UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

TESINA

Nombre del programa: Mejora de la Gestión Ambiental del Transporte

QUE PARA OBTENER EL TÍTULO DE INGENIERO EN COMPUTACIÓN

PRESENTA:

Oscar Roberto Belman Galván

DIRECTOR DE TESINA
Carlos Alberto Román Zamitiz

CIUDAD UNIVERSITARIA

Agradecimientos

Quiero agradecer a la Universidad Nacional Autónoma de México, que me dio la oportunidad de estudiar una carrera y me abrió las puertas al conocimiento. A la honorable Facultad de Ingeniería nido de muchos que como yo eligieron esta extraordinaria carrera y que con mucho orgullo, amor, pasión y respeto representaré.

A la Secretaría de Medio Ambiente y Recursos Naturales por darme la oportunidad de pertenecer a su programa de servicio social, de aprender y emprender con este proyecto.

A todos mis maestros de la carrera por sus conocimientos, consejos, confianza y formación en especial al Ing. Carlos Alberto Román Zamitiz quien me fue guiando por todo este proceso, me apoyó y me tuvo paciencia y para poder terminar este trabajo. Al Dr. Daniel Trejo Medina, Ing. German Santos Jaimes y al Maestro Oscar Sulvaran Rodríguez porque influyeron en mi formación con sus lecciones y experiencias en formarme como una persona de bien y preparada para los retos que pone la vida, y hacerme mejor como ingeniero.

Hoy y siempre a mi familia, mis padres, mis hermanos y mi cuñado por todo su apoyo incondicional, la alegría y la fortaleza necesaria para seguir adelante.

Dedicatorias

Con todo mi cariño y mi amor para mis padres que hicieron todo en la vida para que yo pudiera lograr mis sueños, por motivarme y darme la mano cuando sentía que el camino se terminaba, a ustedes por siempre mi corazón y mi agradecimiento.

A ti mamá porque se que esto significa mucho más para ti, sufriste conmigo constantemente y eres esa fuente interminable de apoyo, y amor incondicional.

A ti papá porque me das ganas de ser un hombre de bien. Me inspiraste a seguir tu ejemplo desde pequeño para convertirme como tú en ingeniero.

A mis hermanos porque siempre han estado junto a mi brindándome todo su apoyo. A mi hermana mayor, Carito, porque siempre has sido un gran ejemplo a seguir. Aprendí mucho de ti y pusiste la barra alta motivándome a siempre dar todo de mi. Tu sabiduría y madurez han sido una fuerte influencia para lograr todos mis objetivos en la vida. Sin ti no sería el hombre que soy hoy. A mi hermano menor, Sebs, porque me haces querer ser el mejor ejemplo a seguir, ser mejor hombre y ser mejor hermano. Me haces querer ser fuente de admiración y a dar mi máximo esfuerzo para no decepcionarte. A mi cuñado, Miguel, otro gran modelo a seguir. Un claro ejemplo de que con trabajo, esfuerzo y dedicación puedes lograr todo lo que te propongas. Gracias por siempre creer en mi.

A mi novia, Denisse, tu paciencia, comprensión y bondad me inspiraron a ser mejor. En momentos de desesperación podía contar contigo para poder encontrar confort y aliento para seguir dando mi máximo esfuerzo, fuiste capaz de contenerme cuando todo iba mal y hacerme sentir que todo iba a estar bien como solo tú lo puedes hacer. Gracias por estar siempre a mi lado. Menorquetres. #teamBelmango

Alex, Juan Carlos, Lalo, Héctor y Paco, cómo no querer dar todo de ti cuando tienes amigos que son élite en cada una de sus áreas de conocimiento. Un gran pilar donde siempre he podido apoyarme en cualquier situación. Siempre han estados listos para brindarme toda su ayuda. Gracias por su amistad. #SkD

Luque, Corro y Luis Omar, porque la unión hace la fuerza. Encontré en ustedes una gran amistad y gran apoyo durante estos 9 semestres. Nos convertimos en un gran equipo de trabajo, una pequeña familia. Porque juntos salimos adelante y aprendí el verdadero significado de trabajar en equipo. #teamDonker

Por último, a las familias Galván Ruiz, Villanueva López y García Amador, porque siempre han creído en mi y me han apoyado a lo largo de toda mi vida. Son un gran ejemplo como familia y como personas, me hicieron parte de su familia. He podido aprender mucho de convivir con ustedes a lo largo de mi vida.

Índice

Nombre del Programa: Mejora de la Gestión Ambiental del Transporte

I. Objetivo

Aplicar los conocimientos adquiridos en la carrera de Ingeniería en Computación, con módulo de Ingeniería de Software, para mejorar el trámite ambiental denominado Cédula de Operación Anual (COA) del programa ambiental Registro de Emisiones y Transferencia de Contaminantes por medio de un sistema de mejores prácticas en el desarrollo de Software en la Secretaría de Medio Ambiente y Recursos Naturales.

II. Marco teórico

II.1 Introducción

La Secretaría de Medio Ambiente y Recursos Naturales (SEMARNAT) es la dependencia del gobierno federal encargada de impulsar la protección, restauración y conservación de los ecosistemas y recursos naturales y bienes y servicios ambientales de México, con el fin de propiciar su aprovechamiento y desarrollo sustentable.

Para cumplir con este mandato, la SEMARNAT, sus tres subsecretarías y los diversos Órganos Desconcentrados y Descentralizados que forman parte del Sector Ambiental Federal, trabajan en cuatro aspectos prioritarios:

- La conservación y aprovechamiento sustentable de los ecosistemas y su biodiversidad.
- La prevención y control de la contaminación.
- La gestión integral de los recursos hídricos.
- El combate al cambio climático.

Durante mi estancia en la SEMARNAT estuve en la subdirección de Información y Divulgación, en el área enfocada en la prevención y control de la contaminación a través de programas como el Registro de Emisiones y Transferencia de Contaminantes y la Cédula de Operación Anual. El área de prevención y control de la contaminación tiene como objetivo prevenir, reducir y controlar la generación de residuos y las emisiones contaminantes que afectan los suelos, el agua y el aire.

Por ello la SEMARNAT desarrolla importantes esfuerzos, impulsa al establecimiento de estrategias estatales y municipales de gestión de residuos, el Registro de Emisiones y Transferencia de Contaminantes, la remediación de sitios contaminados y el manejo integral y seguro de las sustancias químicas y materiales peligrosos, además de vigilar el estricto cumplimiento de la legislación ambiental mediante la realización de acciones de inspección, vigilancia y auditoría ambiental.

La SEMARNAT cuenta con un software denominado “Software de la Cédula de Operación Anual” el cual debe ser instalado por todas las empresas que residen en suelos mexicanos, para reportar por este medio todos los contaminantes que se van al aire, al agua y/o al suelo como resultado de la producción o trabajo de dicha empresa, o bien si estos residuos son enviados a otra empresa para que se deshagan de ellos.

El principal objetivo del servicio social fue enfocarse en el desarrollo de un manual de usuario, el cual se encontrará incrustado en un sistema web. Un manual web intuitivo, fácil de usar y fácil de entender para que los usuarios finales pudieran hacer un correcto uso del software de la Cédula de Operación Anual. También, en la depuración y actualización de las bases de datos que arrojará el software de la Cédula de Operación Anual. Con base a los resultados arrojados por el software de la Cédula de Operación Anual, se lleva un correcto control de los contaminantes que cada empresa arroja al medio ambiente para poder prevenir daños mayores a éste.

II.2 El Software

El software es la parte lógica de la computadora que está compuesta por todos los programas, rutinas y sistemas que permiten a la computadora ejecutar sus funciones.

Los componentes de cualquier sistema de software se muestran en la figura 1.1

Software=Programas + Documentación + Procedimientos de Operación

Figura 1.1

La creación y elaboración de software beneficia toda actividad en los espacios productivos por la optimización de tiempo, recursos y trabajo. La producción, distribución y utilización de cualquier programa computacional nos permite estructurar y organizar un sinfín de actividades que son indispensables.

Con el avance de la tecnología digital el software es más complejo, más poderoso y tiene más probabilidad de fracasar, por eso el desarrollo de software debe seguir reglas que permitan generar un producto que cumpla sus requerimientos.

En los primeros años de las aplicaciones informáticas los programadores solían desarrollar sus proyectos con su propio estilo personal y no seguían ningún método estándar. En este enfoque, el diseño se realizaba de manera implícita en la mente de cada desarrollador, y la documentación era a menudo inexistente, lo cual causaba muchísimos problemas al largo plazo.

Esta forma de hacer las cosas era adecuada para el desarrollo de programas sencillos y para pequeños equipo de trabajo. Sin embargo, con la rápida evolución del software, los programas de computadora se volvieron demasiado complejos y sofisticados para poder ser manejadas sin algún enfoque específico y general. El software fue evolucionado rápidamente, y pronto se hacia cargo de más y más funciones en máquinas del día a día. Fue hasta que surgieron grandes casas de desarrollo de software, que se buscaba generalizar su desarrollo para una amplia distribución.

Un estudio realizado en 1979 por la Contraloría General de los Estados Unidos encontró estos problemas en el desarrollo de proyectos de software:

- 2 % de los proyectos de software trabajó en la entrega
- 3 % de los proyectos trabajaba sólo después de algunas correcciones
- 45 % de los proyectos fueron entregados, pero nunca funcionando al 100% como lo pedía el cliente
- 30 % de los proyectos se pagan, pero nunca completo por entrega tardía

Debido a la ausencia de una forma estandarizada del desarrollo de proyectos de software, era muy difícil encontrar y corregir los errores que surgieran. El costo de desarrollar el software a menudo excedía el presupuesto estimado, lo cual hacía que no fueran rentables. También se notó que los proyectos de software estaban tomando mucho más tiempo de lo previsto inicialmente.

La ausencia de un enfoque estructurado y metódico puede causar problemas como:

Los errores cometidos en una fase de desarrollo llevadas a fases posteriores, provocando que se tenga que volver y empezar o corregir fases que deberían estar completadas y por lo tanto retrasan la entrega del proyecto.
Dificultad en la estimación de honorarios y la medición del progreso causa deslizamiento de horario y exceso de presupuesto.
Dificultades en la incorporación de las modificaciones de diseño e instalación de versiones correctas de software.

Se observó que la mayoría de los proyectos de software fracasaron, particularmente en el cumplimiento costos y tiempo. También se pudo observar que los fracasos se debieron más a causa de factores humanos, que debido a la falta de conocimientos técnicos. Muchos de los problemas con el desarrollo de software se pueden remontar a la metodología defectuosa. Estos problemas de desarrollo de software se hicieron sentir ampliamente en el comienzo de la década de 1970 y se conocen colectivamente como la “Crisis del Software”.

Durante mi estancia en la SEMARNAT me encontré que había mucho problemas en el software de la Cédula de Operación Anual. Esta aplicación la había desarrollado un pequeño equipo de trabajo hace 8 años. El equipo que lo había desarrollado no siguió alguna metodología de desarrollo, ni habían hecho la documentación necesaria. Al software de la Cédula de Operación Anual se le hacían modificaciones constantemente; cada que se modificaba el código para adaptarlo a nuevos requerimientos, no se hacía un control de cambios. Todo esto hizo que se tuviera una herramienta funcional, pero mal gestionada. Usaba códigos de diferentes desarrolladores y por lo general nadie quería meterse en módulos específicos que no se entendía bien que era lo que realizaban y tenían miedo a que dejara de funcionar el software. Todo esto escaló al grado de que tuvieron que llevar un equipo de consultores a tratar de

optimizar el software y quitar módulos que ya no fueran útiles. Fue entonces que sugerí implementar alguna metodología de mejores prácticas. Sugerí usar el Modelo de Procesos para la Industria del Software (MoProSoft), dado que era el modelo con el cual estaba más familiarizado. Más adelante profundizaré en lo que consiste este modelo.

II.3 Introducción a la Ingeniería de Software

La Ingeniería de Software juega un rol muy importante ya que no son sólo programas, es todo aquello que se involucra en el proceso del desarrollo de una sistema de información y su metodología en la administración de información, mediante el planteamiento de diversos problemas relacionados con el desarrollo de software y los diversos modelos de procesos de producción de software tales como los diagramas de flujo, manuales de operación y demás elementos relacionados con la configuración de datos que se necesitan para hacer que estos programas operen de manera correcta.

Cualquier programa es un subconjunto de software y se convierte en el software sólo si se preparan manuales de documentación y procedimiento operativo. El programa es una combinación de código fuente y el código objeto.

La Ingeniería de Software puede definirse como la rama de la Ingeniería encargada del diseño, desarrollo, operación y mantenimiento de productos de software de manera sistemática y cuantificable.

II.4 Importancia de la Ingeniería de Software

Como ya se había mencionado, la experiencia previa en la construcción de estos sistemas mostró que un enfoque informal para el desarrollo de software no era muy bueno, debido a que los grandes proyectos a menudo tenían años de atraso, costaban mucho más de lo presupuestado, eran difíciles de mantener y con un desempeño pobre. Cómo resultado, fueron

necesarias nuevas técnicas y métodos para controlar la complejidad inherente en los sistemas de software. Estas técnicas han llegado a ser parte de la Ingeniería de Software y aunque se utilizan ampliamente no se ha logrado que esto sea universal, ya que hay varios estándares diferentes.

Actualmente, casi todas las empresas y países hacen uso de varios sistemas computacionales para poder trabajar día con día, en esos sistemas el software es el componente predominante, como los son: control de procesos, sistemas de seguridad, aviación civil, redes de comunicación, telefonía, etc. En algunos casos si el software llegara a fallar las consecuencias pueden llegar a ser muy graves e irreversibles.

Con la Ingeniería de Software se puede analizar, diseñar, programar y aplicar un programa de manera correcta y organizada, cumpliendo con todas las especificaciones del cliente y del usuario final, también ayuda a mejorar la calidad de los productos de software y facilita el control en el proceso de desarrollo de software.

La Ingeniería de Software pretende demostrar si el modelo de procesos para la industria del desarrollo del software, el cual fomenta la estandarización de su operación a través de la incorporación de las mejores prácticas en gestión e ingeniería de software, da mejores resultados al aplicarse en organizaciones dedicadas al desarrollo y mantenimiento del software.

La ingeniería es el análisis, diseño, construcción, verificación y gestión de las entidades técnicas. Sin importar la entidad que ha de ser diseñada, siempre se deben formular y contestar las siguientes preguntas:

- ¿Cuál es el problema a resolver?
- ¿Qué hacer?
- ¿Cuáles son las características de la entidad que se utiliza para resolver el problema?

• ¿Cómo hacerlo?

• ¿Quién lo va a hacer?

• ¿Qué se necesita para hacerlo?

• ¿Cuánto va a costar?

• ¿Cuánto va a durar?

• ¿Cómo se afectará la entidad a largo plazo, cuando las correcciones, adaptaciones y mejoras son solicitados por los usuarios de la entidad?

II.5 Desarrollo de software

El desarrollo de software incluye una serie de diferentes actividades durante las cuales tratamos de asegurarnos de que se van entregar las funciones necesarias. Podemos dividir las actividades de desarrollo de software en cuatro grandes categorías:

Análisis de requerimientos: Es una descripción de lo que el software tiene que hacer. Para la mayoría de los sistemas de software, van desde lo que el usuario necesita, una exposición de las necesidades y luego a una especificación precisa. Los requisitos de software expresan las necesidades y las limitaciones de un producto de software que contribuyen a la solución de algún problema del mundo real. Es una tarea difícil y es muy propenso a errores; esta etapa es la más importante en el desarrollo de Software, dado que si no se tiene claro que es lo que el cliente quiere y necesita se le entregará algo inservible.

Una propiedad esencial de todos los requerimientos de software es que sean verificables, ya sea como una característica individual, como requisito funcional o a nivel del sistema como un requerimientos no funcional. Hay diferentes tipos de requerimientos.

Requerimientos funcionales: Estás son las funciones que el software debe ejecutar. También se les conoce como capacidades o características del software. Éstos también se pueden definir como el conjunto finito de pasos de prueba que se puede escribir para validar su comportamiento.

Requerimientos no funcionales: Son conocidos como las restricciones y requisitos de calidad. Adicionalmente pueden ser clasificados de acuerdo a si son los requerimientos de rendimiento, requerimientos de mantenimiento, los requerimientos de seguridad, requerimientos de fiabilidad.

El estudio de la Ingeniería de Requisitos es una parte cada vez más importante de la Ingeniería del Software.

Diseño: Cubre la descripción estructural de alto nivel de la arquitectura del sistema, a través del diseño detallado de los componentes individuales del sistema y la implementación del sistema. También se puede definir como referencia a todas las actividades involucradas en la conceptualización, la elaboración, la implementación, la puesta en marcha de sistemas. El diseño de software por lo general implica la resolución de problemas y la planificación de una solución de software. Esto incluye tanto los componentes de bajo nivel, diseño de algoritmos de alto nivel y diseño de la arquitectura.

Verificación y Validación: La verificación y validación del software tiene como propósito ayudar a las organizaciones de desarrollo a construir software de calidad durante el ciclo de vida de software. Determinar si los procesos y productos de software satisfacen plenamente las necesidades de uso y del usuario según los requerimientos especificados. Esta determinación debe incluir actividades de adquisición, planeación, análisis, diseño, desarrollo, pruebas, instalación, operación y mantenimiento de procesos y productos de software. Verificación se

refiere a si se está construyendo bien aquellos artefactos que se especificaron en una etapa anterior. Se pregunta si se está construyendo bien. La Validación se refiere a si se satisfacen los requerimientos según se establecen en las especificaciones del cliente. Se pregunta si se construye lo correcto. Estas actividades son esenciales para garantizar si un proyecto de software va en la dirección correcta.

Existen varios tipos de pruebas conocidas, que son:

Estática: La evaluación estática es el único modo disponible de evaluación de artefactos para las primeras fases del proceso de desarrollo (requerimientos, análisis y diseño), cuando no existe código. En otras palabras son actividades de revisiones formales e informales, pláticas estructuradas, inspecciones, auditorías, etc.

Dinámica: Únicamente puede dar comienzo cuando finaliza la actividad de codificación, son todas las pruebas que se realizan una vez que el código ya esté presente.

Dentro de estas dos categorías se pueden diferenciar otros tipos de pruebas como los siguientes:

Pruebas de Unidad

Verifica cada módulo que conforma el sistema, este proceso se puede llevar acabo en paralelo para múltiples módulos. Se prueban los siguientes aspectos:

Interfaz del módulo: Para asegurar que la información fluye de forma adecuada hacia y desde el módulo que está siendo probado.

Estructura de datos: Para asegurar que los datos que se mantienen temporalmente conserven su integridad durante la ejecución del programa.

Bifurcaciones: Para asegurar que todas las sentencias del módulo se ejecutan por lo menos una vez.

Manejo de errores: Para asegurar que el sistema no se detenga bruscamente o se salga y mande errores controlados.

Prueba de camino básico

Es un método para ejecutar por lo menos una vez cada sentencia del programa, se basa en el manejo de grados y complejidad ciclomática.

Prueba de integración

Una vez que todos los módulos han sido probados por separado, hay que ponerlos juntos, para probar si la interacción entre módulos puede ocasionar que el sistema no funcione bien ya como un todo.

Pruebas de validación

Son pruebas que tratan de comprobar la conformidad de los requerimientos, la portabilidad, compatibilidad, recuperación de errores, documentación, facilidad de mantenimiento, etc.

Una vez validado el sistema, se pueden dar dos condiciones. La primera es que el software esté correcto y listo a la aceptación del cliente, la segunda es que haya errores o deficiencias, por lo cuál hay que resolverlas, lo que implica tiempo y costos.

Pruebas del sistema

Son pruebas hasta cierta forma realizadas para ver que sucede en situaciones externas al sistema, por ejemplo el hardware, fallos de energía eléctrica que puedan surgir, seguridad de acceso al sistema, resistencia a situaciones anómalas, etc.

Prueba de recuperación

Prueba del sistema que fuerza el fallo del software en muchas formas y verifica que la recuperación se lleve adecuadamente.

Prueba de seguridad

Son mecanismos de protección incorporados al sistema, lo protegen de penetraciones impropias. Se trata de hacer invulnerable al sistema desde cualquier ataque, frontal, retaguardia o por los flancos, externos o internos.

Prueba de resistencia

La prueba se ejecuta en un sistema de forma que demande recursos de cantidad, frecuencia o volúmenes anormales, recursos como: interrupciones, frecuencias de entradas, memoria ram, memoria virtual, excesivas búsquedas de datos, etc. Se intenta tirar el sistema.

Prueba de rendimiento

Las pruebas están diseñadas para probar el rendimiento de software en tiempo de ejecución dentro del contexto de un sistema integrado, posiblemente se requiera tanto de hardware como de software.

Pruebas de función

Verifica que el sistema integrado realiza las funciones especificadas en los requerimientos.

Pruebas de rendimiento

Compara con el rendimiento de los componentes integrados con los requerimientos no funcionales del sistema, que pueden incluir: seguridad, exactitud, velocidad y confiabilidad, dado que restringen la manera en que se realizan las funciones del sistema.

Pruebas de aceptación

Estas pruebas aseguran a los clientes que el sistema que pidieron es el sistema que se construyó para ellos.

Pruebas de instalación

Pruebas que permiten que los usuarios ensayen las funciones del sistema y documentan todos los problemas adicionales surgidos.

De acuerdo con el autor Lawrence Pfleeger, los pasos o secuencias pueden verse claramente en la figura 1.2.

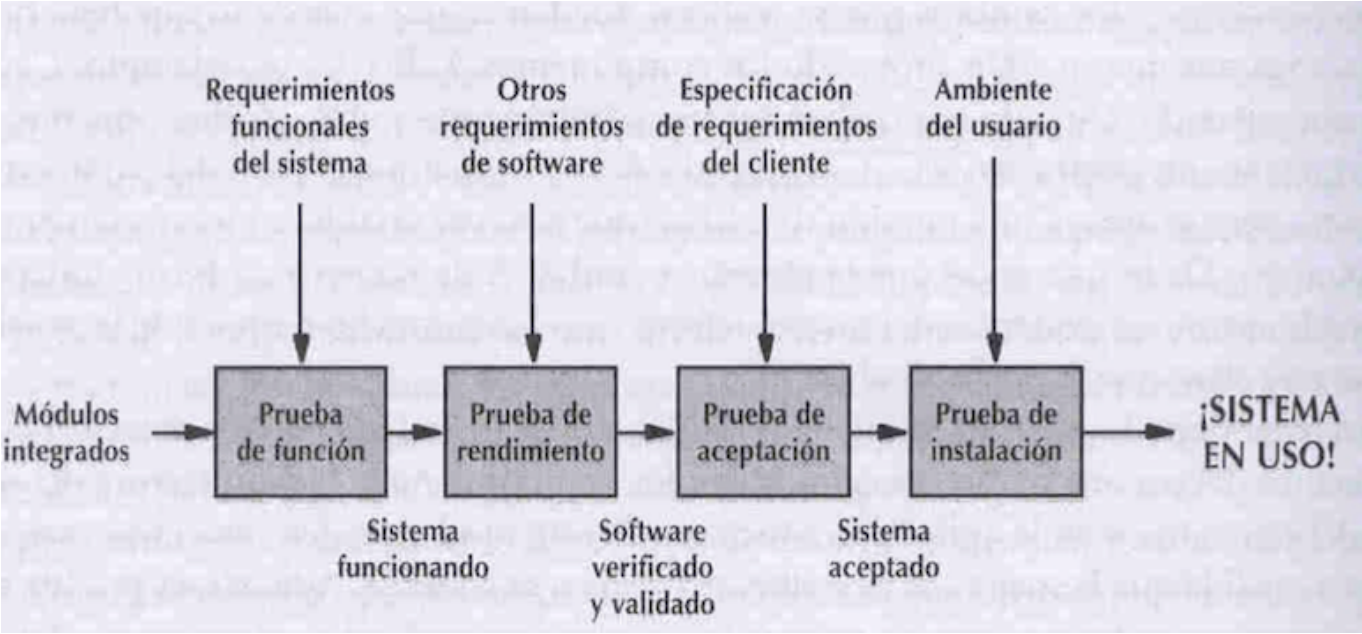


Figura 1.2

Las pruebas de software son una actividad que debe realizarse en cada etapa del proyecto, aunque son de mayor parte en la etapa de desarrollo. Las pruebas son un elemento crítico para la garantía de calidad del software y representa un último repaso de las especificaciones, del diseño y la codificación.

Mantenimiento: A medida que pasa el tiempo, las aplicaciones de software deben ser sometidas a procesos de modificación que extiendan su vida útil o mejoren sus características. Esta actividad garantiza que el sistema realiza un seguimiento de los cambios en su entorno

operativo en toda su vida de uso. Esto implica la corrección de errores que se descubren en el funcionamiento y la modificación del sistema para tener en cuenta los cambios en los requisitos del sistema. Así como todo cambio que vaya surgiendo, para poder asegurar el correcto funcionamiento del sistema a través del tiempo.

Corrección de bugs, adaptación a nuevos entornos tecnológicos o agregado de funcionalidad son algunas de las tareas que incluye el mantenimiento del software, una actividad que se repite periódicamente desde que empieza a utilizarse hasta su abandono definitivo.

Hay diferentes tipos de mantenimiento:

Correctivo: Tiene por objetivo localizar y eliminar los posibles defectos de los programas. Un defecto en un sistema es una característica del sistema con el potencial de provocar un fallo. Un fallo se produce cuando el comportamiento de un sistema difiere con respecto al comportamiento definido en la especificación. Y son todos aquellos cambios precisos para corregir estos errores dentro del producto de software para que funcione como debe.

Los errores en un sistema software pueden ser:

- Procesamiento (salidas incorrectas de un programa).
- Rendimiento (tiempo de respuesta demasiado alto).
- Programación (inconsistencias en el diseño).
- Documentación (inconsistencias entre la funcionalidad de un programa y el manual de usuario).

Adaptativo: Tal como lo dice su nombre, éste sirve para adaptaciones del software a cambios del entorno tecnológico donde se ejecuta como nuevos hardware, otros sistema de gestión de bases de datos, nuevos sistema operativo o nuevo framework. Este tipo de mantenimiento puede ser desde un pequeño retoque hasta una reescritura de todo el código.

Los cambios en el entorno de desarrollo del software pueden ser:

- En el entorno de los datos (p.e. cambiar sistema de archivos por Base de Datos relacional).

- En el entorno de los procesos (p.e. migración a plataforma con procesos distribuidos).

Este mantenimiento es cada vez más frecuente debido a la tendencia actual de la recurrente actualización de hardware y de sistemas operativos.

Perfectivo: Son las acciones llevadas a cabo para mejorar la calidad del software en cualquiera de sus aspectos: Reestructuración del código, optimización de rendimiento y eficiencia ya sea en flexibilidad, reusabilidad o para implementación de nuevos requisitos. También se le conoce como mantenimiento evolutivo.

Se divide en dos:

- Mantenimiento de ampliación: incorporación de nuevas funcionalidades.
- Mantenimiento de eficiencia: mejora de la eficiencia de ejecución.

Preventivo: Modificación del software para mejorar las propiedades de dicho software, calidad y mantenibilidad, sin alterar sus especificaciones funcionales. Incluir sentencias que comprueben la validez de los datos de entrada, reestructuración de los programas para aumentar su legibilidad o incluir nuevos comentarios. Este tipo de mantenimiento utiliza las técnicas de ingeniería inversa y reingeniería. El mantenimiento para la reutilización especializado en mejorar la reusabilidad del software se incluye en este tipo.

Dificultades del mantenimiento

Uno de los principales problemas con el cual se lidiaba a diario en la SEMARNAT, era la gran cantidad de código heredado y no se sabía en su totalidad que hacía, y por ende no querían intervenir en esas secciones. Esto también era porque se contaba con nula documentación, como un control de cambios, empeorando la situación.

La mayor parte del software en la actualidad está formado por código antiguo, se le conoce como código heredado”; esto es, código desarrollado hace tiempo, con técnicas y herramientas

en desuso y, peor aún porque las personas encargadas del desarrollo actualmente no se encargan de su mantenimiento. Además puede que incluso este código haya pasado varias actividades de mantenimiento; y por otra parte, el volver a reescribirlo no compensa por la carga financiera que supuso y la necesidad de su amortización.

II.6 Modelos de ciclo de vida del software

Se denomina ciclo de vida a toda la vida del software, describe el desarrollo de software, desde la fase inicial hasta la fase final. Aunque a veces, se habla de ciclo de desarrollo, para denominar al subconjunto del ciclo de vida que empieza en el análisis y finaliza la entrega del producto.

Un ciclo de vida establece el orden de las etapas del proceso de software y los criterios a tener en cuenta para poder pasar de una etapa a la siguiente.

Estos programas se originan en el hecho de que es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación. El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.

El ciclo de vida básico de un software consta de los siguientes procedimientos:

- **Definición de objetivos**: definir el resultado del proyecto y su papel en la estrategia global.
- **Análisis de los requisitos:** recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.
- **Diseño general:** requisitos generales de la arquitectura de la aplicación.
- **Diseño en detalle:** definición precisa de cada subconjunto de la aplicación.
- **Programación e implementación:** es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.
- **Prueba de unidad:** prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.

• **Integración:** Para garantizar que los diferentes módulos se integren con la aplicación. Éste es el propósito de la prueba de integración que está cuidadosamente documentada.

• **Validación:** Para garantizar que el software cumple con las especificaciones originales.

• **Documentación:** sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.

• **Mantenimiento:** para todos los procedimientos correctivos y las actualizaciones secundarias del software.

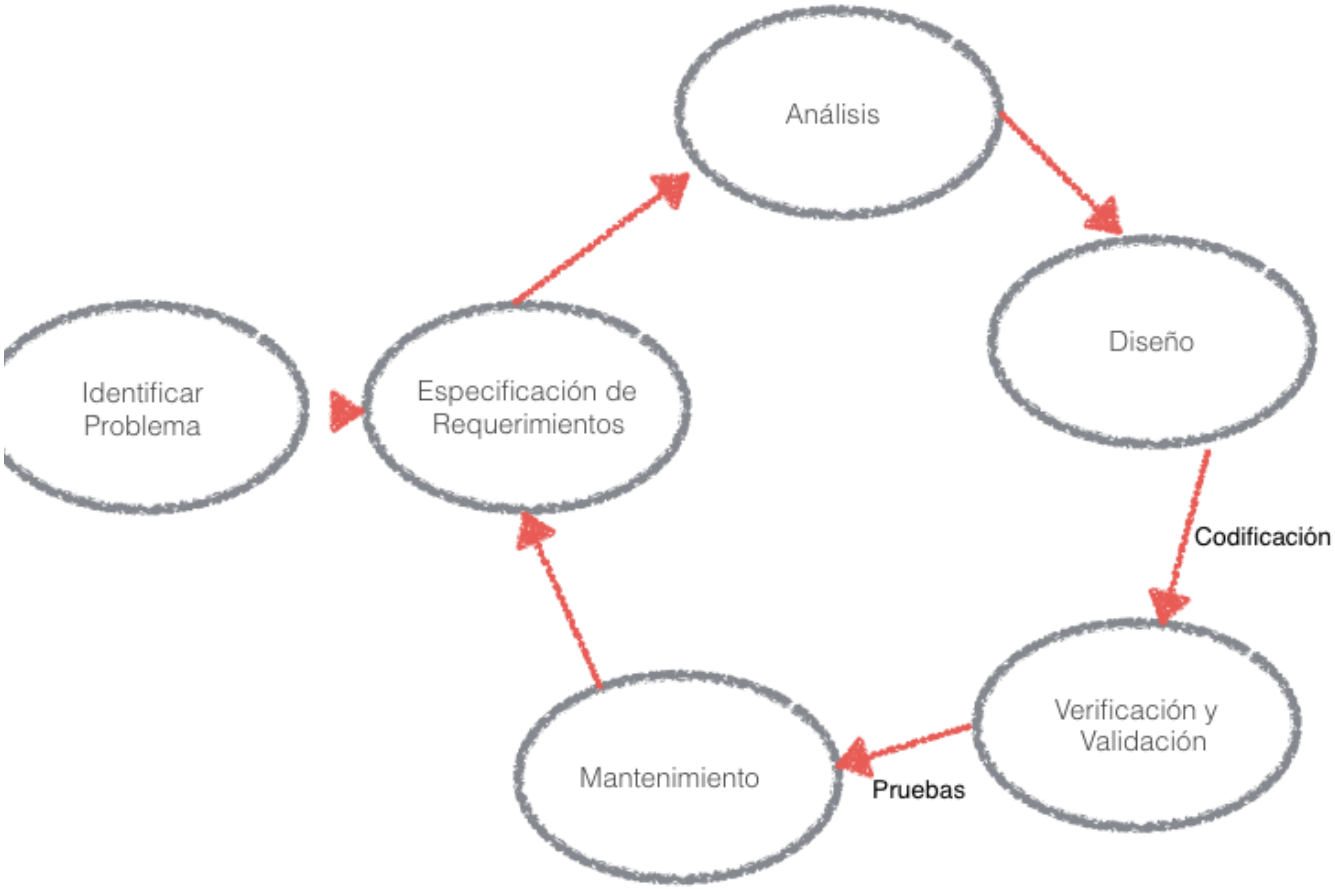


Figura 1.3

El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.

Modelo en Cascada

Este es un modelo secuencial donde cada actividad proporciona la entrada a la siguiente etapa en el proceso. Este proceso por lo general tiene una gran visibilidad debido a que al cierre de cada etapa se debe generar toda la documentación necesaria para esa etapa. Debido a la naturaleza secuencial de este modelo cada que haya un cambio éstos tienden a forzarnos a regresar a alguna etapa anterior y después seguir a través de cada una de las etapas de nuevo. En los primeros días de la producción de software que era el modelo estándar utilizado por la mayoría de los desarrolladores, es el modelo más antiguo, propuesto por Winston Royce en1970.

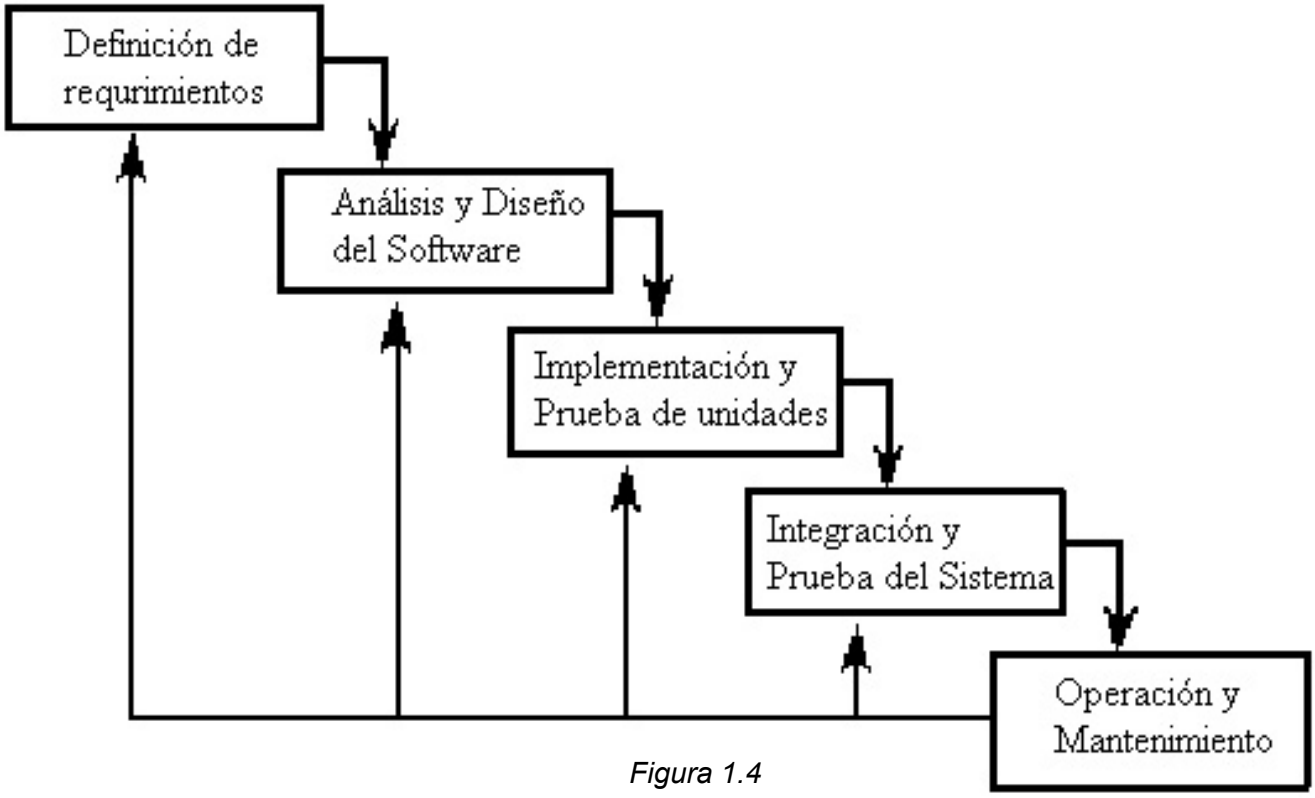


Figura 1.4

Modelo en Cascada

En la actualidad este modelo sigue siendo utilizado ampliamente debido a que es de muy fácil entendimiento e implementación. También refuerza buenos hábitos como definir antes que diseñar, diseñar antes que codificar, y funciona bien en productos maduros y en equipos de trabajo débiles.

Modelo V

Al modelo V se le considera una extensión del modelo en cascada porque es secuancial. La letra V es por Verificación y Validación. Se le conoce así porque en lugar de moverse hacia abajo en una forma lineal, los pasos del proceso se van hacia arriba después de la fase de codificación, para formar la típica V. Cada fase debe ser completada antes de que comience la fase siguiente. Las pruebas del producto están programadas en paralelo con su correspondiente fase de desarrollo. Se busca que con las pruebas en cada etapa se minimicen los riesgos del proyecto. En el modelo V se muestran las relaciones entre cada fase del ciclo de vida de desarrollo y de su fase asociado a la prueba. El eje horizontal representa el tiempo en el cual se realizan todas las fases.

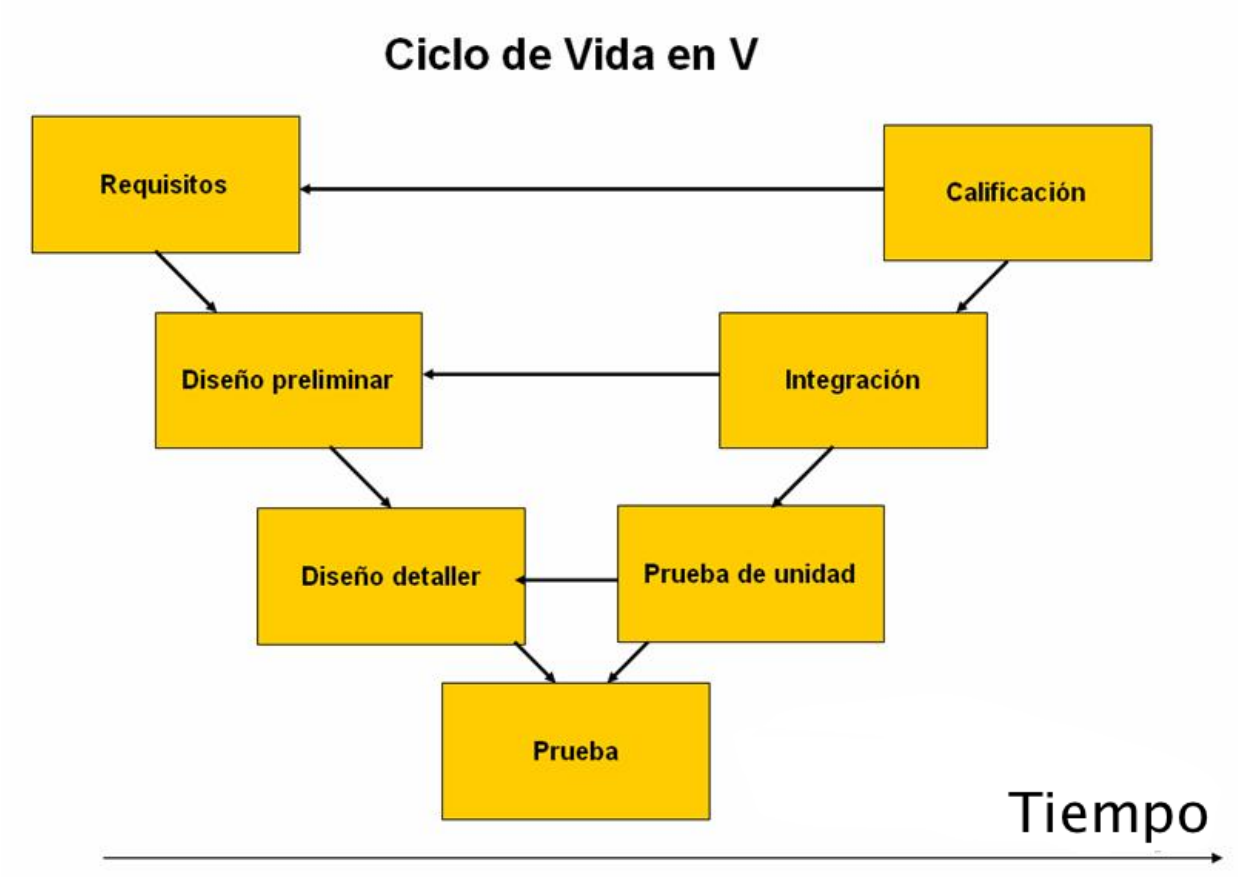


Figura 1.5

Ventajas del Modelo V:

Específica bien los roles de los distintos tipos de pruebas a realizar.

Hace explícito parte de la iteración y trabajo que hay que realizar.

Este método involucra chequeos de cada una de las etapas del modelo Cascada.

Es un método más robusto y completo que el método Cascada y produce software de mayor calidad que con el modelo Cascada.

Es un modelo sencillo y de fácil aprendizaje.

Involucra al usuario en las pruebas.

Desventajas de Modelo V:

Es difícil que el cliente exponga explícitamente todos los requisitos.

El cliente debe tener paciencia, ya que obtendrá el producto al final del ciclo de vida.

El modelo no contempla la posibilidad de retornar a etapas inmediatamente anteriores, cosa que en la realidad puede ocurrir.

Se pierde dinero, ya que si algún proceso fue mal desarrollado, éste debe ser revisado de nuevo, lo que puede traer como consecuencia un "RollBack" de todo un proceso.

Las pruebas pueden ser caras y a veces no lo suficientemente efectivas.

Este modelo es más usado que el modelo en cascada al ser más robusto, y permitir software de mayor calidad, debido al proceso de validación y verificación. El modelo V, es un estándar público y fue desarrollado por la Administración Federal Alemana. Se requiere de una alta confianza con los clientes para elegir el enfoque en forma de V para un proyecto. Debido a que no se producen prototipos, hay un muy alto riesgo de no cumplir con las expectativas del cliente.

Modelo en Espiral

El problema con los modelos de procesos de software es que no se enfrentan lo suficiente con la incertidumbre inherente a los procesos de software. Muchos proyectos importantes de software fallaron porque los riesgos del proyecto se despreciaron y nadie se preparó para algún imprevisto. Fue así como el factor de riesgo se empezó a considerar en el desarrollo de proyectos de software, esto se le agregó a las mejores características de los modelo en Cascada y fue así que surgió el Modelo en Espiral. La dirección de este nuevo modelo fue incorporar los puntos fuertes y evitar las dificultades de otros modelos desplazando el énfasis de administración hacia la evaluación y resolución del riesgo. De esta manera se permite tanto a los desarrolladores como a los clientes detener el proceso cuando se le considere conveniente.

En este modelo no se define en detalle el sistema completo al principio; los diseñadores deben definir e implementar solamente los rasgos de mayor prioridad. Con el conocimiento adquirido, podrían entonces volver atrás y definir e implementar más características en trozos más pequeños. El modelo en espiral es robusto porque la naturaleza iterativa nos permite planificar con flexibilidad, también es visible porque cada prototipo está completamente documentado.

Este modelo fue introducido por Barry Boehm en 1986.

El modelo Espiral define cuatro actividades principales en su ciclo de vida:

- **Planeación**: determinación de los objetivos, alternativas y limitaciones del proyecto.
- **Análisis de riesgo:** análisis de alternativas e identificación y solución de riesgos.
- **Ingeniería:** desarrollo y testeo del producto.
- **Evaluación del cliente:** tasación de los resultados de la ingeniería.

El modelo está representado por una espiral dividida en cuatro cuadrantes, cada una de las cuales representa una de las actividades arriba mencionadas.

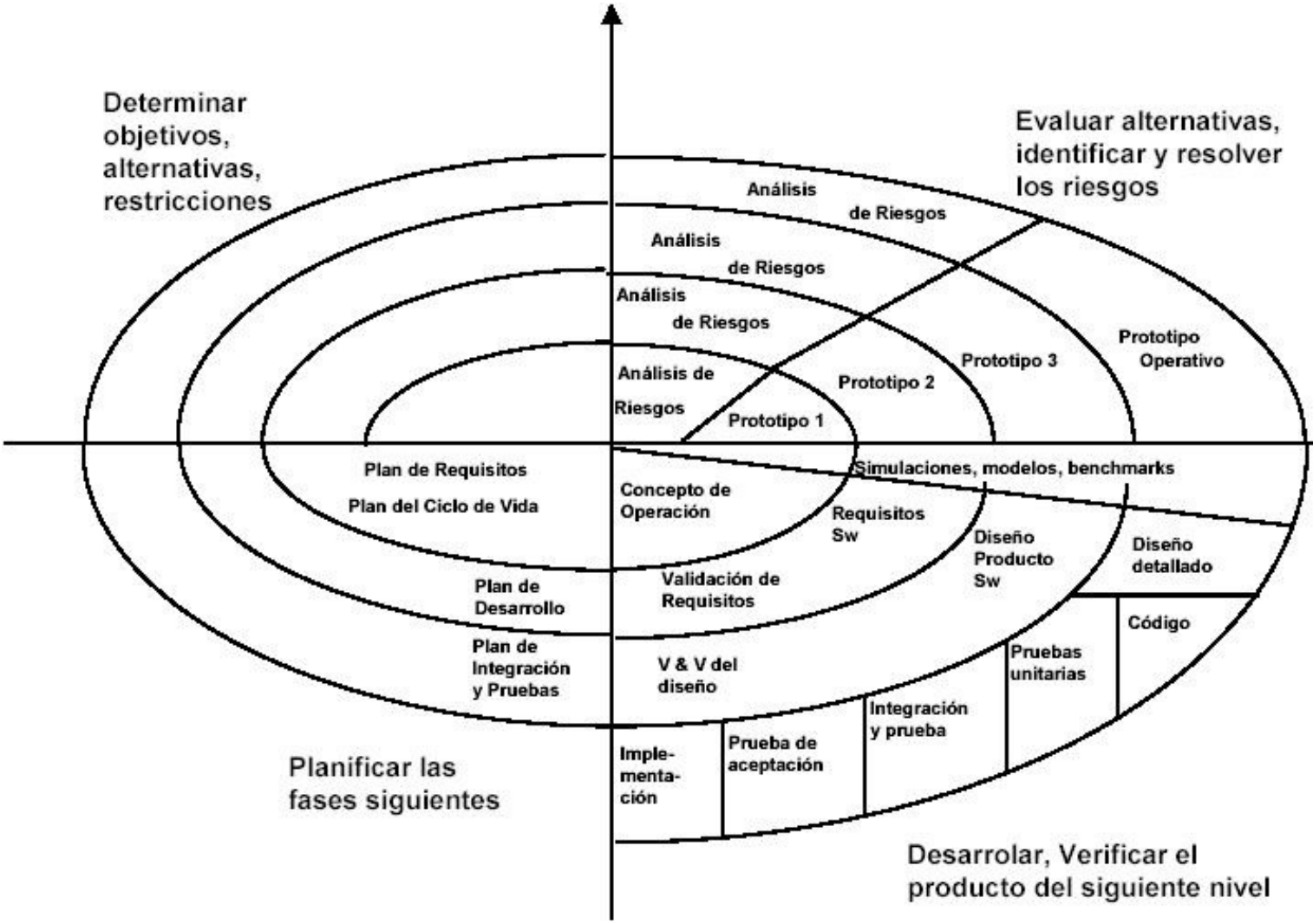


Figura 1.6
Modelo en Espiral

Ventajas del Modelo en Espiral

• Evita las dificultades de los modelos existentes a través de un acercamiento conducido por el riesgo.

• Intenta eliminar errores en las fases tempranas.

• Es el mismo modelo para el desarrollo y el mantenimiento.

• Provee mecanismos para la aseguración de la calidad del software.

• La reevaluación después de cada fase permite cambios en las percepciones de los usuarios, avances tecnológicos o perspectivas financieras.

Desventajas del Modelo en Espiral

- Falta un proceso de guía explícito para determinar objetivos, limitaciones y alternativas.
- Provee más flexibilidad que la conveniente para la mayoría de las aplicaciones.
- La pericia de tasación del riesgo no es una tarea fácil. El autor declara que es necesaria mucha experiencia en proyectos de software para realizar esta tarea exitosamente.

Modelo de Prototipos

Este modelo permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador, el usuario y el cliente estén de acuerdo en lo que se necesita, así como también la solución que se propone para dicha necesidad y de esta forma minimizar el riesgo y la incertidumbre en el desarrollo. También es conocido como evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos, pues a partir de que éste sea aprobado se puede iniciar el verdadero desarrollo del software. Un prototipo es una representación o modelo del sistema a desarrollar que a diferencia de un modelo de simulación, incorpora componentes del producto real, éste será una representación del sistema, aunque no es un sistema completo, posee las características del sistema final o parte de ellas. Un prototipo tiene un funcionamiento limitado en cuenta a capacidades, confiabilidad o eficiencia. este modelo se encarga del desarrollo de diseños para que éstos sean analizados y prescindir de ellos a medida que se adhieran nuevas especificaciones, es ideal para medir el alcance del proyecto. Este modelo principalmente se lo aplica cuando un cliente define un conjunto de objetivos generales para el software a desarrollarse sin delimitar detalladamente los requisitos de entrada procesamiento y salida, es decir cuando el responsable no está seguro de la eficacia de un algoritmo, de la adaptabilidad del sistema o de la forma en que interactúan el hombre y la máquina. Este modelo se usa

principalmente cuando se quiere ayudar al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos.

Se divide en 5 etapas:

- Plan rápido
- Modelado, diseño rápido
- Construcción del prototipo
- Desarrollo, entrega y retroalimentación
- Comunicación
- Entrega del desarrollo final

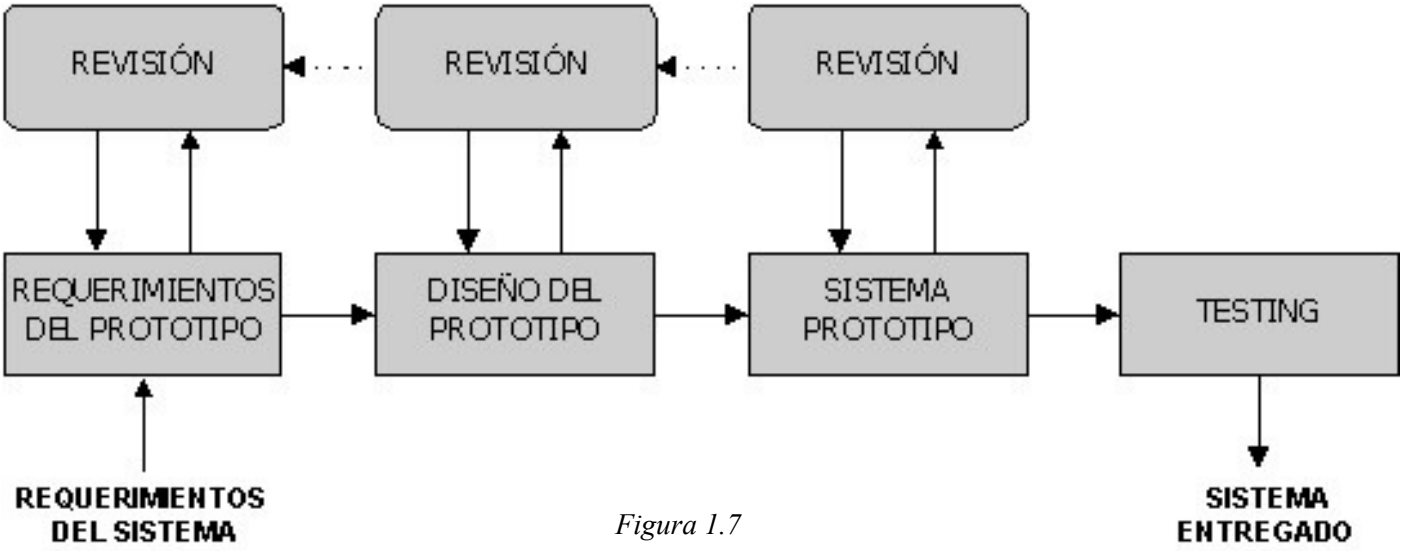


Figura 1.7

Modelo en Prototipos

Este modelo resulta muy útil cuando el cliente no identifica los requisitos detallados, ni tiene bien visualizado qué es lo que quiere. También cuando el responsable del desarrollo no está seguro de la eficiencia de un algoritmo, sistema operativo o de la interface hombre-máquina o parte de algún requerimiento.

II.7 Mejores prácticas para el desarrollo de software

Existen diferentes procesos en el tema Ingeniería de Software, que tienen como objetivo presentar diferentes técnicas que consisten en la combinación de procedimientos que permiten guiar el diseño y el desarrollo de sistemas de software a un producto final de calidad.

Una práctica es un método o técnica utilizada para llevar a cabo una parte de un proceso y describe cómo se realiza. Las mejores prácticas son aquellas técnicas o métodos que permiten incrementar la satisfacción del cliente al incorporar su uso en nuestro proceso. No existe un modelo de proceso de software estándar que se adecue para cualquier tipo de requerimiento. Se denominan mejores prácticas porque se han identificado como el factor común que caracteriza a organizaciones exitosas de software.

La calidad del software se puede definir como el conjunto de características mensurables que debe cumplir el software, para que éste realice lo que debe hacer, fácil de usar, que se entregue en presupuesto y en tiempo estimado, que el software no falle, sea rápido, etc., en pocas palabras que satisfaga plenamente o supere las expectativas del usuario.

Las mejores prácticas siguen los siguientes líneamientos:

Desarrollar software iterativamente

En función de la cada vez mayor complejidad solicitada para los sistemas de software, ya no es posible trabajar secuencialmente: definir primero el problema completo, luego diseñar toda la solución, construir el software y finalmente, probar el producto. Es necesario un enfoque iterativo, que permita una comprensión creciente del problema a través de refinamientos sucesivos, llegando a una solución efectiva luego de múltiples iteraciones acotadas en complejidad.

A través de las iteraciones que generan versiones ejecutables, se logra detectar en forma temprana los desajustes e inconsistencias entre los requerimientos, el diseño, el desarrollo y la implementación del sistema, manteniendo al equipo de desarrollo concentrado en producir resultados.

Administrar los requerimientos

Los requerimientos son las condiciones o capacidades que el sistema debe conformar. La administración de requerimientos es un enfoque sistemático para hallar, documentar, organizar y monitorear los requerimientos cambiantes de un sistema.

La administración de requerimientos permite:

- Que las comunicaciones estén basadas en requerimientos claramente definidos
- Que los requerimientos puedan ser priorizados, filtrados y monitoreados
- Que sea posible realizar evaluaciones objetivas de funcionalidad y desempeño
- Que las inconsistencias se detecten más fácilmente

Utilizar arquitecturas basadas en componentes

El proceso de software debe concentrarse en el desarrollo temprano de una arquitectura robusta ejecutable, antes de comprometer recursos para el desarrollo en gran escala. Se describe como diseñar una arquitectura flexible, que se acomode a los cambios, comprensible intuitivamente y promueve una más efectiva reutilización de software. Soporta el desarrollo de software basado en componentes: módulos no triviales que completan una función clara.

Modelar software visualmente

Modelar software visualmente para capturar la estructura y comportamiento de arquitecturas y componentes. Las abstracciones visuales ayudan a comunicar diferentes aspectos del software; comprender los requerimientos, ver como los elementos del sistema se relacionan

entre sí, mantener la consistencia entre diseño e implementación y promover una comunicación precisa. El estándar UML(Lenguaje de Modelado Unificado), creado por Rational Software, es el cimiento para un modelado visual exitoso.

Verificar la calidad de software

Es necesario evaluar la calidad de un sistema respecto de sus requerimientos de funcionalidad, confiabilidad y desempeño. La actividad fundamental son las pruebas, que permite encontrar las fallas antes de la puesta en producción. El aseguramiento de la calidad se construye dentro del proceso, en todas las actividades, involucrando a todos los participantes, utilizando medidas y criterios objetivos, permitiendo así detectar e identificar los defectos en forma temprana.

Controlar los cambios al software

La capacidad de administrar los cambios es esencial en ambientes en los cuales el cambio es inevitable. Se debe describir como controlar, rastrear y monitorear los cambios para permitir un desarrollo iterativo exitoso. Es también una guía para establecer espacios de trabajo seguros para cada desarrollador, suministrando el aislamiento de los cambios hechos en otros espacios de trabajo y controlando los cambios de todos los elementos de software como modelos, código, documentos, etc. Describe como automatizar la integración y administrar la conformación de versiones.

En mi estancia en la SEMARNAT decidí usar el modelo MoProSoft dado que ya había trabajado con ese modelo y tenía mucha documentación que me serviría de apoyo para el proyecto.

II.8 Modelo de Procesos para la Industria del Software (MoProSoft)

Se define como un modelo de procesos para el desarrollo y mantenimiento de software dirigido a la pequeña y mediana industria y a las áreas internas de desarrollo de software. Es un modelo mexicano de mejores prácticas para la mejora y evaluación de los procesos de desarrollo y mantenimiento de sistemas y productos de software. Este modelo fue desarrollado a solicitud de la Secretaría de Economía para servir de base a la Norma Mexicana para la Industria de Desarrollo y Mantenimiento de Software bajo el convenio con la Facultad de Ciencias de la Universidad Nacional Autónoma de México.

MoProSoft está dividido en 9 procesos, llamados también prácticas, organizados por categorías de acuerdo a sus respectivas áreas de aplicación. Las categorías de procesos coinciden con los tres niveles básicos de la estructura de una organización: Alta Dirección, Gestión y Operación.

Procesos de Dirección

Gestión de Negocios:

Su propósito la razón de ser de la organización, sus objetivos y las condiciones para lograrlos, para lo cual es necesario considerar las necesidades de los clientes, así como evaluar los resultados para poder proponer cambios que permitan la mejora continua.

Procesos de Gestión

Gestión de Proyectos:

Generar proyectos que contribuyan al cumplimiento de los objetivos y estrategias de la organización

Gestión de Procesos:

Establece procesos que apoyen a las estrategias de la organización así como actividades de mejora en los mismos.

Gestión de Recursos:

Consigue y provee a la organización de los recursos para desarrollar las actividades de acuerdo a las necesidades de cada proceso y proyecto.

Recursos Humanos y Ambiente de Trabajo:

Provee y administra los recursos humanos y busca mantener un ambiente de trabajo adecuado en la organización.

Bienes, Servicios e Infraestructura:

Provee, administra y mantiene los recursos de la organización para que la misma pueda operar.

Conocimiento de la Organización:

Provee, administra y mantiene las herramientas y repositorios que conforman la base de conocimiento de la organización.

Procesos de Operación

Administración de Proyectos Específicos:

Administra los proyectos internos y externos en base a los planes de cada uno, genera acciones correctivas.

Desarrollo y Mantenimiento de Software:

Genera los productos a través del ciclo de vida de desarrollo del software buscando satisfacer las necesidades del cliente.

Este estándar fue el que propuse en la SEMARNAT. No pude implementar todo el modelo en la Secretaría , sin embargo si pude tener impacto en el desarrollo y mantenimiento de software. Implementando documentos como la de especificación de requerimientos y el control de cambios, documentos que se mostraran más adelante.

III. Resultados obtenidos

A continuación se presentan los resultados del trabajo realizado como servicio social en el Secretaría de Medio Ambiente y Recursos Naturales, en forma concreta, para garantizar el cumplimiento del objetivo del programa:

Me pude percatar que en el software de la Cédula de Operación Anual, al no contar con ningún protocolo de desarrollo de software o de buenas prácticas, tenía muchísimos problemas con todo su sistema a la hora de modificarlo, y peor aún, ocasionaba más problemas cuando querían hacer cambios para mejorarlo o agregar requerimientos nuevos. Otro gran problemática con el uso de este software es que no contaba con documentación de como usarse, un manual de usuario. El programa es para uso interno, y a veces algunas empresas lo solicitaban para facilitar la elaboración de sus reportes. Muchas veces las personas que usaban la herramienta no contaban con preparación técnica, o amplios conocimientos técnicos para operarlo de manera correcta; y a decir verdad, la herramienta si era complicada de usar. Durante mi estancia, también me percaté que en el área de soporte técnico recibían, en promedio, 30 llamadas al día y todas eran para preguntar por qué no funcionaba de manera correcta el programa, cómo lo instalaban, qué entorno debían tener configurado en su computadora para que el programa operará de forma adecuada, etc.

Fue entonces, que deduje que podía tener iniciativa y proponer cambios en la manera que trabajan. Presenté mi propuesta, primero deberían hacer uso de buenas prácticas de software, como mencioné anteriormente, sugerí MoProSoft porque es ágil en equipos de trabajos pequeños y ya contaba con formatos para implementar. El desarrollo ya estaba hecho, pero podían beneficiarse de esto, como con un control de cambios.

Donde realmente pude tener impacto, fue en la documentación de la herramienta de la COA. Como había mencionado, no se contaba con documentación, no había manual de uso. Fue ahí

cuando propuse la elaboración de un manual en línea, para que los usuarios pudieran consultarlo cuando tuvieran alguna duda operativa; antes de realizar el manual en línea, se necesitaba de un manual de usuario impreso, para después hacer uno que pudieran consultar desde la intranet.

III.1 Importancia de la documentación

La documentación de programas de software es el conjunto de información que nos dice qué hacen los sistemas, cómo lo hacen y para quién lo hacen. La documentación consiste en un material que explica las características técnicas y la operación de un sistema. Es esencial para proporcionar el entendimiento de un sistema a quien lo vaya a usar, así como para darle un correcto mantenimiento.

La documentación de los programas es un aspecto sumamente importante, tanto en el desarrollo de la aplicación como en el mantenimiento de la misma. Mucha gente no hace este parte del desarrollo y no se da cuenta de que pierde la posibilidad de la reutilización de parte del programa en otras aplicaciones,y esto sin necesidad de conocer a la perfección todo el código. La documentación de un programa empieza a la vez que la construcción del mismo y finaliza justo antes de la entrega del programa o aplicación al cliente, así mismo, la documentación que se entrega al cliente tendrá que coincidir con la versión final de los programas que componen la aplicación. Una vez concluido el programa, los documentos que se deben entregar son una guía técnica, una guía de uso y de instalación.

Tipos de documentación

La documentación que se entrega al cliente se divide claramente en dos categorías, interna y externa:

Interna: Es aquella que se crea en el mismo código, ya puede ser en forma de comentarios o de archivos de información dentro de la aplicación.

Externa: Es aquella que se escribe en cuadernos o libros, totalmente ajena a la aplicación en si. Dentro de esta categoría también se encuentra la ayuda electrónica.

La guía técnica

En la guía técnica o manual técnico se reflejan el diseño del proyecto, la codificación de la aplicación y las pruebas realizadas para su correcto funcionamiento. Generalmente este documento esta diseñado para personas con conocimientos técnicos, generalmente está orientado a desarrolladores.

El principal objetivo es el de facilitar el desarrollo, corrección y futuro mantenimiento de la aplicación de una forma rápida y fácil.

Esta guía esta compuesta por tres apartados claramente diferenciados:

Cuaderno de carga: Es donde queda reflejada la solución o diseño de la aplicación.

Esta parte de la guía es únicamente destinada a los programadores. Debe estar realizado de tal forma que permita la división del trabajo

Programa fuente: Es donde se incluye la codificación realizada por los programadores. Este documento puede tener, a su vez, otra documentación para su mejor comprensión y puede ser de gran ayuda para el mantenimiento o desarrollo mejorado de la aplicación. Este documento debe tener una gran claridad en su escritura para su fácil comprensión.

Pruebas: Es el documento donde se especifican el tipo de pruebas realizadas a lo largo de todo el proyecto y los resultados obtenidos.

Manual de uso

Es lo que comúnmente llamamos el manual del usuario. Contiene la información necesaria para que los usuarios utilicen correctamente la aplicación. Este documento se hace desde la

guía técnica pero se suprimen los tecnicismos y se presenta de forma que sea entendible para el usuario que no sea experto en computación. Un punto a tener en cuenta en su creación es que no debe hacer referencia a ningún apartado de la guía técnica y en el caso de que se haga uso de algún tecnicismo debe ir acompañado de un glosario al final de la misma para su fácil comprensión.

La guía de instalación

Es la guía que contiene la información necesaria para implementar dicha aplicación.

Dentro de este documento se encuentran las instrucciones para la puesta en marcha del sistema y las normas de utilización del mismo. Dentro de las normas de utilización se incluyen también las normas de seguridad, tanto las físicas como las referentes al acceso a la información.

En el manual que realicé, empecé por la guía de instalación, seguido del manual de uso. Quedaron los dos documentos incrustados en uno solo.

III.2 Implementación de Buenas Prácticas

Me apoyé de los Procesos de Operación de MoProSoft, específicamente en el Desarrollo y Mantenimiento de Software. Modifiqué uno de los formatos de MoProSoft con los que contaba para poder adaptarlo a la SEMARNAT.

Me centré en dos documentos que sabía que podían tener un impacto inmediato. Formato de Solicitud de Cambio y el Control de Cambios.

Como lo mencioné anteriormente, el software de la Cédula de Operación Anual, sufría cambios constantemente. Al ser un equipo de trabajo reducido, los integrantes no sentían la necesidad de documentar o mencionar que se le harían cambios al código, simplemente se hacían.

El principal objetivo del Control de Cambios es la evaluación y planificación del proceso de cambio para asegurar que, si éste se lleva a cabo, se haga de la forma más eficiente, siguiendo los procedimientos establecidos, asegurando así la calidad del software.

Estos cambios pueden ocurrir durante las fases de elaboración, construcción o mantenimiento de cualquier aplicación o proyecto, es decir en cualquier tiempo antes o después de su implementación.

En la SEMARNAT el origen de los cambios en la aplicación eran principalmente nuevas necesidades de los usuarios del sistema y la redefinición de los requerimientos del sistema.

Para poder elaborar una correcta gestión de cambios, le expliqué a mi equipo de trabajo que se debían seguir los siguientes pasos, junto con el llenado del formato:

1) Identificar necesidad del cambio

2) Realizar solicitud de cambio

3) Anexar nuevo requerimiento en el acta constitutiva del proyecto

4) Realizar cambio y realizar pruebas técnicas

La solicitud debe ir firmado por el líder de proyecto, el encargado de llenar la solicitud y el encargado de realizar los cambios.

El formato de Solicitud de Cambio es el siguiente:

SEMARNAT SECRETARÍA DE MEDIO AMBIENTE Y RECURSOS NATURALES	**Secretaria de Medio Ambiente y Recursos Naturales** **Software de la Cédula de Operación Anual**	
	Formato de Solicitud de Cambio	Versión 1.0
		26/09/13

Formato Solicitud de Cambio	
Nombre de la aplicación:	[Nombre de la aplicación o proyecto]
Fecha:	[Fecha en la cual se solicita el cambio]
Líder de Proyecto:	[Nombre del líder de proyecto]
Responsable de solicitud:	[Nombre de quién llena la solicitud]
Responsable de actividad:	[Nombre del responsable a realizar cambios]

Causas o Beneficios

Descripción del desarrollo

Procedimientos, módulos o archivos afectados

Firma responsable de solicitud	**Firma Responsable de actividad**

Figura 3.1

Ya que estuviera aprobada la solicitud se procedía a realizar el cambio. Y una vez realizado el cambio, éste se debía reportar en el formato de Control de Cambios para evitar problemas en la gestión y funcionamiento de la aplicación. El formato que realicé de Control de Cambios es el siguiente:

SEMARNAT SECRETARÍA DE MEDIO AMBIENTE Y RECURSOS NATURALES	**Secretaria de Medio Ambiente y Recursos Naturales** **Manual de usuario en linea**	
	Control de Cambios	Versión 1.0
		26/09/13

Fecha de cambio	Versión	Responsable	Descripción del cambio	Firma del responsable
dd/Mmm/aa	<x.y>	<Nombre>	<Detalle>	

Figura 3.2

III.3 Elaboración del Manual de Usuario

Tal como lo indiqué, empecé por la guía de instalación. Aquí se le mostraba a los usuarios todos los requisitos de hardware necesarios para poder descargar, instalar y ejecutar el software de la COA.

El manual de usuario consta de 50 páginas, y en su contenido se encuentra toda la información necesaria para descargar, configurar y usar el software.

La tabla de contenido del manual quedó de la siguiente manera:

1) Requisitos y descarga de la COA
2) Instalación de la COA
3) Características de la COA
4) Uso del formato electrónico de la COA
5) Datos de Registro

6) Sección I. Información técnica general

7) Sección II. Registro de emisiones de contaminantes de la atmosfera.

8) Sección III. Registro de descargas

9) Sección IV. Registro de la generación, manejo y transferencia de residuos peligrosos

10) Sección V. Emisiones y transferencia de contaminantes

11) Herramientas COA

12) Atención a dudas de la COA

Tabla de contenido

Figura 3.3

El índice del manual se puede observar en la imagen siguiente:

El software no requiere muchos recursos, pero es necesario saber que se necesitaba para su correcta ejecución, ya que de no ser así causa inestabilidad en el software. Esté fue uno de los principales motivos para realizar el manual, que el usuario pudiera tener una guía paso a paso de cómo configurar su equipo y la aplicación para su correcto funcionamiento.

En la siguiente imagen se puede observar como quedaron en el manual los requisitos mínimos para poder instalar el software de la Cédula de Operación Anual.

REQUISITOS Y DESCARGA DE LA COA

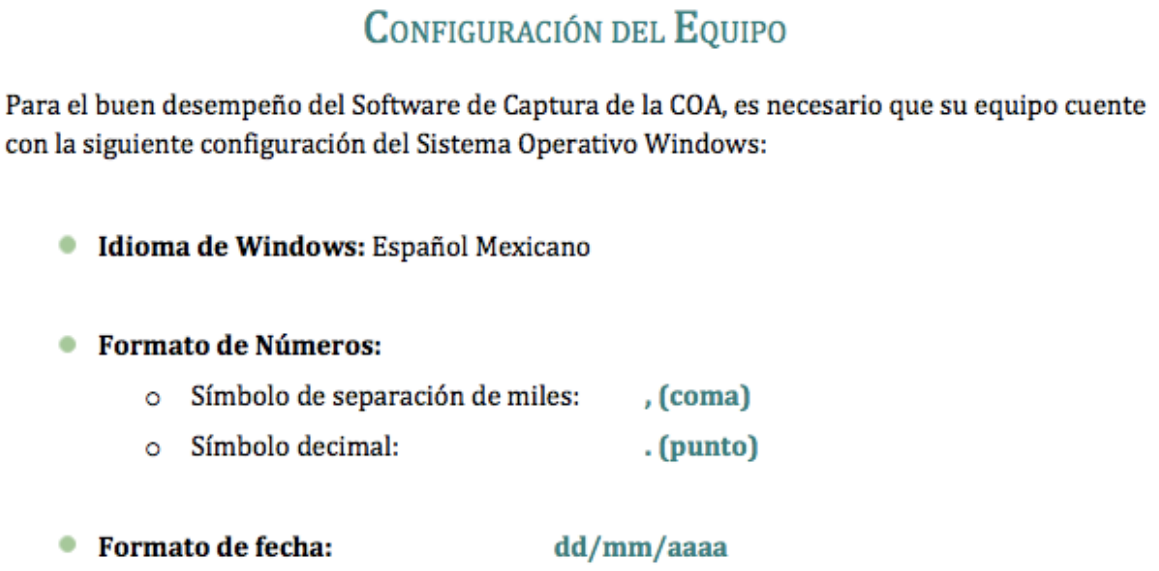
REQUISITOS DEL EQUIPO

El formato electrónico de la Cédula de Operación Anual (COA) opera en ambiente Windows XP, Vista, Windows 7 u 8, los requisitos del equipo, para el funcionamiento adecuado de la herramienta son los siguientes:

- Espacio en disco duro de 100Mb o superior.
- Memoria RAM de 512 Mb o superior
- Instalado el Office 2007 (específicamente Access 2007).
- Un Sistema Operativo (mencionados arriba).
- Conexión a internet.
- JAVA 6 o superior

Figura 3.4

Uno de los errores más comunes que se recibían en el área de soporte técnico, era que no podían instalar el software o no funcionaba correctamente, y era por no tener el equipo configurado de manera correcta. La configuración es sencilla y aquí se puede observar:



CONFIGURACIÓN DEL EQUIPO

Para el buen desempeño del Software de Captura de la COA, es necesario que su equipo cuente con la siguiente configuración del Sistema Operativo Windows:

- **Idioma de Windows:** Español Mexicano
- **Formato de Números:**
  - Símbolo de separación de miles: **, (coma)**
  - Símbolo decimal: **. (punto)**
- **Formato de fecha:** **dd/mm/aaaa**

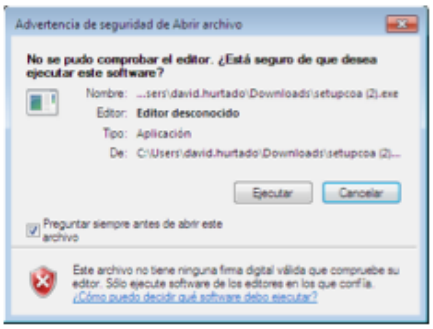
Figura 3.5

Este es un claro ejemplo de por qué la correcta configuración del equipo es muy importante, sin importar que tan sencillo sea, como en este caso que si no se tenía el formato de fecha como indicado en la imagen anterior, el sistema arrojaba resultados “basura”, o simplemente marcaba error al instalarse.

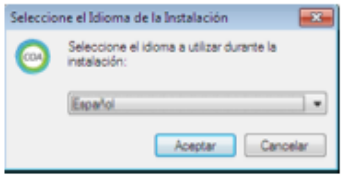
Después de poner los requisitos de hardware y la configuración del sistema continué con los pasos necesarios para poder instalar el software de la COA. En al manual se indican todos los pasos a seguir para instalar el software como se puede mostrar en las siguientes imágenes:

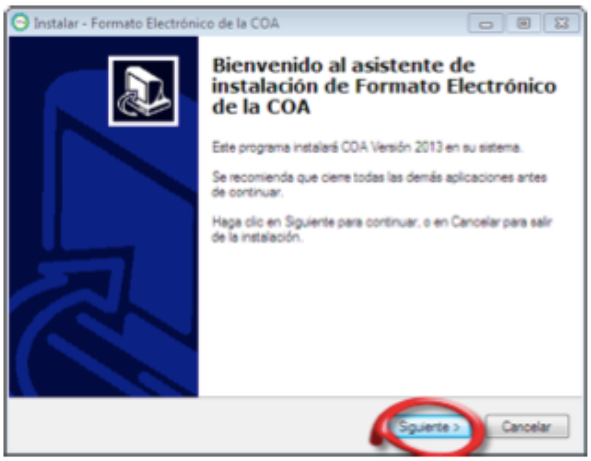
INSTALACIÓN DE LA COA

Ejecutamos nuestro instalador, se va a desplegar una ventana de seguridad de Windows, en la cual deberás permitir la ejecución del software.



Después seleccionamos el lenguaje de instalación de nuestro software en español.



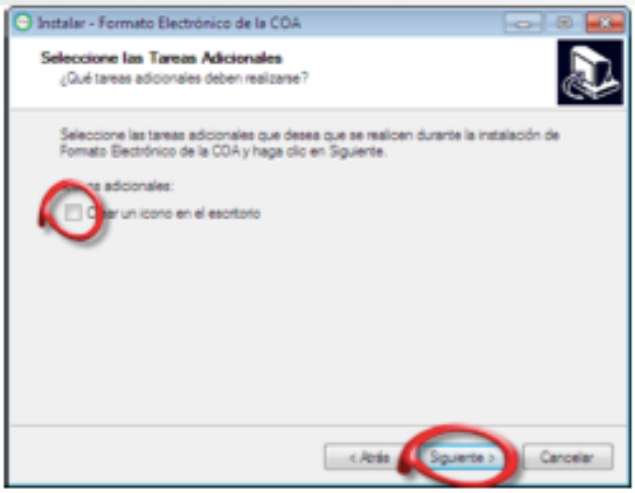


Una vez terminados los pasos anteriores, la ventana del asistente de instalación será mostrada en tu pantalla, has caso a las indicaciones que te dé el asistente y dale clic en ***Siguiente.***

8

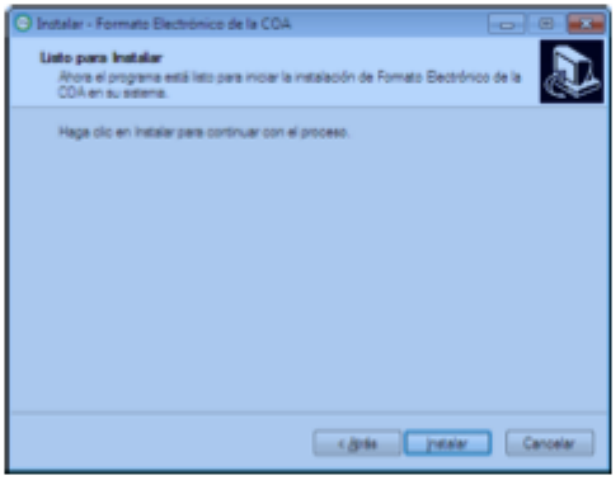
Figura 3.6

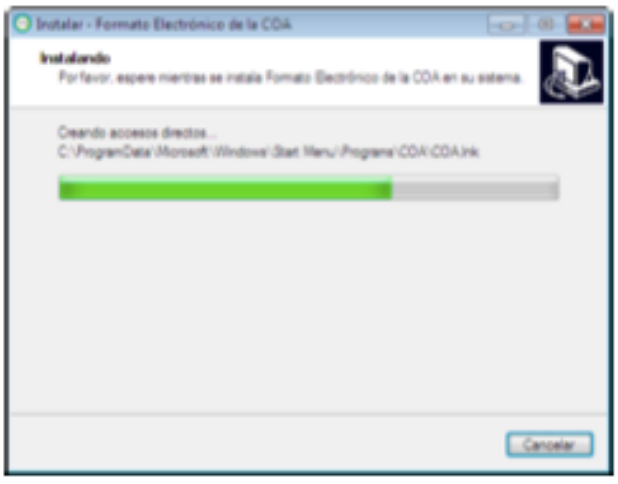
En el manual hay imágenes de todos los pasos por los cuales el usuario tendrá que pasar para instalar la aplicación, y cada imagen con su respectiva explicación.



En esta ventana podrás seleccionar si quieres un icono en el escritorio, o no. Una vez que has elegido darás clic en ***Siguiente*** para continuar con la instalación.

Para comenzar la instalación, sólo deberás dar clic en ***Instalar***, pero si quieres cambiar la configuración, puedes regresarte con el botón "Atrás".





Ahora sólo deberás esperar a que la extracción de la información concluya, para comenzar a utilizar tu herramienta de la COA.

Figura 3.7

Así se muestra el paso final en la guía de instalación en el manual:

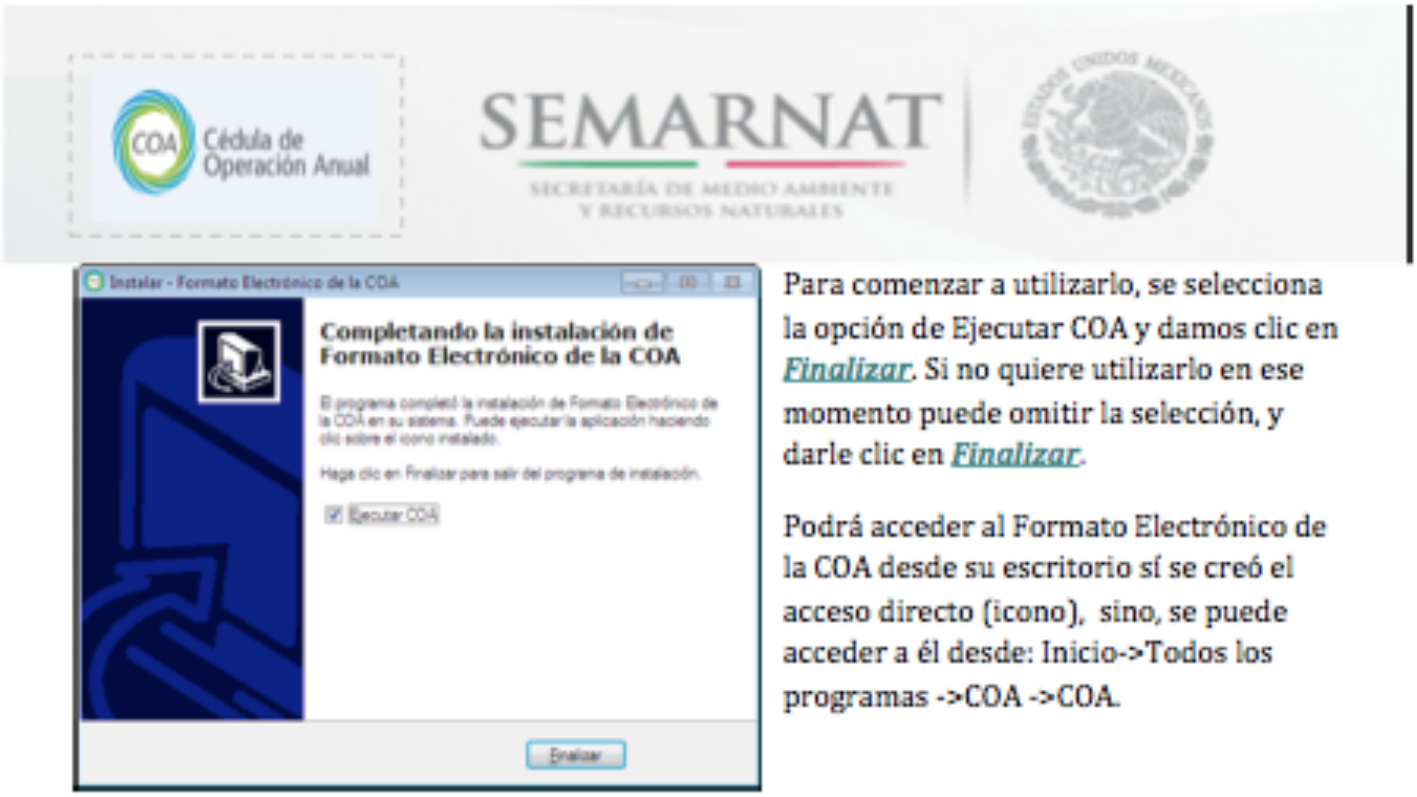
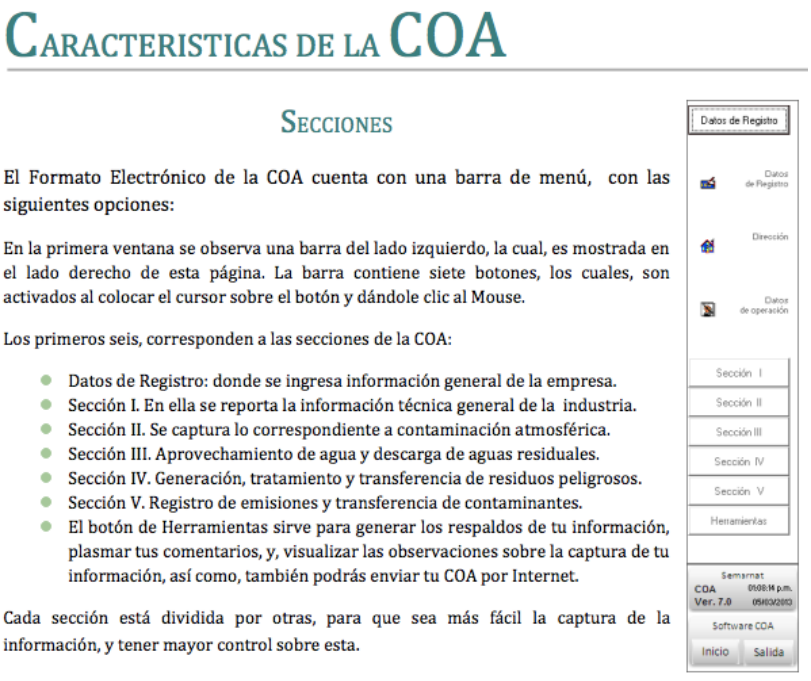


Figura 3.8

El software cuenta con seis secciones, las cuales están todas descritas en el manual. En cada sección se desglosa cada uno de los apartados que hay que llenar, así como qué hace cada uno de los botones que hay en el software. A continuación se ejemplifica como desarrollé una de las secciones en el manual.



CARACTERISTICAS DE LA COA

SECCIONES

El Formato Electrónico de la COA cuenta con una barra de menú, con las siguientes opciones:

En la primera ventana se observa una barra del lado izquierdo, la cual, es mostrada en el lado derecho de esta página. La barra contiene siete botones, los cuales, son activados al colocar el cursor sobre el botón y dándole clic al Mouse.

Los primeros seis, corresponden a las secciones de la COA:

- Datos de Registro: donde se ingresa información general de la empresa.
- Sección I. En ella se reporta la información técnica general de la industria.
- Sección II. Se captura lo correspondiente a contaminación atmosférica.
- Sección III. Aprovechamiento de agua y descarga de aguas residuales.
- Sección IV. Generación, tratamiento y transferencia de residuos peligrosos.
- Sección V. Registro de emisiones y transferencia de contaminantes.
- El botón de Herramientas sirve para generar los respaldos de tu información, plasmar tus comentarios, y, visualizar las observaciones sobre la captura de tu información, así como, también podrás enviar tu COA por Internet.

Cada sección está dividida por otras, para que sea más fácil la captura de la información, y tener mayor control sobre esta.

Figura 3.9

El manual lo hice para que inclusive personas que no tuvieran alguna clase conocimientos técnicos pudieran entenderlos fácilmente. Es por eso que desarrollé cada sección y botón de la manera más específica posible como se observa en la siguiente imagen:

MENSAJES

Durante la captura, usted podrá observar que, si no ha completado el llenado de las celdas o se tiene un error con respecto a los **CRITERIOS DE REVISIÓN**, al querer intentar salvar el programa le envía un aviso indicando que faltan datos, del mismo modo el programa resalta activando un sombreado cuando no sea capturado el dato ó el dato esta repetido.

BOTONES DE LAS SECCIONES

Los Botones más utilizados en las ventanas de las secciones anteriores, serán explicados brevemente, para tener un mejor control de la herramienta.

- Con este botón, podrás guardar el progreso que lleves en tu captura. Al momento de guardar se mostraran avisos con respecto a la información capturada.

- Sirve para crear un nuevo registro (renglón), para colocar más información.

- Sirve para eliminar un registro (renglón), en caso de que así lo decidas.

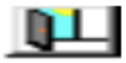
- Cancela todos los registros ó cambios que se hayan realizado desde la última vez que guardo información.

- Cierra la ventana que, en ese momento se encuentre activa.

- Agrega un subrenglón del renglón principal.

- Elimina un subrenglón del renglón principal.

- La función de este icono permite salvar los datos que se ingresen.



- Este botón permite al usuario abandonar la opción de Datos de Registro.

- Una vez que se llene la primera ventana, con la ayuda de este botón, se podrá desplazar a las siguientes ventanas

Nota: Es importante recordar que se debe salvar periódicamente para evitar la pérdida de la información.

Figura 3.10

Una vez realizado el manual de usuario con la guía de instalación, fue cuando propuse hacer el manual en línea, para que no solo los usuarios que lo quisieran impreso pudieran bajar el documento en PDF de la página oficial de la SEMARNAT, si no que también pudieran acceder a un sitio web en donde por medio de videos e imágenes les quedara aún más claro el uso y configuración del software. Una vez aprobada la propuesta me di a la tarea de realizar este proyecto de software.

Primero tenía que saber qué iba a realizar y cómo lo iba a realizar. En este caso los clientes eran mis supervisores en el área de Divulgación e Información de la SEMARNAT.

Era un proyecto sencillo, pero a la vez complicado pues el cliente aún no sabía 100% lo que quería. Para este proyecto decidí usar el ciclo de vida de cascada, dado que tenía un poco de experiencia con este modelo y me parecía el correcto para minimizar errores.

Así como lo marca el ciclo de vida en cascada, empecé con la toma de los requerimientos funcionales y no funcionales del manual en línea, mis supervisores no tenían muy claro que era lo que querían ni lo que tenía que llevar el proyecto, entonces, por medio de entrevistas me platicaron cómo se lo imaginaban y cómo les gustaría que funcionara. A esta manera de toma de requerimientos se le denomina “Historias de Usuario” . Esta técnica puede llegar a complicarse dado que el cliente no tiene bien claro qué es lo que quiere.

Las Historias de Usuario

Es la técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas.

Una vez realizada las historias de usuario, tenía lo que mis asesores querían, los requerimientos funcionales y no funcionales pedidos.

Requerimientos Funcionales:

- Sistema web
- Programado en HTML5, CSS 3 y jQuery 1.9
- Contar con misma información que el manual de usuario

Requerimientos No Funcionales:

- Alojar en servidores locales SEMARNAT
- No exigir muchos recursos a los servidores
- Interfaz minimalista y limpia sin exceso de botones
- Con la misma plantilla de colores de la página oficial de la SEMARNAT (http://www.semarnat.gob.mx/)
- Intuitivo
- Simple de usar
- Contenido Multimedia

Durante la validación de requerimientos, me encontré con un problema: mis supervisores no tenían acceso a los servidores, ni sabían las características de estos; aún peor, los servidores no se encontraban en el edificio. Se sabía que contaba con Sistema operativo Windows Server, Servidor HTTP y Servidor Apache. Aunque no se contara con esto al momento, si se podría hacer el despliegue del sitio. Entonces, lo único que me pidieron, fue que hiciera que el manual en línea y ocupara lo mínimo de recursos, lo cual no sería problema dado que el manual no usaría bases de datos, motor de búsquedas o algún tipo de registro de usuario al ser un sistema web estático.

Seguí con el diseño del manual, el cual lo estructuré de la siguiente manera, con base al manual de usuario con el que ya se contaba:

1) Objetivos del sitio web

2) Estructura y diseño

    a) Página de inicio

    b) Páginas de las principales secciones

    c) Páginas secundarias

    d) Menú principal de navegación

    e) Menú secundario de navegación

    f) Atajos de navegación

El objetivo principal era tener el mismo manual de usuario que se tiene en documento pero de manera más fácil de entender para el usuario, al no tener tanta información y tutoriales de como manejar el software.

Tal y como el documento oficial del manual de usuario, el manual en línea contaría con todas las secciones que el software de la COA cuenta, con una breve descripción de qué datos llenar en esa descripción y un videotutorial de como hacerlo. El sitio se dividió en las siguientes secciones:

- **¿Qué es la COA?**
- **Instalación**

    ¿Cómo descargar la COA?

    ¿Cómo instalar la COA?

    Requisitos de instalación

    ¿Cómo configurar el equipo?

- **Sección I**

  1.1 Diagrama de operación y funcionamiento

  1.2 Insumos

  1.3 Productos y subproductos

  1.4 Combustibles para uso energético

- **Sección II**

  2.1.1 Características de la maquinaria que genera contaminantes

  2.1.2 Ductos y chimeneas

  2.2 Contaminantes Atmosféricos Normados

  2.3 Emisiones Anuales

- **Sección III**

  3.1 Aprovechamiento o uso de agua

  3.2.1 Descargas de aguas residuales

  3.2.2 Volúmen total de descarga

- **Sección IV**

  4.1 Generación y transferencia

  4.2 Almacenamiento de residuos

  4.3 Manejo de residuos peligrosos

- **Sección V**

  5.1 Uso, producción y comercialización de sustancias RETC

  5.2 Emisiones y transferencias de sustancias RETC

  5.3 Emisiones de sustancias derivadas de accidentes, inicio de operaciones y paros programados

  5.4 Prevención contaminación

  5.4.2 Reutilización,tratamiento y prevención de sustancias dentro del establecimiento y disposición final

5.5 Tratamiento y disposición

5.6 Razones de cambios

- **Herramientas COA**

Observaciones y aclaraciones

Comentarios y sugerencias

Impresión y respaldo de la COA

Respaldo y envío de la COA por Internet

Ya una vez estructurado y diseñado el manual en línea, pude empezar con la implementación del mismo. Este fue el resultado preliminar del diseño e implementación, el cual fue validado por mis supervisores y me permitieron seguir con el proyecto. Diseño de manual en línea:

Figura 3.11

Conforme iba programando la página, iba haciendo mi documentación. Todo fue documentación interna y queda incrustado en el mismo código. Cómo no se iba a hacer el despliegue del manual en línea durante mi estancia, esto era muy importante, por si algún futuro desarrollador quería agregar o hacer modificaciones al sistema. Quedó indicado por bloques que era lo que hacía cada bloque. Así podemos evitar futuras crisis. También hice uso de documentación externa:

La documentación del sistema web de la COA quedó de la siguiente manera:

Documentación de Sistema Web

1) Aspectos General:

Tecnologías y/o lenguajes de programación web utilizados:

HTML 5

CSS 3

jQuery 1.9

2) Aplicaciones de desarrollo utilizadas:

Coda 2 (https://panic.com/coda/)

Editor de texto (Default del S.O.)

Navegador Google Chrome o cualquier navegador (https://www.google.com/intl/en/chrome/browser/)

3) Uso de URLs

El sitio tiene muchas URLs que conectan con el sitio oficial de la SEMARNAT y con todas las secciones de uso de la COA. Estás URLs deben ser codificadas de la siguiente manera:

- Sin tildes en caso de que se necesario, hacer uso de carácter especial de la “ñ” que permite usar HTML (&oacute).
- Sin espacios

- Sin “ñ” en caso de que se necesario, hacer uso de carácter especial de la “ñ” que permite usar HTML (&#208).

4) Uso de imágenes

Todas las imágenes usadas en el sitio web, fueron proporcionadas por la SEMARNAT y sacadas de la aplicación de la Cédula de Operación Anual. La imágenes se encuentras alojadas dentro de la carpeta del proyecto que se llama “img”. Todas las imágenes se encuentras en formato PNG para reducir el tamaño de estas.

5) Uso de videos

Para fines de eficiencia, los videos se encuentran alojados en la cuenta oficial de Youtube de la SEMARNAT. Siendo la Ing. Floreida Paz Benito quien cuenta con los permisos necesarios para acceder a la cuenta. Se hace uso del video por medio de incrustación de la URL del video. Ver condiciones de URLs en punto 3.

6) Fuente de información

Toda la información divulgada en este sistema web fue obtenida del sitio oficial de la SEMARNAT, documentación interna y del manual de usuario de la Cédula de Operación Anual.

Para el contenido multimedia, grabé directo de la tarjeta de video de la computadora que tenía asignada, el cómo usar el software de la COA sección por sección. Como uno de los requerimientos era que usara el mínimo de recursos del servidor, decidí subir esos vídeotutoriales a YouTube, y una vez ahí, incrustarlos dentro de las secciones pertinentes del videotutorial, de esta manera logré que el manual en línea fuera más eficiente.

Después, realice las pruebas alfa y de validación. Siendo yo mismo el usuario a probarlas, ya que mis supervisores me pidieron que ya les entregara un sistema final. Lo probé como si fuera un usuario sin conocimientos técnicos y tuviera muchas dudas de cómo usar el software de la Cédula de Operación Anual. El manual en línea cumplía con su objetivo principal, que era informar y sacar de dudas a los usuarios. También, no presentaba errores, corría bien en mi servidor local en la computadora, que estaba muy limitada en hardware. Con una interfaz intuitiva y muy limpia.

Por último, hice la entrega de todo el sistema web. Les enseñe el manual en línea funcionando, mis supervisores lo aprobaron. Habían quedado completamente satisfechos con el proyecto. Les entregué el código fuente con toda la documentación necesaria, y todo el contenido necesario. Lamentablemente durante mi estancia, yo no pude hacer el despliegue del sistema porque había concluido con mi Servicio Social, pero al seguir todos los estándares establecidos por la Ingeniería de Software, y todos los pasos necesarios en el ciclo de vida de un proyecto de software, se puede confiar, que a la hora del despliegue no habrá problemas, ni cuando se le quiera dar mantenimiento de cualquier tipo al sistema, porque quedó perfectamente documentado. El mantenimiento del sistema será mínimo debido a que es un sistema estático, solo se necesitaría realizar cambios, si el sistema de la COA sufre cambios o llegará a haber algún cambio en los servidores.

A continuación se puede ver el resultado final. Es la página de inicio del manual de usuario en línea:

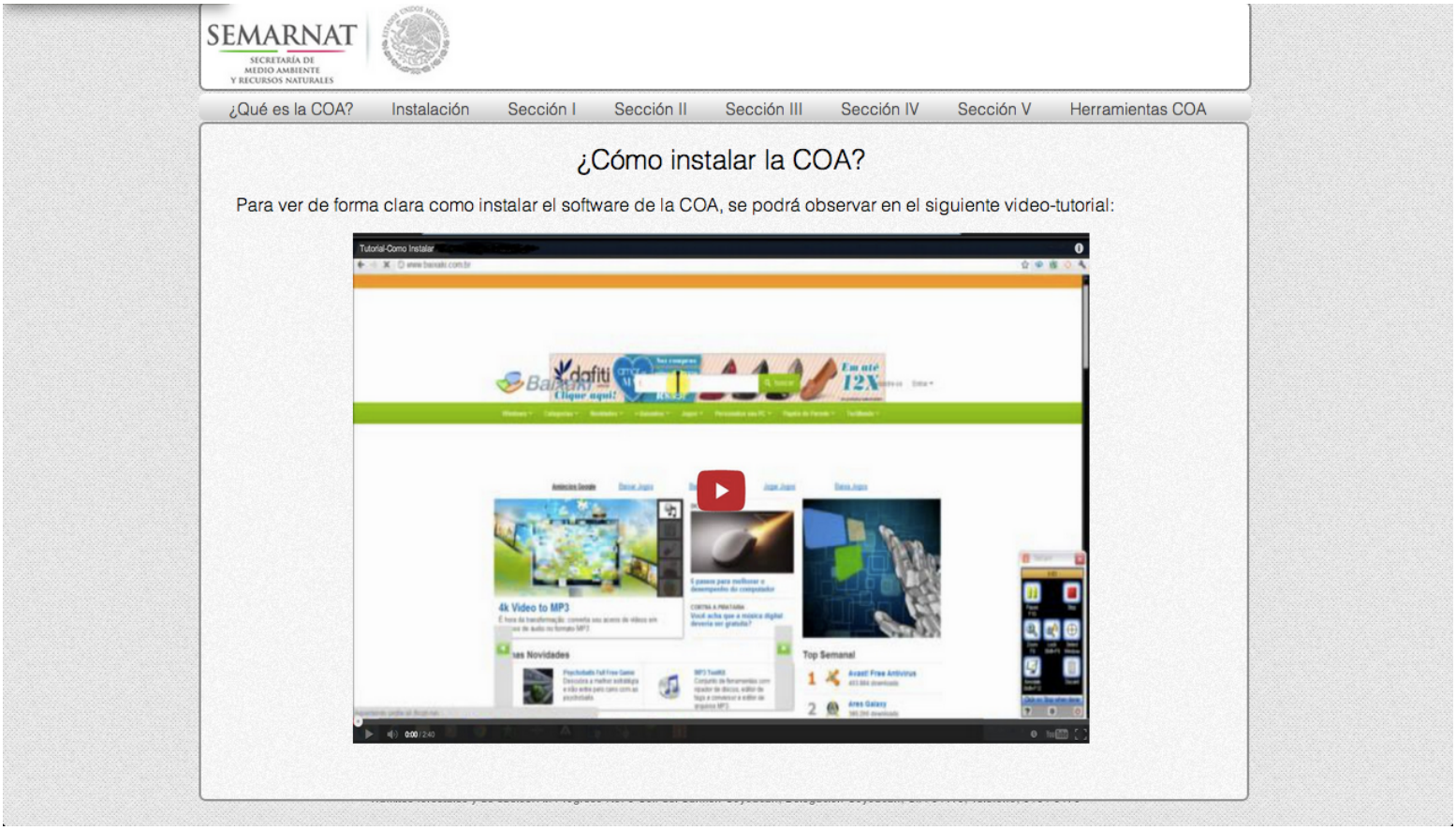


Figura 3.12

El manual en línea no lo pude ver trabajando en el mundo real ayudando a usuarios, sin embargo, si pude ver como el Manual de Usuario fue distribuido por diferentes medio y siendo de gran utilidad para los usuarios finales. Esto fue de gran ayuda para la SEMARNAT. Pude observar cómo un manual de usuario bien hecho, las llamas que recibían el equipo de soporte técnico todos los días disminuyeron en al menos un 50%. Este manual se subió a la página oficial de la COA de la SEMARNAT para poder ser descargado como PDF por cualquier usuario.

IV. Conclusiones

El software se ha convertido en el elemento clave de la evolución de productos informáticos. A través del tiempo, el software ha pasado de ser una resolución de problemas especializadas y una herramienta de análisis de información, a ser una industria por si misma. Pero la temprana cultura e historia de la programación ha creado un conjunto de problemas que persisten todavía. El software se ha convertido en un factor que limita la evolución de los sistemas informáticos. El software se compone de programas, datos y documentos. Cada uno de estos elementos componen una configuración que se crea como parte del proceso de la Ingeniería del Software. El intento de la Ingeniería del Software es proporcionar un marco de trabajo para construir software con mayor calidad. El software los vemos todos los días en nuestro vida cotidiana. Hacemos uso de él aún sin darnos cuenta. Es necesario contar con software de calidad para poder facilitar nuestra calidad de vida. No podríamos tener productos de tan alta calidad si no fuera por los Ingenieros en Computación y la Ingeniería de Software.

El desarrollo de software es un proceso que puede llegar a ser sumamente complejo. Todos los errores durante el desarrollo pueden llegar a ser transcendentes y causar problemas de alto impacto como en desempeño, calidad, así como económicamente.

Sin importar el tamaño de un proyecto se debe seguir con los estándares y procesos estipulados para poder minimizar cualquier tipo de error humano. Las mejores prácticas sólo reducen el riesgo de encontrarnos con problemas realmente serios y garantizar la calidad del software.

Es importante tomarse el tiempo necesario para conocer a nuestros clientes y usuarios, así como su ambiente de trabajo. Sobre todo, porque si no se tiene claro los requerimientos, se puede entrar un producto que no le sirve al cliente. Esto, también ayuda a establecer una buena relación de trabajo y comunicación entre el equipo de desarrollo y los clientes. Los

usuarios participan poco en el desarrollo del software, con el riesgo de que no satisfaga sus necesidades y aumenten los esfuerzos en el mantenimiento.

Los estándares utilizados para el desarrollo son la principal herramienta donde nos debemos basar, para seguir unas normas que harían que el proyecto se entregue con buena calidad.
La importancia de las métricas dentro de la ingeniera, me dan la posibilidad de tener control de los indicadores de los proyectos.

Para desarrollar un producto de software se requiere de todo un proceso. El cual esta predefinido por una diversidad de modelos. Es elemental tener en cuenta los modelos que se adapten para un proyecto requerido, para obtener como producto de ello una funcionalidad óptima del producto. Existen diversos modelos dentro del proceso del software, así mismo, existen diversas metodologías y tipos de modelos para desarrollar un software. Producto de ello cada metodología tendrá un modelo de proceso para su desarrollo, el cual se adecua respecto con las necesidades de un proyecto.

La tecnología evoluciona rápidamente, y claro, el software no es la excepción. Si no se sigue las recomendaciones de la Ingeniería de software para el mantenimiento de software puede ocasionar desastres. Los sistemas que son mantenidos son cada vez más difíciles de cambiar.
Después de cada cambio los programas tienden a ser menos estructurados. Como consecuencia se produce una documentación desfasada, código que no cumple los estándares, incremento en el tiempo de comprensión de los programas o incremento de los efectos secundarios de los cambios.

Pareciera que la documentación de un proyecto de software no es tan importante, pero impregna el ciclo de vida del mismo. Es la parte más visible de su proceso. Sin ella, no se puede dar mantenimiento al software, los usuarios no pueden entender el uso del software y

prácticamente no pueden utilizar el software. Los nuevos desarrolladores tendrían que desaprovechar mucho código en el desarrollo del software, o no podrían hacer mejoras. La documentación del software es la manifestación más importante de la Ingeniería de Software.

Las herramientas utilizadas para una correcta documentación de código y la importancia que esto trasciende a lo largo de toda la vida del software. Se determinó que documentar un código a medida que se realiza su implementación es de vital importancia y trae grandes beneficios.

Bibliografía

• Cuevas, G., Ingeniería del software: Práctica de la programación, Serie Paradigma, México, 1993.

• Dolado Cosín, Javier, Técnicas cuantitativas para la gestión en la ingeniería del software, Net biblo, 2009.

• Pressman, R. Ingeniería del software: un enfoque práctico 6a ed. México

• Somerville, I. Ingeniería del software 7a ed., Rivera de Loira, México , 2005.

Referencias electrónicas

• Secretaría de Medio Ambiente Y Recursos Naturales “Quienes Somos”

http://www.semarnat.gob.mx/conocenos/Paginas/quienessomos

• Introduction to Software Engineering

http://www.newagepublishers.com/samplechapter/001036.pdf

• Diseño conceptual y especificación de requerimientos para el desarrollo y rediseño de sitios web

http://eprints.rclis.org/18666/1/PedrazaBlancoCodinaCavaller2013_EspecificacionRequisitosWeb.pdf

• Paradigm Shift In Programming Techniques

http://my.safaribooksonline.com/book/software-engineering-and-development/9788131758694/introduction/sect1-9#X2ludGVybmFsX0h0bWxWaWV3P3htbGlkPTk3ODgxMzE3NTg2OTQlMkZzZWN0MS04JnF1ZXJ5PQ==

• Modelo Procesos para la Industria del Software (MoProSoft)

http://www.allsoft.com.mx/recursos/AS-Moprosoft.pdf

http://www.moprosoft.com.mx