno a un sistema de gestión local y ha sido creado por la empresa SOLBYTE.

Actualmente en Colombia y particularmente en la ciudad de Bogotá, las compañías que prestan servicios de grúas, carga y transporte no cuentan con sistemas de gestión web, capaces de integrar la parte operativa de su empresa con servicios para sus empleados como: la consulta de pagos de nómina, certificados laborales, solicitudes de vacaciones, registros de servicios realizados a diario (solo personal operativo) y para sus proveedores la creación de una base de datos donde se incluiría la información de contacto del proveedor.

Debido a cada una de estas problemáticas que han surgido cada vez que intentamos ahondar dentro este mundo del transporte de carga, y observando la situación de este gremio vemos la necesidad de llegar a cada una de las empresa para mejorar su parte operativa y de servicio mediante un sistema de gestión web llamado SIGCRANE, el cual al ser implementado mejorara la información hacia el usuario, agilizara tramites operativos y serán más eficientes los procedimientos generados para cada una de las solicitudes requeridas por los empleados..

MISION DEL PORYECTO

Nuestra misión es ser conocidos y reconocidos a nivel local y nacional con nuestro software SIGCRANE, buscamos llegar a implementar nuestro producto a muchas compañías del sector transporte de carga y maquinaria, mediante la elaboración de esta herramienta informática, proporcionamos manejo eficiente y calidad de la información en tiempo real, creando oportunidades de crecimiento y desarrollo. Para las compañías llevándolas a la vanguardia de la tecnología

MISION DE LA EMPRESA

AUTOSERVICIO SALAZAS S.A.S se crea con la finalidad de incursionar en el mercado, con la prestación de servicios de carga y transporte de todo tipo de vehículo y maquinaria tanto liviana como pasada, cubriendo la necesidades de nuestros cliente a nivel local y nacional, nuestro objetivo es llegar a ser una empresa pionera que otorga soluciones a los requerimientos de nuestros cliente, entregando un buen servicio con personal calificado y maquinaria de última generación, enfocada siempre a un mejoramiento continuo, ofreciendo calidad y responsabilidad con el fin de dar una satisfacción total en nuestros productos .

VISION DEL PROYECTO

Ser uno de los principales software de gestión administrativa, ofreciendo soluciones tecnológicas de calidad y responsabilidad, que cumplan las necesidades de las compañías, enfrentándonos a los cambios, siendo reconocidos con la confianza de las persona que requieren nuestro servicios.

VISION DE LA EMPRESA

Ser reconocida como una de las empresas líderes a nivel nacional en la prestación de servicios de soluciones de transporte, teniendo la seguridad que nuestros servicios son de excelente calidad, con el de poder diferenciarnos de nuestra competencia.

OBJETIVOS

OBJETIVO GENERAL

Analizar, Desarrollar e implementar un sistema de gestión web para el manejo de la información de la empresa Grúas Salazar.

OBJETIVOS ESPECÍFICOS

1- Aplicar un módulo de autenticación de usuarios que contemple los roles de gerencia, cliente, operarios y administrativos.
2- Organizar la información de la empresa mediante la creación de una base de datos generada en MySql, la cual contara con las tablas de: proveedores, clientes, empleados y servicios.
3- Desarrollar un módulo de órdenes de trabajo, el cual permitirá el registro de los servicios realizados por los operarios de la compañía.
4- Crear un módulo de soluciones de transporte que controle los servicios que la empresa ofrece a sus clientes
5- Generar un módulo de registro de cliente
6- Módulo de trámites de empleados en donde los funcionarios encuentren: desprendibles de pago, certificados laborales y formulario de solicitud de vacaciones.

Programar un módulo en donde se generen los consolidados de la información ingresada por cada uno de los operarios sobre los servicios realizados.

ESTUDIO DE CAMPO

-VISITA DE CAMPO

Se realizo visita a las instalaciones de la empresa y de empresas afines, para determinar que tan común es el hecho de que estas compañías, no administren la información a través de las tecnologías y el uso de herramientas informáticas, todo esto en pro del beneficio de las mismas y en procura de su crecimiento organizacional.

-ENCUESTA

Se verifica mediante encuestas a empresas del sector, vía presencial y telefónica, que tan cotidiano es el uso de tecnologías de la información; estos estudios arrojan como resultado que a todas ellas les hace falta una herramienta tecnológica, que este creada a medida y que proporcione, los datos en el momento justo; sin embargo muchas llevan algo de información en pequeños programas o hojas de cálculo.
Otras simplemente lo llevan de forma artesanal en papel y lápiz. Generando demoras y reproceso en algunas actividades de su operación diaria.

-ENTREVISTA

Mediante una entrevista a la propietaria de Autoservicio Salazar, se define la puesta en marcha de un plan piloto para crear un Sistema de Gestión Web, capaz de permitir un manejo eficiente de la información y en tiempo real, así como a partir de los datos suministrados por la empresa y con base en sus necesidades se inicia el análisis de requerimientos y de ahí se desprende la creación del proyecto SIGCRAME.

MARCO TEÓRICO

En países como España se ha desarrollado importantes avances en este sector como es el caso de la empresa SOLBYTE que ha implementado un software de gestión de información local, llamado NOVATRANS.
Enfocados en la creación de un sistema de gestión web que le permita a las compañías o empresas que tienen como actividad económica fundamental, la prestación de servicios de cargue acompañamiento y transporte de maquinaria pesada, liviana y formal es necesario empezar a crear una cultura organizacional, que integre nuevas tecnologías de información a sus procesos productivos.
Además en este importante sector de la economía no se ha masificado el uso de software a nivel administrativo, financiero y de gestión del personal. En el mejor de los casos y queriendo ser eficientes cuentan con una página web, que solamente describe los servicios ofrecidos a los usuarios que en la mayoría de los casos carecen de impacto visual, diseño, distribución y hasta de información.
Al generar una perspectiva de manera global referente a esta problemática en la cual se encuentra inmerso este gremio, decidimos generar un enfoque mediante investigaciones basadas en soluciones tecnológicas ya implementadas, es así que
Se generan visitas a empresas como Autoservicios Salazar la cual presta servicios de transporte de carga pesada y vehículos en donde se evidencia la falta de uso de medios tecnológicos los cuales facilitarían y agilizarían los procesos administrativos de esta compañía, además se generan encuestas entre el personal de la empresa permitiéndonos conocer aún más las falencias encontradas dentro de la empresa a nivel tecnológico. Por lo tanto dichos resultados arrojados de esta investigación, evidencia la necesidad de crear un Sistema de Gestión Web el cual ayudara a administrar la información de forma eficaz y eficiente.

MARCO HISTORICO

A nivel de desarrollo de Software para las empresas de transporte de grúa, y particularmente en Colombia; no existe una empresa dedicada al desarrollo de soluciones informáticas para este sector. Es así como se hace necesario dar una mirada a estas compañías de la economía del transporte, investigando e indagando en la creación de herramientas que permitan llevar el control adecuado de las diferentes actividades que las empresas de este ramo pueden desarrollar a diario, buscando hacer más eficientes sus procesos productivos y que se conviertan en mayores ingresos económicos para sus propietarios

MARCO REFERENCIAL

Partiendo de las necesidades del sector, hemos desarrollado SIGCRAME, una herramienta de Software capaz de gestionar, la información de los servicios prestados por la empresa Autoservicios Salazar. Esto le permitirá una mejor administración de la información en tiempo real, mediante el uso de las nuevas tecnologías; en el futuro le dará la posibilidad de adaptarse a los cambios de crecimiento de la empresa y a los futuros requerimientos de la misma.

Solo en países como España, líder en el desarrollo de Software se han generado herramientas informáticas para este sector, con muy buenos resultados y con excelentes alternativas de uso. Tal es el caso del instrumento NOVATRANS, desarrollado por la empresa SOLBYTE que como ya hemos mencionado ofrece, múltiples soluciones para estas compañías; de ahí la importancia que para nuestro país se comience a elaborar soluciones, acordes a las necesidades de nuestras empresas.

MARCO LEGAL

Es necesario tener en cuenta algunas directrices para la ejecución, venta y distribución de Software los cuales son: Derechos de Autor y propiedad intelectual, ley de habeas data y publicidad.

De acuerdo a lo anterior,

La Propiedad Intelectual es una disciplina normativa que protege las creaciones intelectuales provenientes de un esfuerzo, trabajo o destreza humanos, dignos de reconocimiento jurídico. La Propiedad Intelectual comprende:

El derecho de autor y los derechos conexos.

¿Qué es el derecho de autor?

Es la protección que le otorga el Estado al creador de las obras literarias o artísticas desde el momento de su creación. Merece la pena señalar, en este punto, que existen varios tipos de infracción de los derechos de autor del software; y todos ellos deben evitarse. Los tipos de infracción más comunes son los siguientes:

· Utilización sin licencia alguna; por ejemplo, copiar un programa de software de un amigo o de Internet, etc., cuando la licencia del software no lo permita explícitamente.

· Uso abusivo; por ejemplo, comprar un software licenciado para un ordenador e instalarlo en dos.

· Fallo en la transmisión de la licencia o incapacidad para volver a licenciar; cuando se adquiere hardware de segunda mano, no necesariamente se van a transmitir todas las licencias de software, por lo que debemos tomar las medidas oportunas para asegurarnos de que el uso es legítimo.

· Obtención del software de manera fraudulenta; por ejemplo, conseguir una reducción del precio aparentando que su empresa es una institución educativa.

· descargan y usan copias piratas infringen derechos de autor.

· "Ofertas especiales" ilícitas de vendedores de hardware; cuando un vendedor de hardware vende un ordenador con el software instalado, pero el software no está licenciado (lo más habitual es que el cliente no lo sepa).
· Hacer una copia ilícita del software en un CD-ROM grabable u otro soporte similar, con la intención de dársela a otra persona.
· Falsificación, es decir, la realización de copias ilegales de software en CD-ROM grabables, o soportes similares, a escala comercial y su venta como si fuesen copias legales (utilizando carátulas engañosas, etc.). La falsificación es dominio exclusivo de los delincuentes profesionales. Si el software se encuentra a un precio considerablemente reducido, bien podrá tratarse de una falsificación. y por un tiempo determinado.

CICLOS DE VIDA DEL SISTEMA

Es el proceso que se sigue para construir, entregar y hacer evolucionar el software, desde la concepción de una idea hasta la entrega y el retiro del sistema.
Confiable, predecible y eficiente.
"Una aproximación lógica a la adquisición, el suministro, el desarrollo, la explotación y el mantenimiento del software" IEEE 1074

"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" ISO 12207-1

CICLO DE VIDA CLASICO

Este enfoque del desarrollo de software ha sido el más utilizado y en la actualidad, pese a la aparición de metodologías ágiles, sigue siendo la solución predominante, si bien, en cada organización o en cada de proyecto se puede llevar a cabo con ciertas variantes.

Además, en función del autor, las fases en que se divide el ciclo de vida clásico, pueden ser diferentes. Para este artículo voy a utilizar las que estableció Roger Pressman (no obstante para la descripción del alcance de las mismas voy a basarme principalmente en Métrica V.3, pero adaptándome a esta clasificación), si bien otras muchas alternativas podrían ser válidas:

– **Análisis:** Esta fase comprendería desde la posible obtención de unos objetivos o requisitos iniciales para determinar la viabilidad del sistema y escrutar las distintas alternativas de solución, pasando por la elaboración del catálogo de requisitos, hasta la realización de casos de uso, prototipado de pantallas e informes, como una primera especificación del plan de pruebas.

– **Diseño:** El análisis describe el sistema sin entrar en características propias de la implementación, es en esta fase donde se adapta ese análisis generalista a la solución concreta que se quiere llevar a cabo, definiéndose la arquitectura general del sistema de información, su división en subsistemas de diseño, el modelo de datos lógico, el modelo de clases (en el caso de un diseño orientado a objetos), la especificación detallada del plan de pruebas, etc…

– **Codificación:** En esta fase se realiza la construcción del sistema de información y las pruebas relacionadas con dicho proceso, como son las unitarias, integración y de sistema,

así como otras actividades propias de las etapas finales de un desarrollo como es la realización de la carga inicial de datos (si bien en muchos casos se deja esto para cuando el producto está en producción) y/o la construcción del procedimiento de migración.

– Pruebas: En esta etapa se realizaría la instalación del sistema en un entorno de pruebas lo más parecido posible al de producción (entorno de preproducción) donde se realizarían las pruebas de implantación (que verifican principalmente aspectos no funcionales) y las de aceptación, donde los usuarios validan que el sistema hace lo que realmente esperaban (sin que se deba olvidar que los límites los establecen los modelos realizados previamente y que han debido ser validados). Por último se realizaría la implantación del sistema en el entorno de producción.

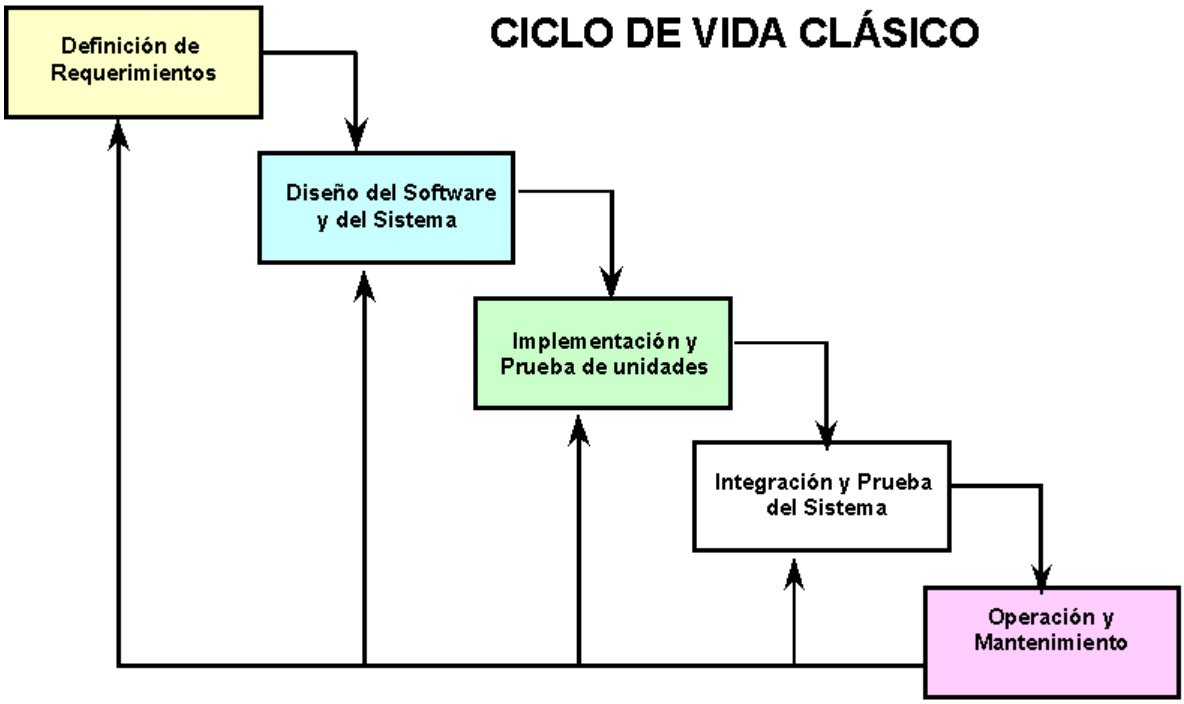
– Mantenimiento: Una vez que el sistema se encuentra en producción, se realizarán sobre el mismo diversas tareas de mantenimiento, que en función de su naturaleza se clasifican en correctivos, evolutivos, adaptativos y perfectivos. Estas tareas de mantenimiento serán consecuencia de incidencias y peticiones reportadas por los usuarios y los directores usuarios.

En el ciclo de vida clásico en función de las modificaciones y/o correcciones que se realizan en una etapa será necesaria la vuelta a fases previas para hacer coherente el proceso de desarrollo y los modelos.

El principal inconveniente que se le ha achacado siempre a este tipo de ciclo de vida es que los usuarios tardan demasiado en ver los resultados, lo que hace que el tiempo transcurrido desde que se define el sistema hasta que está disponible sea lo suficientemente amplio como para que hayan ocurrido muchas cosas: desde que no estén la mayoría de las personas que participaron en la especificación, como cambios en los procesos, cambios de criterio, etc…, lo que provoca a hacer replanteamiento de los requisitos en etapas más tardías del desarrollo con el coste que eso conlleva (un refinamiento de los requisitos es razonable siempre y cuando no se superen unos límites) y a que muy probablemente el sistema finalmente disponible esté alejado de lo que realmente quiere el conjunto de usuarios en estos momentos que es distinto a lo que querían hace meses y a lo que querrán dentro de otros tantos.

Otro inconveniente ligado con el anterior es que al tratarse el sistema como un todo, los modelos generados (catálogo de requisitos, casos de uso, etc…) serán lo suficientemente grandes como para no poder ser revisados y comprendidos en toda su magnitud, sobre todo por personal no informático. Esto crea una incertidumbre sobre los requisitos del sistema que más adelante traerá problemas.

No todo es malo en este modelo, ya que describe un procedimiento racional y ordenado de desarrollo de software, la clave para su éxito o su fracaso es como se gobierne el mismo y las circunstancias que rodeen al proyecto en el momento de su ejecución. Además, existen variantes que ofrecen una mayor flexibilidad al mismo y que permite reducir sus riesgos.



MODELO EN ESPIRAL

El modelo espiral en el desarrollo del software es un modelo meta del ciclo de vida del software donde el esfuerzo del desarrollo es iterativo, tan pronto culmina un esfuerzo del

desarrollo por ahí mismo comienza otro; además en cada ejecución del desarrollo se sigue cuatro pasos principales:

1. Determinar o fijar los objetivos. En este paso se definen los objetivos específicos para posteriormente identifica las limitaciones del proceso y del sistema de software, además se diseña una planificación detallada de gestión y se identifican los riesgos.

2. Análisis del riesgo. En este paso se efectúa un análisis detallado para cada uno de los riesgos identificados del proyecto, se definen los pasos a seguir para reducir los riesgos y luego del análisis de estos riesgos se planean estrategias alternativas.

3. Desarrollar, verificar y validar. En este tercer paso, después del análisis de riesgo, se eligen un paradigma para el desarrollo del sistema de software y se lo desarrolla.

4. Planificar. En este último paso es donde el proyecto se revisa y se toma la decisión si se debe continuar con un ciclo posterior al de la espiral. Si se decide continuar, se desarrollan los planes para la siguiente fase del proyecto.

Con cada iteración alrededor de la espiral, se crean sucesivas versiones del software, cada vez más completas y, al final, el sistema de software ya queda totalmente funcional.

La diferencia principal entre el modelo espiral y los modelos anteriores (ej.: cascada, evolutivo, incremental, etc.) es la evaluación del riesgo. El riesgo es todo lo que pueda salir mal en un proyecto de desarrollo de software. Por ejemplo, si queremos utilizar un lenguaje de programación para desarrollar un sistema operativo, un riesgo posible es que los compiladores utilizables no produzcan un código objeto eficiente. Los riesgos originan problemas en el proyecto, como el exceso de los costos. Es así que, la disminución de los riesgos es una actividad muy importante.

Un modelo espiral comienza con la determinación de los objetivos tanto funcionales como de rendimiento. Después se enumeran algunas formas posibles de alcanzar estos objetivos identificando las fuentes de riesgos posibles. Luego continuamos con el siguiente paso que es resolver estos riesgos y llevar a cabo las actividades de desarrollo, para finalizar con la planificación del siguiente ciclo de la espiral.

CARACTERÍSTICAS DEL MODELO EN ESPIRAL PARA EL DESARROLLO DE SOFTWARE

Es considerado como un modelo evolutivo ya que combina el modelo clásico con el diseño de prototipos.

Contiene una nueva etapa que es el análisis de riesgos, no incluida anteriormente.

Este modelo es el indicado para desarrollar software con diferentes versiones actualizadas como se hace con los programas modernos de PC´s.

La ingeniería puede desarrollarse a través del ciclo de vida clásico o el de construcción de prototipos.

Este es el enfoque más realista actualmente.

El modelo en espiral esta compartida en varias actividades estructurales, también llamadas regiones de tareas.

Existen seis regiones de tareas que son:

Comunicación con el cliente: esta es una tarea requerida para establecer comunicación entre el desarrollador y el cliente.

Planificación: esta tarea es necesaria aplicarla para pode definir los recursos, el tiempo y otras informaciones relacionadas con el proyecto, es decir, son todos los requerimientos.

Análisis de riesgos: esta es una de las tareas principales por lo que se aplica el modelo en espiral, es requerida para evaluar los riesgos técnicos y otras informaciones relacionadas con el proyecto.

Ingeniería: esta es una tarea necesaria ya que se requiere construir una o más representaciones de la aplicación.

Construcción y adaptación: esta tarea es requerida en el modelo espiral porque se necesita construir, probar, instalar y proporcionar soporte al usuario.

Evaluación el cliente: esta también es una tarea principal, necesaria para adquirir la reacción del cliente según la evaluación de las representaciones del software creadas durante la etapa de ingeniería y la de implementación creada durante la etapa de instalación.

VENTAJAS DEL MODELO ESPIRAL

No requiere una definición completa de los requerimientos del software a desarrollar para comenzar su funcionalidad.

En la terminación de un producto desde el final de la primera iteración es muy factible aprobar los requisitos.

Sufrir retrasos corre un riesgo menor, por que se comprueban los conflictos presentados tempranamente y existe la forma de poder corregirlos a tiempo.

DESVENTAJAS DEL MODELO ESPIRAL

Existe complicación cuando se evalúa los riesgos. Se requiere la participación continua por parte del cliente.

Se pierde tiempo al volver producir inicialmente una especificación completa de los requerimientos cuando se modifica o mejora el software.

ACOPLAMIENTOS DEL MODELO ESPIRAL

Los nuevos requerimientos del sistema se definen en todo los detalles posibles, esto implica generalmente el entrevistarse con un número determinado de usuarios que representarán a todos los usuarios tanto externos como internos y otros aspectos del sistema existente.

Un prototipo preliminar se crea para el desarrollo del nuevo software partiendo de un diseño hecho del sistema que se construyó del prototipo inicial. Esto es generalmente un sistema scaled-down, y representa una aproximación de las características del producto final.

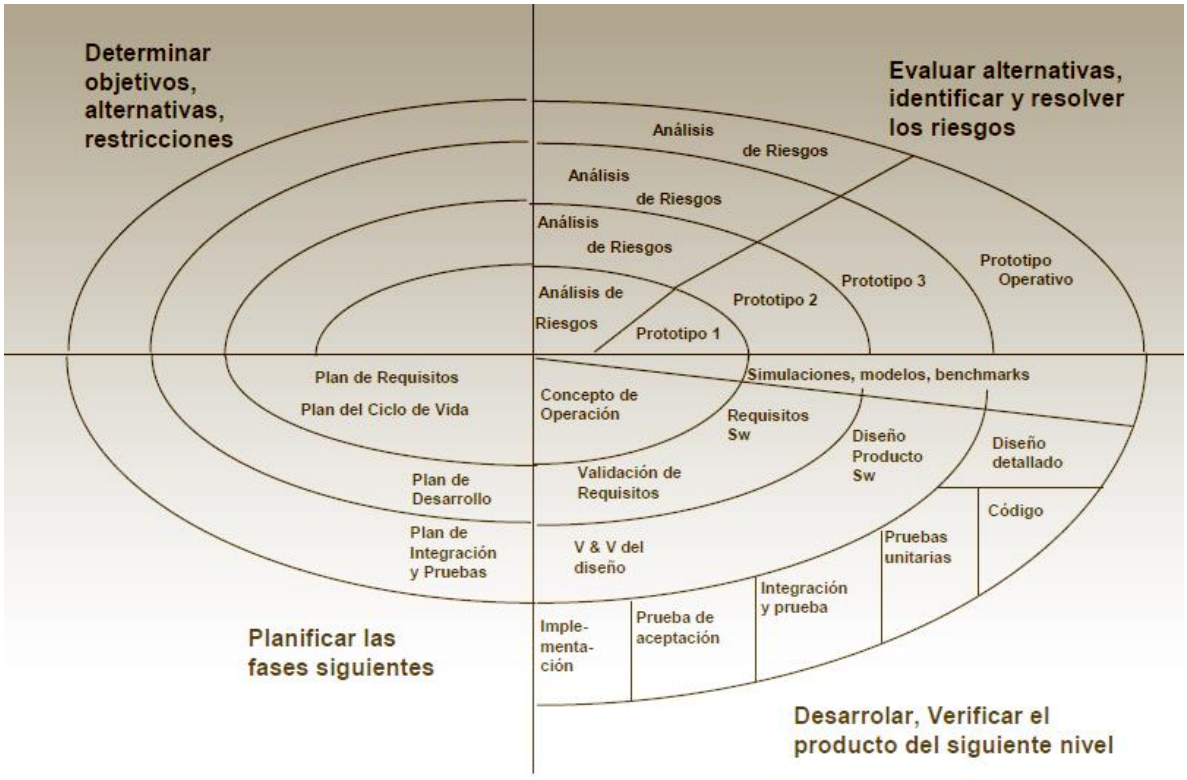
Un segundo diseño de software es desarrollado por un procedimiento cuádruple:

- Evaluación del primer prototipo en términos de sus fuerzas, debilidades, y riesgos;
- Definir los requisitos del segundo prototipo;
- Planeando y desarrollando el segundo prototipo;
- Construyendo y probando el segundo prototipo.

En la opción del cliente, el proyecto completado puede ser abortado si el riesgo se juzga demasiado grande. Los factores de riesgo pudieron implicar los excesos de coste del desarrollo, cálculo erróneo del fusionar los costes, o cualquier otro factor que podría, en el juicio del cliente, dar lugar a un producto final menos que satisfactorio.

El diseño existente se evalúa de manera semejante al igual que el diseño anterior, y, en caso de necesidad, otro prototipo se desarrolla de él según el procedimiento cuádruple expuesto anteriormente.

Se iteran los pasos precedentes hasta que el cliente está satisfecho sabiendo que el diseño mejorado representa el producto final deseado. Además, se construye el sistema final, basado en el diseño mejorado. El sistema final se evalúa y se prueba con todas las de ley. El mantenimiento general se realiza sobre una base continua para prevenir fallas en grande y para reducir al mínimo el tiempo perdido.



MODELO EN CASCADA

El Modelo en Cascada o también conocido como Ciclo de Vida del software da las pautas que permiten la organización en el desarrollo del software a través de la implementación de sus características etapas, esto quiere decir que cuando se esté llevando a cabo todas las tareas pertinentes dentro de esa etapa, no se podrá avanzar a la siguiente etapa hasta no concluir con todas las tareas.

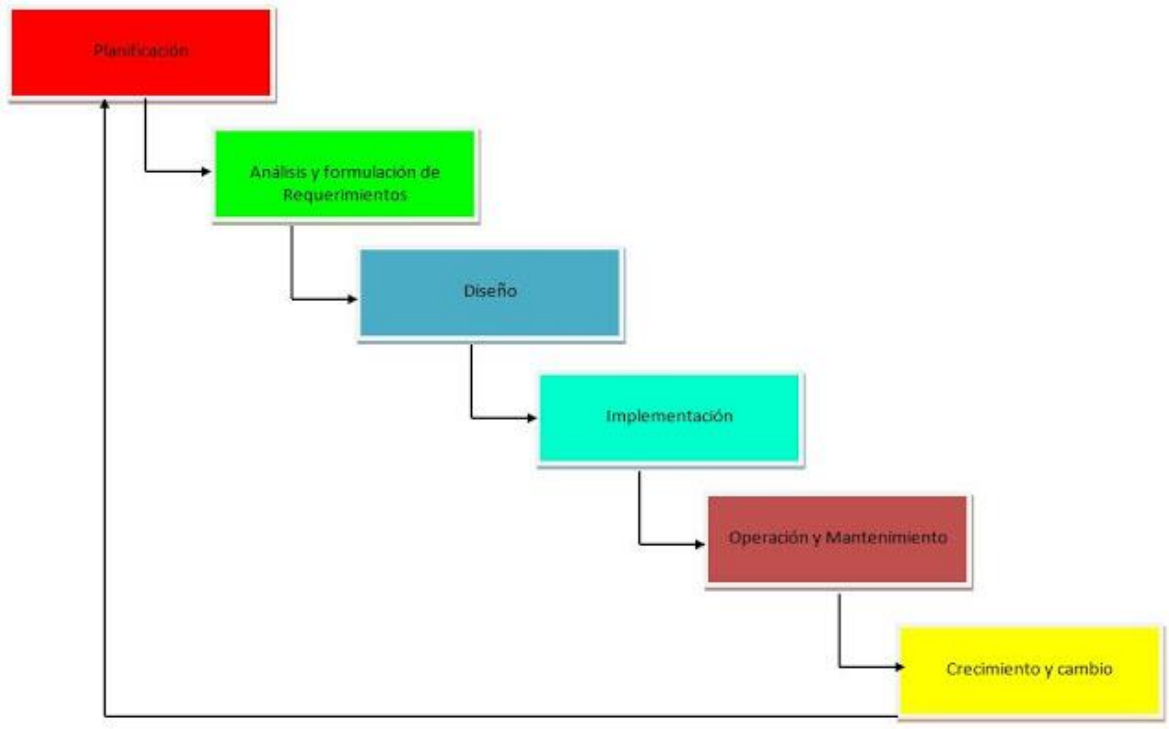
A continuación una breve descripción de cada proceso que constituye este modelo:

1. **Planificación:** Realiza un estudio de factibilidad del software así como contemplar los posibles costos que pueden surgir mediante su implementación.

2. **Análisis y Diseño de Requirimientos**: Involucra la identificación de las características que nos guían para determinar las funcionalidades del software de acuerdo al medio donde se pretende implementar, es muy importante notar que trata de responder a las preguntas ¿Quienes intervienen en el uso del Software?, Qué restricciones tendrá el software?

3. **Diseño:** Se identifica y describe las abstracciones del software y cumplir con los requerimientos plasmando todas esas características en un diseño que permite visualizar y contemplar adicionalmente situaciones no previstas.
4. **Implementación:** Realizar las pruebas pertinentes y verificar que se cumplen con las características identificadas.
5. **Operación y Mantenimiento:** Se instala dentro del ambiente, dependerá que pasará a partir de ahí, ya que esta etapa aun puede considerar nuevamente la existencia de características que no han sido contempladas y/o características innecesarias, implicando la modificación del software para la adaptación de estas anomalías.
6. **Crecimiento y cambio:** Se evalúa el software de modo que se determina si se puede emplear dentro de la nueva tecnología no afectando la integridad del mismo, de modo que si no es posible que exista una adaptación a lo nuevo, el proceso de diseño del software nuevamente se repite desde el principio.

Lamentablemente el uso de este modelo del desarrollo del software pone en jaque la integridad mientras se construye el sistema, ya que si se falla en una etapa, se ve obligado a reiniciar prácticamente el proceso de construcción, otra de las situaciones que pueden llevar al fracaso es precisamente una de sus características esenciales, avanzar hasta que se concluya la etapa anterior, viéndolo de este modo, puede atrasar de manera significativa el proceso de desarrollo de software, quizá tome mucho más tiempo del que realmente necesite, otra desventaja es el mantenimiento del software, ya que se involucra la repetición de sus pasos que se llevaron a cabo para la constitución del software volviendo este método muy tedioso, es recomendable utilizar este modelo siempre y cuando se conozca los requerimientos.



MODELO ORIENTADO A OBJETOS

Un objeto es un concepto, abstracción o cosa que tiene un cierto significado para una aplicación

• **La década de los 90:** la era de la programación orientada a objetos. ◦ Necesidad de este paradigma: los usuarios demandan programas y entornos de trabajo simples y fáciles de usar. ◦

Implicaciones: mayor número de líneas de código que es necesario organizar, gestionar y mantener.

• **Proporciona mejores herramientas para:** ◦ Obtener un modelo del mundo real cercano a la perspectiva del usuario. ◦ Interaccionar fácilmente con un entorno de computación, empleando metáforas familiares. ◦ Facilitar la modificación y la extensión de los componentes sin codificar de nuevo desde cero.

• **La programación tradicional** está orientada a los procedimientos.

• **En la programación orientada a objeto**s las entidades centrales son los datos (objetos).

◦ **Los objetos** se comunican entre sí mediante el uso de mensajes y el conjunto de objetos que responden a los mismos mensajes se implementan mediante clases.

◦**La clase describe e implementa** todos los métodos que capturan el comportamiento de sus instancias.
◦ **La implementación está totalmente oculta** (encapsulada) dentro de la clase, de modo que puede ser extendida y modificada sin afectar al usuario.
◦ **Una clase es como un módulo.** Sin embargo, también es posible extender y especializar una clase (mecanismo de herencia).

Los principios de la tecnología orientada a objetos se aplican a todos los aspectos del proceso de desarrollo del software:
• Metodología.
• Herramientas de diseño y análisis.
• Interfaces de usuario.
• Lenguajes de programación.
• Bases de datos.
• Sistemas operativos.

Terminología común y principios básicos

• Modularización — Módulos fáciles de manejar y que comprenden las estructuras de datos y las operaciones permisibles.
• Encapsulado — Distingue entre la interface a un objeto (qué es lo que hace), de la implementación (cómo lo hace).
• Tipos de datos abstractos — Agrupa todos los objetos que tienen la misma interface y los trata como si fueran del mismo tipo.
• Herencia — Reutilización, ya que permite definir nuevos tipos en funciones de otros tipos. El nuevo tipo hereda las estructuras de datos y los métodos del tipo precedente.

Un objeto es un concepto, abstracción o cosa que tiene un cierto significado para una aplicación

• Mensajes — Un objeto lleva a cabo sus acciones cuando recibe un mensaje concreto, codificado de una forma simple, estándar e independiente de cómo o dónde está implementado el objeto.
• Polimorfismo — Diferentes objetos responden al mismo mensaje. El sistema determina en tiempo de ejecución qué código invocar dependiendo del tipo de objeto (técnicas de Overloading y Dynamic binding).

• Se presentan como nombres propios o referencias específicas en la descripción o discusión de un problema:
• Algunos objetos tienen una entidad real (por ejemplo la empresa Seat).

• Otros son entidades conceptuales (por ejemplo, la fórmula para resolver una ecuación de segundo grado).
• Existen objetos que se introducen por razones de implementación y carecen de equivalencia en la realidad física (por ejemplo, un árbol binario).

Conceptos de objeto y clase

• Cada objeto tiene existencia propia y puede ser identificado. Se ha definido la identidad como: "aquella propiedad de un objeto que lo distingue del resto de objetos"
• En la etapa de análisis de una aplicación es posible dar por supuesto que los objetos tienen identidad.
• En el diseño, es necesario adoptar una metodología para implementar dicha identidad (direcciones de memoria, códigos numéricos o una combinación de los valores de ciertos atributos).Esta noción de identidad es una parte integral y muy importante de la tecnología orientada a objetos.

Objetos

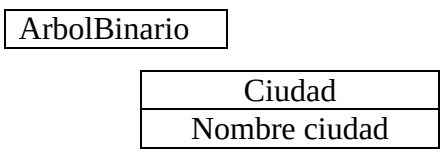
• Los objetos y sus relaciones se describen mediante diagramas de instancias:

• Cuando se construye un modelo es necesario decidir qué objetos mostrar y cuales ignorar.
• Un objeto representa sólo los aspectos relevantes de un problema.
• No resulta útil modelar detalles irrelevantes o extraños, ya que el ámbito de un objeto depende sólo de qué necesita la aplicación.

Clases

• Un objeto es una instancia (u ocurrencia) de una clase.
• Una Clase es la descripción de un grupo de objetos con:
◦ Propiedades similares (atributos del objeto).
◦ Comportamiento (operaciones y diagramas de estado) y semántica común. ◦ Y que establecen el mismo tipo de relaciones con otros objetos.

Las clases proporcionan un mecanismo para compartir la estructura entre objetos similares
• Algunas clases tienen una contrapartida real (persona y empresa por ejemplo).
• Otras son entidades conceptuales (ecuación algebraica).
• Además, existen clases que son sólo artefactos de una implementación específica (por ejemplo, árbol binario).
• Las clases y sus relaciones se describen mediante diagramas de clases:



Población

Simulación
Descripción Fecha ejecución Convergencia
Calcular

Clases

• No se especifican ni los atributos ni las operaciones de la clase ArbolBinario.

• La clase Ciudad tiene los atributos nombre Ciudad y población.

•La clase Aeropuerto cuenta con los atributos código Aeropuerto, nombreAeropuerto y zonaHoraria.

Aeropuerto
codigoAeropuerto nombreAeropuerto zonaHoraria

• No se muestran las operaciones de Ciudad y Aeropuerto. Esto no significa que ambas clases carezcan de operaciones, sino sólo que se ha decidido no mostrarlas.

• La clase Simulación tiene los atributos descripción, fecha Ejecución y convergencia así como la operación calcular. 1301

Diagramas de clases y objetos

• Tanto los diagramas de clases como los de instancias constituyen una expresión del modelo orientado a objetos.

• Del mismo modo que una clase describe un conjunto de objetos, un diagrama de clases describe un conjunto de diagramas de instancias.

• Un diagrama de clases permite representar de forma abstracta el conjunto de diagramas de instancias, documentando la estructura de los datos.

Valores y atributos de objeto

• Un valor es un trozo de información (dato).

• Un atributo de objeto es una propiedad de una clase a la que se le asigna un nombre y que contiene un valor para cada objeto de la clase.

• Los modelos orientados a objetos se construyen sobre estructuras: clases y relaciones. • Los atributos tienen una importancia menor y se utilizan para elaborar las clases y las relaciones.

• Es importante no confundir valores y objetos: Los objetos tienen identidad mientras que los valores no.

Operaciones y métodos

• Una operación es una función o procedimiento que se puede aplicar por un objeto o sobre un objeto.

• Cada operación actúa sobre un objeto que es su argumento implícito.

• La clase Simulación contiene la operación calcular cuyo argumento implícito es un objeto de la clase Simulación.
• Al nombre de la operación se pueden añadir detalles opcionales tales como la lista de argumentos o el tipo del resultado.
• Algunas operaciones son polimórficas, esto es, aplicables a muchas clases. La implementación de una operación para una clase concreta se denomina método.

Conceptos de enlace y asociación

• Un enlace (link) es una conexión física o conceptual entre objetos. Muchos enlaces interconectan dos objetos, pero es posible la existencia de enlaces entre tres o más objetos.
• Una asociación es la descripción de un grupo de enlaces con una estructura y semántica común. De acuerdo con esto:
• Un enlace es una instancia de una asociación. Los enlaces de una asociación relacionan objetos relacionan objetos de las mismas clases y tienen propiedades similares (atributo s del enlace).
• Una asociación describe un conjunto de enlaces potenciales del mismo modo que una clase describe un conjunto potencial de objetos. 13019 – D
• Las asociaciones constituyen un aspecto importante y distintivo del modelado OMT (y UML). ◦ Muchas metodologías de desarrollo y lenguajes orientados a objetos han pasado por alto las asociaciones.
◦ Una asociación no es lo mismo que un puntero.

Multiplicidad

La multiplicidad especifica el número de instancias de una clase que se pueden relacionar con una instancia de la clase relacionada
• Un vuelo puede ser reservado por múltiples reservas de vuelo.
• El código de pasaje de un vuelo reservado determina el precio del pasaje que cobrará la línea aérea.
• Una reserva de viaje consiste en múltiples reservas de vuelo que están ordenadas a medida que un pasajero viaja de una ciudad hasta la siguiente.
• La línea aérea utiliza un localizador de registro para identificar una reserva de viaje en su sistema de información.

Roles

• Un rol es uno de los extremos de una asociación al que se le puede asignar un nombre.
• Los nombres de rol son especialmente útiles en el recorrido de las asociaciones debido a que se pueden tratar como pseudo-atributos:
◦ una DescripciónVuelo.origen hace referencia al aeropuerto de origen del vuelo.
◦ una DescripciónVuelo.destino hace referencia al aeropuerto en el que finaliza el vuelo.

Agregación, composición y propiedad

• La agregación es una relación del tipo parte–todo entre dos clases.
• UML permite definir dos tipos de agregación:
◦ Agregación simple: también denominada agregación compartida. Permite modelar una relación parte–todo en la cual un objeto es propietario de otro objeto pero no en exclusividad. El objeto poseído puede ser a su vez poseído por otros objetos.
◦ Composición. También denominada en ocasiones agregación fuerte o agregación compuesta. Permite modelar una relación parte–todo en la que un objeto es propietario en exclusiva del otro objeto.

METODOLOGIAS

METODOLOGIA RUP

Es una metodología cuyo fin es entregar un producto de software. Se estructura todos los procesos y se mide la eficiencia de la organización.
Es un proceso de desarrollo de software el cual utiliza el lenguaje unificado de modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.
El RUP es un conjunto de metodologías adaptables al contexto y necesidades de cada

organización.
Describe como aplicar enfoques para el desarrollo del software, llevando a cabo unos pasos para su realización.
Se centra en la producción y mantenimiento de modelos del sistema.

Principales características

- Forma disciplinada de asignar tareas y responsabilidades (quién hace qué, cuándo y cómo)
- Pretende implementar las mejores prácticas en Ingeniería de Software
- Desarrollo iterativo
- Administración de requisitos
- Uso de arquitectura basada en componentes
- Control de cambios
- Modelado visual del software
- Verificación de la calidad del software

El RUP es un producto de Rational (IBM). Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

Fases del ciclo de vida del RUP:

1. Fase de Inicio: Esta fase tiene como propósito definir y acordar el alcance del proyecto con los patrocinadores, identificar los riesgos asociados al proyecto, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

2. Fase de elaboración: En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

3. Fase de Desarrollo: El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

4. Fase de Cierre: El propósito de esta fase es asegurar que el software esté disponible

para los usuarios finales, ajustar los errores y defectos encontrados en las pruebas de aceptación, capacitar a los usuarios y proveer el soporte técnico necesario. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

La metodología RUP tiene 6 principios clave:

1. Adaptación del proceso*:* El proceso debe adaptarse a las características de la organización para la que se está desarrollando el software.

2. Balancear prioridades*:* Debe encontrarse un balance que satisfaga a todos los inversores del proyecto.

3. Colaboración entre equipos*:* Debe haber una comunicación fluida para coordinar requerimientos, desarrollo, evaluaciones, planes, resultados, entre otros.
4. Demostrar valor iterativamente*:* Los proyectos se entregan, aunque sea de una forma interna, en etapas iteradas. En cada iteración se evaluará la calidad y estabilidad del producto y analizará la opinión y sugerencias de los inversores.

5. Elevar el nivel de abstracción: Motivar el uso de de conceptos reutilizables.

6. Enfocarse en la calidad*:* La calidad del producto debe verificarse en cada aspecto de la producción.

Disciplina de desarrollo de RUP

Determina las etapas a realizar durante el proyecto de creación del software.

- **Ingeniería o modelado del negocio:** Analizar y entender las necesidades del negocio para el cual se está desarrollando el software.
- **Requisitos:** Proveer una base para estimar los costos y tiempo de desarrollo del sistema.
- **Análisis y diseño:** Trasladar los requisitos analizados anteriormente a un sistema automatizado y desarrollar una arquitectura para el sistema.
- **Implementación:** Crear software que se ajuste a la arquitectura diseñada y que tenga el comportamiento deseado.
- **Pruebas**: Asegurarse de que el comportamiento requerido es correcto y que todo lo solicitado está presente.
- **Despliegue:** Producir distribuciones del producto y distribuirlo a los usuarios.

Disciplina de soporte RUP
Determina la documentación que es necesaria realizar durante el proyecto.

- **Configuración y administración del cambio**: Guardar todas las versiones del proyecto.
- **Administración del proyecto**: Administrar los horarios y recursos que se deben de emplear.
- **Ambiente**: Administrar el ambiente de desarrollo del software.
- **Distribución**: Hacer todo lo necesario para la salida del proyecto.

Elementos del RUP

- **Actividades**: Procesos que se han de realizar en cada etapa/iteración.
- **Trabajadores**: Personas involucradas en cada actividad del proyecto.
- **Artefactos**: Herramientas empleadas para el desarrollo del proyecto. Puede ser un documento, un modelo, un elemento del modelo.

Artefactos
RUP en cada una de sus fases (pertenecientes a la estructura estática) realiza una serie de artefactos que sirven para comprender mejor tanto el análisis como el diseño del sistema (entre otros). Estos artefactos (entre otros) son los siguientes:

Inicio:

- Documento Visión
- Especificación de Requerimientos

Elaboración:

- Diagramas de caso de uso

Construcción:

- Documento Arquitectura que trabaja con las siguientes vistas:

VISTA LOGICA:

- Diagrama de clases
- Modelo E-R (Si el sistema así lo requiere)

VISTA DE IMPLEMENTACION:

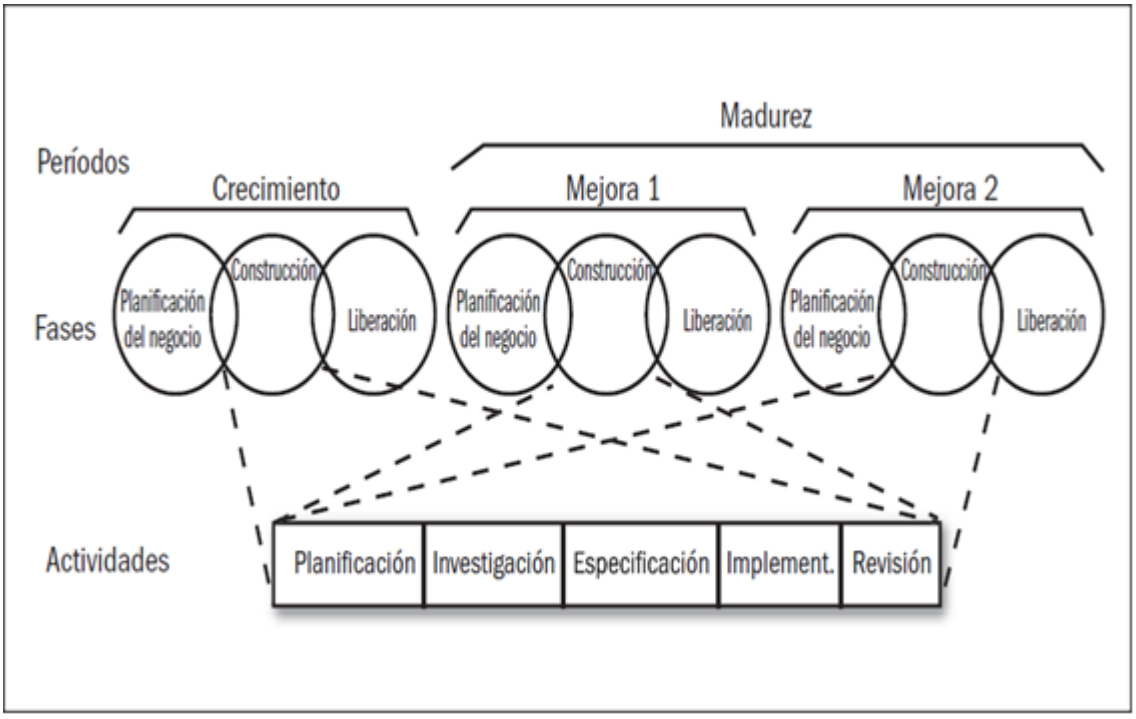
- Diagrama de Secuencia
- Diagrama de estados
- Diagrama de Colaboración

VISTA CONCEPTUAL

- Modelo de dominio

VISTA FISICA

- Mapa de comportamiento a nivel de hardware



METODOLOGIA RAD

El desarrollo rápido de aplicaciones o RAD (acrónimo en inglés de rapid application development) es un proceso de desarrollo de software, desarrollado inicialmente por James Martin en1980. El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades CASE (Computer Aided Software Engineering). Tradicionalmente, el desarrollo rápido de aplicaciones tiende a englobar también la usabilidad, utilidad y la rapidez de ejecución.

Hoy en día se suele utilizar para referirnos al desarrollo rápido de interfaces gráficas de usuario tales como Glade, o entornos de desarrollo integrado completos. Algunas de las plataformas más conocidas son Visual Studio, Lazarus, Gambas, Delphi,Foxpro , Anjuta, Game Maker, Velneo o Clarion. En el área de la autoría multimedia, software como Neosoft Neoboo y MediaChance Multimedia Builder proveen plataformas de desarrollo rápido de aplicaciones, dentro de ciertos límites.

FASES DE RAD

- **Modelado de gestión:** el flujo de información entre las funciones de gestión se modela de forma que responda a las siguientes preguntas: ¿Qué información conduce el proceso de gestión? ¿Qué información se genera? ¿Quién la genera? ¿A dónde va la información? ¿Quién la proceso?
- **Modelado de datos**: el flujo de información definido como parte de la fase de modelado de gestión se refina como un conjunto de objetos de datos necesarios para apoyar la empresa. Se definen las características (llamadas atributos) de cada uno de los objetos y las relaciones entre estos objetos.
- **Modelado de proceso**: los objetos de datos definidos en la fase de modelado de datos quedan transformados para lograr el flujo de información necesario para implementar una función de gestión. Las descripciones del proceso se crean para añadir, modificar, suprimir, o recuperar un objeto de datos. Es la comunicación entre los objetos.
- **Generación de aplicaciones:** El DRA asume la utilización de técnicas de cuarta generación. En lugar de crear software con lenguajes de programación de tercera generación, el proceso DRA trabaja para volver a utilizar componentes de programas ya existentes (cuando es posible) o a crear componentes reutilizables (cuando sea necesario). En todos los casos se utilizan herramientas automáticas para facilitar la construcción del software.
- **Pruebas de entrega**: Como el proceso DRA enfatiza la reutilización, ya se han comprobado muchos de los componentes de los programas. Esto reduce tiempo de pruebas. Sin embargo, se deben probar todos los componentes nuevos y se deben ejercitar todas las interfaces a fondo.

¿PORQUE USAR RAD?

Malas razones

- Prevenir presupuestos rebasados (RAD necesita un equipo disciplinado en manejo de costos).
- Prevenir incumplimiento de fechas (RAD necesita un equipo disciplinado en manejo de tiempo).

Buenas razones

- Convergir tempranamente en un diseño aceptable para el cliente y posible para los desarrolladores.
- Limitar la exposición del proyecto a las fuerzas de cambio.
- Ahorrar tiempo de desarrollo, posiblemente a expensas de dinero o de calidad del producto.

CARACTERISTICAS DE RAD

Equipos Híbridos

- Equipos compuestos por alrededor de seis personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema así como aquellas personas involucradas con los requisitos.
- Los desarrolladores de RAD deben ser "renacentistas": analistas, diseñadores y programadores en uno.

Herramientas Especializadas

- Desarrollo "visual"
- Creación de prototipos falsos (simulación pura)
- Creación de prototipos funcionales
- Múltiples lenguajes
- Calendario grupal
- Herramientas colaborativas y de trabajo en equipo
- Componentes reusables
- Interfaces estándares (API)

"**Timeboxing**"

- Las funciones secundarias son eliminadas como sea necesario para cumplir con el calendario.

Prototipos Iterativos y Evolucionarios.

- Reunion JAD (Joint Application Development):
  - Se reúnen los usuarios finales y los desarrolladores.
  - Lluvia de ideas para obtener un borrador inicial de los requisitos.
- Iterar hasta acabar:
  - Los desarrolladores construyen y depuran el prototipo basado en los requisitos actuales.
  - Los diseñadores revisan el prototipo.
  - Los clientes prueban el prototipo, depuran los requisitos.
  - Los clientes y desarrolladores se reúnen para revisar juntos el producto, refinar los requisitos y generar solicitudes de cambios.

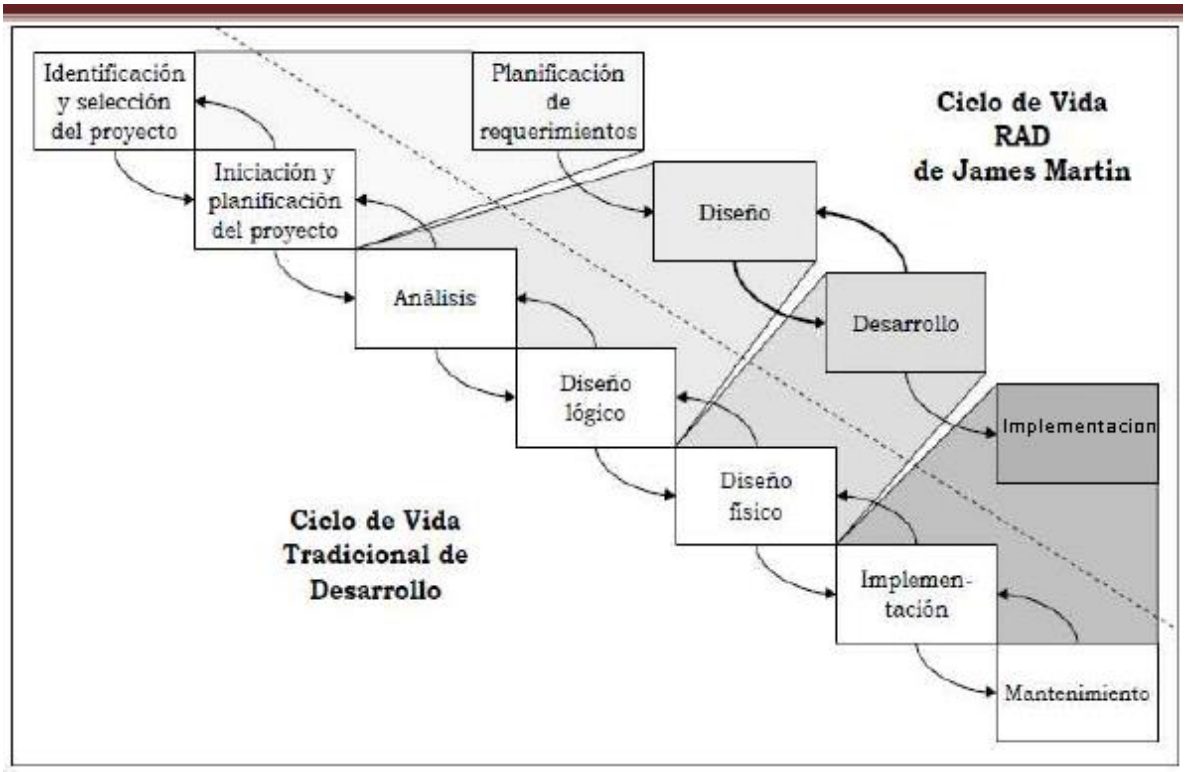
- Los cambios para los que no hay tiempo no se realizan. Los requisitos secundarios se eliminan si es necesario para cumplir el calendario.

VENTAJAS

1. Comprar puede ahorrar dinero en comparación con construir.
2. Los entregables pueden ser fácilmente trasladados a otra plataforma.
3. El desarrollo se realiza a un nivel de abstracción mayor.
4. Visibilidad temprana.
5. Mayor flexibilidad.
6. Menor codificación manual.
7. Mayor involucramiento de los usuarios.
8. Posiblemente menos fallas.
9. Posiblemente menor costo.
10. Ciclos de desarrollo más pequeños.
11. Interfaz gráfica estándar.

DESVENTAJAS

1. Comprar puede ser más caro que construir.
2. Costo de herramientas integradas y equipo necesario.
3. Progreso más difícil de medir.
4. Menos eficiente.
5. Menor precisión científica.
6. Riesgo de revertirse a las prácticas sin control de antaño.
7. Más fallas (por síndrome de "codificar a lo bestia").
8. Prototipos pueden no escalar, un problema mayúsculo.
9. Funciones reducidas (por "timeboxing").
10. Dependencia en componentes de terceros: funcionalidad de más o de menos, problemas legales.



METODOLOGIA XP

Entregas pequeñas: colocan un sistema sencillo en producción rápidamente que se actualiza de forma rápida y constante permitiendo que el verdadero valor de negocio del producto sea evaluado en un ambiente real. Estas entregas no pueden pasar las 2 o 3 semanas como máximo.

Entendimiento

1. Diseño simple: se basa en la filosofía de que el mayor valor de negocio es entregado por el programa más sencillo que cumpla los requerimientos. *Simple Design* se enfoca

en proporcionar un sistema que cubra las necesidades inmediatas del cliente, ni más ni menos. Este proceso permite eliminar redundancias y rejuvenecer los diseños obsoletos de forma sencilla.

2. Metáfora: desarrollada por los programadores al inicio del proyecto, define una historia de cómo funciona el sistema completo. XP estimula historias, que son breves descripciones de un trabajo de un sistema en lugar de los tradicionales diagramas y modelos UML (*Unified Modeling Language*). La metáfora expresa la visión evolutiva del proyecto que define el alcance y propósito del sistema. Las tarjetas CRC (Clase, Responsabilidad y Colaboración) también ayudarán al equipo a definir actividades durante el diseño del sistema. Cada tarjeta representa una clase en la programación orientada a objetos y define sus responsabilidades (lo que ha de hacer) y las colaboraciones con las otras clases (cómo se comunica con ellas).

3. Propiedad colectiva del código: un código con propiedad compartida. Nadie es el propietario de nada, todos son el propietario de todo. Este método difiere en mucho a los métodos tradicionales en los que un simple programador posee un conjunto de código. Los defensores de XP argumentan que mientras haya más gente trabajando en una pieza, menos errores aparecerán.

4. Estándar de codificación: define la propiedad del código compartido así como las reglas para escribir y documentar el código y la comunicación entre diferentes piezas de código desarrolladas por diferentes equipos. Los programadores las han de seguir de tal manera que el código en el sistema se vea como si hubiera estado escrito por una sola persona.

Bienestar del programador.

1. La semana de 40 horas: la programación extrema sostiene que los programadores cansados escriben código de menor cualidad. Minimizar las horas extras y mantener los programadores frescos, generará código de mayor calidad. Como dice *Beck*, está bien trabajar tiempos extra cuando es necesario, pero no se ha de hacer durante dos semanas seguidas.

PROCESO DE DESARROLLO

La programación extrema parte del caso habitual de una compañía que desarrolla software normalmente a medida, en la que hay diferentes roles: un equipo de gestión (o diseño), uno de desarrollo y los clientes finales. La relación entre el equipo de diseño, los que desarrollan el software y clientes es totalmente diferente al que se ha producido en las metodologías tradicionales, que se basaba en una fase de captura de los requisitos previa al desarrollo, y de una fase de validación posterior al mismo.

1. **Interacción con el cliente.**
En este tipo de programación el cliente pasa a ser parte implicada en el equipo de desarrollo. Su importancia es máxima en el momento de tratar con los usuarios y en efectuar las reuniones de planificación. Tiene un papel importante de interacción con el equipo de programadores, sobre todo después de cada cambio, y de cada posible problema localizado, mostrando las prioridades, expresando sus sensaciones... En este tipo de programación existirán pruebas de aceptación de la programación que ayudarán a que su labor sea lo más provechosa posible.
Al fin y al cabo, el cliente se encuentra mucho más cerca del proceso de desarrollo. Se elimina la fase inicial de recopilación de requerimientos, y se permite que éstos se vayan cogiendo a lo largo del proyecto, de una manera ordenada. De esta forma se posibilita que el cliente pueda ir cambiando de opinión sobre la marcha, pero a cambio han de estar siempre disponibles para solucionar las dudas del equipo de desarrollo.
En XP aparece un nuevo concepto llamado "*Historia de usuario*". Se trata de una lista de características que el cliente necesita que existan en el producto final. Estas constan de dos fases:
o En la primera fase, el cliente describe con sus propias palabras las características y, es el responsable del equipo, el encargado de informarlo de las dificultades técnicas de cada una de ellas y de su coste. A consecuencia de este diálogo, el cliente deja por escrito un conjunto de historias y las ordena en función de la prioridad que tiene pera él. De esta manera ya es posible definir unas fechas aproximadas para ellos.
o En la segunda fase, el cliente cogerá las primeras historias a implementar y las dividirá en trabajos a realizar. El cliente también participa, pero hay más peso por parte del equipo de desarrolladores, esto dará como resultado una planificación más exacta. Este método se repetirá para cada historia.

A diferencia de otras técnicas, como puede ser UML, en el caso de XP, se exige que sea el cliente el encargado de escribir los documentos con las especificaciones de lo que realmente quiere, como un documento de requisitos de usuario.
En esta fase, el equipo técnico será el 'encargado de catalogar las historias del cliente y asignarles una duración. La norma es que cada historia de usuario tiene que poder ser realizable en un espacio entre una y tres semanas de programación. Las que requieran menos tiempo serán agrupadas, y las que necesiten más serán modificadas o divididas.
Finalmente decir que las historias de los usuarios serán escritas en tarjetas, lo que facilitará que el cliente pueda especificar la importancia relativa entre las diferentes historias de usuario, así como el trabajo de los programadores que podrán catalogarlas correctamente. Este formato también es muy útil en el momento de las pruebas de aceptación.

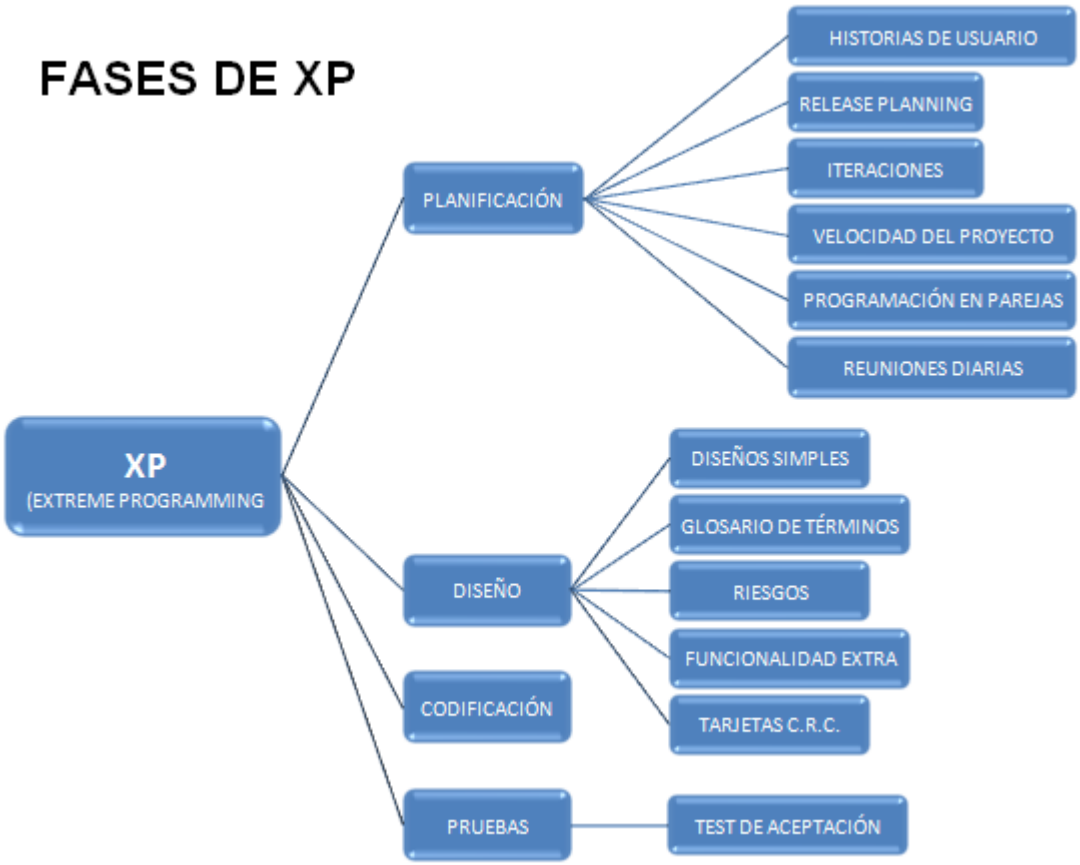
PLANIFICACIÓN DEL PROYECTO

En este punto se tendrá que elaborar la planificación por etapas, donde se aplicarán diferentes iteraciones. Para hacerlo será necesaria la existencia de reglas que se han de seguir por las partes implicadas en el proyecto para que todas las partes tengan voz y se sientan realmente partícipes de la decisión tomada.
Las entregas se tienen que hacer cuanto antes mejor, y con cada iteración, el cliente ha de recibir una nueva versión. Cuanto más tiempo se tarde en introducir una parte esencial, menos tiempo se tendrá para trabajar con ella después. Se aconseja muchas entregas y muy frecuentes. De esta manera un error en la parte inicial del sistema tiene más posibilidades de detectarse rápidamente.
Una de las máximas a aplicar es, los cambios, no han de suponer más horas de programación para el programador, ya que el que no se termina en un día, se deja para el día siguiente.
Se ha de tener asumido que en el proceso de planificación habrán errores, es más, serán comunes, y por esto esta metodología ya los tiene previstos, por lo tanto se establecerán mecanismos de revisión. Cada tres o cinco iteraciones es normal revisar las historias de los usuarios, y renegociar la planificación.
Cada iteración necesita también ser planificada, es lo que se llama *planificación iterativa*, en la que se anotarán las historias de usuarios que se consideren esenciales y las que no han pasado las pruebas de aceptación. Estas planificaciones también se harán en tarjetas, en las que se escribirán los trabajos que durarán entre uno y tres días.
Es por esto que el diseño se puede clasificar como continuo. Añade agilidad al proceso de desarrollo y evita que se mire demasiado hacia delante, desarrollando trabajos que aún no han estado programados.
Este tipo de planificación en iteraciones y el diseño iterativo, hace que aparezca una práctica que no existía en la programación tradicional. Se trata de las discusiones diarias informales, para fomentar la comunicación, y para hacer que los desarrolladores tengan tiempo de hablar de los problemas a los que se enfrentan y de ver cómo van con sus trabajos.

DISEÑO, DESARROLLO Y PRUEBAS

El desarrollo es la parte más importante en el proceso de la programación extrema. Todos los trabajos tienen como objetivo que se programen lo más rápidamente posible, sin interrupciones y en dirección correcta.
También es muy importante el diseño, y se establecen los mecanismos, para que éste sea revisado y mejorado de manera continuada a lo largo del proyecto, según se van añadiendo funcionalidades al mismo.
La clave del proceso de desarrollar XP es la comunicación. La mayoría de los problemas en los proyectos son por falta de comunicación en el equipo.
En XP, aparece un nuevo concepto llamado Metáfora. Su principal objetivo es mejorar la comunicación entre todos los integrantes del equipo, al crear una visión global y

común de lo que se quiere desarrollar. La metáfora tiene que ser expresada en términos conocidos por los integrantes del equipo, por ejemplo comparando el sistema que se desarrollará con alguna cosa de la vida real.
Antes de empezar a codificar se tienen que hacer pruebas unitarias, es decir: Cada vez que se quiere implementar una parte de código, en XP, se tiene que escribir una prueba sencilla, y después escribir el código para que la pase. Una vez pasada se amplía y se continúa. En XP hay una máxima que dice "Todo el código que puede fallar tiene que tener una prueba".
Con estas normas se obtiene un código simple y funcional de manera bastante rápida. Por esto es importante pasar las pruebas al 100%.
Respecto a la integración del soft, en XP se ha de hacer una integración continua, es decir, cada vez se tienen que ir integrando pequeños fragmentos de código, para evitar que al finalizar el proyecto se tenga que invertir grandes esfuerzos en la integración final. En todo buen proyecto de XP, tendría que existir una versión al día integrada, de manera que los cambios siempre se realicen en esta última versión.
Otra peculiaridad de XP es que cada programador puede trabajar en cualquier parte del programa.
De esta manera se evita que haya partes "propietarias de cada programador". Por esto es tan importante la integración diaria.
Para terminar, otra peculiaridad que tiene la XP. La de fomentar la programación en parejas, es decir, hacer que los programadores no trabajen en solitario, sino que siempre estarán con otra persona. Una pareja de programadores ha de compartir el teclado, el monitor y el ratón. El principio fundamental de este hecho es realizar de manera continua y sin parar el desarrollo de código. Las parejas tienen que ir cambiando de manera periódica, para hacer que el conocimiento se difunda en el grupo. Está demostrado que de esta manera el trabajo es más eficaz y también se consigue más y mejor código.


FASES DE XP
XP
(EXTREME PROGRAMMING
PLANIFICACIÓN
HISTORIAS DE USUARIO
RELEASE PLANNING
ITERACIONES
VELOCIDAD DEL PROYECTO
PROGRAMACIÓN EN PAREJAS
REUNIONES DIARIAS
DISEÑO
DISEÑOS SIMPLES
GLOSARIO DE TÉRMINOS
RIESGOS
FUNCIONALIDAD EXTRA
TARJETAS C.R.C.
CODIFICACIÓN
PRUEBAS
TEST DE ACEPTACIÓN

METODOLOGIA UML

Unified Modeling Languaje

UML [UML] es un lenguaje para especificar, construir, visualizar y documentar los artefactos de un sistema de software orientado a objetos (OO). Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software.

UML se quiere convertir en un lenguaje estándar con el que sea posible modelar todos los componentes del proceso de desarrollo de aplicaciones. Sin embargo, hay que tener en cuenta un aspecto importante del modelo: no pretende definir un modelo estándar de desarrollo, sino únicamente un lenguaje de modelado. Otros métodos de modelaje como OMT (Object Modeling Technique) o Booch sí definen procesos concretos. En UML los procesos de desarrollo son diferentes según los distintos dominios de trabajo; no puede ser el mismo el proceso para crear una aplicación en tiempo real, que el proceso de desarrollo de una aplicación orientada a gestión, por poner un ejemplo.

Las diferencias son muy marcadas y afectan a todas las fases del proceso. El método del UML recomienda utilizar los procesos que otras metodologías tienen definidos.

Los Inicios

A partir del año 1994, Grady Booch [Booch96](precursor de Booch '93) y Jim Rumbaugh (creador de OMT) se unen en una empresa común, Rational Software Corporation, y comienzan a unificar sus dos métodos. Un año más tarde, en octubre de 1995, aparece UML (Unified Modeling Language) 0.8, la que se considera como la primera versión del UML. A finales de ese mismo año, Ivan Jacobson, creador de OOSE (Object Oriented Software Engineer) se añade al grupo.

Como objetivos principales de la consecución de un nuevo método que aunara los mejores aspectos de sus predecesores, sus protagonistas se propusieron lo siguiente:

- El método debía ser capaz de modelar no sólo sistemas de software sino otro tipo de sistemas reales de la empresa, siempre utilizando los conceptos de la orientación a objetos (OO).
- Crear un lenguaje para modelado utilizable a la vez por máquinas y por personas.
- Establecer un acoplamiento explícito de los conceptos y los artefactos ejecutables.
- Manejar los problemas típicos de los sistemas complejos de misión crítica.

Lo que se intenta es lograr con esto que los lenguajes que se aplican siguiendo los métodos más utilizados sigan evolucionando en conjunto y no por separado. Y además, unificar las perspectivas entre diferentes tipos de sistemas (no sólo software, sino también en el ámbito de los negocios), al aclarar las fases de desarrollo, los requerimientos de análisis, el diseño, la implementación y los conceptos internos de la OO.

Modelado de objetos

En la especificación del UML podemos comprobar que una de las partes que lo componen es un meta modelo formal. Un meta modelo es un modelo que define el lenguaje para expresar otros modelos. Un modelo en OO es una abstracción cerrada semánticamente de un sistema y un sistema es una colección de unidades conectadas que son organizadas para realizar un propósito específico. Un sistema puede ser descripto por uno o más modelos, posiblemente desde distintos puntos de vista.

Una parte del UML define, entonces, una abstracción con significado de un lenguaje para expresar otros modelos (es decir, otras abstracciones de un sistema, o conjunto de unidades conectadas que se organizan para conseguir un propósito). Lo que en principio puede parecer complicado no lo es tanto si pensamos que uno de los objetivos del UML es llegar a convertirse en una manera de definir modelos, no sólo establecer una forma de modelo, de esta forma simplemente estamos diciendo que UML, además, define un lenguaje con el que podemos abstraer cualquier tipo de modelo.

El UML es una técnica de modelado de objetos y como tal supone una abstracción de un sistema para llegar a construirlo en términos concretos. El modelado no es más que la construcción de un modelo a partir de una especificación.

Un modelo es una abstracción de algo, que se elabora para comprender ese algo antes de construirlo. El modelo omite detalles que no resultan esenciales para la comprensión del original y por lo tanto facilita dicha comprensión.

Los modelos se utilizan en muchas actividades de la vida humana: antes de construir una casa el arquitecto utiliza un plano, los músicos representan la música en forma de notas musicales, los artistas pintan sobre el lienzo con carboncillos antes de empezar a utilizar los óleos, etc. Unos y otros abstraen una realidad compleja sobre unos bocetos, modelos al fin y al cabo. La OMT, por ejemplo, intenta abstraer la realidad utilizando tres clases de modelos OO: el modelo de objetos, que describe la estructura estática; el modelo dinámico, con el que describe las relaciones temporales entre objetos; y el modelo funcional que describe las relaciones funcionales entre valores. Mediante estas tres fases de construcción de modelos, se consigue una abstracción de la realidad que tiene en sí misma información sobre las principales características de ésta.

Los modelos además, al no ser una representación que incluya todos los detalles de los originales, permiten probar más fácilmente los sistemas que modelan y determinar los errores. Según se indica en la Metodología OMT (Rumbaugh), los modelos permiten una mejor comunicación con el cliente por distintas razones:

- Es posible enseñar al cliente una posible aproximación de lo que será el producto final.
- Proporcionan una primera aproximación al problema que permite visualizar cómo quedará el resultado.
- Reducen la complejidad del original en subconjuntos que son fácilmente tratables por separado.

Se consigue un modelo completo de la realidad cuando el modelo captura los aspectos importantes del problema y omite el resto. Los lenguajes de programación que estamos

acostumbrados a utilizar no son adecuados para realizar modelos completos de sistemas reales porque necesitan una especificación total con detalles que no son importantes para el algoritmo que están implementando. En OMT se modela un sistema desde tres puntos de vista diferentes donde cada uno representa una parte del sistema y una unión lo describe de forma completa. En esta técnica de modelado se utilizó una aproximación al proceso de implementación de software habitual donde se utilizan estructuras de datos (modelo de objetos), las operaciones que se realizan con ellos tienen una secuencia en el tiempo (modelo dinámico) y se realiza una transformación sobre sus valores (modelo funcional).

UML utiliza parte de este planteamiento obteniendo distintos puntos de vista de la realidad que modela mediante los distintos tipos de diagramas que posee. Con la creación del UML se persigue obtener un lenguaje que sea capaz de abstraer cualquier tipo de sistema, sea informático o no, mediante los diagramas, es decir, mediante representaciones gráficas que contienen toda la información relevante del sistema. Un diagrama es una representación gráfica de una colección de elementos del modelo, que habitualmente toma forma de grafo donde los arcos que conectan sus vértices son las relaciones entre los objetos y los vértices se corresponden con los elementos del modelo. Los distintos puntos de vista de un sistema real que se quieren representar para obtener el modelo se dibuja dé forma que se resaltan los detalles necesarios para entender el sistema.

3. Artefactos para el Desarrollo de Proyectos

Un artefacto es una información que es utilizada o producida mediante un proceso de desarrollo de software. Pueden ser artefactos un modelo, una descripción o un software. Los artefactos de UML se especifican en forma de diagramas, éstos, junto con la documentación sobre el sistema constituyen los artefactos principales que el modelador puede observar.

Se necesita más de un punto de vista para llegar a representar un sistema. UML utiliza los diagramas gráficos para obtener estos distintos puntos de vista de un sistema:

- Diagramas de Implementación.
- Diagramas de Comportamiento o Interacción.
- Diagramas de Casos de uso.
- Diagramas de Clases.

Ejemplo de algunos de los diagramas que utiliza UML.
Diagramas de Implementación

Se derivan de los diagramas de proceso y módulos de la metodología de Booch, aunque presentan algunas modificaciones. Los diagramas de implementación muestran los aspectos físicos del sistema. Incluyen la estructura del código fuente y la implementación, en tiempo de implementación. Existen dos tipos:
Diagramas de componentes
Diagrama de plataformas despliegue

4. Diagramas de componentes

Muestra la dependencia entre los distintos componentes de software, incluyendo componentes de código fuente, binario y ejecutable. Un componente es un fragmento de código software (un fuente, binario o ejecutable) que se utiliza para mostrar dependencias en tiempo de compilación.

Diagrama de plataformas o despliegue

Muestra la configuración de los componentes hardware, los procesos, los elementos de procesamiento en tiempo de ejecución y los objetos que existen en tiempo de ejecución. En este tipo de diagramas intervienen nodos, asociaciones de comunicación, componentes dentro de los nodos y objetos que se encuentran a su vez dentro de los componentes. Un nodo es un objeto físico en tiempo de ejecución, es decir una máquina que se compone habitualmente de, por lo menos, memoria y capacidad de procesamiento, a su vez puede estar formada por otros componentes.

Diagramas de Interacción o Comportamiento

Muestran las interacciones entre objetos ocurridas en un escenario (parte) del sistema. Hay varios tipos:

Diagrama de secuencia.
Diagrama de colaboración.
Diagrama de estado.
Diagrama de actividad.

Diagrama de secuencia

Muestran las interacciones entre un conjunto de objetos, ordenadas según el tiempo en que tienen lugar. En los diagramas de este tipo intervienen objetos, que tienen un significado parecido al de los objetos representados en los diagramas de colaboración, es decir son instancias concretas de una clase que participa en la interacción. El objeto puede existir sólo durante la ejecución de la interacción, se puede crear o puede ser destruido durante la ejecución de la interacción. Un diagrama de secuencia representa una forma de indicar el período durante el que un objeto está desarrollando una acción directamente o a través de un procedimiento.

En este tipo de diagramas también intervienen los mensajes, que son la forma en que se comunican los objetos: el objeto origen solicita (llama a) una operación del objeto destino. Existen distintos tipos de mensajes según cómo se producen en el tiempo: simples, síncronos, y asíncronos.

Los diagramas de secuencia permiten indicar cuál es el momento en el que se envía o se completa un mensaje mediante el tiempo de transición, que se especifica en el diagrama.

Diagrama de colaboración

Muestra la interacción entre varios objetos y los enlaces que existen entre ellos. Representa las interacciones entre objetos organizadas alrededor de los objetos y sus vinculaciones. A diferencia de un diagrama de secuencias, un diagrama de colaboraciones muestra las

relaciones entre los objetos, no la secuencia en el tiempo en que se producen los mensajes. Los diagramas de secuencias y los diagramas de colaboraciones expresan información similar, pero en una forma diferente.
Formando parte de los diagramas de colaboración nos encontramos con objetos, enlaces y mensajes. Un objeto es una instancia de una clase que participa como una interacción, existen objetos simples y complejos. Un objeto es activo si posee un thread o hilo de control y es capaz de iniciar la actividad de control, mientras que un objeto es pasivo si mantiene datos pero no inicia la actividad.

Un enlace es una instancia de una asociación que conecta dos objetos de un diagrama de colaboración. El enlace puede ser reflexivo si conecta a un elemento consigo mismo. La existencia de un enlace entre dos objetos indica que puede existir un intercambio de mensajes entre los objetos conectados.

Los diagramas de interacción indican el flujo de mensajes entre elementos del modelo, el flujo de mensajes representa el envío de un mensaje desde un objeto a otro si entre ellos existe un enlace. Los mensajes que se envían entre objetos pueden ser de distintos tipos, también según como se producen en el tiempo; existen mensajes simples, sincrónicos, balking, timeout y asíncronos. Durante la ejecución de un diagrama de colaboración se crean y destruyen objetos y enlaces.

Diagramas de actividad

Son similares a los diagramas de flujo de otras metodologías OO. En realidad se corresponden con un caso especial de los diagramas de estado donde los estados son estados de acción (estados con una acción interna y una o más transiciones que suceden al finalizar esta acción, o lo que es lo mismo, un paso en la ejecución de lo que será un procedimiento) y las transiciones vienen provocadas por la finalización de lasacciones que tienen lugar en los estados de origen. Siempre van unidos a una clase o a la implementación de un caso de uso o de un método (que tiene el mismo significado que en cualquier otra metodología OO). Los diagramas de actividad se utilizan para mostrar el flujo de operaciones que se desencadenan en un procedimiento interno del sistema.

Diagramas de estado

Representan la secuencia de estados por los que un objeto o una interacción entre objetos pasa durante su tiempo de vida en respuesta a estímulos (eventos) recibidos. Representa lo que podemos denominar en conjunto una máquina de estados. Un estado en UML es cuando un objeto o una interacción satisfacen una condición, desarrolla alguna acción o se encuentra esperando un evento.
Cuando un objeto o una interacción pasa de un estado a otro por la ocurrencia de un evento se dice que ha sufrido una transición, existen varios tipos de transiciones entre objetos: simples (normales y reflexivas) y complejas. Además una transición puede ser interna si el estado del que parte el objeto o interacción es el mismo que al que llega, no se provoca un cambio de estado y se representan dentro del estado, no de la transición. Como en todas las metodologías OO se envían mensajes, en este caso es la acción de la que puede enviar

mensajes a uno o varios objetos destino.

Diagramas de Casos de Uso

Los casos de uso es una secuencia de transacciones que son desarrolladas por un sistema en respuesta a un evento que inicia un actor sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/o otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la relación y la generalización son relaciones.

Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el mismo. En este tipo de diagrama intervienen algunos conceptos nuevos: un actor es una entidad externa al sistema que se modela y que puede interactuar con él; un ejemplo de actor podría ser un usuario o cualquier otro sistema. Las relaciones entre casos de uso y actores pueden ser las siguientes:

- Un actor se comunica con un caso de uso.
- Un caso de uso extiende otro caso de uso.
- Un caso de uso usa otro caso de uso

5. Diagramas de Clases

Los diagramas de clases representan un conjunto de elementos del modelo que son estáticos, como las clases y los tipos, sus contenidos y las relaciones que se establecen entre ellos.
Algunos de los elementos que se pueden clasificar como estáticos son los siguientes:

Paquete: Es el mecanismo de que dispone UML para organizar sus elementos en grupos, se representa un grupo de elementos del modelo. Un sistema es un único paquete que contiene el resto del sistema, por lo tanto, un paquete debe poder anidarse, permitiéndose que un paquete contenga otro paquete.

Clases: Una clase representa un conjunto de objetos que tienen una estructura, un comportamiento y unas relaciones con propiedades parecidas. Describe un conjunto de objetos que comparte los mismos atributos, operaciones, métodos, relaciones y significado. En UML una clase es una implementación de un tipo. Los componentes de una clase son:
Atributo. Se corresponde con las propiedades de una clase o un tipo. Se identifica mediante un nombre. Existen atributos simples y complejos.
Operación. También conocido como método, es un servicio proporcionado por la clase que puede ser solicitado por otras clases y que produce un comportamiento en ellas cuando se realiza.
Las clases pueden tener varios parámetros formales, son las clases denominadas plantillas. Sus atributos y operaciones vendrán definidos según sus parámetros formales. Las plantillas pueden tener especificados los valores reales para los parámetros formales, entonces reciben el nombre de clase parametrizada instanciada. Se puede usar en cualquier lugar en

el que se podría aparecer su plantilla.

Relacionando con las clases nos encontramos con el término utilidad, que se corresponde con una agrupación de variables y procedimientos globales en forma de declaración de clase, también puede definirse como un estereotipo (o nueva clase generada a partir de otra ya existente) de un tipo que agrupa variables globales y procedimientos en una declaración de clase. Los atributos y operaciones que se agrupan en una utilidad se convierten en variables y operaciones globales. Una utilidad no es fundamental para el modelado, pero puede ser conveniente durante la programación.

Metaclase: Es una clase cuyas instancias son clases. Sirven como depósito para mantener las variables de clase y proporcionan operaciones (método de clase) para inicializar estas variables. Se utilizan para construir metamodelos (modelos que se utilizan para definir otros modelos).

Tipos: Es un descriptor de objetos que tiene un estado abstracto y especificaciones de operaciones pero no su implementación. Un tipo establece una especificación de comportamiento para las clases.

Interfaz: Representa el uso de un tipo para describir el comportamiento visible externamente de cualquier elemento del modelo.

Relación entre clases: Las clases se relacionan entre sí de distintas formas, que marcan los tipos de relaciones existentes:
Asociación:
Es una relación que describe un conjunto de vínculos entre clases. Pueden ser binarias o n-arias, según se implican a dos clases o más. Las relaciones de asociación vienen identificadas por los roles, que son los nombres que indican el comportamiento que tienen los tipos o las clases, en el caso del rol de asociación (existen otros tipos de roles según la relación a la que identifiquen). Indican la información más importante de las asociaciones. Es posible indicar el número de instancias de una clase que participan en una relación mediante la llamada multiplicidad. Cuando la multiplicidad de un rol es mayor que 1, el conjunto de elementos que se relacionan puede estar ordenado. Las relaciones de asociación permiten especificar qué objetos van a estar asociados con otro objeto mediante un calificador. El calificador es un atributo o conjunto de atributos de una asociación que determina los valores que indican cuales son los valores que se asociarán.
Una asociación se dirige desde una clase a otra (o un objeto a otro), el concepto de navegabilidad se refiere al sentido en el que se recorre la asociación.
Existe una forma especial de asociación, la agregación, que especifica una relación entre las clases donde el llamado "agregado" indica él todo y el "componente" es una parte del mismo.
Composición:
Es un tipo de agregación donde la relación de posesión es tan fuerte como para marcar otro tipo de relación. Las clases en UML tienen un tiempo de vida determinado, en las relaciones de composición, el tiempo de vida de la clase que es parte del todo (o agregado) viene determinado por el tiempo de vida de la clase que representa el todo, por tanto es equivalente a un atributo, aunque no lo es porque es una clase y puede funcionar como tal

en otros casos.
Generalización:
Cuando se establece una relación de este tipo entre dos clases, una es una Superclase y la otra es una Subclase. La subclase comparte la estructura y el comportamiento de la superclase. Puede haber más de una clase que se comporte como subclase.
Dependencia:
Una relación de dependencia se establece entre clases (u objetos) cuando un cambio en el elemento independiente del modelo puede requerir un cambio en el elemento dependiente.
6. Relación de Refinamiento
Es una relación entre dos elementos donde uno de ellos especifica de forma completa al otro que ya ha sido especificado con cierto detalle.

Nuevas características del UML

Además de los conceptos extraídos de métodos anteriores, se han añadido otros nuevos que vienen a suplir carencias antiguas de la metodología de modelado. Estos nuevos conceptos son los siguientes:

- Definición de estereotipos: un estereotipo es una nueva clase de elemento de modelado que debe basarse en ciertas clases ya existentes en la meta modelo y constituye un mecanismo de extensión del modelo.
- Responsabilidades.
- Mecanismos de extensibilidad: estereotipos, valores etiquetados y restricciones.
- Tareas y procesos.
- Distribución y concurrencia (para modelar por ejemplo ActiveX/DCOM y CORBA).
- Patrones/Colaboraciones.
- Diagramas de actividad (para reingeniería de proceso de negocios)
- Clara separación de tipo, clase e instancia.
- Refinamiento (para manejar relaciones entre niveles de abstracción).
- Interfaces y componentes.

7. El Proceso de Desarrollo

UML no define un proceso concreto que determine las fases de desarrollo de un sistema, las empresas pueden utilizar UML como el lenguaje para definir sus propios procesos y lo único que tendrán en común con otras organizaciones que utilicen UML serán los tipos de diagramas.

UML es un método independiente del proceso. Los procesos de desarrollo deben ser definidos dentro del contexto donde se van a implementar los sistemas.

Herramientas CASE

Rational Rose es la herramienta CASE que comercializan los desarrolladores de UML y que soporta de forma completa la especificación del UML 1.1.
Esta herramienta propone la utilización de cuatro tipos de modelo para realizar un diseño del sistema, utilizando una vista estática y otra dinámica de los modelos del sistema, uno lógico y otro físico. Permite crear y refinar estas vistas creando de esta forma un modelo

completo que representa el dominio del problema y el sistema de software.

Desarrollo Iterativo

Rational Rose utiliza un proceso de desarrollo iterativo controlado (controlled iterative process development), donde el desarrollo se lleva a cabo en una secuencia de iteraciones. Cada iteración comienza con una primera aproximación del análisis, diseño e implementación para identificar los riesgos del diseño, los cuales se utilizan para conducir la iteración, primero se identifican los riesgos y después se prueba la aplicación para que éstos se hagan mínimos.
Cuando la implementación pasa todas las pruebas que se determinan en el proceso, ésta se revisa y se añaden los elementos modificados al modelo de análisis y diseño. Una vez que la actualización del modelo se ha modificado, se realiza la siguiente iteración.

Trabajo en Grupo

Rose permite que haya varias personas trabajando a la vez en el proceso iterativo controlado, para ello posibilita que cada desarrollador opere en un espacio de trabajo privado que contiene el modelo completo y tenga un control exclusivo sobre la propagación de los cambios en ese espacio de trabajo.

También es posible descomponer el modelo en unidades controladas e integrarlas con un sistema para realizar el control de proyectos que permite mantener la integridad de dichas unidades.

Generador de Código

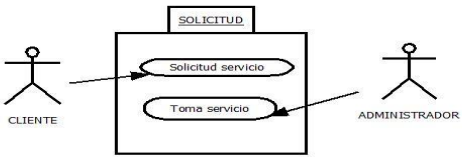
Se puede generar código en distintos lenguajes de programación a partir de un diseño en UML.

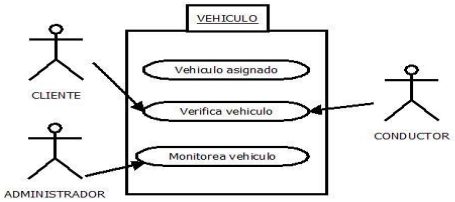
Ingeniería Inversa

Rational Rose proporciona mecanismos para realizar la denominada Ingeniería Inversa, es decir, a partir del código de un programa, se puede obtener información sobre su diseño.

MODELOS

CASOS DE USO





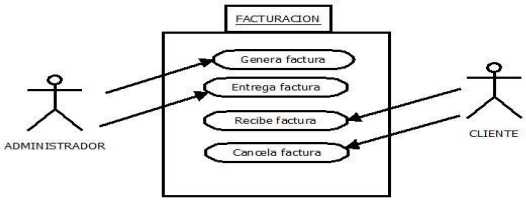


DIAGRAMA DE CLASES

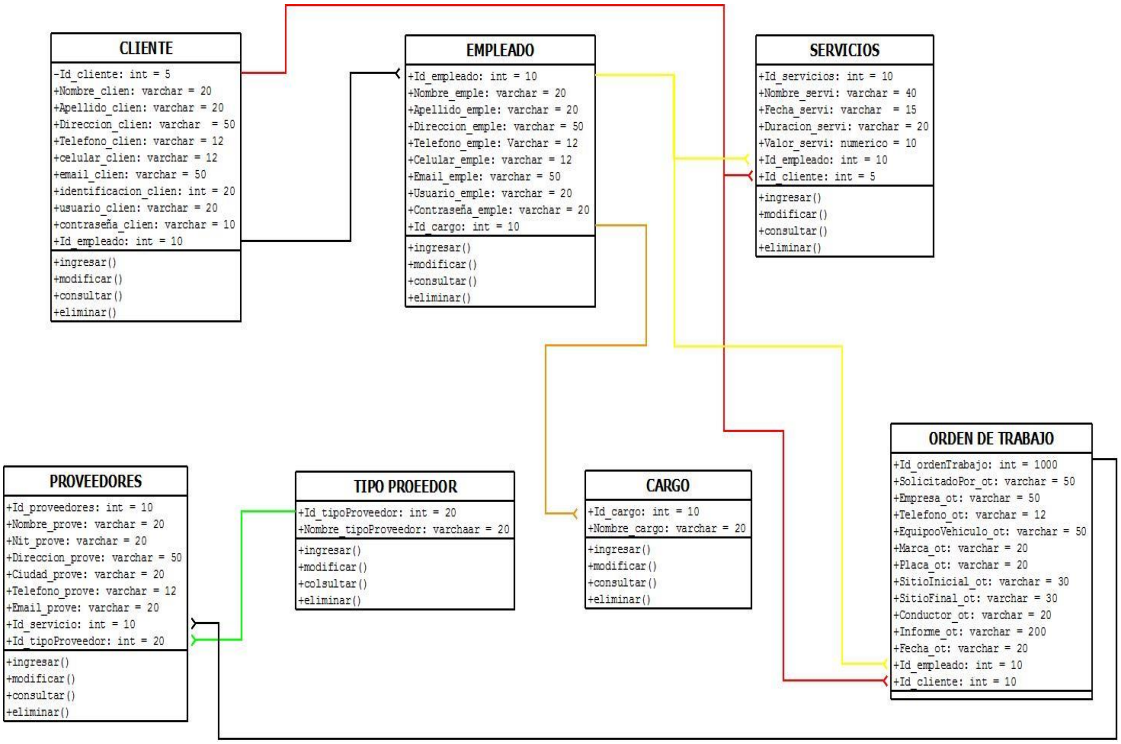


DIAGRAMA DE ESTADO

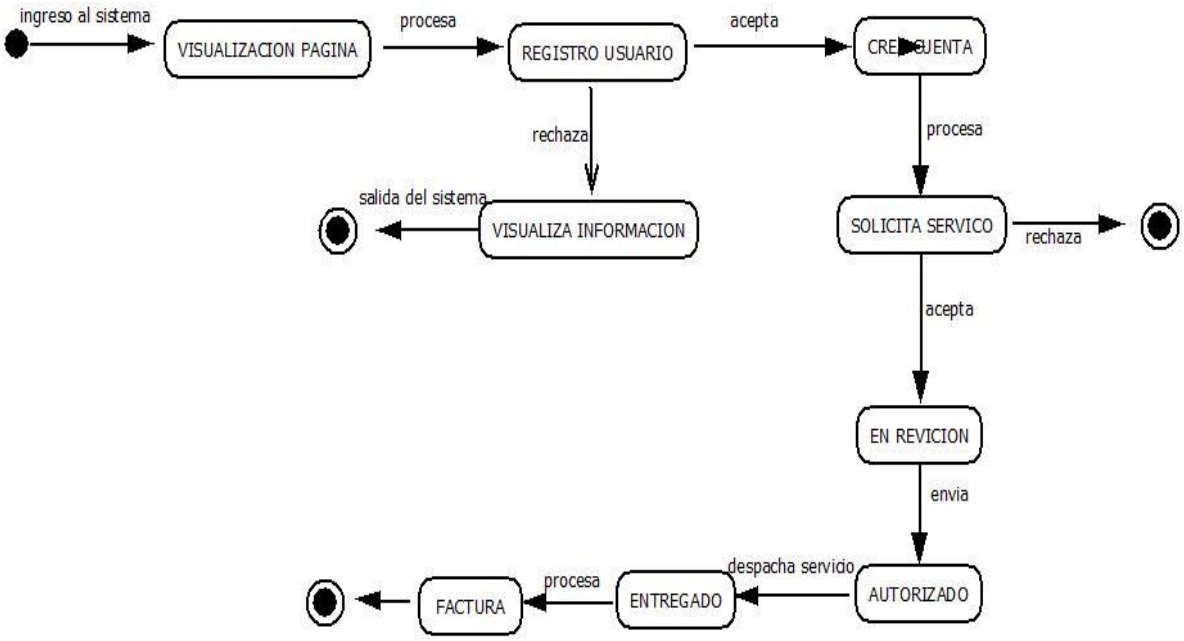
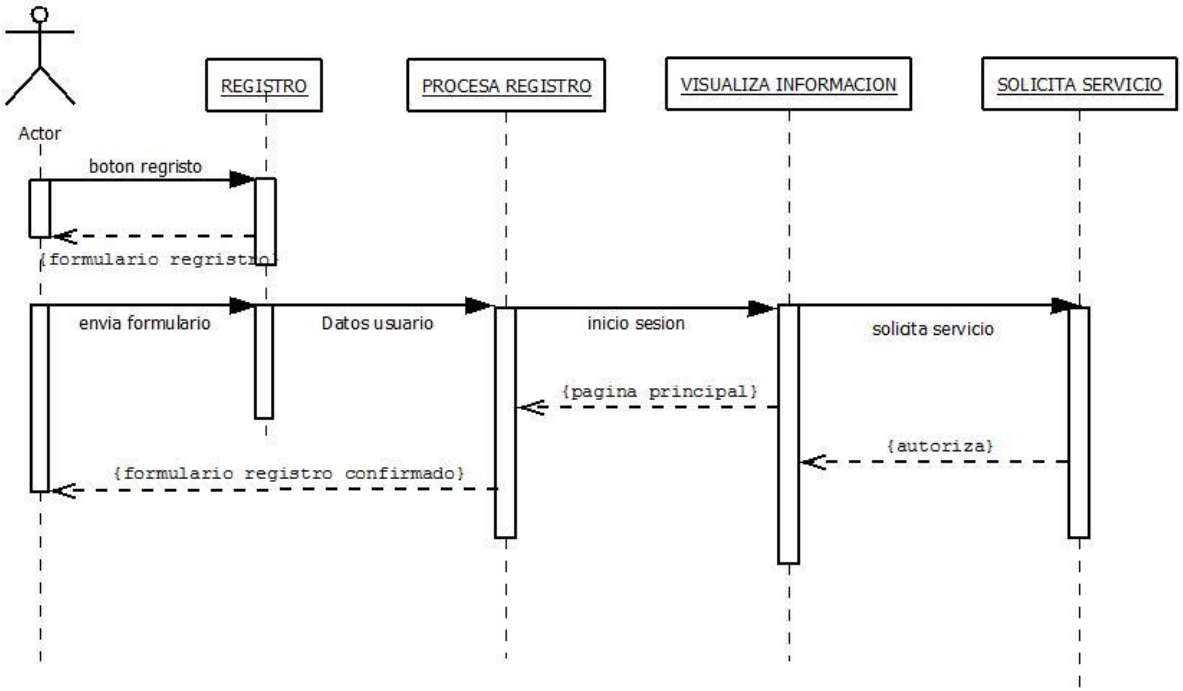


DIAGRAMA DE SECUENCIA



FLUJO GRAMA DE PROCESOS

ORGANIGRAMA DE EMPRESA

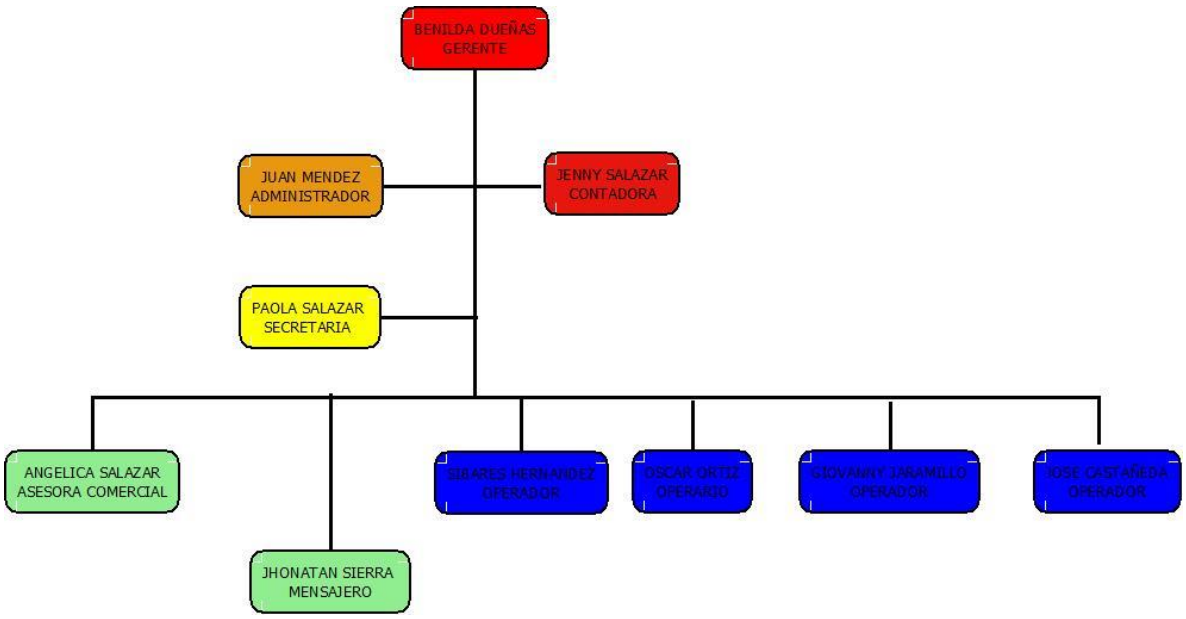


DIAGRAMA DE FLUJO DE PROCESOS

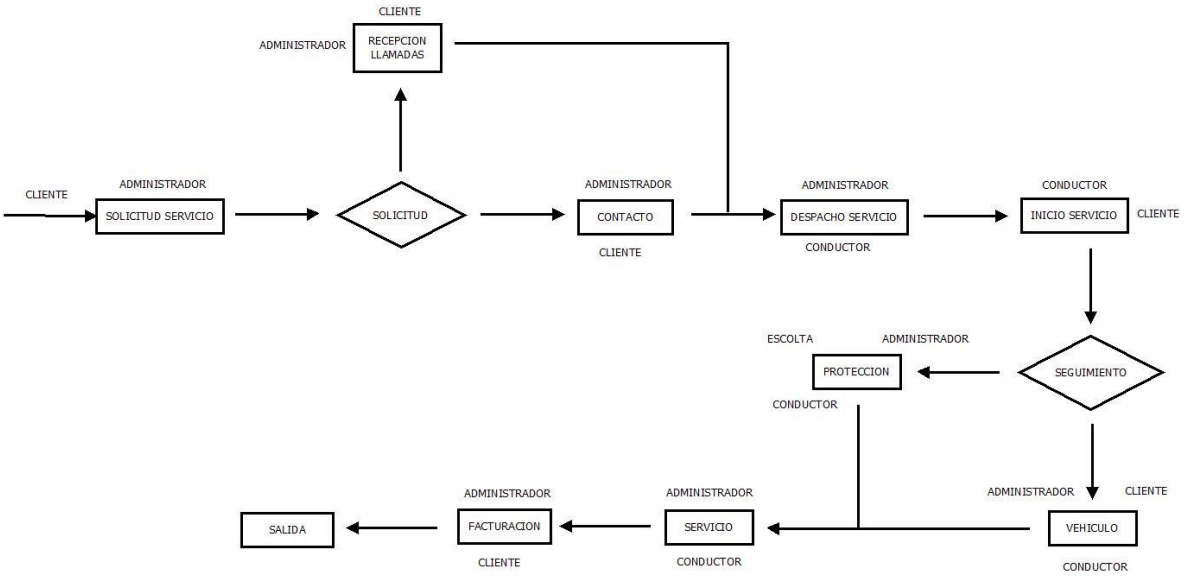
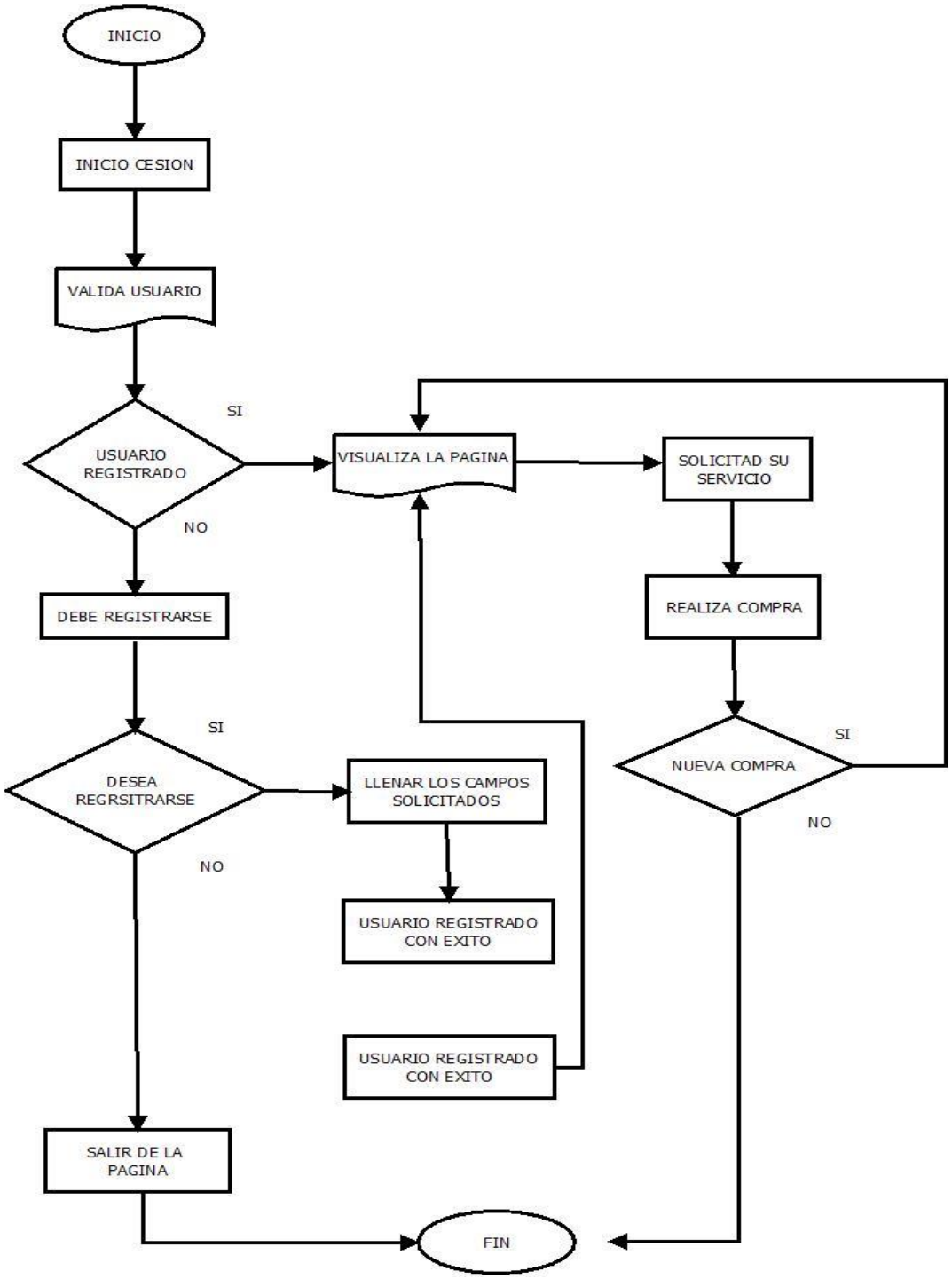
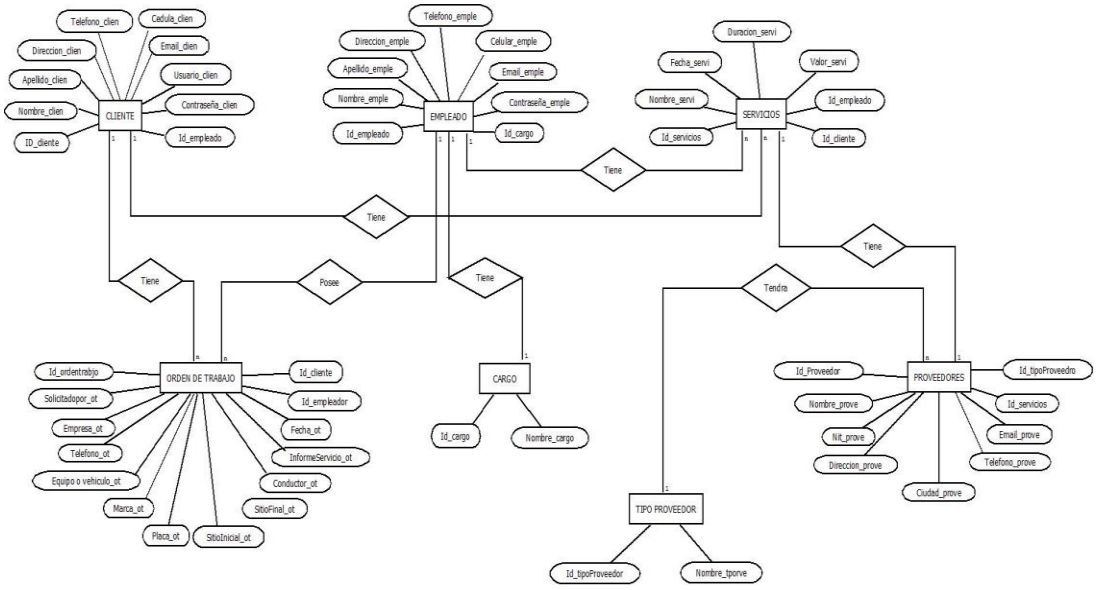


DIAGRAMA DE FLUJO DE DATOS

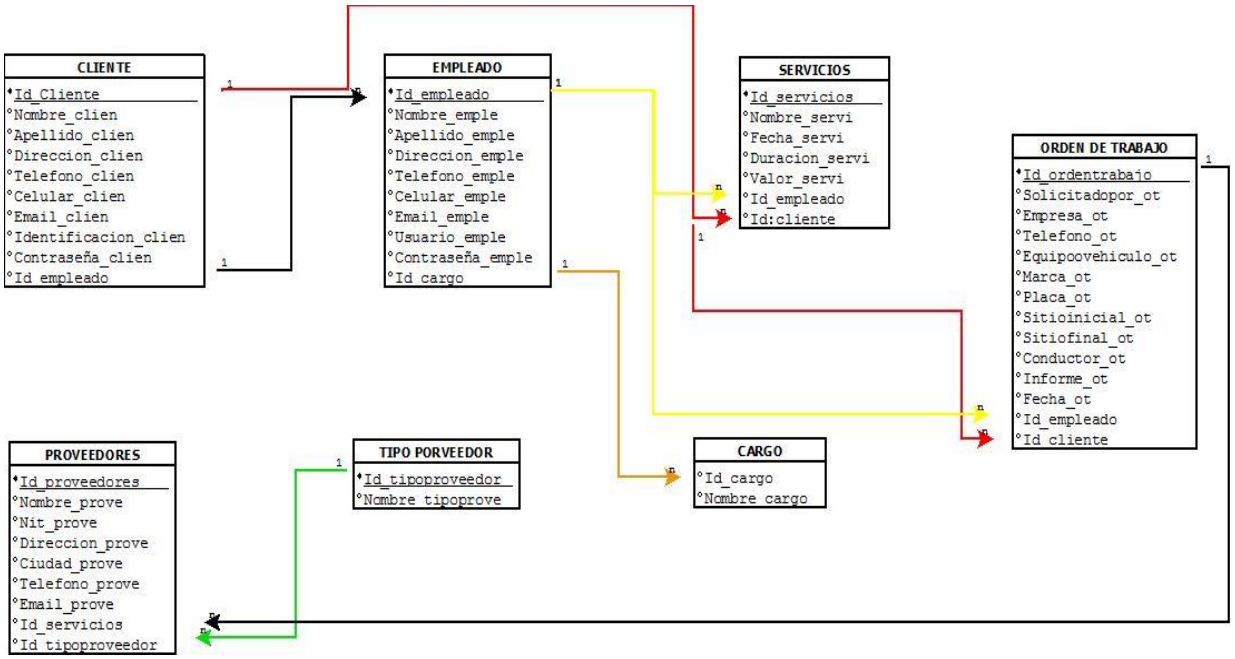


MODELO DE DATOS

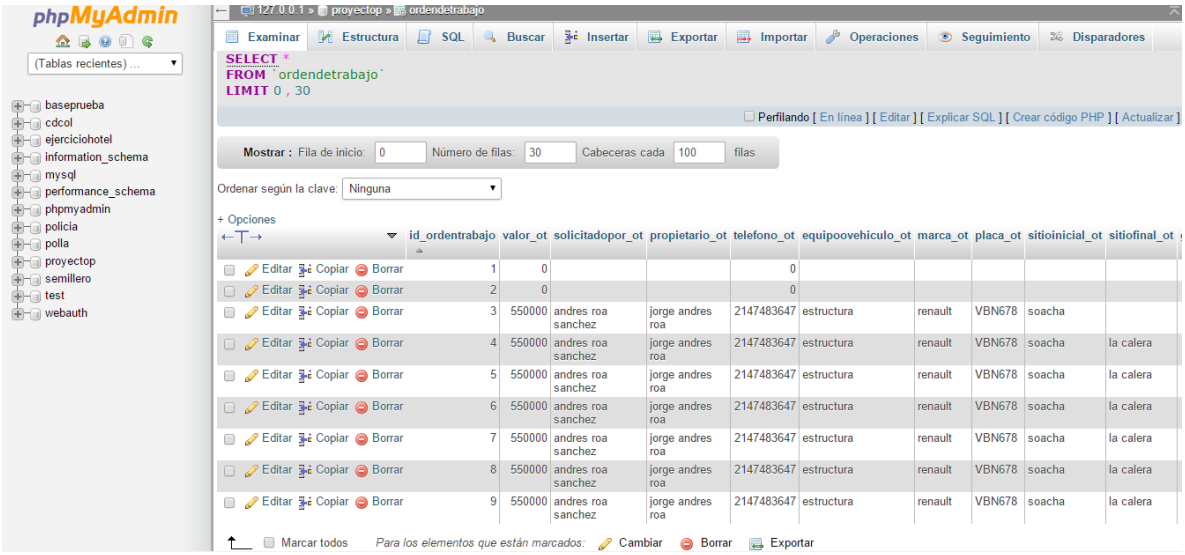
MODELO ENTIDAD RELACION



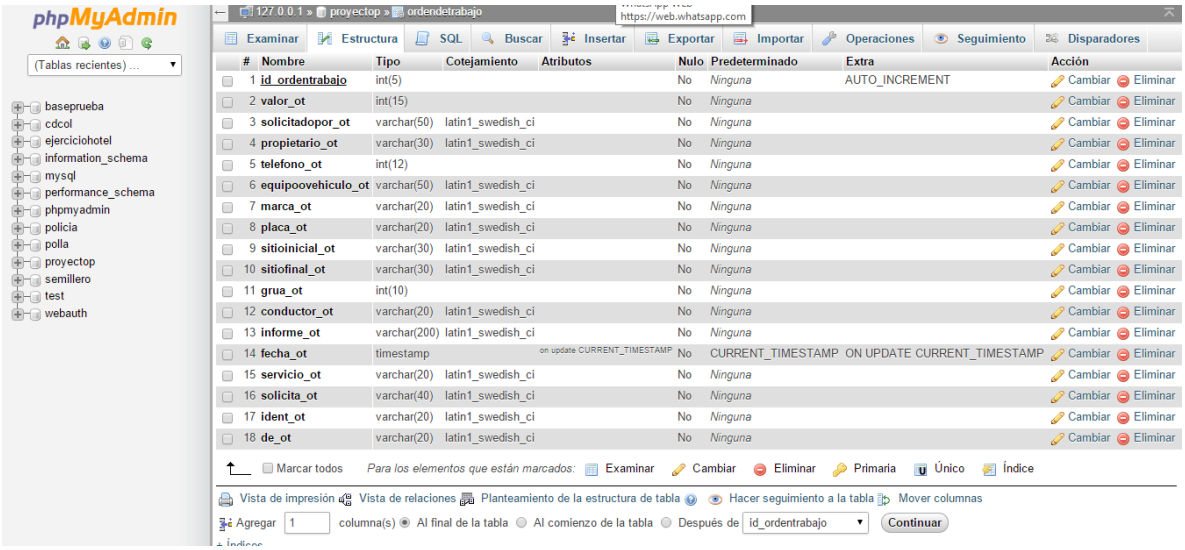
MODELO RELACIONAL



MODELO TABULAR



MODELO TABULAR



VIABILIDAD O FACTIBILIDAD

Es el proceso de análisis de viabilidad al estudio que intenta conocer previamente el éxito o fracaso de un proyecto. Para lograr esto se necesitan conocer datos empíricos que pueden llegar a ser variables, los cuales se dan a ilustrar a través de diversos tipos de investigaciones (estudios, encuestas, estadísticas, etc.)

El medio factible son las posibilidades que tiene de lograrse el desarrollo de un determinado proyecto. El análisis de factibilidad es el estudio que realiza una empresa para determinar si el negocio que se propone será de gran interés al público o de lo contrario será un fracaso total, y cuáles serán los medios estratégicos que se deben desarrollar para que el proyecto sea un éxito.

TECNICA

El sistema de gestión web SIGCRANE, requiere un software y un hardware con los elementos básicos de un equipo cómputo, podrá ser implementado en una lapto o equipo de escritorio, que cuente con un sistema operativo Microsoft Windows siete en cualquier versión de 64 y 32 bits, Procesador Intel corre i3, y una memoria RAM de 2 gb, es necesario tener un hosting o dominio con acceso a internet.

HUMANA

Para el manejo adecuado del sistema de gestión web SIGCRANE se realizaran capacitaciones al personal administrativo y logístico de la compañía, se indicara el manejo, las utilidades y comodidades que permitirá el sistema.

LEGAL

SIGCRANE es un software desarrollado en herramientas de uso libre; sin embargo para su ejecución, es necesario tener en cuenta los derechos de autor, la compra de su licencia para

su utilización legal, al igual que los sistemas operativos de los equipos en los cuales se ejecute.

FINANCIERA

La compañía AUTOSERVICIO SALAZAR S.A.S hacer una inversión de ocho millones quinientos aproximadamente que es el valor del software únicamente, ya que cuenta con los equipos y la infraestructura adecuada para su funcionamiento al igual que un plan de internet para el uso del sistema.

COSTOS

Los valores relacionados en las siguientes tablas se deriva de la sumas de los tres participantes durante un término en tiempo de dos meses de desarrollo.

COSTOS FIJOS		
REF	**DETALLE**	**VALOR**
1	SERVICIO DE LUZ X3	140000
2	SERVICIO DE INTERNET X3	270000
3	SERVICIO DE AGUA X3	300000
4	SALARIOS MINIMO X 3	4080000
	CFT	4790000

COSTOS VARIABLES		
REF	**DETALLE**	**VALOR**
1	ALIMENTACION X3	1200000
2	TRANSPORTE X3	600000
3	SUMINISTROS	300000
	CVT	2100000

UTILIDAD DE 20% = 6.890.000 * 20%= 1.378.000

VALOR TOTAL DEL PROYECTO = 8.268.000

MATRIZ DOFA

FACTORES INTERNOS / FACTORES EXTERNOS	**LISTA DE FORTALEZAS** **F1**: EXCELENTE IMAGEN **F2**: DISEÑO EXCLUSIVO **F3**: ESPACIO PARA TODO TIPO DE USUARIOS **F4**: CLARIDA DE DATOS **F5:** SE VE COMO UNA NECESIDAD **F6:** INTERESES EN LA COMUNIDAD **F7:** APLICA PARA MUCHOS CAMPOS	**LISTA DE DEBILIDADES** **D1**: ESCASA VINCULACION CON EL SECTOR **D2:** LA OFERTA DEL SISTEMA ESTA DE LA MANO CON LA DEMANDA **D3:** LOS INDICES DE EFICIENCIA SON BAJOS **D4:** FALTA DE CONECTIVIDAD **D5:** EQUIPOS OBSOLETOS
LISTA DE OPORTINIDADES **O1:** MEJORAR LOS INGRESOS DE LAS COMPAÑIAS **O2:** DAR A CONOCER LOS SERVICIOS QUE OFRECEN LAS COMPAÑIAS **O3**. ESTAR A LA VANGUARDIA TECNOLOGICA **O4:** CREAR UN SOFTWARE UNICO **O5:** TRABAJAR EN RED **O6:** AHORRAR TIEMPO Y DINERO **O7:** MEJORAR EL RENDIMIENTO DE LA EMPRESA	**FO** **ESTRATEGIA PARA MAXIMIXAR TANTO LAS F COMO LAS O** **1: CREAR UN SOFTWARE UNICO QUE PERMITA MOSTRAR AL 100% LAS CUALIDADES DE LA COMPAÑÍA Y ASI PODER MEJORAR E IMPULSAR LA ECONOMIA DE SI MISMA PARA SER PIONERA EN EL SECTOR DE PRODUCTIVIDAD**	**DO** **ESTRATEGIA PARA MINIMIZAR LAS D Y MAXIMIZAR LAS O** **1: DAR A CONOCER EL PRODUCTO, Y BRINDAR CAPACITACIONES PERMITIENDO QUE LAS PERSONAS SE INTERESEN MAS POR ESTAR A LA VANGUARDIA CON SUS COMPAÑIAS**
LISTA DE AMENAZAS **A1:** ESCASES DE VISITAS A LA PAGINA **A2:** POCO INTERES DE LOS CLIENTES **A3:** OFERTAS DE OTROS SISTEMAS **A4:** FALLAS DE INTERNET **A5:** VIRUS EN LA RED **A6:** FALLAS DE SEGURIDAD	**FA** **ESTRATEGIA PARA MINIMIZAR LAS A Y MEJORAR LA F** **1: BRINDAR UN SISTEMA SEGURO CAPAZ DE MANTENERSE EL LINEA DESDE CUALQUIER TERMINAL EN CASO DE FALLAS DE EQUIPOS**	**DA** **ESTRATEGIA PARA MINIMIZAR LAS D COMO LAS A** **1: FACILITAR EL MANEJO DEL SISTEMA PARA QUE SEA UTIL Y SENCILLO, Y QUE CUALQUIER PERSONA SEA CAPAZ DE SOLUCIONAR PROBLEMAS**

DICCIONARIO DE DATOS

TABLA DE DATOS

CLIENTE			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_cliente	int	5	identificador del cliente
Nombre_clien	varchar	20	nombre del cliente
Apellido_clien	varchar	20	apellidos del cliente
Direccion_clien	varchar	50	direccion ubicación del cliente
Telefono_clien	varchar	12	telefono fijo del cliente
Celular_clien	varchar	12	celular del cliente
Email-clien	varchar	50	correo electronico del cliente
Identificacion_clien	varchar	20	nit o cedula del cliente
Usuario_clien	varchar	20	crear seudinimo de la cuenta
Contraseña_clien	varchar	10	clave de la cuenta cliente

EMPLEADO			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_empleado	Int	5	identificador del empleado
Nombre_emple	Varchar	20	nombre del empleado
Apellido_emple	Varchar	20	apellidos del empleado
Direccion_emple	Varchar	50	direccion ubicación del empleado
Telefono_emple	Varchar	12	telefono fijo del empleado
Celular_emple	Varchar	12	celular del empleado
Email-emple	Varchar	50	correo electronico del empleado
Identificacion_emple	Varchar	20	cedula del empleado
Usuario_emple	Varchar	20	crear seudinimo de la cuenta
Contraseña_emple	Varchar	10	clave de la cuenta empleado

SERVICIOS			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_servicios	int	10	numero identificador de servicios
Nombre_servi	varchar	40	nombre del servicio
Fecha_servi	varchar	15	fecha de realizacion de servicio
Duracion_servi	varchar	20	duracion en tiempo del servicio
Valor_servi	numerico	10	precio del servicio

PROVEDORES			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_proveedores	int	10	numero identificador del proveedor
Nombre_prove	varchar	20	nombre del proveedor
Nit_prove	varchar	20	nit o cedula del proveedor
Direccion_prove	varchar	50	direccion ubicación del proveedor
Ciudad-prove	varchar	20	ciudad de ubicación del proveedor
Telefono-prove	varchar	12	telefono fijo del proveedor
Email-prove	varchar	20	correo electronico del proveedor

ORDEN DE TRABAJO			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_ordenTrabajo	int	1000	numero iden orden de trabajo
SolicitadoPor_ot	varchar	50	nombre de la persona que solicita el servicio
Empresa_ot	varchar	50	empresa que solicita el servicio
Telefono-ot	varchar	12	telefono de la empresa o persona que solicita el servicio
EquipooVehiculo	varchar	50	descripcion del vehículo que se transporta
Marca_ot	varchar	20	marca del equipo que se transporta
Placa_ot	varchar	20	placas del equipo que se transporta
SitioInicial_ot	varchar	30	lugar de inicio del servicio
SitioFinal_ot	varchar	30	lugar final del servicio
Conducto_ot	varchar	20	nombre del empleado que presta el servicio
Informe_ot	varchar	200	descripcion del servicio que se presto
Fecha_ot	varchar	20	fecha en la cual se presta el servicio

TIPO DE PROVEEDOR			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_tipoProveedor	Int	20	numero identificador del tipo proveedor
Nombre_tipoProveedor	varchar	20	nombre del tipo de proveedor

CARGO			
ATRIBUTO	TIPO	TAMAÑO	DESCRIPCION
Id_cargo	Int	10	numero identificador del cargo
Nombre_cargo	Varchar	20	nombre del cargo

GLOSARIO GENERAL

CORREO ELECTRÓNICO:
Servicio de internet que permite el intercambio rápido de mensajes entre personas remotas que no necesariamente han de estar conectadas a la vez. Para poder hacer uso, ambas personas deben disponer de una cuenta, ofrecida por un proveedor de estos servicios
DESCARGAR:
Transferir información desde un ordenador de la red Internet al ordenador propio. También se le suele llamar "bajar" o "download".
DISCO DURO:
Disco que se encuentra en el interior de la CPU y que almacena, tanto la información generada por el usuario, como los archivos necesarios para que los programas funcionen
DOMINIO DE INTERNET:
Es un grupo de entre dos y tres letras que agrupa a un conjunto de servidores con ciertas características comunes. Forma parte del nombre de dominio, que aparece en la dirección de las páginas web
ENLACE:
Elemento de una página web que da acceso a otro documento (o a otra parte del mismo documento) al hacer clic sobre él con el botón izquierdo del ratón. Es la base del acceso a la información en la World Wide Web
HARDWARE:
Cualquiera de los elementos físicos que componen un ordenador: disco duro, placa base, tarjeta gráfica, puertos...
HTML:
Lenguaje de marcas de hipertexto. Es el lenguaje en el que están escritas las páginas web. Realmente se trata de un texto en el que hay insertadas etiquetas que comienzan con el símbolo < y acaban con los símbolos />
INTERNET:
Es una red de ordenadores conectados entre sí que intercambian información a través de las líneas telefónicas. Ofrece multitud de servicios como la world wide web (www), la transferencia de ficheros, las charlas en tiempo real o chats, los foros, el correo electrónico.

MEMORIA RAM:
Parte de la memoria de un ordenador en la que éste almacena información de modo temporal y automático para poder realizar sus operaciones.

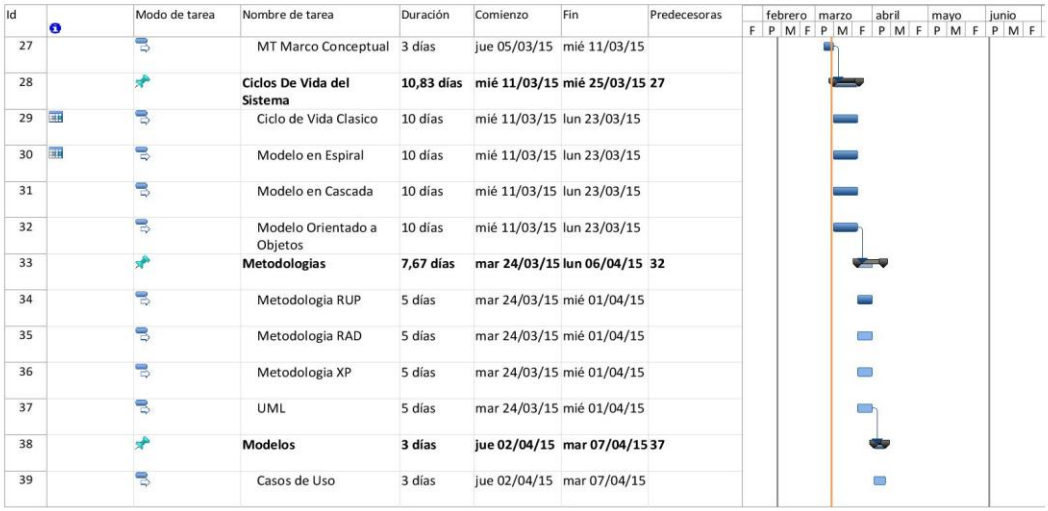
NAVEGADOR:
Programa que interpreta el código (HTML y más) en el que están escritas las páginas web y nos las muestra tal y como las vemos en el monitor
ON-LINE:
Estado de un ordenador cando está conectado a internet. La traducción literal es "en línea". Esta expresión se usa cuando se habla de actividades en las que se obtiene una respuesta del servidor que nos da acceso a internet.
PÁGINA WEB:
Es un archivo, escrito en código HTML, que se encuentra en el disco duro de un ordenador conectado a la red internet. Estos archivos se transfieren por la línea telefónica y, si se ven a través de un programa navegador (como Internet Explorer o FireFox) muestran texto, imágenes, animaciones, sonidos, vídeos... Pero lo más característico es que contienen enlaces (también llamados vínculo o hipervínculos) a otras páginas o documentos, de manera que podemos ir "saltando" por la información que nos interesa.
PASSWORD:
Literalmente "palabra de paso". También se le llama contraseña. Es una palabra, conocida sólo por el usuario, que le permite el acceso y uso a zonas privadas, bien de la web, bien de programas específicos-
PROCESADOR:
Circuitos electrónicos incluidos en una pastilla que ejecutan las instrucciones básicas de un ordenador. Es el núcleo que le permite realizar todas las operaciones que nosotros utilizamos. También se le llama microprocesador, debido a su pequeño tamaño.
WINDOWS:
Sistema operativo de Microsoft, que ha ido evolucionando en diferentes versiones para los distintos tipos de ordenadores.
AUTOMATIZAR:
Aplicar la automática a un proceso, a un dispositivo, etc.
ORGANIGRAMA:
Sinopsis o esquema de la organización de una entidad, de una empresa o de una tarea.

WEBGRAFIA

http://www.fceia.unr.edu.ar/asist/intro-cuali-t.pdf
http://148.202.105.18/webcucsur/sites/default/files/CICLO%20DE%20VIDA%20DEL%20SW%20JAIME.pdf
http://ingenexescom.blogspot.com/2012/02/modelo-en-cascada.html
http://informatica.uv.es/iiguia/DBD/Teoria/capitulo_2a.pdf
http://procesosdesoftware.wikispaces.com/METODOLOGIA+XP
http://4.bp.blogspot.com/-NhkT3CG-kWo/TZIju379zaI/AAAAAAAAAAg/XgUWv6kvv7o/s1600/ciclo-de-vida-clasico-de-desarrollo-de-sistemas1.jpg
http://3.bp.blogspot.com/-Gju-pCvrGFs/UIsQ3gSaNsI/AAAAAAAAABg/WT0qPIT0x1Y/s1600/espiral.png
http://2.bp.blogspot.com/-41FxuAYiTi0/ULqPQYtK0fI/AAAAAAAAADQ/Ixu9tFNc9co/s1600/36-b04b7deeb0.png
http://derechosdeautorsoftware.blogspot.com/2008/09/los-diversos-tipos-de-infraccin.html

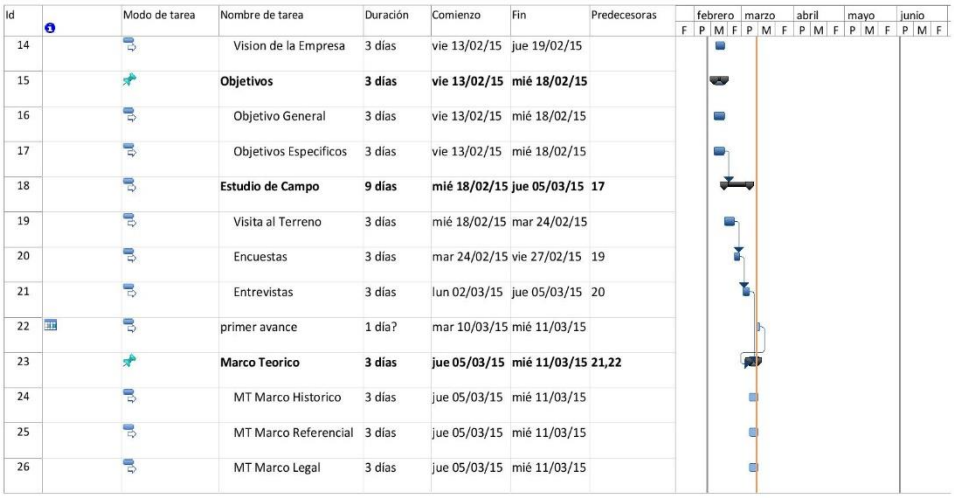
CRONOGRAMA DE ACTIVIDADES

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
27		MT Marco Conceptual	3 días	jue 05/03/15	mié 11/03/15	
28		**Ciclos De Vida del Sistema**	**10,83 días**	**mié 11/03/15**	**mié 25/03/15**	**27**
29		Ciclo de Vida Clasico	10 días	mié 11/03/15	lun 23/03/15	
30		Modelo en Espiral	10 días	mié 11/03/15	lun 23/03/15	
31		Modelo en Cascada	10 días	mié 11/03/15	lun 23/03/15	
32		Modelo Orientado a Objetos	10 días	mié 11/03/15	lun 23/03/15	
33		**Metodologias**	**7,67 días**	**mar 24/03/15**	**lun 06/04/15**	**32**
34		Metodologia RUP	5 días	mar 24/03/15	mié 01/04/15	
35		Metodologia RAD	5 días	mar 24/03/15	mié 01/04/15	
36		Metodologia XP	5 días	mar 24/03/15	mié 01/04/15	
37		UML	5 días	mar 24/03/15	mié 01/04/15	
38		**Modelos**	**3 días**	**jue 02/04/15**	**mar 07/04/15**	**37**
39		Casos de Uso	3 días	jue 02/04/15	mar 07/04/15	



Página 3

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
14		Vision de la Empresa	3 días	vie 13/02/15	jue 19/02/15	
15		**Objetivos**	**3 días**	**vie 13/02/15**	**mié 18/02/15**	
16		Objetivo General	3 días	vie 13/02/15	mié 18/02/15	
17		Objetivos Especificos	3 días	vie 13/02/15	mié 18/02/15	
18		**Estudio de Campo**	**9 días**	**mié 18/02/15**	**jue 05/03/15**	**17**
19		Visita al Terreno	3 días	mié 18/02/15	mar 24/02/15	
20		Encuestas	3 días	mar 24/02/15	vie 27/02/15	19
21		Entrevistas	3 días	lun 02/03/15	jue 05/03/15	20
22		primer avance	1 día?	mar 10/03/15	mié 11/03/15	
23		**Marco Teorico**	**3 días**	**jue 05/03/15**	**mié 11/03/15**	**21,22**
24		MT Marco Historico	3 días	jue 05/03/15	mié 11/03/15	
25		MT Marco Referencial	3 días	jue 05/03/15	mié 11/03/15	
26		MT Marco Legal	3 días	jue 05/03/15	mié 11/03/15	



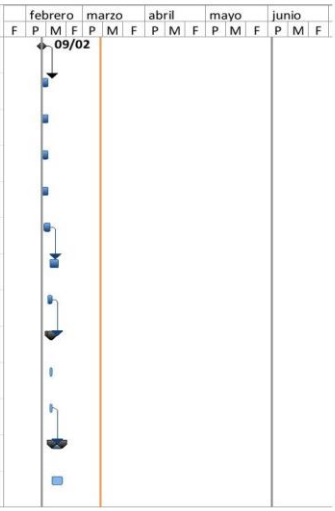
Página 2

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
66		Anexos	2 días	vie 29/05/15	mar 02/06/15	65
67		Fin	0 días	mar 02/06/15	mar 02/06/15	66

Proyecto: Proyecto de Grado
Fecha: mar 10/03/15

Tarea	Hito externo		Informe de resumen manual
División	Tarea inactiva		Resumen manual
Hito	Hito inactivo		Sólo el comienzo
Resumen	Resumen inactivo		Sólo fin
Resumen del proyecto	Tarea manual		Fecha límite
Tareas externas	Sólo duración		Progreso

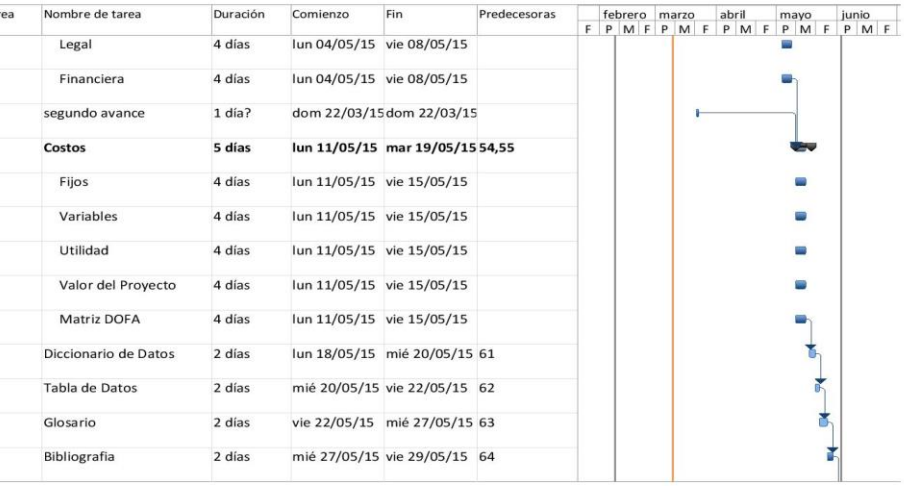
Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1		Inicio	0 días	lun 09/02/15	lun 09/02/15	
2		Portada y Contraportada	2 días	lun 09/02/15	mié 11/02/15	1
3		Nota de Aceptacion	2 días	lun 09/02/15	mié 11/02/15	
4		Agradecimientos	2 días	lun 09/02/15	mié 11/02/15	
5		Dedicatoria	2 días	lun 09/02/15	mié 11/02/15	
6		Introducción	2 días	mar 10/02/15	jue 12/02/15	
7		Tabla de Contenido	2 días	jue 12/02/15	lun 16/02/15	6
8		Justificacion	1 día	jue 12/02/15	vie 13/02/15	
9		**Mision**	**1 día**	**jue 12/02/15**	**vie 13/02/15**	**8**
10		Mision del Proyecto	1 día?	jue 12/02/15	vie 13/02/15	
11		Mision de la Empresa	1 día?	jue 12/02/15	vie 13/02/15	
12		**Vision**	**3 días**	**vie 13/02/15**	**jue 19/02/15**	**11**
13		Vision del Proyecto	3 días	vie 13/02/15	jue 19/02/15	



Proyecto: Proyecto de Grado
Fecha: mar 10/03/15

Tarea	Hito externo		Informe de resumen manual
División	Tarea inactiva		Resumen manual
Hito	Hito inactivo		Sólo el comienzo
Resumen	Resumen inactivo		Sólo fin
Resumen del proyecto	Tarea manual		Fecha límite
Tareas externas	Sólo duración		Progreso

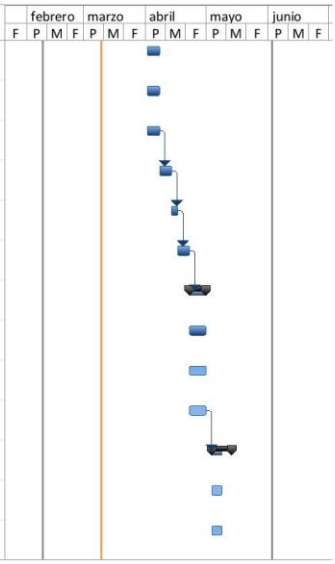
Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
53		Legal	4 días	lun 04/05/15	vie 08/05/15	
54		Financiera	4 días	lun 04/05/15	vie 08/05/15	
55		segundo avance	1 día?	dom 22/03/15	dom 22/03/15	
56		**Costos**	**5 días**	**lun 11/05/15**	**mar 19/05/15**	**54,55**
57		Fijos	4 días	lun 11/05/15	vie 15/05/15	
58		Variables	4 días	lun 11/05/15	vie 15/05/15	
59		Utilidad	4 días	lun 11/05/15	vie 15/05/15	
60		Valor del Proyecto	4 días	lun 11/05/15	vie 15/05/15	
61		Matriz DOFA	4 días	lun 11/05/15	vie 15/05/15	
62		Diccionario de Datos	2 días	lun 18/05/15	mié 20/05/15	61
63		Tabla de Datos	2 días	mié 20/05/15	vie 22/05/15	62
64		Glosario	2 días	vie 22/05/15	mié 27/05/15	63
65		Bibliografia	2 días	mié 27/05/15	vie 29/05/15	64



Proyecto: Proyecto de Grado
Fecha: mar 10/03/15

Tarea
División
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
Sólo duración
Informe de resumen manual
Resumen manual
Sólo el comienzo
Sólo fin
Fecha límite
Progreso

Id	Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
40		Diagrama de Clases	3 días	jue 02/04/15	mar 07/04/15	
41		Diagrama De Estados	3 días	jue 02/04/15	mar 07/04/15	
42		Diagramas de Secuencias	3 días	jue 02/04/15	mar 07/04/15	
43		Flujograma de Procesos	3 días	mar 07/04/15	lun 13/04/15	42
44		Organigrama de La Empresa	3 días	lun 13/04/15	jue 16/04/15	43
45		Diagrama de Flujo de Datos	3 días	vie 17/04/15	mié 22/04/15	44
46		**Modelo de Datos**	**5 días**	**jue 23/04/15**	**vie 01/05/15**	**45**
47		Modelo de E-R	5 días	jue 23/04/15	vie 01/05/15	
48		Modelo Relacional	5 días	jue 23/04/15	vie 01/05/15	
49		Modelo Tabular	5 días	jue 23/04/15	vie 01/05/15	
50		**Viabiidad o Factibilidad**	**6,67 días**	**lun 04/05/15**	**jue 14/05/15**	**49**
51		Tecnica	4 días	lun 04/05/15	vie 08/05/15	
52		Humana	4 días	lun 04/05/15	vie 08/05/15	



Proyecto: Proyecto de Grado
Fecha: mar 10/03/15

Tarea
División
Hito
Resumen
Resumen del proyecto
Tareas externas
Hito externo
Tarea inactiva
Hito inactivo
Resumen inactivo
Tarea manual
Sólo duración
Informe de resumen manual
Resumen manual
Sólo el comienzo
Sólo fin
Fecha límite
Progreso

ANEXO1

REQUERIMIENTOS

Número de requisito	RF01
Nombre de requisito	Crear una interfaz grafica de usuario
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Sistema web
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF02
Nombre de requisito	Crear un formulario de registro de usuarios
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Base Datos Tablas: Empleados, Clientes, Servicios, Orden de trabajo, Proveedores, Cargo. Sistema Web
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF03
Nombre de requisito	Base Datos
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Sistema web BD
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF04
Nombre de requisito	Menús desplegables para búsqueda o consulta, ejemplo Servicios.
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Interfaz y base de datos
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF05
Nombre de requisito	Formulario de Contacto para Clientes
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Clientes, Administrador
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF06
Nombre de requisito	Galería de Fotos
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador,
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF07
Nombre de requisito	Modulo de Empleados
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional
Número de requisito	RF08
Nombre de requisito	Reportes de Nomina
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF09
Nombre de requisito	Certificados Laborales del personal
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF10
Nombre de requisito	Solicitudes de Vacaciones
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF11
Nombre de requisito	Ordenes de Trabajo
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Número de requisito	RF12
Nombre de requisito	Reporte de ordenes de trabajo
Tipo	X☐ Requisito ☐ Restricción
Fuente del requisito	Administrador, Empleados.
Prioridad del requisito	X☐ Alta/Esencial ☐ Media/Deseado ☐ Baja/ Opcional

Requisitos comunes de los interfaces

Interfaces de usuario

- La interfaz diseñada para el usuario tendrá un conjunto de ventanas y botones, listas y campos de texto, la cual deberá tener todas las opciones del sistema propuesto y su visualización se realizara a través de internet.

Interfaces de hardware

- En los computadores que se use la aplicación web deberán estar en perfecto estado, con soporte de internet y conexión a la misma.
- Adaptadores de red alambrica o inalámbrica.
- Procesadores de 1.66GHz y superior.
- Memoria mínima de 256Mb.
- Mouse.
- Teclado.

Interfaces de software

- Nombre de Dominio : el sitio web deberá contra con un nombre único, para su identificación en la red ; de modo tal que los usuarios puedan acceder.

Interfaces de comunicación

- Tanto los servidores como los clientes se comunicaran entre sí, a través de protocolos de estándares de internet, cuando sea posible. Ejemplo para transferencia de archivos o documentos, imágenes, audio etc., de modo seguro.(FTP, HTTPS u otros convenientes).

REQUISITOS FUNCIONALES

Requisito funcional 1

- **Interfaz Grafica de Usuario:** Se deberá construir una interfaz, la cual permita a los usuarios navegar en la página de forma fácil e intuitiva.

- ✓ Los usuarios podrán encontrar de forma fácil y rápida la información de consulta, dado que la interfaz de usuarios contribuirá con este fin.

Requisito funcional 2

- **Registrar usuarios:** El sistema de Gestión web dará a (Clientes, Empleados, Desarrolladores, Administrador, Visitantes), la opción de registrarse. Mediante la inserción de datos como: Nombre. Apellido, email
  - ✓ La información se almacenara en la base datos, para que cuando el usuario ingrese, el sistema realice la respectiva validación.

Requisito funcional 3

- **Base de Datos:** permitirá el registro, validación, de usuarios según su perfil.
  - ✓ Tomara la información la almacenara para luego, permitir las respectivas consultas o modificaciones.

Requisito funcional 4

- **Consultar Información en los menús:** El sistema de Gestión web ofrecerá a los usuarios la posibilidad de consultar acerca de los Servicios de Grúa, para aplicaciones de transporte.
-
  - ✓ **Consultar Servicios**: Permitirá conocer las ultimas aplicaciones de transporte de Grúa ofrecidos por la empresa.
  - ✓ **Consultar Quienes Somos**: Permitirá der a conocer la información pertinente de la empresa.
  - ✓ **Consultar Empleados**: Dara la capacidad de consultar la información respectiva de cada empleado.

Requisito funcional 5

- ✓ **Formulario de contacto para clientes:** destino a un espacio donde los clientes pueden publicar sus inquietudes, así como solicitar servicios vía web y a la vez plantear sus inquietudes.

Requisito funcional 6

- **Galería de fotos**: Permitirá la publicación de imágenes.

- ✓ El administrador publica las imágenes, suministradas por los empleados y o usuarios.
- ✓ Basado en información veraz y respetando los derechos de la fuente. En su propiedad intelectual o material.

Requisito funcional 8

- **Modulo de empleados**: Permitirá la visualizar la información de los empleados.

  - ✓ El administrador podrá publicar la información pertinente a cada empleado, al igual que generar reportes de los mismos.

Requisito funcional 9

- **Certificados laborales del personal**: Permitirá la visualizar la información del vinculo laboral, como la impresión del mismo.

  - ✓ El administrador podrá publicar la información pertinente a cada empleado, al igual que la modificación de la misma, cuando lo estime conveniente.

Requisito funcional 10

- **Solicitudes de Vacaciones**: Permitirá a los empleados a través del sistema hacer su solicitud de periodo de vacaciones.

  - ✓ El administrador podrá publicar la información pertinente a cada empleado, mediante la validación de la información dará la aprobación de la solicitud.

Requisito funcional 11

- **Ordenes de trabajo**: Permitirá a los empleados a través del sistema de Gestión Web, poder descargar la información de los servicios.

  - ✓ El administrador podrá visualizar e imprimir la información de los servicios prestados para su posterior cobro y liquidación a los empleados.

Requisito funcional 12

- **Reporte de ordenes de trabajo**: Permitirá a los empleados a través del sistema de Gestión Web, poder descargar la información de los servicios.

- ✓ El administrador podrá visualizar e imprimir la información de los servicios prestados para su posterior cobro y liquidación a los empleados.
- ✓ Además permite al administrador saber el estado de los servicios y la disponibilidad de los vehículos.

REQUISITOS NO FUNCIONALES

Requisitos de rendimiento

- Garantizar que el diseño de las consultas o cualquier otro proceso no afecten el desempeño de las bases de datos o de la red por saturación de tráfico.

Seguridad

- Garantizar a los usuarios seguridad y confiabilidad en el desempeño del sistema de Gestión Web, para esto los registros de consultas pueden ser consultados y actualizados de forma permanente y simultanea, sin que el tiempo de respuesta se vea afectado.

- Garantizar la seguridad del sistema con respecto a la información y datos que se manejan, ya sean documentos, archivos y contraseñas de usuarios.

- Calidad y control de accesos a los usuarios de internet, con la posibilidad de subir información y consultar la misma.

Fiabilidad

- Para hacer de la pagina fiable se requiere una interfaz de uso intuitivo muy sencilla, fácil para cualquier usuario.

- La interfaz deberá ajustarse a otros en tornos web, para cualquier consulta acerca de los mismos.

Disponibilidad

La aplicación web. Deberá estar el 100% del tiempo disponible, las 24 horas del día 7, días a la semana, con la capacidad de ofrecer una detección temprana de posibles fallos a fin de corregirlos previamente.

Mantenibilidad

- Tener un completo manual documentado, de cada una de las actividades de mantenimiento y control, para minimizar el esfuerzo y los costos.
- Dentro de la interfaz deberá tener un completo manual de ayudad, ya que la administración puede eventualmente recaer sobre personal de poca experiencia, en uso de la informática.

Portabilidad

- Se realizara con la opción de correr en cualquier navegador web y con cualquier motor de búsqueda, además deberá ser responsiva, para su correcta visualización a través de dispositivos móviles y tabletas.
- Básicamente para sistemas operativos Windows, Linux, Android entre otros.

Otros requisitos

- Cualquier información que se publique, deberá provenir de fuentes confiables y respetando siempre, o reconociendo el derecho a sus autores o creadores.
- Fomentar el respeto por las personas y la propiedad intelectual.

ANEXO 2

CASO DE USO GENERAL