Universidad Autónoma de Querétaro
Facultad de Informática
Especialidad en Maestría en Sistemas de Información

Metodología de desarrollo para sistemas de información basados en web.

Opción de titulación
Tesis o Publicación de artículos

Que como parte de los requisitos para obtener el Grado de
Maestría en Sistemas de Información

Presenta:
I.T. Eduardo Aguirre Caracheo

Dirigido por:
MISD. Juan Salvador Hernández Valerio

MISD. Juan Salvador Hernández Valerio
Presidente

Firma

Dra. Sandra Luz Canchola Magdaleno
Secretario

Firma

M. en C. Ruth Angélica Rico Hernández
Vocal

Firma

MISD. Carlos Alberto Olmos Trejo
Suplente

Firma

MISD. Jesús Armando Rincones
Suplente

Firma

M. en C. Ruth Angélica Rico Hernández
Director de la Facultad

Dr. Iñaki Torres Pacheco
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Octubre 2014

RESUMEN

Desde su aparición la plataforma web se ha vuelto omnipresente en cada aspecto de nuestra cotidianidad en sus distintas formas y categorías. Su evolución la ha llevado a volverse una plataforma robusta y segura que algunos desarrolladores de software y la gran mayoría de usuarios encuentran idónea para la solución a sus necesidades. Sin embargo, desarrollar sobre esta nueva alternativa resultaba caótico y sin un proceso claro para quienes trabajaban con ella, justo como la Ingeniería de Software en sus comienzos. Para desarrollar en la web, se comenzaron a utilizar y adaptar metodologías de desarrollo de software tradicional. El resultado de esto eran sistemas web que fallaban en cumplir sus requerimientos y las metodologías adaptadas que si cumplían con los requerimientos tenían problemas en volver a tener éxito. Como respuesta a esto surge una nueva disciplina llamada Ingeniería Web. La Ingeniería Web es la aplicación de enfoques sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de sitios y sistemas web. Junto con ella comienzan a surgir metodologías de desarrollo enfocadas en la plataforma web. Sin embargo, muchas de ellas no son metodologías enfocadas en el ciclo de vida o proceso de desarrollo de software, sino en buenas prácticas de diseño. En la presente investigación se analizaron las distintas metodologías que se aplican en el desarrollo web, tanto de software tradicional como de software web. Esto con el objetivo de generar una metodología enfocada en el desarrollo web en base a la comparación y análisis de las metodologías estudiadas. La metodología desarrollada es aplicada en dos casos prácticos, para finalmente realizar un análisis de la misma.

(Palabras clave: Sistemas de Información Web, Desarrollo web, Ingeniería Web, Metodología de Desarrollo, Ingeniería de Software.)

ABSTRACT

Since its arrival, the web platform in its shapes and categories has become omnipresent in every aspect of our daily lives. Its evolution has led it to become a safe and robust platform that some software developers and most users find ideal for fulfilling their needs. However, developing for this new alternative was chaotic and lacked a clear procedure for whoever worked on it, just like Software Engineering on its early days. To develop for the web, traditional software development methodologies were used and adapted. This resulted on web systems that failed to meet their requirements, and the adapted methodologies that did meet the requirements had problems in succeeding again. As an answer to this issue, a new discipline called Web Engineering emerged. Web Engineering is the application of systematic, disciplined and quantifiable approaches on the development, operation and maintenance of web sites and systems. Along with it, other web-oriented development methodologies began to surface. However, many of them are not methodologies that focus on the software development process or life cycle, but on the best design practices. This research analyzes different methodologies applied to web development, being it either traditional software or web software, The purpose being to generate a methodology focused on web design, based on the comparison and analysis of the studied methodologies. The developed methodology is applied on two case studies to, in the end, make an analysis of said methodology.

Key Words: Web Information Systems, Web development, Web Engineering, Development Methodology, Software Engineering.)

DEDICATORIA

A mis padres, que siempre me han apoyado a cumplir mis metas brindándome su apoyo y amor incondicional. Gracias por ser el mejor ejemplo de trabajo y mi inspiración en cada momento. Gracias a ustedes soy lo que soy.

A Pablo, Isaac y Marianita, quienes fueron y serán siempre mi impulso y mi motivo para crecer como profesionista y persona, por siempre estar aquí conmigo.

A Shadia, mi compañera en momentos de aventura y mi motor en momentos de debilidad, gracias por todo tu amor, apoyo y confianza.

A mis primos, abuelitas y tíos por su apoyo y sinceridad, gracias a sus consejos crecí como persona.

A mi amigo Moisés, mi hermano durante todo el tiempo que ha durado esta travesía.

A mis amigos Diego, Alejandro, Carlos Olmos y Juan Salvador quienes siempre me han apoyado y me han brindado su amistad, gracias por alegrarme siempre los días.

A mis amigos David, José Luis, Luis Eduardo, Paulina, Alonso y Reiji por las muestras de amistad, cariño y confianza.

AGRADECIMIENTOS

Al MISD. Juan Salvador Hernández Valerio el apoyo y paciencia brindada desde siempre y por el tiempo y trabajo dedicados durante el desarrollo de esta investigación.

A la M. en C. Ruth Angélica Rico Hernández por brindarme la oportunidad de formar parte de su equipo de trabajo y por apoyarme y permitirme continuar con mí desarrollo profesional.

Al MISD. Carlos Alberto Olmos Trejo por el apoyo brindado durante el desarrollo de los sistemas en los que se evaluó la metodología y por fomentar mi crecimiento como persona y profesionista.

Al L.I. Alejandro Vargas Díaz y el I.S. Diego Octavio Ibarra Corona por su apoyo incondicional como compañeros de trabajo y amigos y por su colaboración en el desarrollo de la investigación.

A mis maestros por compartir su tiempo y conocimiento conmigo.

TABLA DE CONTENIDOS

INDICE DE TABLAS

INDICE DE FIGURAS

1. INTRODUCCIÓN

La web o *World Wide Web* es una plataforma que ha evolucionado desde hace veinticinco años posicionándose como el mayor medio de difusión de intercambio personal en la historia de la humanidad. Por medio de esta plataforma, usuarios que se encuentran separados por tiempo y espacio han podido interactuar, buscar y compartir.

Si bien, al igual que cualquier otro medio de comunicación, esta plataforma tiene sus desventajas. A pesar de que su existencia se basa en medios físicos, como servidores de almacenamiento web y medios de comunicación, la naturaleza virtual de la información en la web permite que esta pueda ser buscada y accedida más fácil y eficientemente que con cualquier otro medio físico existente.

En el 2005 los primeros mil millones de usuarios fueron alcanzados. Para el 2010 el número de usuarios fue duplicado y se espera que los tres mil millones de usuarios sean alcanzados para finales del 2014. Mientras en 1995 solamente el 1% de personas participaban como usuarios en internet, hoy día poco más del 40% de la población mundial tiene conexión a internet.

Tan solo en México la cantidad de usuarios en internet creció en este año 13% con respecto al 2013, esto quiere decir que poco más de 50 millones de personas, lo que representa el 40% de la población, son usuarios de servicios en la web.

Ha sido tanto el impacto en la sociedad que servicios y modelos de negocio completos son endémicos de la web. Los sitios web de comercio electrónico, redes sociales, *streaming* de audio o video, banca electrónica, correo electrónico, blogs, almacenamiento en la nube, son solo algunos ejemplos de la variedad de servicios basados en internet.

El desarrollo de estas aplicaciones en la web es llevado a cabo por la ingeniería web. Esta rama de la ingeniería aun no llega al mismo nivel de madurez de la ingeniería de software que ha vivido una constante evolución por más de cincuenta años.

A sí mismo, encontrar una metodología que respalde el trabajo del equipo durante el desarrollo de un proyecto para la plataforma web se vuelve una tarea complicada, ya que como la ingeniería web, las metodologías para desarrollo de software basado en esta plataforma son aun inmaduras o difíciles de implementar y muchos desarrolladores optan por utilizar una metodología de software tradicional y adaptarla a las necesidades de la metodología web.

Bajo la problemática anterior, esta investigación busca analizar y criticar las metodologías de desarrollo de software tradicional y de software web para proponer una metodología lo suficientemente compatible y adaptada para servir de respaldo durante todo el ciclo de desarrollo de un proyecto web.

2. OBJETIVOS

2.1. Temas a tratar

En la presente investigación, los temas principales que se contemplarán son los siguientes:

- Ingeniería de Software.
- Ingeniería Web.
- Metodologías de desarrollo de software tradicional.
- Metodologías de desarrollo de software web.
- Metodologías agiles.
- Metodologías pesadas.
- Ingeniería de Requerimientos.

Algunos temas adicionales pueden ser tratados y desarrollados dentro de cada uno de los temas principales.

2.2. Objetivo Generales

Diseñar una metodología de desarrollo de aplicaciones web que adapte las metodologías de desarrollo existentes complementándolas con las etapas necesarias para que puedan aplicarse a los sistemas de información web.

Objetivos Específicos.

- Proponer una metodología de desarrollo de sistemas de información que se adapte a las necesidades de las tecnologías web.
- Definir cada una de las etapas para el desarrollo de sistemas de información web en base a un análisis de los métodos existentes de desarrollo de software.
- Aplicar la metodología propuesta durante el desarrollo de un proyecto (sistema asistente de carga horaria) para la Facultad de Informática.
- Adecuar la metodología propuesta con base los resultados observados en el desarrollo de la aplicación.

3. INGENERÍA DE SOFTWARE E INGENIERÍA WEB

El origen de la Ingeniería de Software data de inicios de los años 60's, que fue un periodo esencial en la era de la computación con la aparición de enormes computadoras con fines científicos en distintas universidades y centros de investigación (Wirth, 2007). Las empresas no tardaron en darse cuenta del potencial para los negocios que las computadoras tenían.

De las necesidades de las empresas por llevar a cabo ciertas soluciones surgió una nueva profesión. Al comienzo, la programación y las computadoras eran tareas por separado (Wirth, 2007), ya que el programador no tenía acceso a la computadora, solamente entregaba el programa y un contador era quien se encargaba de ponerlo en la cola de procesos.

Esto quiere decir que desde la aparición de las primeras computadoras digitales se ha tenido la necesidad de desarrollar software para distintas aplicaciones en las organizaciones para cubrir sus necesidades de tener control sobre la información y el negocio de manera más rápida y segura. Esto ha provocado que a lo largo del tiempo hayan surgido diferentes enfoques, técnicas y lenguajes para el desarrollo de programas o sistemas.

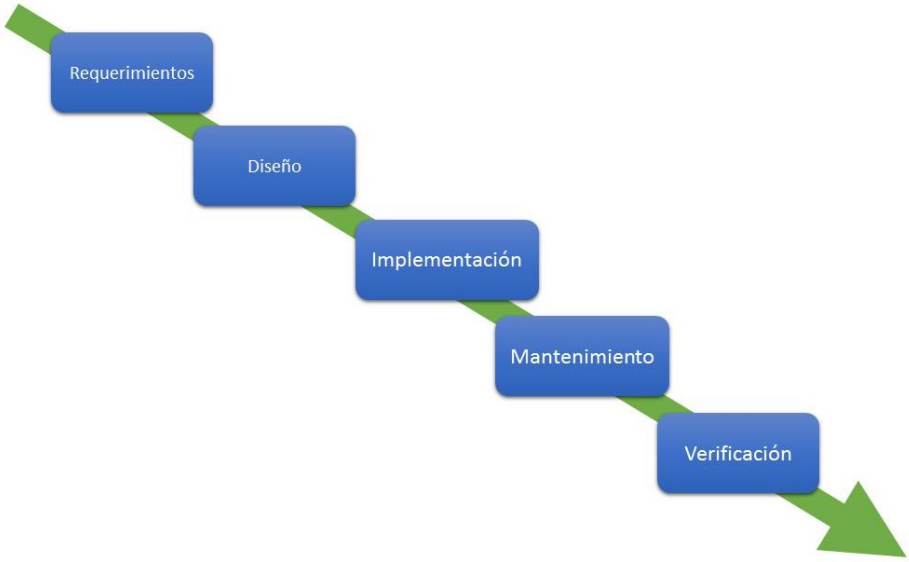
Sin embargo, la historia del desarrollo de software se vio afectada por la falta de disciplina de los programadores, ya que solamente el 16% de los proyectos desarrollados se lograban de manera exitosa (Sacha, 2007); el resto terminaban con en fallas en tiempos de entrega, proyectos desarrollados sin ningún orden que generaban costos exponenciales en los desarrollos y mantenimientos de código ocasionando pérdidas en los costos de las empresas y, por increíble que parezca, incluso vidas.

El software siguió evolucionando y con esto llegaron los sistemas de información. En los sistemas anteriores, que eran los sistemas transaccionales, se tenía una pila de programas esperando ser ejecutados uno después del otro hasta terminar cierta tarea. Los sistemas de información permiten tener una mayor interacción con el usuario e interconexión con diferentes sistemas dentro o fuera de la empresa (Cornford, 2013). El problema es que mientras mayor sea la interacción y más profunda la interconexión nos enfrentamos con problemas como: casos de uso no contemplados, la seguridad en los sistemas y al no tener un orden en su ejecución, una mayor posibilidad de errores debido a que no existe un flujo de trabajo definido.

Con el objetivo de producir software de calidad, es decir, que no solo no tuviera errores sino que también cumpliera con tiempos de entrega y presupuestos, comenzaron a implementarse las metodologías de desarrollo. Estas metodologías son un conjunto de pasos que tienen como objetivo llevar a cabo el proyecto de una forma más estructurada y metódica, utilizando herramientas de análisis y diseño para lograr desarrollar software de calidad de manera exitosa (Laboratorio Nacional de Calidad del Software, 2009).

Así como el desarrollo de software tuvo un proceso de evolución, las metodologías también tuvieron que madurar conforme eran aplicadas. La primera metodología de desarrollo de software, por ejemplo, era la de cascada. Esta es una metodóloga en la que una fase del desarrollo comienza hasta que la anterior es completada (ver figura 3-1). La metodología en cascada no contemplaba una fase de pruebas, esto provocaba que en los proyectos desarrollados los errores de las primeras etapas eran encontradas al final en el momento de su uso.

Figura 3-1 Ciclo de Vida del Modelo de Desarrollo de Software en Cascada. Fuente: Elaboración Propia.



Con el paso del tiempo y gracias a la experiencia ganada desarrollando proyectos utilizando esta metodología, se logró madurar el proceso proponiendo nuevas etapas o complementando las existentes.

En una de las metodologías posteriores ya se contemplaba la fase de pruebas, aunque ahora existía la problemática de tener que emplear ingenieros para desarrollar software e ingenieros para realizar las pruebas en el código (Petersen & C., 2009). Esto también requería herramientas e infraestructura para efectuar el control de calidad de la forma adecuada. Después de años de evolución nacen diferentes metodologías, herramientas y diferentes disciplinas adicionales a la programación como lo es el análisis, definición de requerimientos, pruebas, gerencia de proyecto, arquitectura de software, diseño experiencia del usuario y usabilidad, entre otras.

La cantidad enorme de procesos de estas nuevas metodologías cuando se implementan dejan la impresión a los clientes y las empresas de que son muy complejas, lentas, y con exceso de documentación.

Para hacer el proceso de desarrollo de proyecto más amigable, tiempo después se generaron metodologías mucho más ligeras, denominadas ágiles, en donde la prioridad está en las interacciones e individuos en lugar de los procesos y herramientas (Balaji, 2012). Esto quiere decir software funcionando en lugar de documentación extensa, colaboración con el cliente sobre negociación contractual y respuesta eficiente ante cambios en lugar de seguir un plan (Serena Software, 2007). Esta nueva corriente tiene como objetivo llevar a cabo desarrollos más rápido y con buena calidad.

3.1. Ingeniería Web

3.1.1. Breve Historia de la Web

El Internet nace en 1969, pero no era como lo conocemos hoy en día. En aquel entonces era una plataforma utilizada completa y únicamente con propósitos militares. Después de esto, se comenzó a utilizar por universidades, pero solo como una herramienta para compartir datos con fines científicos a través de redes de computadoras por medio del protocolo de comunicación TCP/IP (Protocolo de control de transferencia sobre el protocolo de Internet), pero el lenguaje y la interfaz con la que se comunicaban era nada amigable para el usuario y poco funcional.

No fue hasta 20 años después en 1989 que surgió el primer sitio web. El ingeniero en software Tim Berners-Lee del CERN, laboratorio de Geneva, Suiza especializado en física de partículas, inventó la *World Wide Web* ó Red Informática Global, mejor conocida como la web. En ese tiempo muchos científicos participaban en experimentos que se llevaban a cabo en el CERN por periodos largos de tiempo para después regresar a sus laboratorios en otro lugar del mundo. Los científicos estaban ansiosos por compartir e intercambiar su información con sus colegas, y esto era una tarea difícil y costosa de lograr, sin saber que desde hace veinte años

contaban con la plataforma para llevarlo a cabo (About: World Wide Web Foundation, Consultado en el 2014).

Tim Berners-Lee entendió la problemática que se tenía y se dio cuenta del potencial que nadie había notado de tener millones de computadoras conectadas entre sí a través del Internet, sin embargo, el Internet aún no estaba listo para llevar a cabo la comunicación con la que se planeaba resolver los problemas.

En 1989 Berners-Lee sometió una propuesta a revisión a la administración del CERN en la que especificaba las tecnologías que se necesitaría para poder hacer del Internet una herramienta realmente accesible y útil para las personas, propuesta que establecería las bases de lo que a la larga se transformaría en la web (About: World Wide Web Foundation, Consultado en el 2014).

Después de ser rechazado la primera vez volvió a someter la propuesta a revisión en 1990, donde especificaba tres tecnologías fundamentales para la web y que en la actualidad siguen siendo sus bases.

La primera de estas tecnologias es el lenguaje en el que se escriben las páginas web, el HTML (HyperText Markup Language) que es el lenguaje de etiquetado de hipertexto que sirve para dar estructura, contenido y vinculos a otros recursos o páginas. La segunda tecnología es el URI (Uniform Resource Indentifier) o Identificador Uniforme del Recurso, que es básicamente la dirección única con la cual podemos localizar algo en Internet. Finalmente, la última tecnología es la manera de navegar a través de estos documentos, el HTTP (Hypertext Transfer Protocol), o protocolo de transferencia de hipertexto.

Aparte de establecer este conjunto de tecnologías para que la web fuera funcional, Tim Berners-Lee también construyó el software con el cual se podría acceder a estas páginas en la red, el navegador llamado *“World Wide Web”* y la

computadora que almacenaba las páginas y recursos a los cuales se accedería, el servidor web llamado “HTTP” ó *“HyperText Transfer Protocol Daemon”* (Leiner, Cerf, & Clark, 2009).

A finales de 1990 la primera página web se encontraba en línea en el primer y único servidor web funcional en el mundo, que se encontraba en el CERN. En abril de 1993 CERN anuncio que la tecnología de la web estaría bajo licencia *royalty-free*, que quiere decir que estaría disponible para toda persona que quisiera utilizarla (Leiner, Cerf, & Clark, 2009). Con la posibilidad de que cualquier persona pudiera crear sitios web debido a su código abierto el crecimiento de la web comenzó a notarse, tan solo en 1995 ya había poco más de 50 servidores web en distintas universidades y centros de investigación.

Que la web sea tecnología abierta ha sido su principal factor de crecimiento desde entonces, ya que da la libertad al creador de las páginas de hacer lo que fuera en la web, como utilizar sus propias etiquetas, estructuras y contenidos. Tim Berners-Lee y otras personas se dieron cuenta de que para poder alcanzar el potencial verdadero de la web debería de haber algo que estandarizara y regulara la forma en que se creaban contenidos.

Con este objetivo en 1994 se fundó el *World Wide Web Consortium* (W3C), en el que las personas interesadas en el desarrollo de la web trabajaban en construir lineamientos y especificaciones que serían la base para asegurar que el contenido la web se construyera y funcionara de manera correcta y responsable.

3.1.2. Ingeniería Web

Después de poco más de dos décadas desde la aparición de la web, esta se ha convertido en un elemento omnipresente en nuestra vida cotidiana y continua creciendo exponencialmente y en la complejidad de los sistemas y aplicaciones de esta plataforma. La web se ha vuelto la plataforma favorita tanto de desarrolladores

como de usuarios y organizaciones que cada vez con mayor frecuencia encuentran la solución a sus problemas gracias a la web.

Debido a la manera incremental en que ahora dependemos en las aplicaciones y sistemas basados en la web, su rendimiento, seguridad y calidad se convirtieron en una importancia primordial, y las expectativas y demandas puestas en las aplicaciones web han incrementado significativamente a través de los años (Murugesan & Ginige, 2005).

Sin embargo, es algo común que sitios web que comenzaron con algunas páginas crezcan y se vuelvan inmanejables y hacer cambios se vuelve una tarea difícil, si no es que imposible mientras más grande sea el sitio web. Esto debido a que los desarrolladores web muchas veces piensan que el desarrollo de aplicaciones web es solamente la creación de páginas de HTML (*HyperText Markup Language* – Lenguaje de Etiquetado de HiperTexto) con algunos enlaces o imágenes o inclusive utilizando software para la cración de páginas web como *Dreamweaver* o *Frontpage*.

La llegada de herramientas de creación y manejo fácil de un sitio web parece haber trivializado la necesidad de una planeación cuidadosa, previsión, y una metodología sistemática de diseño. (Balasubramanian & Bashian, 1998).

Si bien existen sitios web que entren en la categoría antes mencionada, también existen sitios web más complejos llamados sistemas de información y requieren que exista planeación, arquitectura y diseño del sistema web, pruebas, evaluaciones de seguridad y de rendimiento y mantenimiento continuo de los sistemas mientras el uso y los requerimientos crecen y se desarrollan (Murugesan & Ginige, 2005).

Existen muchas personas desarrollando sitios web y muchos de ellos fuera de las funciones de los sistemas de información tradicionales, por lo tanto sin utilizar una metodología formalizada. Muchos desarrolladores web inventan sus propias metodologías durante el desarrollo con la esperanza de que el producto sirva a las metas de la organización. (Russo & Graham, 1998)

La necesidad por la Ingeniería Web hace falta según las percepciones de los desarrolladores y gerentes, sus experiencias en la creación de aplicaciones de manera factible por las nuevas tecnologías, y la complejidad de las aplicaciones web (Deshpande Y., 2002). Para poder construir sistemas y aplicaciones complejas y de larga escala, los desarrolladores web necesitan adoptar un proceso disciplinado e desarrollo y metodología, usar mejor herramientas de desarrollo y seguir un conjunto de buenos lineamientos (Murugesan & Ginige, 2005).

Lo primero que debemos reconocer cuando trabajamos con la Ingeniería Web es el hecho de que un buen desarrollo web requiere esfuerzos multidisciplinarios y no encaja por completo en cualquier otra disciplina existente, como la Ingeniería de Software.

Otro elemento que debemos de analizar es que un sitio web no es un evento, sino un proceso. No podemos desarrollarlo y después olvidarlo. Necesitamos construir en el diseño formas de mantener la información actualizada y una vez desarrollada, las actualizaciones se vuelven un proceso continuo (Ginige).

La Ingeniería Web como nueva disciplina busca resolver estas necesidades y se enfoca en el desarrollo exitoso de sistemas y aplicaciones basadas en la web, mientras utiliza un enfoque disciplinado y ordenado para el desarrollo web para evitar fallas potenciales ya que mantiene el desarrollo bajo control, minimiza los riesgos y mejora la calidad, mantenimiento y escalabilidad de las aplicaciones web.

La Ingeniería Web es la aplicación de enfoques sistemáticos, disciplinados y cuantificables para el desarrollo, operación y mantenimiento de las aplicaciones basadas en la web (Deshpande Y., 2002).

3.1.3. Sistemas de información web

La evolución de la web ha unido algunas disciplinas que se encontraban separadas como la multimedia, ciencias de la información y tecnologías de información y comunicación que juntas facilitan la creación, mantenimiento, uso y compartición de diferentes tipos de información en cualquier lugar, cualquier momento y utilizando una variedad de dispositivos como computadoras de escritorio y portátiles, pc de bolsillo, y teléfonos móviles (Murugesan & Ginige, 2005).

Cuando se habla de multimedia en la web se refiere a los diferentes tipos de media, como datos, textos, imágenes, video y audio, y los diferentes canales de comunicación como uno a uno, muchos a uno y muchos a muchos. Ciencias de la información se refiere a la colaboración, distribución, organización, presentación, administración y distribución de la información para la creación de contenido. Finalmente las tecnologías de información y comunicación se refieren a la infraestructura de comunicación que facilita compartir y facilitar la información, ya sea por medio de comunicación cableada o inalámbrica.

Con el paso de los años aparecen muchas nuevas tecnologías y estándares web para dar un mejor soporte a los sistemas de información web, como XMM, los *web services*, la web semántica y multimedia, *web mining*, *web intelligence,* y servicios basados en las tecnologías móviles tan recientes.

3.1.4. Diferencias entre software tradicional y software web

Una de las principales problemáticas de los sistemas de información web es en termino del entendimiento usuarios finales. Si un sistema está limitado a una intranet

las dificultades de definir al usuario se minimizan pero si el sistema va mas allá puede dirigirse a usuarios en todo el mundo si es que así se desea que la aplicación funcione, y si este fuera el caso, se deben de definir estrategias y pólizas para entender mejor a los potenciales usuarios desconocidos y establecer los parámetros necesarios para que los sistemas tengan los parámetros de calidad adecuados.

A diferencia de los sistemas tradicionales, los sistemas de información basados en la web no siempre están enfocados a grupos de usuarios específicos o una organización (Deshpande Y., 2002). Pero aparte de esto existen otras diferencias entre los sistemas de información web y el software convencional. A continuación se listan algunas de ellas:

- Las aplicaciones web se encuentran en constante evolución. Si bien, el software tradicional también cuenta con un periodo de revisión planeado y discreto en momentos específicos de su ciclo de vida, las aplicaciones web se encuentran en continua evolución en términos de sus requerimientos y funcionalidad. Administrar el cambio y la evolución de una aplicación web es un reto técnico, organizacional y de administración mucho más demandante que en el desarrollo de software tradicional.

- La cantidad de usuarios variable, desde un sitio internacional como eBay que recibe millones de usuarios o el portal de la Universidad Autónoma de Querétaro con miles. Estos usuarios puede tener diferentes requerimientos, expectativas y habilidades, por lo tanto, la usabilidad e interfaz de usuario tiene que pensarse para una cantidad de usuarios promedio.

- En los sistemas web hay mucho más énfasis en la creatividad visual y experiencia del usuario.

- La seguridad, privacidad y estabilidad son un punto crítico en los sistemas de información web, ya que las aplicaciones pueden presentar ramificaciones

de errores o la insatisfacción de los usuarios y esto en los sistemas web causa mucho más impacto que en el software tradicional.

- El tiempo y los equipos de desarrollo son más compactos y la presión es mucha más. Por esto, una metodología o proceso de desarrollo que pase de un par de meses a más de un año no es apropiada.

- La tecnología y estándares con la que se desarrollan los sistemas web se encuentra cambiando rápidamente, que si no se tiene el cuidado correcto en las actualizaciones y mantenimiento del sistema, causa inestabilidad tecnológica.

- La presentación de los sistemas de información es diferente a la forma en que se presenta el software tradicional. Con el software web se tiene una variedad de pantallas, dispositivos, hardware, software y redes que tenemos que contemplar en el diseño y desarrollo del sistema.

Estas características multidisciplinarias y necesidades únicas de la web son las que provocan que se requiera que una disciplina especial se encargue de su desarrollo. Pero no solamente una nueva disciplina es necesaria para llevar a cabo el desarrollo de sistemas de información web.

Una encuesta reciente sobre el desarrollo de proyectos basados en la web llevado acabo por Cutter Consortium reporta serios errores (Deshpande Y., 2002):

- Los sistemas entregados no cumplen con las necesidades del negocio 84% de las veces.

- Los tiempos de la planeación se retrasan 79% de las veces.

- Los proyectos exceden el presupuesto 63% de las veces.

- El 53% de las veces, los sistemas entregados no tienen los requerimientos de funcionalidad.

- Finalmente, el 52% de los proyectos no tenían la calidad necesaria.

Esto se debe en su mayoría a que no existen un enfoque uniforme definido para el desarrollo y que los desarrolladores necesitan nuevas tecnologías para hacer su trabajo. Para solucionar esto, muchas metodologías se han propuesto.

4. METODOLOGIAS DE DESARROLLO DE SOFTWARE TRADICIONAL

4.1. Definición

Para entender de manera clara qué son las metodologías de desarrollo, debemos comprender primero qué es un método. Un método, sin contexto, es un conjunto de técnicas, herramientas y procesos que brindan soporte a quien les emplea y facilita lograr un objetivo o alcanzar una meta (Laboratorio Nacional de Calidad del Software, 2009).

Las metodologías son la combinación de métodos y procesos que buscan generar un resultado exitoso y que este resultado pueda ser predecible y repetible las veces que se aplique la metodología.

Podemos entonces, definir una metodología de software como un proceso detallado, planificado y completo, que se lleva a cabo con el fin de formalizar y optimizar este proceso, sirve para saber que hacer durante el ciclo de vida de un proyecto para lograr un objetivo, que por lo regular es un producto de calidad que se desarrolló dentro del tiempo y presupuesto establecidos para cubrir una necesidad.

4.2. Importancia

Pero ¿Por qué son necesarias las metodologías en el desarrollo de software? Para terminar de entender cuál es la necesidad de una metodología tenemos que entrar más a detalle. Primero definiendo cuál es el proceso que se lleva a cabo para crear software.

A este proceso se le llama ciclo de vida del software. Este contempla las etapas por las que tenemos que pasar para obtener un producto final, entre ellas podemos encontrar de manera general: la ingeniería de requerimientos, la fase de diseño, implementación, pruebas, validación, entrega, mantenimiento, etc.

Las metodologías son importantes durante el desarrollo de software, ya que sirven como guías para los equipos de desarrollo durante cada una de las fases del proyecto. Dependiendo de la metodología que se utilice, se contemplan diferentes fases y diferentes productos entregados en cada fase. Estos productos pueden ser documentos de requerimientos, documentación del desarrollo, planes de pruebas o el software como tal.

En algunas de las metodologías una etapa generalmente depende de que la etapa anterior esté finalizada. A este tipo de metodología se le conoce como lineal, y existen muchas otras metodologías que están basadas en este concepto, con algunas variantes.

Esto quiere decir que el programador no puede comenzar la etapa de desarrollo si el diseñador aún no ha terminado las interfaces, o que el ingeniero de pruebas no puede comenzar a hacerlas hasta que el software que probará esté terminado.

En este tipo de metodologías, un error al inicio del ciclo de vida, como un requerimiento mal definido, a la larga puede traer consecuencias enormes. Imaginemos el error inicial como una pequeña bola de nieve que, al final de la colina, termina siendo una bola enorme. El error de una mala definición de requerimiento que es descubierto en la etapa de pruebas puede significar mucho más que un simple acarreo de error. Mientras más se avanza en el ciclo de vida de software, es mucho más costoso el cambio, ya que muchas veces impacta a etapas anteriores. Un error en etapas avanzadas puede provocar que etapas que habían cumplido con sus tiempos, tengan que volver a realizarse para corregir el error, esto ocasiona que todas las etapas por las que ya había pasado el producto tengan que repetirse, extendiendo el tiempo de desarrollo.

Pero el tiempo como tal no es el mayor de nuestros problemas, sino el costo que representa, ya que este tiempo extra provocado por errores de proyecto no es

absorbido por el cliente si el error no fue de él. Esto nos deja con un proyecto en el que se hizo el doble de trabajo, doble de tiempo de desarrollo y el mismo costo con menor ganancia.

En resumen estos pequeños errores, al final resultan en costos y tiempos de proyecto que exceden los planteados en un inicio, que muchas veces la empresa de software tiene que absorber. Como se planteó anteriormente, el objetivo de una metodología es precisamente generar un producto de calidad, en tiempo y costo.

4.3. Beneficios

Como se mencionó anteriormente, la principal ventaja de una metodología es llegar a un buen resultado de manera exitosa dentro del tiempo y costo establecido. Aunque esto es correcto, podemos visualizar las ventajas que tiene adoptar una metodología desde los diferentes enfoques en los que impacta.

El primero de ellos es el enfoque de administración o gestión, ya que al aplicar una metodología es mucho más fácil planificar, controlar y dar seguimiento a un proyecto, debido a que se establecen fases que podemos pueden ser supervisadas durante el desarrollo del proyecto, desde las peticiones del cliente hasta los resultados del desarrollo. (Laboratorio Nacional de Calidad del Software, 2009)

Otro punto de gestión que impacta el uso de una metodología es en el uso de recursos disponibles, ya que se establecen tareas a grupos de trabajo en específico, que al ser entregadas liberan al equipo de trabajo para realizar más actividades en ese o en otro proyecto.

Esto a su vez nos ayuda a mejorar la relación del costo/beneficio. Finalmente, al ser un trabajo dividido en etapas, se mejora la comunicación de los diferentes grupos de trabajo entre ellos mismos y con el cliente.

Desde el punto de vista de los desarrolladores de software, ayuda a mejorar la visión y comprensión del problema que resolverán. Debido a que una buena metodología refleja una buena documentación, al final del desarrollo del proyecto las fases posteriores de mantenimiento y corrección se vuelven una tarea más fácil, y en muchos de los casos, les permite tener módulos o código que puede reutilizarse en proyectos posteriores.

Y finalmente, desde el punto de vista del cliente o los usuarios, una metodología representa un buen nivel de calidad del producto al finalizar el proyecto, y que aparte se entregará en los tiempos y presupuesto acordados desde el comienzo del proyecto.

Estas ventajas nos indican el impacto que tiene elegir una metodología al momento de trabajan un proyecto, ya que se vuelve transcendental para el éxito del producto. La primera decisión que debemos de hacer al momento de pensar en una metodología es si queremos trabajar con una metodología tradicional o una metodología ágil.

4.4. Categorías

El desarrollo de software no es un trabajo fácil y podríamos decir que las metodologías son propuestas de etapas y pasos que podemos seguir para llevar a cabo un proyecto. Existen diferentes tipos de propuestas, o metodologías pero podemos clasificarlas en dos tipos principales: las pesadas y las ágiles.

Por una parte tenemos aquellas más tradicionales o pesadas que se enfocan esencialmente en el control del proceso, que define de manera estricta las actividades a desarrollar, los productos que deben de ser producidos en cada una de ellas y las herramientas y notaciones que contemplarán. Si bien este tipo de metodologías han demostrado su efectividad en un gran número de proyectos desarrollados, también han sido inútiles en otros.

Observando los puntos débiles de las metodologías tradicionales surgieron versiones modificadas que buscaron mejorar incluyendo más actividades que no se contemplaban en el proceso de desarrollo, más restricciones y productos entregables para cada etapa hasta llegar a un punto en el que se cubran cada una de las diferentes etapas del ciclo de vida del proyecto de manera detallada (Laboratorio Nacional de Calidad del Software, 2009). Sin embargo, el resultado de esto sería una metodología de desarrollo más robusta que inclusive podría entorpecer el ciclo de vida del proyecto.

Por otra parte, las metodologías denominadas ágiles se centran en otros aspectos, como el factor humano y el producto final. Estas nuevas filosofías de desarrollo le dan un peso mayor a los miembros del equipo individualmente y a la relación con el cliente, y se basan en desarrollar de manera incremental el software con pequeñas iteraciones. (Mohammed & A., 2010)

Este nuevo enfoque demuestra su efectividad en proyectos en los que los requisitos se encuentran cambiando de manera constante, o cuando se necesita reducir los tiempos de desarrollo sin afectar los requisitos de una calidad alta.

Las metodologías ágiles son un resultado de la evolución del proceso de software que ha traído consigo diferentes propuestas para mejorar el desarrollo de software y está generando una revolución en la manera en que se construye el software, pero algunos escépticos siguen sin verla como alternativa a las metodologías tradicionales (Laboratorio Nacional de Calidad del Software, 2009).

En resumen podemos encontrar dos filosofías diferentes para el desarrollo de software. La tradicional, que hace énfasis en la planificación, documentación y control de proyecto, y las ágiles, que hacen énfasis en la generación de software en ciclos cortos, involucran al cliente en el proceso y se enfocan en la habilidad del factor humano.

4.5. Metodologías Pesadas

Las metodologías tradicionales o pesadas son las primeras metodologías que surgieron. Estas centran su atención en la documentación de todo el proyecto, que suele ser bastante exhaustiva, y en llevar a cabo el desarrollo del proyecto conforme a un plan establecido previamente en la fase inicial del proyecto. Debido a la naturaleza nada flexible de este tipo metodologías, realizar un cambio en el proyecto impacta de manera fuerte en los costes de este (Laboratorio Nacional de Calidad del Software, 2009).

Los programadores en los años sesentas trabajaban bajo el modelo de codificar y arreglar (Wirth, 2007). No existía diseño formal o análisis en los proyectos. A medida que las exigencias de desarrollo de software en el sector aeroespacial fueron incrementando, los proyectos se volvían más complejos y se les exigía mayor seguridad en la entrega del proyecto. Para esto, Winston Royce en 1970 propuso la primera metodología formal de desarrollo de software, la metodología en cascada (Petersen & C., 2009).

En esta metodología se definen fases, que tienen establecido su tiempo de desarrollo y lo que se entregará al final de la fase. La siguiente fase solo puede comenzar una vez que la fase anterior haya sido completada. Los requerimientos son la tarea al inicio del proyecto, seguidos del análisis y diseño. Esto vuelve al desarrollo del proyecto un proceso disciplinado.

Una de las principales problemáticas con el usuario, es que difícilmente puede dar a entender de manera precisa sus necesidades para que los desarrolladores puedan entenderlas. Esto, combinado con la poca flexibilidad que esta metodología representa, hace que la respuesta a cambios en el proyecto sea bastante lenta y costosa.

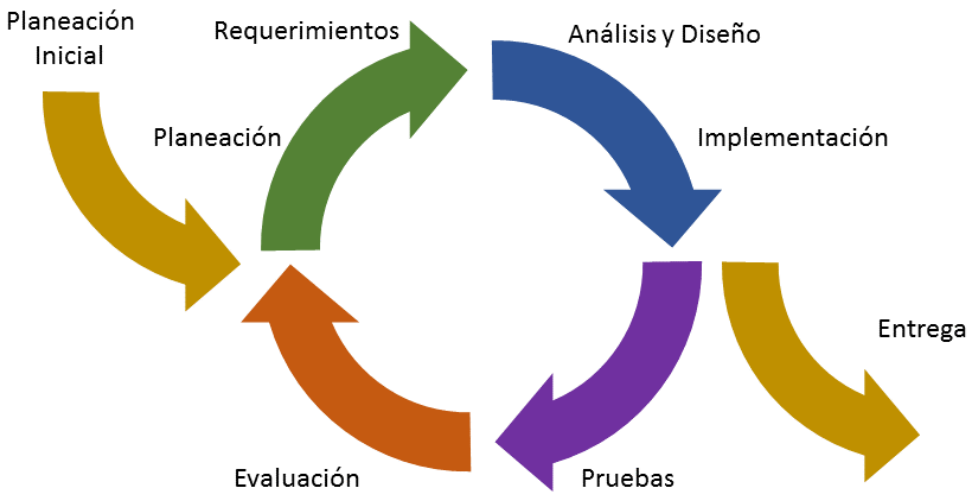
Otro factor que hace difícil adoptar esta metodología es el crecimiento de tamaño y complejidad los sistemas que impacta de distintas formas al desarrollo del proyecto: primero, demanda un grupo de desarrolladores mucho más grande y se vuelve más compleja la comunicación y coordinación; segundo, mientras más grande sea el proyecto, hay muchas más combinaciones de caminos y posibilidades para los usuarios, que resultan en una fase de pruebas bastante exhaustiva y larga. (Laboratorio Nacional de Calidad del Software, 2009)

Lo anterior provoco que las metodologías tradicionales hayan evolucionado para optimizar el modelo en cascada para proveer de una fase llamada prototipado de proyecto, en la que la interacción del usuario con el resultado del análisis del proyecto se encuentra en las primeras etapas en lugar de las últimas, lo que hace que exista un mejor entendimiento de los requerimientos.

También con la evolución de las metodologías tradicionales se busca reducir el tiempo para que el producto se lance al mercado y reducir la presión para la mejora de la calidad y finalmente, lograr proyectos que puedan ser reusables en el futuro.

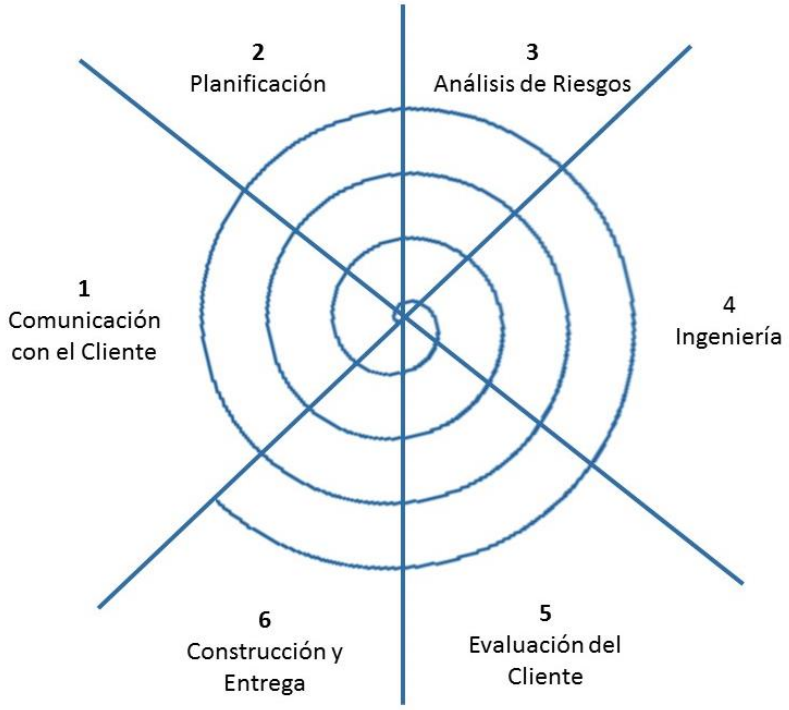
Uno de los resultados de esta evolución es el modelo iterativo e incremental de desarrollo o IID por sus siglas en inglés (Iterative and Incremental Development). Este enfoque fue desarrollado en respuesta a las debilidades de la metodología en cascada básicamente comienza con un plan inicial y termina con la implementación de interacciones cíclicas intermedias (ver figura 4-1). Es iterativo por que se repiten los ciclos de diseño, construcción y pruebas y es incremental por que en cada iteración se agrega funcionalidad hasta que se logre la funcionalidad deseada.

Figura 4-1 Gráfica del ciclo de vida de la metodología *Iterative and Incremental Development.* Fuente: Elaboración Propia.



A partir de la metodóloga IID surge una metodología llamada modelo espiral propuesta por Larry Boehm en 1988, la cual también consistía en iteraciones. Sin embargo, esta metodología está pensada en proyectos largos, caros y complicados, por lo tanto las iteraciones son típicamente de 6 meses a 2 años. Esta metodología combina el modelo en cascada y el prototipado, pero su objetivo es cambiar el énfasis a la evaluación y resolución de riesgos (Boehm, 2000).

Figura 4-2 Gráfica del ciclo de vida de la metodología Espiral. Fuente: Elaboración Propia



Las metodologías mencionadas anteriormente son algunas de las principales metodologías tradicionales o pesadas, y aunque pareciera que tienen enfoques diferentes, todas las metodologías pesadas tienen características en común.

Todas tienen un enfoque predictivo, debido a que tienen la tendencia de primero planear el proceso de software en gran detalle en un largo periodo de tiempo, este enfoque sigue una disciplina de ingeniería donde el desarrollo es predecible y repetible. Otra característica que comparten es la documentación, como el documento de requerimientos que es visto por este tipo de metodologías como una pieza clave en la documentación (Balaji, 2012). El objetivo de las metodologías tradicionales es definir un proceso en el que se deban realizar ciertas

tareas que tienen un procedimiento bien definido y que funcione a quien quiera que aplique de manera correcta la metodología. Finalmente, se enfocan en las herramientas, ya sea de administración de proyectos o desarrollo de software, pero se deben de aplicar para la finalización y entrega de cada una de las tareas.

4.6. Metodologías Ligeras

Las metodologías ligeras o ágiles nacen como respuesta a los problemas ocasionados por las metodologías tradicionales en febrero del 2001, cuando un grupo de desarrolladores de software se reunieron para discutir metodologías de desarrollo ligeras. Después de esta reunión publicaron el "Manifiesto para el Desarrollo de Software Ágil" (*Manifesto for Agile Software Delevopment*) (Serena Software, 2007). En él se definía el nuevo enfoque de desarrollo llamado "ágil", en el que según ellos, estaban descubriendo mejores maneras de desarrollar software haciéndolo y ayudando a otros a hacerlo.

Esta nueva filosofía de desarrollo se basa en la adaptabilidad de los procesos de desarrollo y dan más relevancia a la capacidad de respuesta a un cambio que al seguimiento estricto de un plan. Quien usa esta metodología tiene que entender que: son más valiosos los individuos e interacciones involucradas en el proyecto que los procesos y herramientas, se prefiere trabajar directamente en el desarrollo de software que en documentación robusta y la colaboración del cliente es mucho más valiosa que un contrato (Lindstrom & Jeffires, 2003).

Estas metodologías cuentan con algunos principios clave que hacen la diferencia con las tradicionales. Primero que nada, la satisfacción del cliente es lograda médiate las entregas rápidas y continuas de software útil en semanas en lugar de meses como en las tradicionales, siendo el mismo software la principal medida del progreso del proyecto.

Otro principio clave es la flexibilidad a cambios y la comunicación cercana cara a cara y diaria entre usuarios o clientes y los desarrolladores. Como se mencionó anteriormente, esta metodología se basa en el individuo en lugar de los procesos, por lo tanto el individuo se vuelve un elemento clave que debe ser motivado y en el que se tiene que tener plena confianza, ya que depende de cada uno de ellos la atención que se tenga a la excelencia técnica y buen diseño, además de que la mayoría de las veces los grupos de trabajo se encuentran organizados y liderados por ellos mismos.

Una de las principales metodologías agiles es Scrum que es un proceso de administración para proyectos de desarrollo de software iterativos e incrementales en donde al final de cada iteración se produce un conjunto potencial de funcionalidad (Shwaber & J., 2011). La principal característica de Scrum son los equipos organizados por sí mismos decidiendo qué hacer, mientras el administrador solamente se encarga de remover los obstáculos que puedan encontrar.

Esta metodología cuenta con algunas características clave que la distinguen, una de ellas es que los clientes o usuarios del sistema desarrollado se vuelven una parte importante en el equipo de desarrollo ya que de manera frecuente se le entregan partes funcionales del sistema para recibir retroalimentación inmediata.

Cada miembro del equipo con una tarea es revisado a diario para tener un estatus real del avance del proyecto, y como se comentó anteriormente, si es que hubiera alguna dificultad que se le haya presentado al miembro del equipo, el administrador del proyecto busca la manera de solucionar los obstáculos. Finalmente, debido a que puede haber cambios importantes en cualquier etapa del proyecto, el mismo equipo prepara planes para mitigar riesgos (Schwaber).

La metodología Scrum se basa principalmente en la administración del proyecto pero esto no quiere decir que todas las metodologías ágiles funcionen de

la misma forma. Una metodología ágil que funciona de manera diferente es la de Extreme Programming.

Esta metodología, también llamada XP, es iterativa e incremental, pero a diferencia de Scrum esta se basa en la programación directamente. Muchos equipos de trabajo optan por utilizar estas dos metodologías al mismo tiempo pero no son del todo compatibles (Lindstrom & Jeffires, 2003). Mientras las iteraciones de Scrum son de 4 semanas, las iteraciones en XP son por lo regular de dos. Otra diferencia es que Scrum no permite cambios en los tiempos de sus iteraciones, mientras XP es mucho más flexible para hacer estos cambios.

5. ANÁLISIS METODOLOGÍAS TRADICIONALES (PESADAS Y LIGERAS)

El desarrollo y la Ingeniería Web se han convertido en la plataforma en progreso en esta década, y muchos usuarios y compañías están apostando cada vez más por esta como medio para alcanzar sus objetivos en línea. La manera en que esto afecta a los desarrolladores es que los proyectos tienen que ser logrados bajo la presión de entregar un sistema web de calidad de manera rápida y en realidad hay muy pocas metodologías que respalden este trabajo.

Existen muchas personas desarrollando sitios web y muchos de ellos fuera de las funciones de los sistemas de información tradicionales, por lo tanto sin utilizar una metodología formalizada. Muchos desarrolladores web inventan sus propias metodologías durante el desarrollo con la esperanza de que el producto sirva a las metas de la organización. (Russo & Graham, 1998)

Según investigaciones, muchas metodologías de desarrollo de sistemas de información están basadas en conceptos fuera de moda de los años 80's y estas metodologías son utilizadas para desarrollar sitios web, no sorprendentemente, están limitadas debido a que nunca tuvieron la intención de ser usadas para este propósito. (Howcroft & Carroll).

La llegada de herramientas de creación y manejo fácil de un sitio web parece haber trivializado la necesidad de una planeación cuidadosa, previsión, y una metodología sistemática de diseño. (Balasubramanian & Bashian, 1998). Pero escoger una metodología de desarrollo se vuelve algo complicado debido a la falta de metodologías web formales y escoger metodologías tradicionales tampoco es la alternativa correcta.

Tratar de utilizar las metodologías de desarrollo de software tradicional en un proyecto web de manera completa no es funcional, debido al conjunto de características y requisitos especiales que representa la web. Sin embargo, algunos

de los principios clave de estas pueden ser aplicados de manera exitosa. Por esto, como primer paso en la creación de una nueva metodología de desarrollo de sistemas de información web debemos de considerar y analizar las metodologías de desarrollo de software tradicional y sus etapas para verificar su aplicabilidad en el proceso de creación de un sistema web.

5.1. Análisis Metodologías Tradicionales Pesadas

51.1. Cascada

La primera publicación del modelo en cascada es acreditada a un artículo de Walter Royce en 1970. (Petersen & C., 2009). Este es también llamado modelo de ciclo de vida lineal-secuencial y es el primer modelo en ser aplicado a la Ingeniería de Software. Este consiste en etapas o fases que deben de ser completadas antes de que la siguiente fase pueda comenzar y no existe algún tipo de superposición entre etapas.

El modelo en cascada hace énfasis en la planeación en las primeras etapas y de esta manera asegura que no haya fallas en el diseño antes de comenzar a desarrollar. En adición a esto, su documentación y planeación intensiva hacen que funcione bien para proyecto en los cuales el control de calidad es un tema importante. (Mohammed & A., 2010).

Etapa 1: Requerimientos del Sistema.

En esta etapa se establecen los componentes que serán necesarios para construir el sistema, desde el hardware, herramientas de software y otros componentes. Algunos ejemplos incluyen las decisiones del hardware más adecuado para el sistema, y decisiones de piezas de software externos, como librerías o bases de datos. (Mohammed & A., 2010)

Etapa 2: Requerimientos de Software.

En esta etapa se establecen las expectativas para la funcionalidad del software y se identifican los requerimientos de sistema que el software puede afectar. En esta etapa se analiza la interacción con otros sistemas de software o aplicaciones como bases de datos, interfaces de usuario, requerimientos de rendimiento, entre algunos otros.

Etapa 3: Diseño de la Arquitectura.

En esta etapa se determina la arquitectura del software de un sistema para cumplir con requerimientos específicos. Este diseño define los componentes más grandes y su interacción, pero no define la estructura de cada componente. Las interfaces externas y herramientas usadas en el proyecto pueden ser determinadas por el diseñador. (Mohammed & A., 2010)

Etapa 4: Diseño Detallado.

Examina los componentes de software que se definieron de manera previa en el diseño de la arquitectura y produce una especificación de cómo debe de implementarse cada componente de software. (Mohammed & A., 2010)

Etapa 5: Codificación.

Se implementa la especificación del diseño detallado.

Etapa 6: Pruebas.

Se determina si el software cumple con los requerimientos especificados y encuentra algún error presente en el código.

Etapa 7: Mantenimiento.

Resuelve problemas y peticiones de crecimiento después de que el software es lanzado.

Para cada una de las etapas anteriores se crea un documento que explica cuáles son los objetivos y requerimientos. Al final de cada una de las etapas se hace una revisión que determina si el proyecto puede seguir a la siguiente etapa o necesita ser revisado y corregido para cumplir con sus objetivos. Debido a la naturaleza pesada y lenta de este modelo, muchos proyectos no pueden utilizarlo. A pesar de ser un modelo secuencial, es posible regresar a etapas previas en caso de que se encuentre un error que tenga que ser corregido, sin embargo, estos cambios son bastante costosos en tiempo y presupuesto.

5.1.1.1. Ventajas

- La facilidad en la comprensión e implementación del modelo.
- Establece las bases de buenos hábitos en el desarrollo, como comenzar con el análisis, luego diseño y después la codificación.
- Se documentan los procesos del ciclo de vida del proyecto.
- Es ampliamente usado y conocido.
- Funciona bien en equipos con poca experiencia.

5.1.1.2. Desventajas

- Es irrealista esperar que los requerimientos sean lo suficientemente claros en el inicio del proyecto.
- Al ser la codificación una etapa avanzada en el ciclo de vida, los errores son encontrados ya recorridas etapas en el proyecto.
- Respuesta difícil, lenta y costosa ante cambios en la definición de los requerimientos.

- El exceso de documentación para cada etapa puede entorpecer el desarrollo del proyecto y puede significar exceso de costo en pequeños equipos y proyectos.

5.1.1.3. Matriz FODA

En la siguiente matriz se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-1 Matriz FODA Modelo en Cascada. Fuente: Elaboración propia.

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas:** Facilidad de Aplicación Etapas sencillas y completas Análisis como primera fase Documentación en todas las etapas	**Debilidades:** Etapas forzosamente secuenciales Etapa de pruebas tardía Mala respuesta ante cambios Documentación extensa
EXTERNO	**Oportunidades:** Iteración entre etapas para revisión Pruebas sobre requerimientos Etapas trabajadas de manera paralela	**Amenazas:** Cambios en los requerimientos del cliente Los cambios en los requerimientos técnicos de la web

5.1.1.5. Conclusión

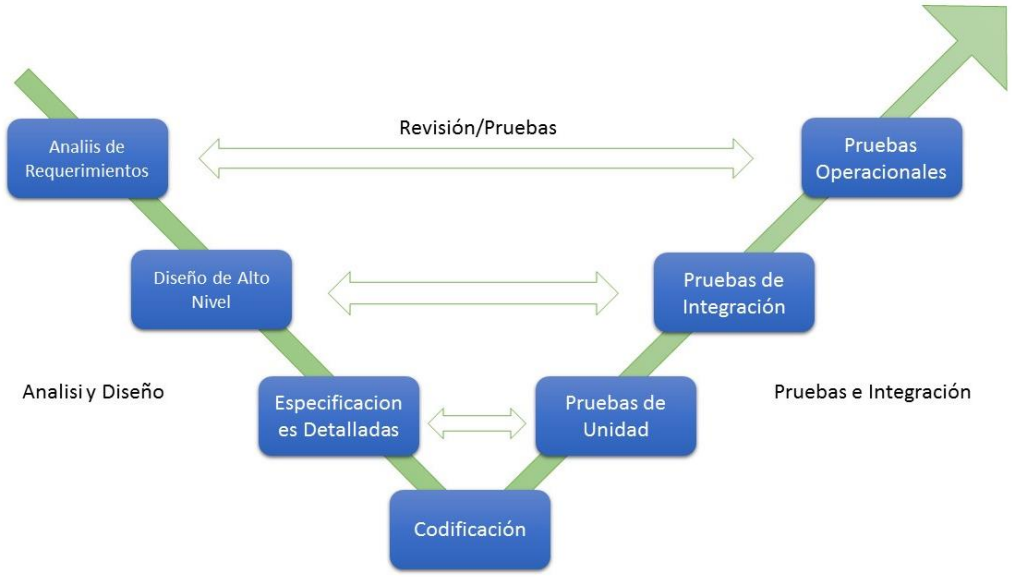
Es un modelo de los más fáciles de implementar y establece las bases de un buen desarrollo debido a su organización en etapas comenzando por el análisis, pero debido a la naturaleza cambiante de la web, este modelo se encuentra bastante limitado en su respuesta al cambio resultando algo costoso y difícil de lograr.

5.1.2. Modelo V

El modelo V, o modelo de validación y verificación, es una versión modificada del modelo en cascada. Al contrario del modelo en cascada, este modelo no fue diseñado en un eje lineal, en lugar de esto las etapas regresan hacia arriba después de que la fase de codificación termina (Balaji, 2012). Otra diferencia entre estos dos modelos es el énfasis que existe en la etapa de pruebas que es llevada a cabo en etapas tempranas del proyecto desde antes de comenzar a codificar. Para las pruebas se crea un plan de pruebas antes de que comience el desarrollo y justo después de la fase de requerimientos. Este plan se enfoca en cubrir las funcionalidades especificadas en los requerimientos.

La fase de diseño de alto nivel se enfoca en el diseño de la arquitectura y del sistema. Un plan de pruebas de integración es creado en esta fase para probar la habilidad de las piezas del sistema de software para trabajar juntas. Sin embargo, la fase de diseño de bajo nivel es donde los componentes de software verdadero son diseñados, y las pruebas de unidad también son creadas en esta fase. (Mohammed & A., 2010).

Figura 5-2 Modelo V de Ciclo de Vida del Software. Fuente: Elaboración propia.



5.1.2.1. Ventajas

- Al ser basado en el modelo en cascada es fácil de usar e implementar en un proyecto.
- Para proyectos pequeños funciona de mejor manera, ya que los requerimientos son fáciles de entender.
- Cada etapa del ciclo de vida tiene sus propios entregables del producto, plan de pruebas y documentación.
- Debido a los planes de pruebas desarrollados al comienzo del proyecto, tiene una mayor oportunidad de ser exitoso a comparación del modelo en cascada.

5.1.2.2. Desventajas

- La principal desventaja de este modelo es que es bastante rígido y poco flexible a cambios en requerimientos en cualquier etapa.
- Los cambios impactan a todo el proyecto. Si es que algún cambio surgiera en cualquier etapa, no solamente se cambiaría el producto y la documentación de requerimientos, también afecta directamente al plan de pruebas.
- El software se produce durante la etapa de implementación, así que no existen prototipos en etapas anteriores.
- A pesar de que este modelo se basa fuertemente en las pruebas del software desde etapas tempranas, no ofrece una solución clara para resolverlos.

5.1.2.3. Matriz FODA

En la siguiente matriz (ver figura 5-3) se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-3 Matriz FODA Modelo V. Fuente: Elaboración Propia

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas:** Facilidad de Aplicación Análisis como primera fase Plan de pruebas sobre requerimientos Documentación y entregables en todas las etapas Pruebas y desarrollo de manera paralela	**Debilidades:** Etapas forzosamente secuenciales Mala respuesta ante cambios de requerimientos A pesar del plan de pruebas no existen una solución clara
EXTERNO	**Oportunidades:** Iteración entre etapas para revisión Desarrollo de prototipos en las etapas tempranas	**Amenazas:** Cambios en los requerimientos del cliente Los cambios en los requerimientos técnicos de la web

5.1.2.4. Conclusión

Una de las características más importantes de este modelo son las pruebas diseñadas desde la fase de requerimientos y se llevan a cabo en cada una de las etapas. Al igual que el modelo en cascada, este modelo no se adapta de manera correcta a la naturaleza cambiante de la web y tiene mala respuesta a los cambios de requerimientos establecidos por el cliente al comienzo del proyecto, ya que no existe algún otro tipo de evento que involucre de nuevo al cliente.

5.1.3. Modelo Espiral

El modelo de desarrollo en espiral es un generador de modelo de procesos basado en riesgos. Tiene dos características principales que lo distinguen. Una es un enfoque cíclico para el crecimiento incremental del grado de decisión e implementación de un sistema mientras se reduce el grado de riesgo. (Boehm, 2000)

En este se hace énfasis principalmente en el análisis de los riesgos. Los riesgos para Boehm (2000) son situaciones o posibles eventos que puede ocasionar

que un proyecto falle en sus metas. Su rango de impacto va desde trivial hasta fatal y de seguro a improbable. Un plan de manejo de riesgos enumera los riesgos y los prioriza por grado de importancia.

El modelo en espiral cuenta con cuatro fases: Planeación, Análisis de Riesgos, Ingeniería y Evaluación. Durante el ciclo de vida del proyecto se hacen iteraciones, llamadas espirales en este modelo, por estas cuatro fases.

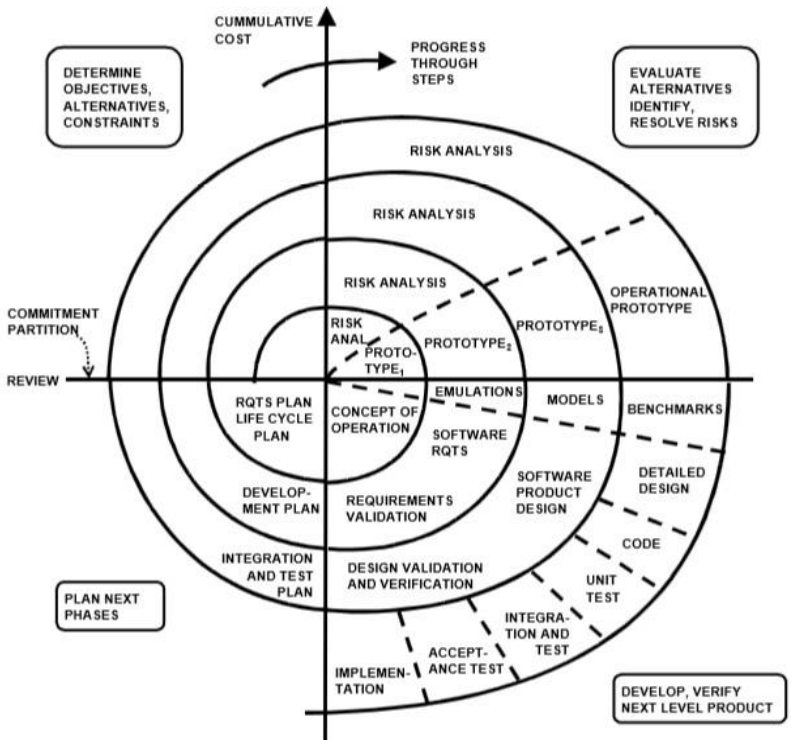
En la espiral base comenzando por la fase de planeación se reúnen los requerimientos y se miden los riesgos. Cada espiral subsecuente se construye sobre la espiral base. Los requerimientos son reunidos durante la fase de planeación. En la fase de análisis de riesgo se lleva a cabo un proceso para identificar los riesgos y sus posibles soluciones y al final de esta fase se crea un prototipo.

El software es producido en la fase de Ingeniería y al final de esta se llevan a cabo las pruebas en la fase de evaluación, necesarias para determinar si el producto cumple con los objetivos o si tiene que ser sometido de nuevo a otra espiral de ciclo de vida.

En este modelo el componente angular representa progreso del proyecto y el radio de la espiral representa el costo. (Mohammed & A., 2010)

Por un número de razones el modelo espiral no es comprendido de manera universal. Algunas de los malentendidos más significativos de este modelo son: La espiral no es más que una secuencia de cascadas incrementales, todo en el proyecto sigue una secuencia en espiral sencilla, cada elemento en el diagrama necesita ser visitado en el orden indicado, y que no se pueden hacer seguimiento para revisar decisiones previas. (Boehm, 2000)

Figura 5-4 Modelo en Espiral del Ciclo de Vida del Software. Fuente: Boehm (2000, p.2)



5.1.3.1. Ventajas

- El análisis de los riesgos se hace de manera intensiva, por lo que se generan planes de contingencia en caso de que sucedieran.
- Es excelente para proyectos de largo alcance.
- El software es producido desde la primera espiral del proyecto aunque en sí el ciclo de vida completo del proyecto sea de varias espirales.
- Las espirales o iteraciones a lo largo del proyecto permiten refinar y madurar el producto con cada ciclo.

5.1.3.2. Desventajas

- Debido a las iteraciones o espirales completas se vuelve un modelo costoso en tiempo y presupuesto.

- Para llevar a cabo el análisis de riesgos de la forma correcta se requiere mucha experiencia.
- No funciona de manera correcta en proyectos pequeños debido al largo tiempo de las espirales.
- El éxito del proyecto depende en gran manera del análisis de riesgos.

5.1.3.3. Matriz FODA

En la siguiente matriz (ver figura 5-4) se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-5 Matriz FODA Modelo Espiral. Fuente: Elaboración Propia

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas**: Iteraciones o espirales de las fases del ciclo de vida Análisis de posibles riesgos y sus soluciones Maduración del producto en cada espiral Software producido desde la primer espiral	**Debilidades:** Análisis de riesgos requiere experiencia para ser bien diseñado Espirales largas y costosas no factibles para proyectos cortos Modelo complicado de aplicar a un proyecto
EXTERNO	**Oportunidades:** Tiempos de espiral cortos o puntos de fuga para proyectos pequeños Análisis de riesgos diseñado de manera paralela o incremental con base pequeña	**Amenazas:** Riesgos no contemplados en el plan afectan al proyecto Proyectos con maduración extensa si no es bien planeado Cambios en requerimientos revisados en etapa final de la espiral

5.1.3.4. Conclusión

Esta metodología basa gran parte de su éxito en el análisis de riesgos, que tiene que ser diseñado por alguien con la experiencia necesaria para reconocer las posibles fallas desde antes de que sucedan, por lo tanto, esta metodología no es factible para ser usada por alguien que comenzara sin experiencia alguna a desarrollar proyectos. Tampoco es factible para proyectos pequeños con espirales más reducidas, ya que el tiempo entre cada espiral es alto.

5.1.4. Rapid Application Development

En 1990 James Martin idea el término RAD o *Rapid Application Development* que se refiere a un ciclo de vida diseñado para desarrollar con resultados de manera más rápida y de más alta calidad que los ciclos de vida tradicionales. (Agarwal, Prasad, Tanniru, & Lynch, 2000). El factor que detiene a las organizaciones para implementar soluciones para incrementar la productividad y ganancias en sus procesos es el alto costo, presupuestos impredecibles y tiempos de entrega largos. (Zeinoun, 2005)

Para solucionar estos problemas surge la metodología RAD, que está basada en el uso de prototipos e iteraciones, lo que significa que el usuario está involucrado en el desarrollo del proyecto y es quien presenta los requerimientos del sistema. El uso de los prototipos representa también que el resultado del proyecto representa con fidelidad la solución a los problemas presentados por el usuario y que cumple satisfactoriamente los requerimientos definidos.

Esta metodología funciona mejor en casos donde los datos son conocidos, los requerimientos pueden ser definidos y se mantienen sin cambios durante el desarrollo, y los requerimientos funcionales se pueden encontrar dentro de un marco de tiempo corto con un equipo pequeño (3-4 meses con 5-6 recursos técnicos) (Zeinoun, 2005).

En caso de que los requerimientos del proyecto sean largos, basta con realizar la división del proyecto en varios y tratarlos con la metodología RAD por separado. Con este enfoque se puede descomponer una solución grande y compleja en pequeñas soluciones más sencillas, lo que es benéfico en cuanto a tiempos de entrega más cortos con los que se puede demostrar el progreso del proyecto y a su vez la aprobación del presupuesto y requerimientos si es que necesitan incrementarse.

Las fases de esta metodología son similares a las de las otras tradicionales, pero en esta se hace énfasis en el prototipado y análisis del proyecto. Ambas fases son anteriores al resto de las fases en el ciclo de vida de un proyecto. A continuación se presenta el orden cronológico de las fases de esta metodología:

- Fase de Prototipo:
  Durante esta fase el cliente o usuario se involucra por completo para construir un caso de negocio, en el que se especifican y solidifican las metas que se espera se cumplan con el desarrollo del sistema de información.

- Fase de Análisis y Planeación
  Se crea un contrato llamado Declaración del Trabajo o en si siglas en ingles SOW (Statement of Work) en el que se incluyen las firmas de la aprobación del cliente para comenzar el proyecto bajo el entendimiento del equipo de desarrollo. También se construye la matriz de funcionalidad.

- Fase de Diseño
  Esta incluye los diseños técnicos y funcionales del proyecto, al finalizar esta fase la matriz de rastreo es completada.

- Fase de Construcción

Esta fase es desarrollada haciendo por lo menos tres iteraciones en las que se presenta los avances del proyecto al cliente al finalizar la primera iteración para recibir realimentación y continuar con la siguiente iteración.

- Fase de Pruebas
  Al llegar a esta fase se hacen iteraciones pequeñas para corregir errores de código y se toma en cuenta más realimentación del usuario solo para mejoras menores.

- Fase de Implementación
  Durante esta fase se hacen las últimas revisiones y correcciones de entrega e incluye la interfaz a los datos de producción.

La siguiente tabla (ver tabla 4-1) muestra el porcentaje y tiempo de duración típico de un proyecto cuando la metodología RAD es implementada.

Tabla 5-1 Duración típica de las fases de proyectos con metodología RAD. Fuente: Zeinoun (2005).

FASE	LONGITUD EN %	SEMANAS (20 TOTAL)
PROTOTIPO	15%	3 semanas
ANALSIS Y PLANEACIÓN	10%	2 semanas
DISEÑO	15%	3 semanas
CONSTRUCCIÓN	30%	6 semanas
PRUEBAS	15%	3 semanas

IMPLEMENTACIÓN	15%	3 semanas

Existen diferentes tipos de roles con diferentes responsabilidades en las distitnas etapas de la metodología RAD. CASEMaker Totem (2000) describe los siguientes roles:

- Patrocinador del Proyecto:
  Es quien o quienes definen las metas del proyecto y las comunican al equipo cliente y al gerente del proyecto. También es quien aprueba el comienzo y el final del proyecto en el documento SOW y asigna y financia el presupuesto para el proyecto.

- Gerente del Proyecto:
  Trabaja con el patrocinador para definir las metas y el criterio de éxito. Junto con el líder técnico asigna los recursos, calcula el presupuesto para el proyecto, y planea cada fase y entrega para mitigar riesgos de calendarización. Es también quien se encarga de dar seguimiento al progreso del proyecto diariamente y se asegura de que se cumplan las metas en el tiempo indicado.

- Analista de Negocios:
  Su rol es el de ayudar a los usuarios a analizar su problema de negocios aprendiendo y entendiendo los procesos y cultura del negocio del cliente para poder ofrecerles la solución correcta. De esta manera, el analista ayuda a definir y refinar los requerimientos funcionales, las pantallas y flujos del sistema. En manos de esta persona se encuentra la comunicación entre el usuario y el equipo técnico, los documentos de definición de la aplicación y los documentos de casos y planes de pruebas.

- Líder técnico:

Es su responsabilidad producir liderar el equipo técnico durante la producción del prototipo y el producto final, estableciendo la arquitectura y los estándares de codificación. Durante los procesos de desarrollo el líder técnico es el encargado de supervisar el progreso y rendimiento de los desarrolladores también participa activamente si es que el proyecto lo requiere, ya sea asesorando o inclusive codificando módulos y llevando a cabo pruebas de integración.

- Equipo Técnico:
  El equipo técnico consiste en un grupo de personas con diferentes áreas de experiencia. Este equipo lleva a cabo el desarrollo del proyecto bajo la supervisión directa del líder técnico. Este equipo también es llamado "equipo SWAT" debido a las siglas de su significado en inglés (*Skilled Workers with Advanced Tools* – Trabajadores Habiles con Herramientas Avanzadas).

- Equipo de Usuario:
  Este equipo debe de representar todos los tipos de usuarios de la aplicación. Este equipo cumple un rol muy importante durante todo el proyecto, desde la definición de requerimientos funcionales, pasando por las revisiones durante la construcción y finalmente las pruebas de aceptación y la entrega del proyecto. Sin la participación de este equipo, la usabilidad, funcionabilidad y producto final se ven afectados.

En el siguiente cuadro (ver tabla 4-2) se muestra las actividades y responsabilidades de los roles en las diferentes fases a lo largo del ciclo de vida de la metodología RAD.

Tabla 5-2 Responsabilidades de los distintos roles en la metodología RAD en las distintas fases de desarrollo. P=Patrocinador, EU=Equipo de usuarios, GP=Gerente de Proyecto, AN=Analista de Negocio, LT=Líder Técnico y ET=Equipo Técnico. Fuente: Zeinoun (2005).

FASE	P	EU	GP	AN	LT Y ET
PROTOTIPO	Revisión del prototipo final	Definición del prototipo	-	Obtener requerimien-tos	Traducción de requerimien-tos para construir prototipo
ANÁLISIS Y PLANEA-CIÓN	Firma del documento SOW y del plan	Refinar requerimien-tos	Planeación del Proyecto	Liderar el análisis y refinamientos	Refinar prototipo
DISEÑO	-	Verificar arquitectura y estructura del sistema	Seguimiento de progreso	Punto de contacto entre usuario y desarrollador	Diseñar arquitectura y solución
CONSTRU-CCIÓN	Revisión en Iteraciones	Revisión y Realimenta-ción en las iteraciones	Seguimiento del progreso y resolución de problemas	Pruebas y definición de casos de prueba	Construir solución
PRUEBAS	Firma	Pruebas y Registro de cambios	Obtener firma	Pruebas y Producción de Resultados de Pruebas	Arreglar errores
IMPLEMEN-TACIÓN	-	Pruebas de la implementa-ción	Resolver problemas técnicos y firma en documentos	Prueba aplicación desarrollada y entrenamien-to del usuario	Implementar y arreglar errores y creación de documento de manual de usuario e instalación

5.1.4.1. Ventajas

- La participación del usuario o grupos de usuarios y el patrocinador del sistema durante todo el ciclo de vida del proyecto por medio de los prototipos, hace que el resultado del producto se enfoque en lo que los usuarios definieron a lo largo de todo el proceso en los aspectos de funcionalidad y diseño en lugar de aspectos técnicos de interés del equipo de desarrollo.

- Los requerimientos evolucionan durante el desarrollo del proyecto, por lo tanto, no es necesario comenzar a trabajar si el usuario no tiene claro el todo lo que se requiere para que el sistema funcione.

- El control y mitigación de riesgos son otra característica importante de la metodología RAD, ya que se llevan a cabo pruebas en las etapas tempranas basadas en la evidencia empírica de los requerimientos. El uso de los prototipos incrementales y las pruebas sobre estas hacen que las pequeñas partes de software entregadas estén libre de error.

- Las técnicas utilizadas por RAD aseguran que exista una buena administración y control de tiempos y presupuesto, esto debido a las entregas de unidades incrementales de software que garantizan minimizar el impacto en tiempo y presupuesto ocasionado por fallas a lo largo del tiempo.

5.1.4.2 Desventajas

- Si no existe el compromiso del cliente con el proyecto, este puede fallar. Esto debido a que los usuarios son el elemento clave durante el desarrollo de un proyecto que esté utilizando la metodología RAD, por lo tanto, la organización tiene que prescindir de los servicios de usuarios o grupos de usuarios que directamente interactuarán con el sistema. Una ironía de esto, es que mientras más experto en el área sea el usuario que participa, mejores son los resultados del proyecto, pero debido a que es una persona experta, la

empresa está menos dispuesta a ceder al usuario para trabajar en el desarrollo del proyecto.

- La flexibilidad de la metodología ofrece la posibilidad de adaptarse a cambios rápidamente para resolver problemas o mejorar el producto, sin embargo, el exceso de flexibilidad puede resultar en falta de control de la dirección y estatus real del proyecto, pero si por el contrario hay más control que flexibilidad, el proceso de desarrollo puede ser inmutable. El balance entre control y flexibilidad representa que más de una significa menos de otra.

- Debido a la participación de un grupo de usuarios clave para el desarrollo del proyecto, puede existir el caso en que un proyecto que involucre una cantidad grande de diferentes tipos de usuario y por motivos del costo que representa ceder usuarios para la organización, algún tipo de usuario en específico que no fue requerido podría tener problemas al adaptarse al sistema

- No todo tipo de proyectos puede adaptarse a esta metodología. Por la naturaleza de esta metodología, los proyectos medianos o pequeños encajan mejor para ser desarrollados con RAD que proyectos grandes en los que el control de los procesos y la documentación es crítico.

5.1.4.3 Matriz FODA

En la siguiente matriz (ver figura 5-6) se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-6 Matriz FODA Metodología RAD. Fuente: Elaboración Propia.

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas**: Libre de documentación excesiva Participación del usuario final en el ciclo de vida del proyecto Iteraciones y entregables funcionales Equipo de trabajo compenetrado	**Debilidades:** No aplicable para proyectos de gran escala Falta de documentación Se necesita buena disciplina y habilidad al aplicarlo Técnicas de programación costosas
EXTERNO	**Oportunidades:** Documentación mínima Diseño de análisis de riesgos o pruebas en base a requerimientos	**Amenazas:** Crecimiento inesperado del proyecto Cambios funcionales sobre la entrega base Falta de elementos en el equipo incrementan el tiempo de entrega

5.1.4.4 Conclusiones

RAD es una metodología tradicional que tiene características que trabajan en armonía con el desarrollo web, sobre todo los procesos iterativos que involucran al usuario junto con las porciones funcionales e incrementales de software que son entregados periódicamente. Sin embargo, la flexibilidad en el desarrollo puede salirse de control y volverse un proyecto sin un alcance establecido en el que el usuario tiene el control del estatus del proyecto.

5.2. Análisis Metodologías Tradicionales Ligeras

5.2.1. Metodología Extreme Programming.

Esta es de las más metodologías ágiles más utilizadas. Extreme Programming o XP comparte los valores enunciados en el manifiesto de las metodologías ágiles pero va más allá especificando un conjunto simple de principios. Mientras muchas metodologías tratan de resolver la pregunta “¿Cuáles son todos los principios que alguna vez necesitaré en un proyecto de software?”, Extreme Programming simplemente pregunta, “¿Cuál es el conjunto más simple de principios que podría necesitar y qué es lo que necesito para limitar mis necesidades a esos principios?” (Lindstrom & Jeffires, 2003).

Esta metodología toma cuatro variables como la base del desarrollo de todo proyecto: Costo, Tiempo, Calidad y Alcance. El costo es la cantidad de dinero que será gastada, por ejemplo: los recursos disponibles para el proyecto (programadores, licencias, equipo). El tiempo determina cuándo el sistema debería de ser entregado. La calidad es la exactitud del sistema definida por el usuario. Finalmente el alcance define qué y cuánto tiene que hacerse en cuanto a la funcionalidad del sistema.

Para Dudziak (2000) el tiempo es el elemento central en XP. Las dependencias fundamentales entre esta variable y las otras son:

- Incremento de la calidad puede incrementar el tiempo que es necesario para las pruebas. Restar la calidad puede reducir el tiempo en cierto grado (reduciendo el número de pruebas de funcionalidad).
- Incrementar el costo (contratar más desarrolladores o proveerles mejor equipo) puede significar menos tiempo pero también el efecto opuesto es posible. Restar el costo incrementa el tiempo dramáticamente.

- Incrementar el alcance significa más tiempo necesitado porque hay más trabajo que hacer. Restar el alcance reduce el tiempo. Esta relación es la que tiene mayor control en XP.

Una de las principales prácticas de XP son las pequeñas entregas. Los conjuntos de funcionalidades útiles más pequeñas son desarrolladas primero, después de esto se hacen pequeñas entregas del sistema de manera frecuente que incrementan la funcionalidad de la primera entrega. Estas piezas de software son desarrolladas por pares de programadores que constantemente revisan el trabajo del otro y dan soporte para hacer un buen trabajo.

Antes de que estas entregas sean implementadas al sistema principal, se llevan a cabo pruebas de unidad para cada pieza de funcionalidad que se tenga que agregar. Después de ser revisadas e integradas en el sistema, un representante de los usuarios finales que siempre está disponible para el equipo de XP revisa que lo entregado cumpla con los requerimientos iniciales.

5.2.1.1. Ventajas

1. Los métodos ágiles o ligeros como XP son los indicados para trabajar proyectos pequeños o medianos.
2. Hacen énfasis en el producto final más que en la documentación.
3. El equipo de trabajo y el cliente se integran por completo en el proyecto.
4. Es un proceso iterativo con entregas funcionales.
5. El cliente es una parte importante del equipo de trabajo y se encuentra presente en todo el ciclo de vida del proyecto.

5.2.1.2. Desventajas

1. No se puede escalar a proyectos grandes donde la documentación sea esencial.

2. Puede degenerarse hasta llegar a las metodologías “Programar y Arreglar” si no se tienen las habilidades y experiencias necesarias.
3. La programación en pares es costosa.
4. La construcción de los casos de prueba es una tarea difícil ya que el proyecto se encuentra en constante evolución.

5.2.1.3. Matriz FODA.

En la siguiente matriz (ver figura 5-7) se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-7 Matriz FODA Metodología XP (Extreme Programming). Fuente: Elaboración Propia.

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas:** Libre de documentación excesiva Participación del usuario final en el ciclo de vida del proyecto Iteraciones y entregables funcionales Equipo de trabajo compenetrado	**Debilidades:** No aplicable para proyectos de gran escala Falta de documentación Se necesita buena disciplina y habilidad al aplicarlo Técnicas de programación costosas
EXTERNO	**Oportunidades:** Documentación mínima Diseño de análisis de riesgos o pruebas en base a requerimientos	**Amenazas:** Crecimiento inesperado del proyecto Cambios funcionales sobre la entrega base Falta de elementos en el equipo incrementan el tiempo de entrega

5.2.1.4. Conclusión.

Por su naturaleza corta e iterativa, esta metodología tiene gran porcentaje de adaptación e integración en el entorno web. Las entregas de pequeñas partes de software funcionales ofrecen la posibilidad al cliente y al usuario de ver un avance

del proyecto cercano a su estatus real. Sin embargo, se vuelve una metodología costosa de aplicar por la técnica de programación en pares, y se requiere cierto nivel de experiencia para los casos de prueba que se encuentran en constante evolución y adaptación por los cambios que involucra tener a un representante del usuario final como parte del equipo de desarrollo.

5.2.2. Metodología SCRUM.

En rugby, “scrum” (relacionado con “Scrimmage”) es un término para una masa de jugadores amontonados y comprometidos entre ellos para llevar a cabo un trabajo. En el desarrollo de software, el trabajo es hacer una entrega (Serena Software, 2007). El origen de Scrum está fundado en la teoría de control de procesos empíricos, o empirismo. El empirismo afirma que el conocimiento proviene de la experiencia y tomar decisiones basadas en lo que sabes. Scrum emplea un enfoque incremental e iterativo que optimiza la predictibilidad y el control de riesgos. (Shwaber & J., 2011).

Scrum es un marco de trabajo o *framework* de procesos que ha sido usado para administrar el desarrollo de productos complejos desde 1990. Para Schwaber y Sutherland (2011) Scrum no es un proceso o una técnica para construir productos, sino un *framework* en el cual puedes emplear varios procesos o técnicas.

La principal diferencia entre las metodologías definidas (cascada, espiral, v) y las empíricas como Scrum es que el enfoque que utilizan asume que el análisis, diseño y desarrollo son procesos que pueden llegar a ser impredecibles. Para controlar esta imprevisibilidad se establece un control de mecanismo y control de riesgos, que dan como resultado una metodología de desarrollo flexible, con capacidad de respuesta y confiable.

La primera fase (planeación) y la última (cierre) consisten en procesos, entradas y salidas bien definidas con instrucciones explícitas para llevarse a cabo.

En estas dos fases el flujo es lineal, con algunas iteraciones pequeñas en la fase de planeación.

Entre estas dos fases se encuentra la fase de *sprint* que es la parte del proceso empírico. Dentro de esta fase los procesos son indefinidos y sin control, por lo que requiere que el control y manejo de riesgos se aplique de manera externa en cada iteración del *sprint* para evitar caos pero manteniendo al máximo la flexibilidad.

Los *sprints* son no lineales y flexibles, en donde si existe conocimiento explícito de un proceso se utiliza, pero si no lo hay se utiliza el conocimiento tácito y prueba y error para llevar a cabo el proceso. Cada iteración del *sprint* construye una parte del producto final. Para Schwaber un *Sprint* es un conjunto de actividades de desarrollo conducidas sobre un periodo predefinido, usualmente de una a cuatro semanas. Este intervalo es basado en la complejidad del producto, valoración de los riesgos y grado de supervisión deseado.

Cada *sprint* consiste en uno o más equipos haciendo lo siguiente:

- Desarrollo:
  Definir cambios para la implementación de los requerimientos del *backlog*, o registro de cambios en requerimientos, en los paquetes, abrir los paquetes, hacer un análisis de dominio, diseño, desarrollo, implementación, pruebas, y documentación. El desarrollo consiste en micro procesos de descubrimiento, invención e implementación.

- Envoltura:
  En inglés es *wrap* y es el cierre de los paquetes, creando una versión ejecutable de cambios y cómo se implementan los requerimientos del *backlog*.

- Revisión:
  Todos los equipos se reúnen para presentar el trabajo y revisar el progreso, levantando y resolviendo problemas y cuestiones, añadiendo al *backlog* nuevos elementos. El riesgo es revisado y las respuestas son definidas.

- Ajuste:
  Es la consolidación de la información reunida por la junta de revisión en los paquetes afectados.

Debido a que el proyecto se encuentra abierto en las fases de planeación y *sprint*, el producto entregable es modelado bajo los constantes cambios en la complejidad ambiental y presiones de competitividad, tiempo, calidad y presupuesto. Cuando se llega al objetivo deseado del producto el proyecto entra a la fase de cierre, en donde el producto se cierra a los cambios fuera del proyecto, se hacen las pruebas y se prepara para ser entregado junto con la documentación final. Algunos conceptos importantes de Scrum son los siguientes:

- Gráfico *Burndown:*
  En esta tabla hay actualizaciones cada día, muestra el trabajo restante del *sprint.* La tabla *burndown* es utilizada para dar seguimiento al progreso del sprint y para decidir cuándo algunos elementos deben de ser removidos del *backlog* del *sprint* y diferidos al siguiente sprint.

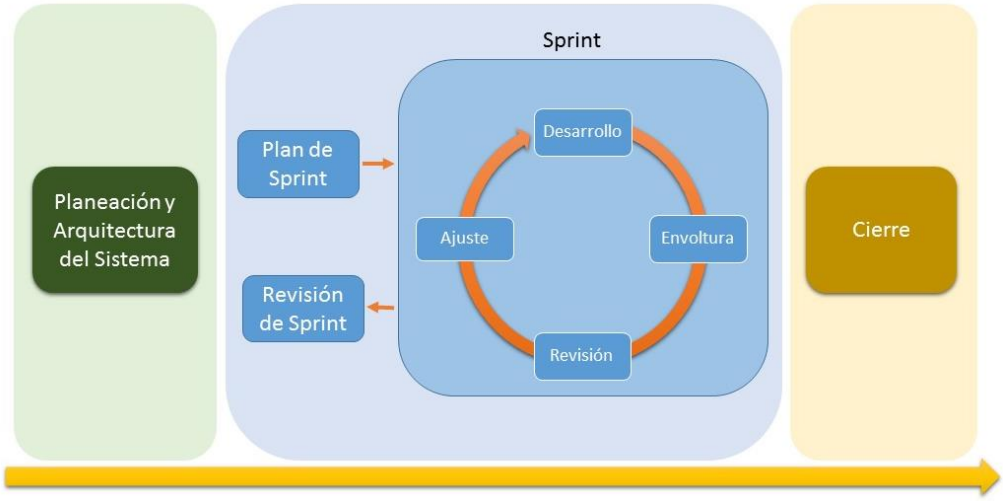
- *Backlog* del Producto:
  Este *backlog* o registro es la lista completa de requerimientos, incluyendo bugs, mejoras, peticiones y mejoras de usabilidad y rendimiento, que no se encuentran en la liberación del producto.

- *ScrumMaster*:

Esta persona es la responsable de administrar el proyecto Scrum. Algunas veces se refiere a la persona que se ha certificado como *ScrumMaster,* tomando el entrenamiento adecuado.

- *Sprint Backlog*:
  Es la lista de elementos del registro asignados a un sprint, pero que aún no han sido completados. Como práctica común, ningún elemento en el *backlog* debe de tardar más de dos días en completarse. Este *backlog* ayuda al equipo a predecir el nivel de esfuerzo requerido para llevar a cabo el sprint.

Figura 5-8 Modelo Scrum del Ciclo de Vida de Software. Fuente: Elaboración Propia.



5.2.2.1. Ventajas

- La metodología está diseñada para ser flexible a cambios durante todo el proceso del ciclo de vida del proyecto.

- Provee un mecanismo de control para la planeación de la liberación de un producto y manejo de variables una vez que el proyecto comienza, lo que permite al cliente hacer cambios en cualquier momento.
- Pequeños equipos de desarrollo colaborativos que son capaces de compartir el conocimiento tácito acerca del desarrollo del proyecto.

5.2.2.2. Desventajas.

- Problemas con los requerimientos no funcionales.
- Debido a la falta de documentación por fase, las decisiones tomadas sobre los cambios en el desarrollo son difíciles de rastrear.
- La planeación, estructura y organización de un proyecto se vuelven difíciles ya que no existe una definición clara desde el comienzo.
- Los cambios constantes durante todo el ciclo de vida del proyecto vuelven incierta la naturaleza precisa del producto terminado, por lo que se vuelve un proceso intenso para los involucrados.

5.2.2.3. Matriz FODA.

En la siguiente matriz (ver figura 5-8) se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-9 Matriz FODA Metodología Scrum

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas**: Flexibilidad durante todo el desarrollo Las iteraciones dentro de los sprint hacen que el producto madure exitosamente Transparencia durante el desarrollo acerca del estatus del producto Mejoras en la comunicación del equipo por las reuniones constantes	**Debilidades:** La falta de documentación durante el desarrollo Cambios por parte del cliente en cualquier momento No se sabe precisamente que tan completado esta el proyecto No es posible crear una planeación debido a que no existe una definición clara
EXTERNO	**Oportunidades:** Histograma de cambios Establecer tiempo de cambios del cliente Trabajar sobre un plan inicial general al que se le puedan hacer modificaciones no tan cruciales	**Amenazas:** Cambios críticos importantes por el cliente Crecimiento del proyecto a gran escala debido a la falta de limites Mala comunicación entre *sprint*s. Sin historial de cambios se puede repetir trabajo hecho previamente.

5.2.2.4. Conclusión.

Esta metodología se adapta muy bien a los cambios que puedan ocurrir durante todo el tiempo de desarrollo, por lo que es adecuada para trabajar con sistemas de información web. Sin embargo, los constantes cambios durante el ciclo de vida del proyecto pueden representar un problema al momento de definir el estado del producto final si no se tienen las bases de una carta de alcance del proyecto, tanto que los proyectos pueden llegar a crecer en tiempo y complejidad.

5.2.3. Metodología DSDM

La metodología DSDM, por las siglas en ingles de Dynamic System Development Method, o Metodología de Desarrollo de Sistemas Dinámicos, se centra en entregar un producto tan rápido como sea posible y contiene una guía de cómo controlar el proceso al mismo tiempo. Para Gal “esta metodología no tiene

técnicas definidas, en lugar de esto se sugieren caminos de enfoques estructurados y orientados a objetos. Esta metodología consiste de cada parte del desarrollo de sistemas, desde la idea del proyecto hasta la última solución del proyecto".

Esta metodología se basa en nueve principios que deben de ser seguidos. Estos principios son una guía para el proyecto hacia el éxito basándose en la integración del usuario y las necesidades del negocio.

- Principio 1: Participación activa del usuario.
  El principio más importante es la participación de manera activa durante todo el proyecto de un usuario con conocimientos del propósito del sistema y de las necesidades del negocio, de esta manera ellos se asegurarán de que no existan malos entendidos de lo que el sistema debería hacer.

- Principio 2: El equipo debe de estar empoderado para tomar decisiones.
  De esta manera el equipo de desarrollo puede tomar decisiones por sí mismo en cualquier momento para no entorpecer el flujo de trabajo. Para que esto se lleve a cabo de forma correcta se necesitan lineamientos y reglas y no pueden sobrepasar el presupuesto asignado del proyecto. Algunas decisiones que se pueden hacer son: cambios en requerimientos, aumentos de funcionalidades, priorización de entregables, entre algunos otros.

- Principio 3: Enfocarse en entregas frecuentes.
  Las entregas frecuentes de resultados aseguran que todos los errores fueron encontrados de manera rápida, son fácilmente reversibles y son más cercanas a la fuente del error. Esto aplica tanto a código del programa como a los requerimientos.

- Principio 4: La aptitud para los negocios es esencial para la aceptación de entregables.

El propósito de la metodología es entregar software que sea lo suficientemente bueno en su funcionalidad operacional para resolver las necesidades del negocio y mejoras posteriores. Para Voigt (2004) cuando se desarrollan sistemas en esta metodología “es más importante asegurarse de que se desarrolla lo correcto y después que se está desarrollando de la mejor manera y con la mejor calidad”.

- Principio 5: Desarrollo Iterativo e Incremental obligatorio.
  Para mantener la complejidad del proyecto lo más manejable posible es necesario que fragmentemos el proyecto en pequeñas partes y en cada entrega se añaden las nueva características hasta que el proyecto cumpla su objetivo deseado. Como menciona Voigt (2014) “este principio requiere aceptar el hecho de que cualquier sistema de software está sujeto al cambio”.

- Principio 6: Todos los cambios durante el desarrollo deben de ser reversibles.
  Poder responder antes los cambios requiere que las configuraciones del sistema puedan estar cambiando durante el desarrollo de cualquier incremento debido a los cambios en las prioridades de los requerimientos (Voigt, 2004). Normalmente el ejercicio de regresar o revertir parte del proceso de desarrollo es temido debido a que puede haber perdidas en el trabajo realizado, sin embargo, esta metodología especifica que el código debe desarrollarse en pequeñas porciones, por lo que el trabajo que se pudiera llegar a perder es muy poco.

- Principio 7: Los requerimientos están basados en un nivel alto.
  Todos los requerimientos obtenidos durante las primeras etapas del desarrollo se convierten en requerimiento de alto nivel para el proyecto, de esta manera se pueden cumplir de mejor manera durante los procesos iterativos del proyecto. Estos requerimientos de alto nivel no pueden ser alterados durante ninguna etapa del proyecto después de ser establecidos.

- Principio 8: Pruebas integradas durante el ciclo de vida.
  Muchas metodologías implementan las pruebas como parte final de la etapa de diseño o implementación, pero DSDM requiere que las pruebas se hagan en una etapa temprana en el desarrollo del proyecto. Las pruebas que se llevan a cabo en esta metodología son: las pruebas con propósitos técnicos llevadas a cabo por los desarrolladores, las pruebas con propósitos funcionales llevas a cabo por los usuarios, las pruebas de aceptación durante el desarrollo de manera incremental y finalmente las pruebas de integración de los componentes desarrollados en cada iteración.

- Principio 9: Enfoque colaborativo y cooperativo.
  Este principio de DSDM busca la completa cooperación e integración entre los desarrolladores y todas las partes interesadas. Esta metodología se centra en entregar productos de manera rápida y el desarrollo de manera rápida no es bueno si el resto de la organización no trabaja junto con todo el equipo (Gall).

5.2.3.1. Ventajas.

- Se minimizan los riesgos al tener entregas de pequeñas porciones del producto funcional que se incrementan en cada iteración hasta formar el producto completo.
- El usuario esta siempre involucrado en cada crecimiento del proyecto, por lo tanto, el resultado final debería de satisfacer las necesidades de los requerimientos.
- El usuario está familiarizado con el sistema desde antes de que sea la entrega oficial debido a las pruebas funcionales que hay a lo largo del ciclo de vida del proyecto.
- La implementación del sistema es bastante sencilla debido a que a lo largo del proyecto esta fase se repitió varias veces.
- A final de cuentas el usuario es el que brinda la solución que el equipo de desarrollo construiría a lo largo del proyecto.

- Las pruebas desde la etapa de requerimientos dan como resultado requerimientos sin malentendidos.

5.2.3.2. Desventajas:

- La comunicación con el cliente es la base del desarrollo de esta metodología, y así como puede dar rapidez al desarrollo también puede entorpecerlo.
- El usuario que participa en el proyecto debe de tener conocimiento del problema que se debe de resolver, en caso contrario el proyecto puede fallar o llegar a objetivos no deseados.

5.2.3.3. Matriz Foda

En la siguiente matriz se muestran las Fuerzas (factores críticos positivos), Oportunidades (aspectos positivos que podemos aprovechar), Debilidades (factores críticos negativos) y Amenazas (aspectos negativos externos).

Figura 5-10 Matriz FODA Metodología DSDM

	POSITIVOS	NEGATIVOS
INTERNO	**Fortalezas**: Participación activa del usuario durante el desarrollo El equipo de desarrollo es capaz de tomar decisiones Enfocarse en los requerimientos principales Los cambios no impacta fuertemente al proyecto desarrollado	**Debilidades:** Los requerimientos posteriores a los principales son dados sobre la marcha
EXTERNO	**Oportunidades:** Elaborar mecanismo de pruebas y plan de contingencia	**Amenazas:** La falta de conocimiento de las necesidades o indisponibilidad por parte del usuario que es parte del equipo de desarrollo

5.2.3.4. Conclusión.

De todas las metodologías de desarrollo de software tradicional agiles mencionadas en el análisis, esta es la que mejor encaja debido a sus principios base que coincide con las características principales del desarrollo web. Las desventajas mencionadas para esta metodología son completamente circunstanciales y no son falla del equipo de desarrollo. En general, estas fallas no deberían de ocurrir si es que existe la participación y compromiso de manera adecuada de los usuarios u organizaciones interesadas en el desarrollo del proyecto.

6. ANALISIS METODOLOGÍAS PARA EL DESARROLLO DE SOFTWARE WEB.

6.1. Definición

Existen muchos artículos, académicos y de periodismo, proponiendo una metodología de desarrollo web. Pero después de una revisión más extensa, estas son más parecidas a ideas para mejores prácticas en el diseño del *"look and feel"* de un sitio web (Howcroft & Carroll).

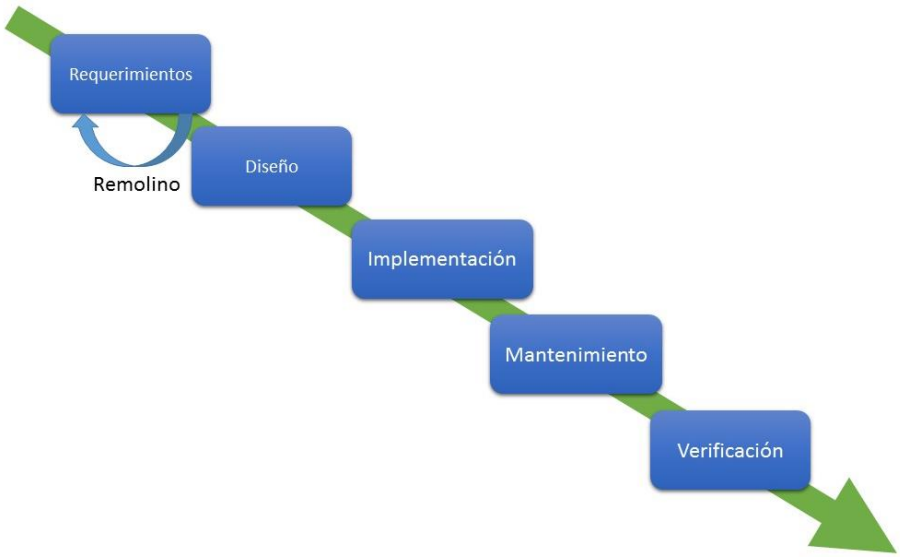
Si bien, las metodologías de desarrollo como el modelo en cascada y prototipado pueden ser efectivos para el desarrollo web, necesitan ser adaptados y enriquecidos para el nuevo ambiente de desarrollo para cumplir con los retos del desarrollo web (Suh, 2005).

Durante los primeros años de desarrollo web el panorama era muy parecido al de la ingeniería de software en los años setentas, en donde no había un enfoque definido para llevar a cabo un proyecto y cada programador tenía su propio método.

En la mayoría de los casos, el enfoque de desarrollo utilizado para los sistemas basados en la web ha sido ad hoc, y en la práctica el desarrollo de software para la web carece de un enfoque riguroso y sistemático (Achebe, 2002). Debido a eso, Powell, Jones y Cutts recomendaron la necesidad de formalizar el procesos en el desarrollo web (Suh, 2005).

La primera metodología sugerida para esto fue una modificación de la metodología en cascada en la que se implementaban los llamados remolinos (*Whirlpool).* Al igual que el modelo en cascada esta metodología consistía de las mismas etapas pero durante las primeras etapas de requerimientos de sistema y de usuario se hacían iteraciones (Figura 6-1) unas cuantas veces, por eso el nombre de remolinos, para refinar los requerimientos del usuario hasta que el proyecto se encontrara listo para arrancar.

Figura 6-1 Metodología en Cascada con remolinos. Fuente: Elaboración Propia.



Tiempo más tarde, el desarrollo de las aplicaciones web se expandió para incluir formulaciones de estrategias de negocio y reingeniería de procesos de negocio. Esta era una nueva metodología ligera que incluía cinco etapas: desarrollo de la estrategia de *e-business,* reingeniería del proceso de negocio, requerimientos de desarrollo del sistema, diseño/codificación y pruebas. Junto con esta metodología surge la llamada ICDM (*Internet Commerce Development Methodology* – Metodología de Desarrollo de Comercio en Internet) que es una metodología similar pero que comienza con el análisis del ambiente de negocio y la reingeniería de procesos.

6.2. Análisis

6.2.1. ICDM

La metodología de desarrollo llamada ICDM (Internet Commerce Development Methodology) se enfoca en el comercio electrónico o *e-business* y lo ve como iniciativas organizacionales y como tal tomo en cuenta la necesidad de dirigirse a las problemáticas estratégicas, de negocios, administrativas y de cultura organizacional así como también los detalles técnicos del diseño y la implementación (Standing, 2002).

Definir la estrategia de *e-business* de una organización involucra lidiar con un rango de fuentes de información y opiniones. La pregunta "¿Cómo una organización puede emplear efectivamente el *e-business?"* es inherentemente subjetiva en naturaleza y cualquier definición de "efectiva" será socialmente construida en gran medida (Standing, The Requirements of Methodologies For Developing Web Applications, 2001).

ICDM hace uso del análisis competitivo para ayudar a dar forma a la dirección que debería de tomar la organización al momento de emplear el *e-business.* Para esto, esta metodología hace un énfasis en el ambiente organizacional utilizando métodos de naturaleza social para la definición de requerimientos y estrategias de negocio ya que reconoce que la estrategia es un proceso constituido por la sociedad con un aprendizaje del ambiente dinámico.

Podemos decir que ICDM es una metodología de análisis de negocios así como también una metodología de desarrollo de sistemas. Muchas metodologías de sistemas de información tradicionales solo cubren los aspectos más técnicos del desarrollo de sistemas y no comienzan con alguna forma de análisis del negocio (Suh, 2005).

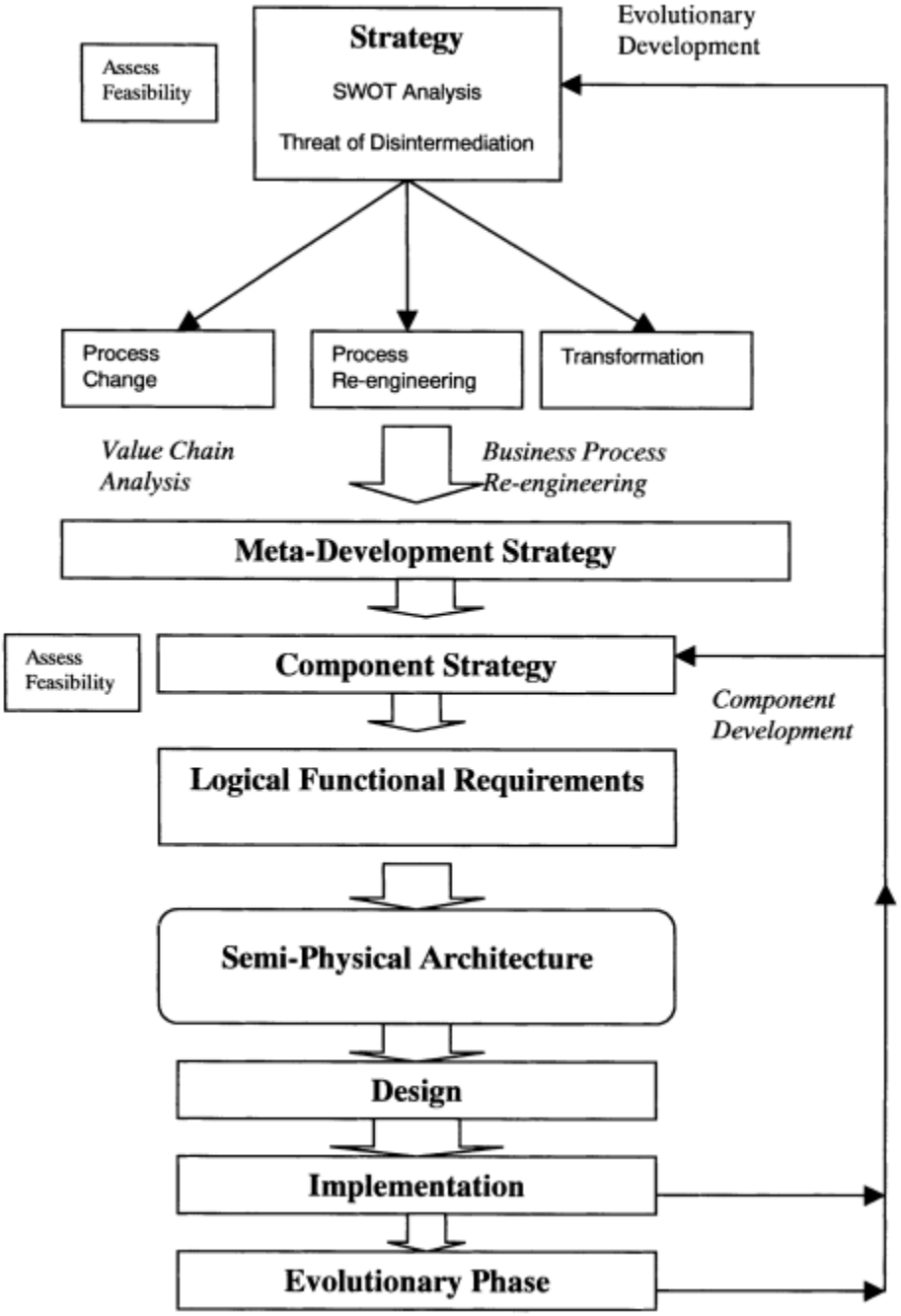
Con la tendencia hacia la globalización de los mercados económicos, una organización debe de estar buscando continuamente oportunidades y aprendizaje en un nivel global (Standing, Methodologies for Developing Web Applications, 2002). Debido a esto, para una metodología web ya no es suficiente enfocarse hacia el interior del desarrollo, también debe de proveer al equipo de desarrollo las herramientas y mecanismos necesarios para analizar el ambiente de negocios de manera amplia. Para esto, ICDM lleva a cabo un análisis FODA en su fase de estrategia de negocios donde se toman en cuenta tendencias de la sociedad y el mundo de negocios.

Se debe de considerar que ICDM no es una metodología con un gran número de pasos para ser llevada a cabo, sino como holgada para el desarrollo de estrategias y para el desarrollo evolutivo de los sistemas basados en la web (Standing, Methodologies for Developing Web Applications, 2002). Como resultado de esto, esta metodología es aplicable a un gran número de situaciones donde las organizaciones están buscando ganancias por invertir en el comercio en Internet.

Esta metodología tiene tanto una estrategia de administración como una estrategia de desarrollo que son dirigidas por las necesidades del negocio. Al contrario de otras metodologías que son escogidas por el equipo de desarrollo que no se enfocan en los intereses del negocio, IDCM se enfoca de manera particular en proveer un enfoque de negocio.

Las fases que componen a esta metodología (Figura 6-2) tanto en la parte de análisis de negocio y de desarrollo del proyecto son las siguientes:

Figura 6-2 Fases de la metodología ICDM. Fuente: Standing. (2002, p. 154).



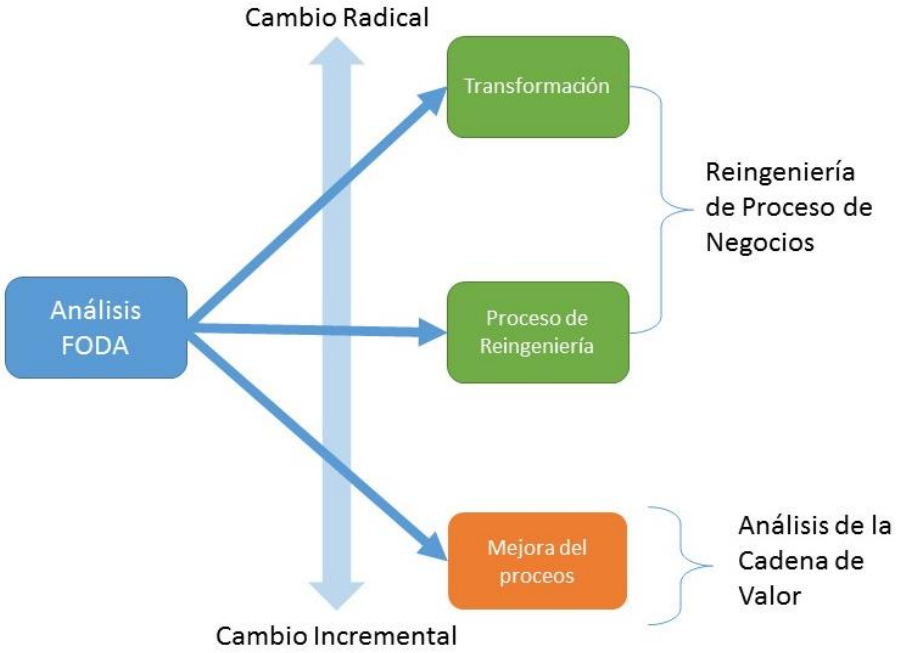
- Estrategia de Administración Web

  Esta metodología recomienda que la administración y desarrollo de los sistemas de *e-business* en tres niveles. La administración y desarrollo en general de la entera estrategia web puede ser visco como una tarea en curso como el desarrollo de los componentes funcionales de la aplicación web. El primer de estos niveles es una meta-desarrollo y una perspectiva de administración que proveen un marco para el desarrollo del proyecto. El segundo de estos niveles es el que le concierne al desarrollo tal cual es de los componentes de un sitio web. Finalmente el tercer nivel le compete al desarrollo e implementación del sistema e incluye a los equipos de desarrollo, analistas, especialistas de contenido, etc.

- Desarrollo de la Estrategia

  El uso del Internet en los negocios puede encontrarse de muchas formas y esta metodología provee un enfoque de planeación estratégica que considera que puede haber diversas opciones para escoger la más apropiada para un negocio. Esta metodología tiene el proceso de reingeniería de negocio y el análisis de la cadena de valor. Para poder decidir sobre alguna de las estrategias mencionadas, se puede emplear un análisis FODA (Fuerzas, Oportunidades, Debilidades y Amenazas) en el que se analiza y detalla el estatus del negocio frente a la competencia.

Figura 6-3 Fase de Planeación Estratégica de la Metodología ICDM



- Estrategia de Meta-Desarrollo.

  Hay un número de estrategias que pueden ser empleadas por una compañía cuando administran el desarrollo de un sitio web. Las opciones dependen de la cantidad de regulación o control que es deseado, ambos para el contenido y el diseño (Standing, Methodologies for Developing Web Applications, 2002). Una de ellas el planear el sitio completamente y regular su desarrollo distribuido en consulta con las unidades de negocio. Otra estrategia es planear el núcleo del sitio web y permitir a las unidades de negocio desarrollar su propio entorno.

- Participación del Usuario.

  El cliente o usuarios del sistema deberían involucrarse en varias de las etapas de las operaciones del *e-business* e incluirse en revisiones periódicas. En las etapas en las que el usuario es realmente útil son la del desarrollo de la estrategia y el análisis de negocio. También pueden participar en la evaluación del diseño en la etapa del prototipo y sin duda alguna deben de incluirse en la etapa de pruebas y evaluación del sitio web, ya que de esta manera se puede obtener la realimentación antes pero también después de que el sistema sea lanzado en revisiones de cumplimiento de objetivo .

- Desarrollo del sitio y los componentes.

  Es la construcción de componentes funcionales y puede ser llevada a cabo como pequeños proyectos. En esta etapa el equipo necesario tiene que ser multidisciplinario ya que no solamente hablamos del desarrollo tecnológico, sino también la implementación de estrategias de negocio.

- Técnicas de Análisis de Requerimientos.

  Esta metodología utiliza dos técnicas principales para la definición de los requerimientos del proyecto, lluvia de ideas (*brainstorming*) y Sesiones de Requerimientos en Grupo. La primera de estas es utilizada para definir caminos alternativos del uso del comercio en Internet en el negocio y el segundo e para obtener los requerimientos detallados en un tiempo relativamente corto con los grupos de usuarios involucrados. Otra herramienta para ayudar a definir los requerimientos es el uso de los prototipos, que serán usados después en gran manera en la fase de diseño y desarrollo.

- Arquitectura Física.

  Las técnicas utilizadas para definir los requerimientos de un proyecto web dependen en su mayoría del tipo de sistema y su funcionalidad. Hay tres tipos principales de sistemas de información web: los sistemas de publicación de

documentos, sistemas de interacción básica y los sistemas de transacciones complejas. Se debe de definir correctamente el tipo de sistema, ya que no siempre un sistema complejo es la solución a los problemas.

- Fase de diseño.
  La fase de diseño involucra el diseño de la infraestructura de la red, desarrollar el sitio web y los controles de seguridad. El diseño del sitio web debe de considerar la construcción de la imagen deseada, la usabilidad del sistema, la estrategia de promoción y mercadeo y finalmente la evaluación de los resultados con los usuarios finales.

- Fase de Implementación y Evaluación.
  Esta fase se relaciona con la fase de meta-desarrollo, y casi nunca, a menos de que el proyecto sea pequeño, se libera todo el proyecto de forma definitiva en un solo ciclo. Debido a que los sistemas de información web no tienen de manera bien definida en qué momento se encuentra finalizado el proyecto por su naturaleza evolutiva, en esta etapa se entrega al equipo de tecnologías de la información a la empresa y se libera de trabajo posterior al equipo de desarrollo. Debido a esto, esta etapa es supervisada por el equipo de la empresa y se establecen pólizas sobre de quien tendrá la responsabilidad de hacer las actualizaciones y modificaciones posteriores.

6.2.1.1. Ventajas

- La participación del cliente y el equipo de desarrollo es el punto fundamental de la metodología, ya que no solamente conocen las necesidades del cliente y del proyecto, si no también brinda la posibilidad de conocer acerca del negocio y proponer el mejor enfoque para que se cumplan los objetivos planteados.

- Las técnicas de análisis de requerimientos especificadas en la metodología son las adecuadas cuando se quieren conocer de manera completa los objetivos y requerimientos de un grupo clientes, y gracias a la lluvia de ideas, muchas veces el cliente mismo encuentra y propone la solución a su problema.

- Gracias a los análisis de la competencia que se hacen por medio del análisis FODA para conocer el status de la organización frente a sus competidores, se puede desarrollar una estrategia de negocio que favorece las metas del cliente, y por lo tanto, el trabajo del equipo de desarrollo.

6.2.1.2. Desventajas

- Una de las principales ventajas de la metodóloga es el enfoque que tiene sobre el desarrollo de una cultura organizacional, sin embargo, esta tarea tiene que ser llevada a cabo por alguien que aparte de estar involucrado en el negocio y conocer sus objetivos, debe de conocer acerca del estado del negocio a nivel mundial para poder proponer cambios en su enfoque.

- Esta metodología se basa en exclusiva en modelos de negocio en Internet por medio de lineamientos flexibles que son insuficientes para factores específicos de la industria.

- Debido a la necesidad de la participación de personal con diferentes áreas de experiencia, se vuelve una metodología cara para ser aplicada en proyectos de negocios que comienzan desde cero a participar en el mundo del *e-business.*

- El ciclo de vida de esta metodología es bastante complejo y largo, ya que los análisis y planeación y reestructura de negocio ocupan una gran parte del proceso de desarrollo, cuando en realidad este proceso no es completamente competencia del desarrollo web.

6.2.1.4. Conclusión

Esta metodología introduce conceptos nuevos a los antes vistos en esta investigación y tiene la característica de enfocarse de manera más fuerte en el cliente u organización y hacia el exterior del desarrollo en lugar de al interior como todas las demás. Sin embargo, es una metodología cara y lenta, que para ser aplicada requiere la suficiente experiencia en los negocios en internet para poder establecer las estrategias de negocios que los clientes requieren para lograr el éxito del proyecto.

6.2.2. OOHDM

La metodología OOHDM (*Objetc-Oriented Hypermedia Design Method* – Método de Diseño de Hipermedia Orientado a Objetos) es un enfoque basado en el modelado para la construcción de aplicaciones de hipermedia de gran tamaño (Shwabe & Moura). Esta metodología es una extensión de una ya existente llamada HDM (*Hypermedia Design Method* – Método de Diseño de Hipermedia).

Esta metodología está compuesta de cuatro diferentes actividades: diseño conceptual, diseño de navegación, diseño de la interfaz abstracta, e implementación. El proceso del ciclo de vida es una combinación de prototipado, incremental e iterativo en el que en cada actividad se construye o enriquece un conjunto de modelos orientados a objetos describiendo preocupaciones de diseño.

Una de las características de esta metodología es la separación del diseño de navegación y el diseño de interface. Esto nos ayuda a concentrarnos en cada una de las tareas una a la vez. Como resultado de esto se obtienen dueños modulares y reusables fases encapsuladas con el diseño de la experiencia para cada actividad.

- Diseño Conceptual

La meta del diseño conceptual es construir un modelo del dominio de la aplicación utilizando los principios de los modelos orientados a objetos con una notación similar a la de UML (Unified Modeling Language – Lenguaje de Modelado Unificado), la diferencia es el uso de atributos con valores múltiples y el uso de direcciones explicitas en las relaciones, dando como resultado un esquema de clases construido por subsistemas, clases y relaciones.

- Diseño de navegación

  En esta metodología, una aplicación es vista como una vista de navegación sobre el modelo conceptual (Shwabe & Moura). Esta es una de las mayores innovaciones de OOHDM, indicando que los objetos que el usuario navega nos son objetos conceptuales, si no otra clase de objetos que son construidos de uno o más objetos conceptuales. Las clases de navegación son llamadas nodos y sus atributos son definidos como vistas orientadas a objetos de clases conceptuales.

- Diseño de la interfaz abstracta

  En esta fase especificamos cuales son los objetos de la interfaz que el usuario puede ver. Esta fase tiene que ser definida claramente ya que si bien el diseño de la navegación podría tomarse como parte de la interfaz, no todo lo que hay en la interfaz sirve para navegar. En esta fase se especifica de qué manera se verán los elementos de navegación, que elemento sirve para navegar, y los cambios que sucederán cuando se navegue. Esta actividad es crítica en las aplicaciones de hipermedia. Para describir la interfaz de usuario se utiliza los modelos formales de interfaz de objetos llamada ADV (Abstract Data View – Vista de Datos Abstractos).

- Implementación

  En esta fase el diseñador implementara los elementos del diseño. Hasta ahora todos los modelos se habían construido en cierta manera para ser independientes de la plataforma de implementación, en esta fase en

particular se toma en cuenta el entorno de desarrollo (Shwabe & Moura). A pesar de que OOHDM es orientado a objetos, no se requiere que su implementación sea orientada a objetos.

6.2.2.1. Ventajas

- La separación por módulos que la metodología OOHDM utiliza permite tener clara cada una de las actividades en la metodología, por ejemplo la separación de la navegación y el diseño de interfaz nos permite tener clara la relación entre las distintas unidades de nuestro sistema de información para después simplemente establecer cuál es el elemento que sirve para navegar.

- La fase de implementación puede utilizarse bajo cualquier lenguaje o paradigma de programación ya que lo que se trabajó hasta ahora fue solo el diseño de las partes abstractas del sistema.

6.2.2.2. Desventajas

- No existe una fase de pruebas con mecanismos que sirvan para llevarlas a cabo, ya que después de la fase de implementación no se especifican los pasos que se debe de seguir para dar mantenimiento o crecimiento a la aplicación web.

- A pesar de ser una metodología incremental, iterativa y con prototipado, no especifica muchas etapas del proyecto, como la petición de requerimientos o revisión con el cliente.

6.2.2.4. Conclusión

Esta metodología tiene excelentes bases para el análisis del proyecto y para la construcción conceptual. Sin embargo, en lo general esta metodología carece de muchos procesos dentro del ciclo de vida de un proyecto que son importantes para la web, como por ejemplo el mantenimiento y las pruebas del producto, sin contar el levantamiento de requerimientos y la participación continua del usuario.

6.2.3. WSDM

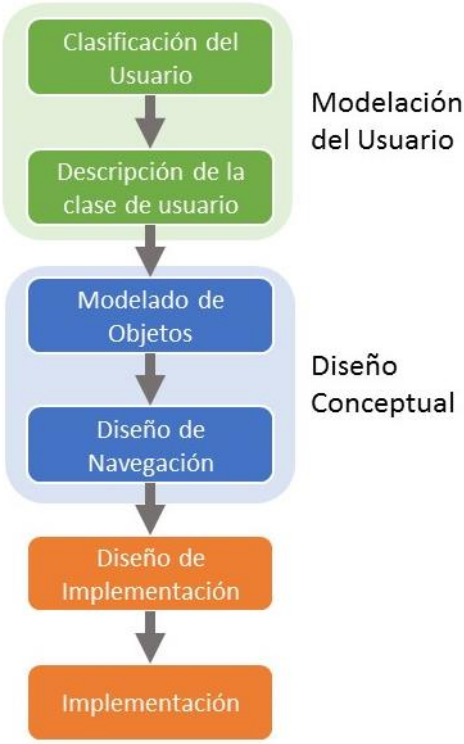
La metodología WSDM (Web Site Design Method – Metodo de Diseño de Sitios Web), también pronunciada *Wisdom* (De Troyer & K., 2000) se centra enfoca en el usuario en lugar de en los datos como las metodologías pasadas. Por esta razón, lo primero que se tiene que hacer es identificar los diferentes tipos de usuarios ya que para crear comunicación de manera efectiva no solo es necesario planear que se va comunicar también se requiere saber a quién.

Lo siguiente que hacemos es describir sus características y sus requerimientos de información. Esto resulta en algo llamado perspectivas. Solo después que eso se realizó, el diseño conceptual del sitio web comienza, tomando como entrada las perspectivas y el modelo de objetos de negocio de la organización (De Troyer & K., 2000).

Hay que diferenciar que esta es una metodología centrada en el usuario, no guiada por el usuario. En un enfoque guiado por el usuario los requerimientos de los usuarios guían de gran manera el diseño. Esto no es posible para aplicaciones web que son distribuidas a nivel mundial ya que los usuarios son desconocidos y no pueden ser entrevistados para conocer sus necesidades. El punto clave de esto es hacerlo de tal manera que tanto el proveedor y el usuario se vean beneficiados por esto.

El método WSDM consiste de las siguientes fases: Modelado del usuario, que está compuesta por las sub-fases clasificación del usuario y descripción de la clase del usuario; diseño conceptual, que al igual tiene las sub-fases modelado de objetos y diseño de navegación; diseño de la implementación; y la implementación final.

Figura 6-4 Ciclo de vida del modelo WSDM. Fuente: Elaboración Propia.



- Modelación del usuario

  Los usuarios normalmente visitan los sitios web buscando una respuesta a algo, y lo más correcto es que el sito anticipe esa respuesta y le de la información que necesita. Por lo tanto, la primera fase de esta metodología es concentrarse en los usuarios potenciales del sitio. A diferencia de otras

metodologías que se basan en primer instancia en reunir la mayor información posible y luego presentarla de forma elegante, esta se basa en el usuario.

- Clasificación del Usuario

  En este paso se identifican y clasifican los tipos de usuarios del sitio web, una manera de hacer esto es analizar la organización y los procesos para quien será construido el sitio web. Cada organización o proceso puede ser divido en actividades y cada actividad involucra personas que pueden ser usuarios potenciales.

- Descripción de la clase de usuario.

  Se analizan las clases de usuarios de manera detallada con dos enfoques. El primero de estos son los requerimientos de información de las diferentes clases de usuario. Para esto necesitamos encontrar las características de las clases de usurarios, ya que con ellas obtendremos conocimiento acerca de cómo se debería presentar la información a cierta clase de usuarios.

- Diseño Conceptual

  Esta fase consiste en dos sub-fases, la de modelado de objetos y diseño de navegación. Durante el modelado de objetos los requerimientos de información d las diferentes clases de usuarios y sus perspectivas son formalmente descritas. En la fase de diseño de navegación describimos como los diferentes usuarios pueden navegar a través del sitio web. Cada perspectiva tendrá su propia pista de navegación (De Troyer & K., 2000).

  - Modelado del Objeto

    En esta fase se modela formalmente los requerimientos de información de los usuarios definido con anterioridad mediante la construcción de modelos de objetos para cada clase de usuario

llamados Modelo de Objetos del Usuario. En estos modelos se definen las relaciones entre diferentes tipos de objetos, reglas y límites.

- Diseño de navegación
  En esta fase un modelo de navegación conceptual es construido, en el podemos encontrar las pistas de navegación para cada perspectiva. Estas pistas describen como los usuarios de una perspectiva en particular pueden navegar a través de la información disponible. En el modelo podemos encontrar diferentes simbologías para cada uno de los términos, ya sea componente de información, componente de navegación y componentes externos.

- Diseño de Implementación
  En este paso esencialmente diseñamos el *"look and feel"* o aspecto y comportamiento de un sitio web. El objetivo es crear un *"look and feel"* consistente, agradable y eficiente para el diseño conceptual hecho en las previas fases (De Troyer & K., 2000). Esta fase dependerá del ambiente de implementación elegido.

- Implementación
  En esta fase se construye el sitio web usando el ambiente de implementación elegido. Esto quiere decir, que si en la fase anterior se escogió HTML para ser el lenguaje del sitio web, el resultado de esta fase debe de ser paquetes de archivos HTML listos para entregarse. Para mejorar el mantenimiento cada vez más sitios web son construidos basándose en bases de datos.

6.2.3.1. Ventajas

- Esta metodología tiene como base la metodología OOHDM, que tiene buenas bases de construcción del modelado, pero esta va un poco más allá

definiendo los requerimientos de información basados en un análisis del usuario.

- Al ser una metodología centrada en el usuario garantiza que al final del proyecto los usuarios recibirán la información que requieren de manera rápida para satisfacer sus necesidades.
- La clasificación de grupos de usuarios permite que una aplicación web pueda ser enfocada a usuarios que cumplen con ciertas características para desplegar diferente información dependiendo de su clase.

6.2.3.2. Desventajas

- Esta metodología no implementa una fase de pruebas, mantenimiento o innovación después de que el sitio es lanzado, por lo tanto la información en el sitio puede volverse inservible e inclusive el sitio puede dejar de estar dentro de estándares de calidad o tecnológicos debido a que no hay actualizaciones.

- No se enfoca en dar una solución de negocios o un diseño de modelo de negocios porque supone que esto ya ha sido implementado por la organización.

6.2.3.4. Conclusión

Esta metodología sirve como complementa para la metodología OOHDM ya que agrega nuevas fases y cambia si enfoque a centrado en el usuario. Al ser una metodología centrada en el usuario la calidad de la experiencia final del usuario incrementa bastante ya que la información fue diseñada para ser mostrada bajo ciertas características de usuario, sin embargo se vuelve una tarea difícil ya que la mayoría de sistemas de información son distribuidos a gran escala y el estudio y modelado de un usuario promedio puede volverse complicado.

6.2.4. Howcroft & Carroll

Esta metodología fue propuesta por Debra Howcroft y John Carroll, ambos del centro de investigación de sistemas de información a universidad de Salford. Esta metodología fue basada en el estudio de metodologías previas, una de las que más impacto tuvo fue la de Russo & Graham (Howcroft & Carroll).

Esta metodología cuenta con cuatro fases diferentes: análisis, diseño, generación, e implementación. Cada fase contiene un número definido de pasos a llevar a cabo para realizar el proceso de ciclo de vida del proyecto. A continuación se describen las fases de esta metodología.

- Fase Uno: Análisis

  La fase uno es la concerniente al desarrollo de la estrategia web y el análisis de como el sitio web lograra esta estrategia (Howcroft & Carroll). Esto es debido a que las dos principales razones para que un proyecto de software falle son la falta de compromiso de mantenimiento hacia un proyecto y el malentendido de requerimientos. Esta fase busca reducir los riesgos estableciendo algunas metas y objetivos estratégicos y después diseñar unos sistemas para cumplirlos.

  - Paso uno: Desarrollo de la estrategia web

    La estrategia desarrollada es realizada en el Documento de Planeación Estratégica, el cual delimita tres elementos nucleares que describen la meta del sitio. El primero de ellos es definir la declaración de en donde la organización desea estar, el segundo es una aseveración de donde está ahora la empresa y finalmente un breve plan de implementación de lo que se hará para llegar del segundo elemento al primero. Este paso es iterativo y es la fase más crucial de la metodología, ya que los errores u omisiones en este momento

pueden resultar costosos más tarde, por lo tanto durante esta fase es conveniente tener un consultor/desarrollador apoyando.

- Paso dos: definiendo los objetivos

  Una vez que la estrategia web ha sido definida y aceptada por la organización y se ha generado el Documento de Planeación Estratégica la forma en que se cumplirán esas metas ahora puede ser identificado. Durante esta fase el desarrollador web debe de completamente involucrado con el proyecto y debe de conocer las tecnologías y servicios actuales en internet para poder dar la solución correcta a estos objetivos. El documento generado por esta fase es el Documento de Objetivos, en el cual se delimitan los objetivos del sitio y otros factores que podrían permitir a los desarrolladores evaluar la viabilidad de la implementación posterior del sitio.

- Paso 3: Análisis de Objetivos

  Durante este paso, se analizan los objetivos descritos anteriormente junto con los recursos para determinar en qué medida pueden ser logrados. Este análisis puede ser subdividido en seis tareas: tecnología, información, habilidades, usuarios, costos y riesgos. Una vez que estos análisis han sido completados un conjunto más refinado de objetivos son el resultado. Todos los objetivos que no puedan ser satisfechos son documentados en una Lista de Deseos que forma parte del Documento de Objetivos y formaran parte del proceso de iteraciones en el siguiente ciclo.

- Fase dos: Diseño

  Esta fase es dirigida por el Documento de Objetivos. El sitio debería ser diseñado con el conocimiento de que es probable que tengamos secciones y procesos agregados durante el tiempo de vida mientras los requerimientos cambian y nuevas tecnologías emergen. Esto debido a que muchos sitios

web crecen de manera exponencial y al no ser bien diseñados en su arquitectura desde el inicio se vuelven problemáticos y no manejables. Los pasos en esta fase son iterativos e incrementales y el resultado de esto es un Documento de Diseño refinado.

- Pasó uno: Diseño

  El diseño de un sitio web puede dividirse en dos tareas principales: Diseño de información y Diseño gráfico. El primero de estos es tan simple como diseñar un conjunto de páginas vinculadas o con contenido extraído de una base de datos con estructuras y procesos más complejos. El segundo, el diseño gráfico, es donde el *"look and feel"* de la aplicación es diseñado para los usuarios finales. El resultado de esta fase es un Documento de Diseño detallado que describe la estructura del sitio web y, si es necesario, la estructura de la base de datos.

- Paso dos: Pruebas de Diseño

  Las pruebas durante las primeras etapas de desarrollo son más efectivas en costo que las pruebas en el software codificado. Por esta razón las pruebas se hacen en la fase de diseño contra los objetivos y las metas establecidos para encontrar problemáticas e incongruencias.

- Fase 3: Generación.

  Esta fase de la metodología se enfoca en la generación de los sitios web y esta guiado por el Documento de Diseño. Esta fase está dividida en cuatro pasos.

  - Paso uno: Selección de los recursos

    En este paso se seleccionan todos los recursos para el desarrollo, desde software y hardware hasta el personal necesario.

- Paso dos: Revisión del diseño

  En este paso el Documento de Diseño de la fase dos se compara con los recursos del paso uno para verificar que los objetivos pueda ser logrados con los recursos que se disponen. Este es un proceso iterativo y si existen problemas, la fase uno puede ser revisada.

- Paso tres: Generación e Instalación del Código

  En este paso la generación del código es realizada y la conexión con el software y servicios necesarios para que el sitio pueda ser lanzado.

- Paso cuatro: Pruebas

  Las pruebas son la parte más compleja de cualquier proyecto web. Esto debido a que las pruebas deben de contemplar un mayor número de usuario que las pruebas en el software tradicional y en un mayor número de dispositivos y variables posibles de diferentes entornos sociales y tecnológicos.

- Fase Cuatro: Implementación
- La fase de implementación es tal vez la más simple pero la más importante de las fases. Para asegurar que los usuarios regresen la presencia del sitio debe de sentirse y el contenido debe de ser de valor.

  - Paso uno: Implementación

    Para implementar de manera completa el sitio web el usuario objetivo deben de saber de su existencia y notar su presencia. Durante esta fase el sitio debe de registrarse en distintos motores de búsqueda y otros métodos de promoción.

  - Paso dos: Mantenimiento

    Los sitios web particularmente tienen la necesidad de contar con una fase de mantenimiento ya que su contenido y su estructura se

encuentran en constante evolución. Es necesario que el sitio sea monitoreado de manera regular para asegurar que la información y los enlaces estén actualizados.

- Paso tres: Revisión de Objetivos
  Este proceso busca evaluar nuevas tecnologías en cuanto están disponibles. Estas pueden ser evaluadas con respecto a los objetivos delimitados en la fase uno, particularmente los objetivos que no fueron implementados y fueron documentados en la Lista de Deseos. Esta fase demuestra que toda metodología debe de ser iterativa y utilizada en una forma no lineal.

6.2.4.1. Ventajas

- Se incluye la etapa de pruebas sobre el diseño, lo que hace que las pruebas sean más eficientes en costo y tiempo.
- Una vez terminado el ciclo de vida hay revisiones de los objetivos para verificar que se cumplan aun después de que es lanzado el sitio.
- Todo el proceso de la metodología busca cumplir con los objetivos planteados, inclusive si es necesario habrá varias iteraciones

6.2.4.2. Desventajas

- No se especifica el proceso de diseño de la fase dos y la forma de evaluación del diseño durante la fase de pruebas de diseño.
- La evaluación durante la fase cuatro no especifica el proceso para realizar las pruebas.
- Como parte del análisis de objetivos debería de incluirse un análisis de tiempo de proyecto dentro de la administración del proyecto.

6.2.4.4. Conclusión

Esta metodología de todas las metodologías web es la que mejor se encuentra preparada para los cambios que pueden ocurrir en el desarrollo del proyecto. En esta las etapas son mucho más claras y definidas que las anteriores. El único punto en desventaja es que esta metodología está centrada en la información y no en el usuario, que es el enfoque más correcto para trabajar en la web.

7. PROPUESTA DE METODOLOGIA

En base a la serie de análisis anteriores sobre las metodologías de desarrollo de software tradicional y las metodologías de desarrollo de software web y por medio de la experiencia obtenida en el desarrollo de proyectos basados en la plataforma web, la siguiente metodología se desarrolló con el propósito de servir como guía completa en el desarrollo de un proyecto web.

7.1. Introducción

La principal problemática de los sistemas de información web es que se centran en dos puntos principales: el usuario y el contenido. En lo que le compete al contenido, gracias al análisis y requerimientos de la información es mucho más fácil de obtener y diseñar el contenido.

En cuanto a la definición de los usuarios, esta es una tarea difícil. Esto se debe a que existen dos tipos de sistemas de información en la web, los distribuidos de manera general, como aplicaciones en línea a la que cualquier persona puede acceder, y los que tendrán usuarios en específico, como sistemas para empresas o en intranet.

Debido a que esta metodología está centrada en el usuario como principal elemento, lo primero que se tiene que hacer es especificar qué tipo de sistema se construirá para poder analizar el tipo de usuario. Uno de los pasos más importantes es la definición del usuario final.

7.2. Etapa 1: Requerimientos

Durante la etapa uno, el líder de proyecto trabajará junto con el cliente para conocer los usuarios potenciales del sistema y el modelo de negocio de la organización. Durante esta fase, el líder de proyecto tendrá las funciones de un consultor de tecnologías o en caso de que se requiera, se podrá utilizar a uno de

los miembros del equipo de desarrollo como consultor. Se requiere un consultor debido a que debe de existir una persona que entienda lo que el cliente necesita en funciones de tecnología para poder guiar al cliente al rumbo que dará solución a sus problemas. Dentro de esta fase hay tres pasos importantes, el del conocimiento del usuario final, el del conocimiento del cliente y el conocimiento de la solución.

7.2.1. Paso 1: Conocimiento del usuario final

Durante este paso el objetivo es encontrar el tipo de usuario final del sistema. Este proceso es importante ya que la metodología es centrada en el usuario y a partir de este punto, es donde se conoce quien utilizará el sistema. En este momento lo único importante es definir qué tipo de usuario existirá en el sistema ya que el análisis y diseño del usuario se hará en etapas posteriores.

7.2.2. Paso 2: Conocimiento del cliente

Durante este paso se busca conocer de manera concreta lo que competa al proyecto acerca del modelo de negocios de la organización. Muchos proyectos desarrollados en la web tienen sus bases en procesos reales dentro de una organización o en sistemas de información antiguos. Por lo tanto, en este paso es necesario conocer los usos y costumbres de la organización para que al final del proyecto el cliente se sienta familiarizado con el resultado.

7.2.3. Paso 3: Conocimiento de la Solución

Este último paso define dos aspectos importantes: el objetivo de la solución y los medios de la solución. En esta etapa se define qué es lo que el usuario espera recibir como producto final y la manera en que espera que funcione. Esta etapa es básica en un proyecto web ya que en este momento es donde se conocen ciertos requerimientos de funcionalidad como aspectos técnicos del servidor, de los servicios web, la capacidad multiplataforma y multidispositivo del proyecto, etc.

Durante este paso se conocen las necesidades técnicas del proyecto para poder llevar a cabo los objetivos que se esperan resolver.

7.2.2. Resultados de la Etapa 1

Como resultado de esta etapa se obtiene un conjunto de información acerca del usuario y del modelo de negocios, pueden ser entrevistas escritas, archivos de audio, video, etc. Con esta información se puede trabajar en la siguiente etapa.

7.3. Etapa 2: Análisis

Durante esta etapa del proyecto se revisa la información obtenidas en la etapa anterior para generar conocimiento acerca del usuario y del negocio. Durante esta etapa se definen los objetivos, tipos de usuarios, clasificación de los usuarios y reglas de negocio. Durante esta etapa existen 3 pasos que se tienen que llevar a cabo y que son los pilares de la construcción de un sistema de información web. Estos pasos se describen a continuación.

7.3.1. Paso 1: Definición de Usuarios

Una vez que se tiene la información acerca del público a quien es dirigido el sistema, ya sea distribuido o a la medida, se comienza con el análisis de esta información para generar características clave en las que la fase de diseño se basará para la construcción de “persona” y en el diseño del sistema en general.

7.3.2. Paso 2: Definición del Modelo de Negocio

Una vez definido el usuario, se comenzará a construir el modelo de negocio en base a las clases de usuario que se generaron. Durante esta etapa se clasifican las actividades y procesos que se pueden llevar a cabo por los usuarios. Este paso se basa en la información de los usuarios para saber qué actividades e información se deben de presentar a los distintos tipos de usuario.

7.3.3. Paso 3: Definición de Objetivos

En esta etapa se analizan los objetivos del sistema en base a los requerimientos del cliente, pero se clasifican en dos categorías. Los objetivos de alto nivel y los objetivos secundarios. Los objetivos de alto nivel son aquellos que son la funcionalidad principal del sistema y son sobre los que se comenzara la construcción, los objetivos secundarios son metas agregadas a los objetivos principales y se construirán una vez que las bases son construidas.

7.3.4. Paso 4: Definición de Requerimientos Técnicos.

El último paso de esta etapa contiene los aspectos técnicos necesarios para llevar a cabo la solución. Si bien el objetivo de esto es de mayor utilidad para el equipo de desarrollo, es importante que el cliente revise las tecnologías tanto de software como de hardware que se piensan implementar como solución del proyecto, ya que esto muchas veces puede generar costos extra que el cliente debe aceptar antes de continuar. En esta definición se describen los resultados esperados del producto.

7.3.5. Resultados de la Etapa 2.

Al finalizar esta etapa, se hace una pequeña iteración la etapa anterior con el cliente para que revise la información analizada. Esta iteración sirve para eliminar malentendidos en alguna de las tres etapas bases del proyecto. Si es necesario, se harán las iteraciones hasta que el entendimiento del proyecto coincida con las necesidades del cliente. Al terminar esta etapa se ambas partes, cliente y equipo de desarrollo, firma el Documento de Creación del proyecto. Este documento es importante para ambas partes porque garantiza al cliente que los objetivos descritos serán cumplidos, pero también protege al equipo de desarrollo ya que establece los límites del proyecto. Básicamente, este documento contiene completa y únicamente lo que se desarrollara. A parte de este documento donde se establecen los limites,

se genera el documento de Definición de Requerimientos, en el que la etapa de diseño se basará.

7.4. Etapa 3: Diseño

Durante esta etapa se llevan a cabo la construcción de los modelos conceptuales de la información recibida de la etapa anterior. Para finalizar esta etapa se llevan a cabo – pasos que se describen a continuación.

7.4.1. Paso 1: Diseño del Funcionamiento:

En este paso se describe de manera detallada como es que el sistema debería de funcionar a nivel técnico. El paso 4 de la etapa 2 es la base de este paso, ya que se describirán los puntos la funcionalidad del sistema. Este diseño conceptual contempla la arquitectura para sitios de escritorio y para sitios móviles si este fuera el caso. También se diseña la arquitectura con la que funcionará el sistema de información.

7.4.2. Paso 2: Diseño de la “Persona”

Una persona en términos de diseño se refiere a la creación de usuarios promedio genéricos para clase de usuario que el sistema contemple.

7.4.3. Paso 3: Diseño de los Componentes y Actividades.

En este paso se modelaran los componentes del sistema de información y las distintas actividades que pueden ser llevadas a cabo en el sistema. Este paso será la base de la construcción del proyecto.

7.4.4. Paso 4: Diseño de la Navegación

Este paso más que presentar las opciones de navegación de las páginas, será un diagrama de flujo de las principales funciones del sistema. En este diagrama

se muestra de qué manera se relacionan los componentes con los usuarios y las actividades que se pueden desarrollar. Este diseño ayudará a la etapa de pruebas, ya que se realizarán sobre de las actividades que pueden realizar los usuarios y el flujo lógico del sistema.

7.4.5. Paso 5: Diseño de la Interfaz:

El diseño de la interfaz se lleva a cabo después del diseño de la “persona”, en este paso se describen los aspectos de usabilidad y accesibilidad necesarios que la “persona” requiere. Este diseño contempla los módulos y actividades antes definidos y es la base de la construcción del sistema.

7.4.6. Resultado de la Etapa 3

Como resultado del proceso de diseño de esta etapa, se generan varios documentos, uno por cada una de los pasos llevados a cabo que se engloban en un documento llamado Especificaciones de Construcción. En este documento se encuentran los modelos, listas, diseños y diagramas de flujo que fueron resultado de los pasos anteriores. Sobre de este documento, la etapa de construcción se basará.

7.5. Etapa 4: Construcción

Durante la fase de la construcción se elaboran los archivos que se implementarán en el sistema y entregarán al cliente. Debido a que esta es la única etapa en la que el cliente puede observar y medir los avances del proyecto, la manera en que se construyen los componentes se describe en los siguientes pasos. Un aspecto recomendado de la etapa de construcción es el uso del este patrón de arquitectura de software Modelo Vista Controlador. Esta metodología supone que el desarrollador utilizará el MVC, sin embargo, puede ser implementada sin necesidad de este patrón de arquitectura de software.

7.5.1. Paso 1: Construcción de los objetivos principales

Durante el paso uno se lleva a cabo la construcción de las funcionalidades principales que reflejan los objetivos primarios del sistema de información. Esto con el objetivo de que el cliente pueda medir el avance del proyecto por medio de las funciones principales en cuanto estén terminadas.

7.5.1.1. Paso 1.1: Construcción de la Vista

Este paso se enfoca en la construcción de la interfaz de usuario de manera visual. Algunas tecnologías que se pueden utilizar dentro de este paso son: HTML para la organización y estructura, y CSS para el diseño de la interfaz. En este paso se generan las plantillas que posteriormente se utilizarán con el resto de los pasos de construcción. Este paso se basa en el paso 5 de la etapa 3.

7.5.1.2. Paso 1.2: Construcción del Controlador o de las Funciones.

Este paso y el paso 1 no son necesariamente secuenciales. Mientras se construye la interfaz de usuario se puede trabajar sobre las funcionalidades del sistema. Este paso se basa en el paso 3 y 4 de la etapa 3. Algunas tecnologías en las que se puede construir la funcionalidad son: JavaScript, PHP, JSP, etc.

7.5.1.3. Paso 1.3: Construcción del Modelo u Origen de los datos.

Si el sistema de información así lo requiere, en este paso se construyen las consultas y el modelo de datos que utiliza el sistema para generar el contenido de la aplicación.

7.5.1.4. Paso 1.4: Revisión de Objetivo principal.

Durante este paso se revisa que los objetivos principales del negocio se hayan cumplido, si no se cumplieron se vuelve a realizar el Paso 1 de la Etapa 4: Construcción de los Objetivos Principales. En caso de que los objetivos principales se hayan cumplido se hará una revisión con el cliente para revisar los avances del

proyecto. Una vez cumplidos los objetivos, las funcionalidades principales son puestas a prueba por el equipo de desarrollo (etapa 5, paso 1) y por el usuario (etapa 5, paso 2).

7.5.2. Paso 2: Construcción de los Objetos Secundarios

La construcción de los elementos secundarios se hace dividiendo los objetivos secundarios en pequeños proyectos de construcción, de esta manera, a los objetivos principales construidos con anterioridad se le agregan funciones pequeñas que no impactan en el funcionamiento del objetivo principal en caso de que existan errores.

7.5.2.1. Paso 2.2: Construcción de la Vista

Este paso se enfoca en la construcción de la interfaz de usuario de manera visual. Algunas tecnologías que se pueden utilizar dentro de este paso son: HTML para la organización y estructura, y CSS para el diseño de la interfaz. En este paso se generan las plantillas que posteriormente se utilizarán con el resto de los pasos de construcción. Este paso se basa en el paso 5 de la etapa 3.

7.5.2.2. Paso 2.2: Construcción del Controlador o de las Funciones.

Este paso y el paso 1 no son necesariamente secuenciales. Mientras se construye la interfaz de usuario se puede trabajar sobre las funcionalidades del sistema. Este paso se basa en el paso 3 y 4 de la etapa 3. Algunas tecnologías en las que se puede construir la funcionalidad son: JavaScript, PHP, JSP, etc.

7.5.2.3. Paso 2.3: Construcción del Modelo u Origen de los datos.

Si el sistema de información así lo requiere, en este paso se construyen las consultas y el modelo de datos que utiliza el sistema para generar el contenido de la aplicación.

7.5.2.4. Paso 2.4: Revisión de Objetivos Secundarios

Cada vez que una funcionalidad secundaria termine de construirse, se debe de hacer la revisión con los objetivos secundarios. En caso de que no los cumpla, se vuelve a repetir el paso 2 de la Etapa 4: Construcción de los Objetivos Secundarios sobre el mismo objetivo secundario. Si el objetivo se cumplió, el paso 2 de la Etapa 4 se lleva a cabo con el siguiente objetivo secundario.

7.5.3. Resultado de la Etapa 4.

El resultado de ambos pasos con subtareas de esta etapa es un documento parecido a una lista de verificación de las funcionalidades y objetivos que debe de cumplir el proyecto. Este documento llamado Verificación de Funcionalidad, junto con los archivos construidos, pasa a la etapa de pruebas, en donde se verificará que se construyó de manera correcta y se harán las pruebas pertinentes.

7.6. Etapa 5: Pruebas

La fase de pruebas de esta metodología se hace en dos pasos: pruebas de producción y pruebas de usuario. Esta fase de pruebas se lleva a cabo de manera paralela con la construcción de los objetivos secundarios (etapa 4 paso 2). Al finalizar la prueba en caso de que errores hayan sido encontrados, se vuelve a la fase de construcción a reparar los módulos, ya sea vista, controlador, o modelo, que se encuentren dañados. Si no se encuentran errores, se procede a la etapa de implementación.

7.6.1. Paso 1: Pruebas de Producción.

Las pruebas de producción son llevadas a cabo inmediatamente después de que los objetivos principales o secundarios son terminados. Estas pruebas evalúan el correcto funcionamiento de los componentes del sistema, ya sea en la interfaz o funcionalidad. Este paso se basa en lo obtenido en el paso 1 y 4 de la etapa 3. Estas pruebas son llevadas a cabo por el equipo de desarrollo. Las pruebas

de producción son llevadas a cabo con el documento Verificación de Funcionalidad, donde se enuncian los puntos importantes de la funcionalidad del proyecto.

7.6.2. Paso 2: Pruebas de usuario.

Estas pruebas son llevadas a cabo por los usuarios. La técnica utilizada para las pruebas de los usuarios es por medio de una liberación de fase *beta* del proyecto. Sobre de este producto aun no terminado el usuario puede probar las funcionalidades principales y las funcionalidades secundarias conforme se agreguen, de esta manera, el equipo de desarrollo puede recibir información real del usuario en forma de realimentación.

7.7.3. Resultados de la Etapa 5.

Después de que el usuario y los desarrolladores no encontraron errores en el proyecto, se firma un documento llamado Acuerdo de Conformidad para poder pasar a la etapa de implementación.

7.7. Etapa 6: Implementación

Después de que la fase de pruebas fue liberada y el sistema con sus funcionalidades principales y secundarias se encuentra libre de errores y completamente funcional, el sistema de información puede pasar de su versión *beta* a su versión oficial. En esta etapa solamente hay un paso que seguir.

7.7.1. Paso 1: Instalación

Una vez que el proyecto ha sido probado y encontrado completamente funcional, se procede a instalar el proyecto en el servidor web junto con los servicios que se requiera. Para algunos proyectos esta instalación será tan sencilla como cargar los archivos en el servidor, y en otros, instalaciones y configuraciones son requeridas.

7.7.2. Resultado de la Etapa 6.

Como resultado de la implementación se genera la documentación necesaria acerca del proyecto y de su instalación para ceder las responsabilidades al equipo al que corresponda, si es que esto fuera necesario. Si el proyecto parará a manos de otra administración, es necesario que el nuevo equipo esté presente en el momento de la entrega e instalación del proyecto para verificar que todo sea entregado de manera correcta y completa. Una vez hecho eso se firma una carta llamada Liberación de Responsabilidades, en el que se especifica que el equipo de desarrollo queda exento de las fallas o mala administración que exista posterior a la entrega. La documentación del proyecto es llamada Manual de Usuario.

7.8. Etapa 7: Mantenimiento

Esta etapa es llevada a cabo por el equipo responsable del proyecto, ya sea el mismo que lo desarrollo o el equipo que lo recibió. Esta etapa busca mantener actualizada la información y las bases de datos. De esta manera el proyecto se vuelve manejable aun después de que fue lanzado.

Figura 7-1 Ciclo de vida de la metodología propuesta. Fuente: Elaboración Propia.


REQUERIMIENTOS
Conocimiento del Usuario Final
Conocimiento del Cliente
Conocimiento de la Solución
Información
ANÁLISIS
Definición de Usuarios
Definición del Modelo de Negocio
Definición de Objetivos
Definición de Req. Técnicos
Documento de Creación del Proyecto
Definición de Requerimientos
DISEÑO
Diseño del Funcionamiento
Diseño de la Persona
D. Componentes y Actividades
Diseño de la Navegación
Diseño de la Interfaz
Especificaciones de Construcción
CONSTRUCCIÓN
Construcción de Objetivo Principal
Construcción de la Vista
Construcción del Controlador
Construcción del Modelo
Revisión de Objetivo
Proyecto Beta
Construcción de Objetivo Secundario
Construcción de la Vista
Construcción del Controlador
Construcción del Modelo
Revisión de Objetivo
Verificación de Funcionalidad
PRUEBAS
Pruebas de Producción
Pruebas de Usuario
IMPLEMENTACIÓN
Acuerdo de Conformidad
Instalación
MANTENIMIENTO
Liberación de Responsabilidades y Manual
Mantenimiento

8. CASO PRÁCTICO

8.1. Caso práctico: Sistema de Administración de “Auditor Social”.

Para evaluar el funcionamiento de esta metodología se aplicó en dos casos de estudio diferentes en sistemas desarrollados por el Centro de Desarrollo de la Facultad de Informática de la Universidad Autónoma de Querétaro. El primero de estos desarrollos fue un sistema de información web para el departamento de Auditoria Superior de Fiscalización del Gobierno del Municipio de Querétaro, Querétaro, México.

El sistema de información desarrollado es un sistema de administración de una aplicación móvil de denuncias desarrollada también por el Centro de Desarrollo. Esta aplicación publica encuestas acerca del servicio municipal y recibe denuncias de los denominados Auditores Sociales.

El desarrollo de este sistema de información presento dos retos. El primero es que ya existía un sistema encargado de la administración de la aplicación, sin embargo, los usuarios del sistema encontraban difícil su uso. Lo que nos lleva al segundo problema. La variedad de usuarios.

Para rediseñar el sistema, utilizamos la metodología diseñada en la tesis, ya que está centrada en el usuario. A continuación se presentan algunas impresiones de pantalla del sistema después de ser terminado.

Figura 8-1 Pantalla de Encuestas del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

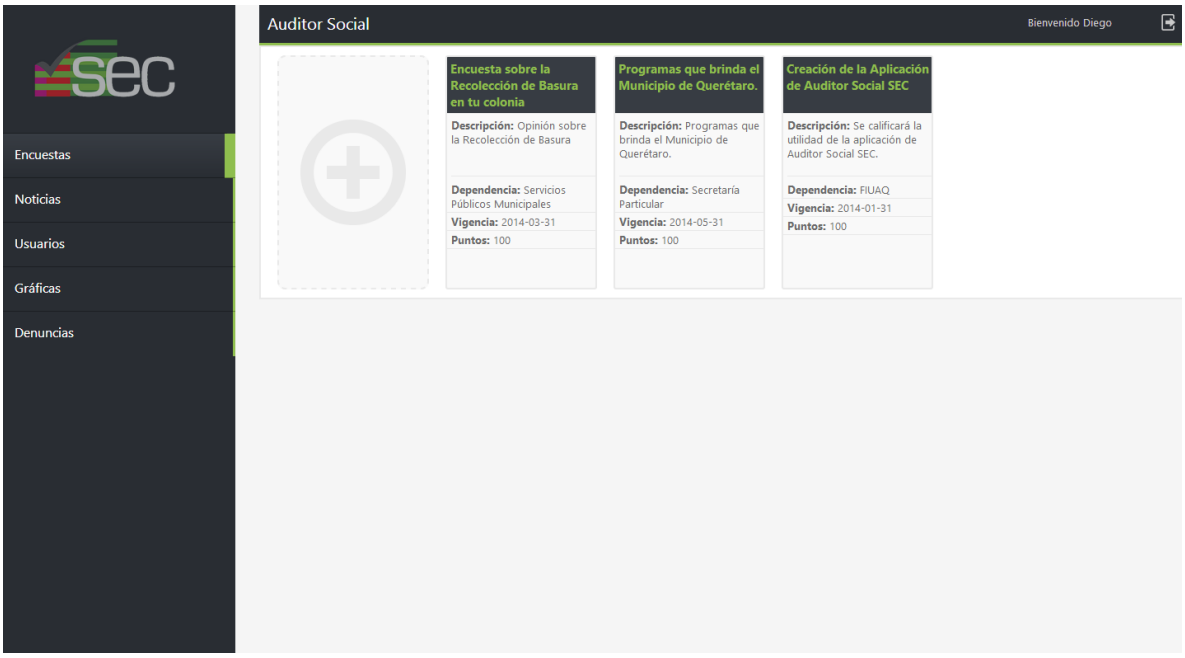


Figura 8-2 Pantalla de Nueva Encuestas del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

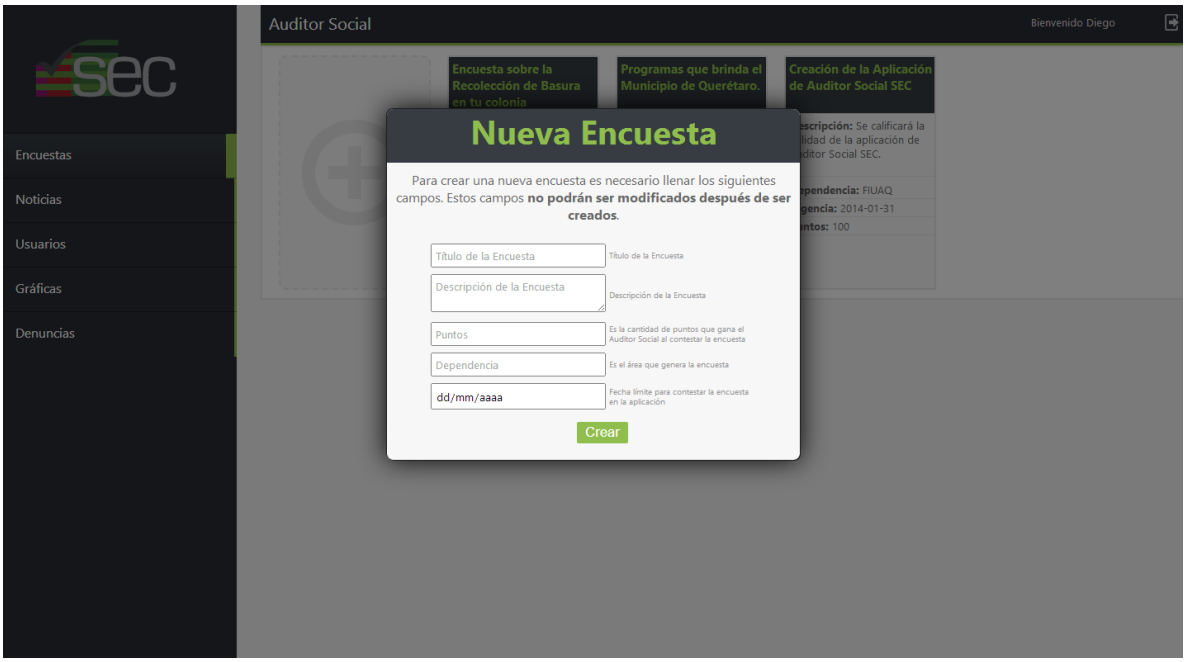


Figura 8-3 Pantalla de Nueva Pregunta del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

Figura 8-4 Pantalla de Ver Encuestas del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

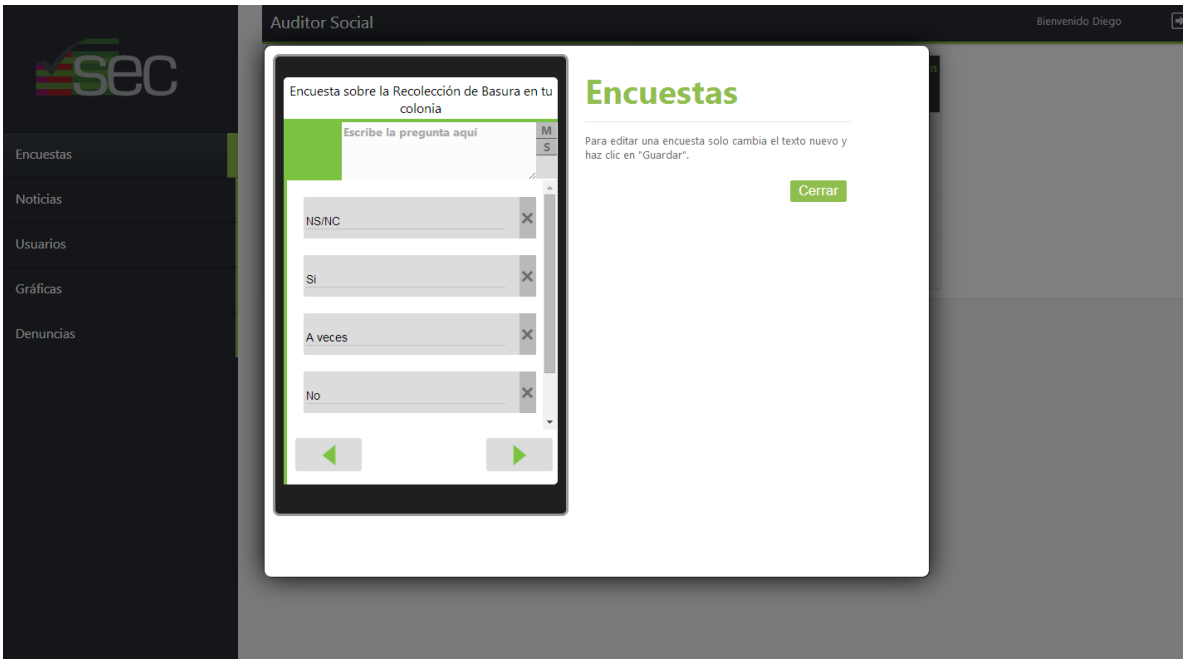


Figura 8-5 Pantalla de Noticias del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

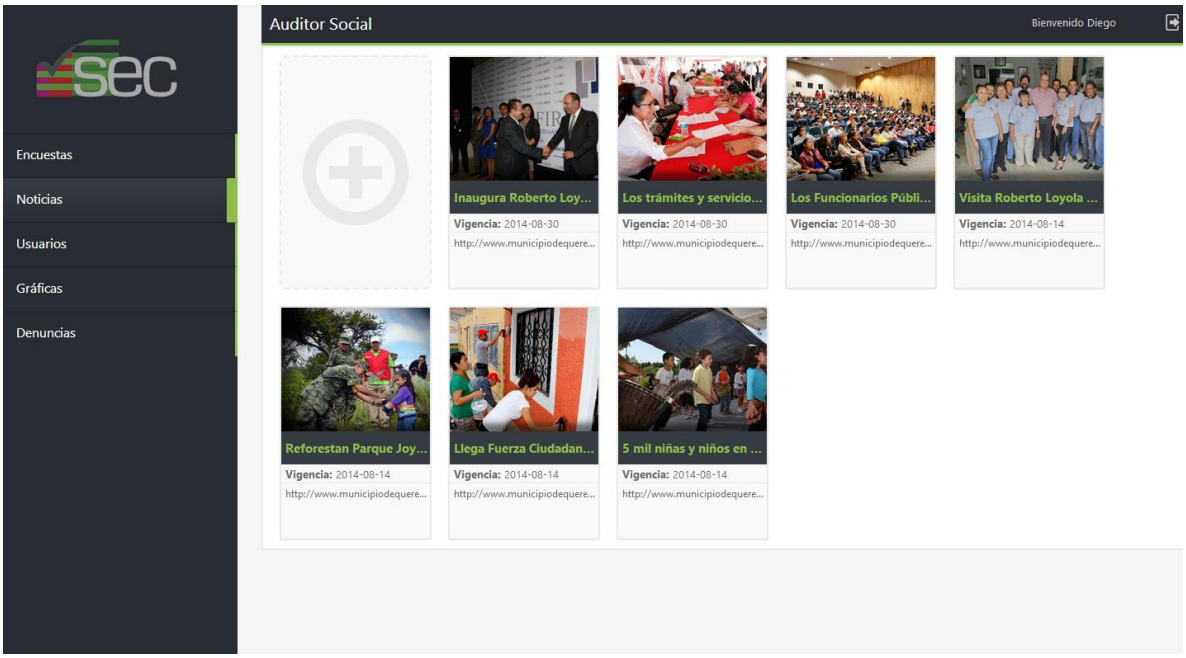


Figura 8-6 Pantalla de Nueva Noticia del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

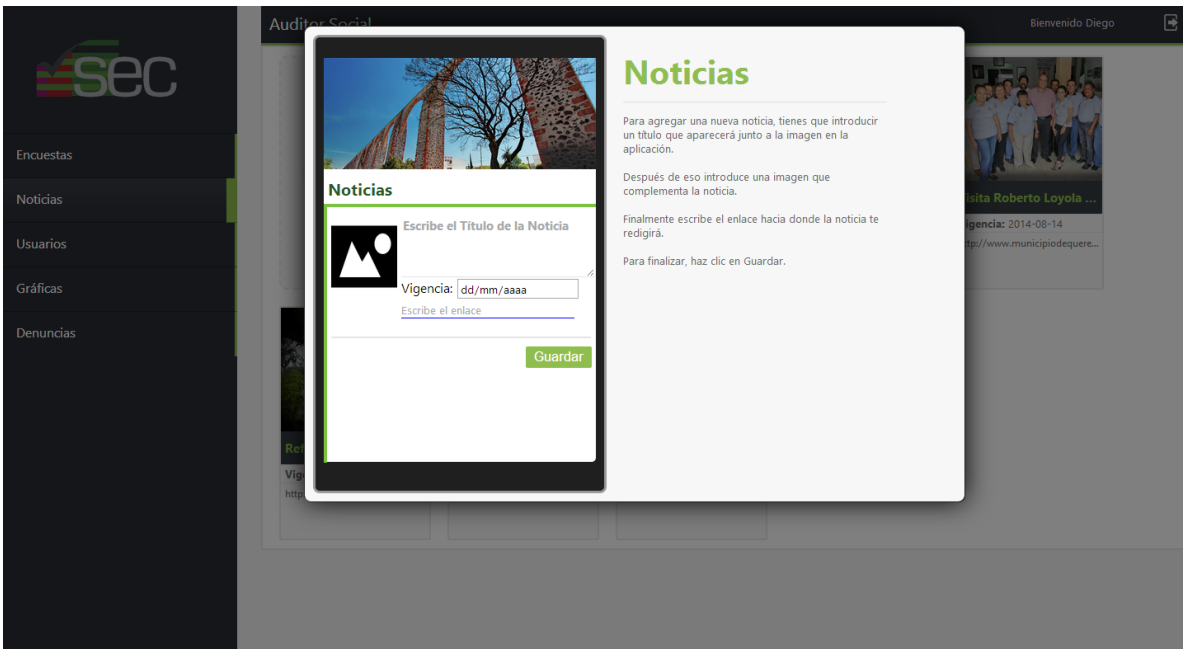


Figura 8-7 Pantalla de Editar Noticia del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

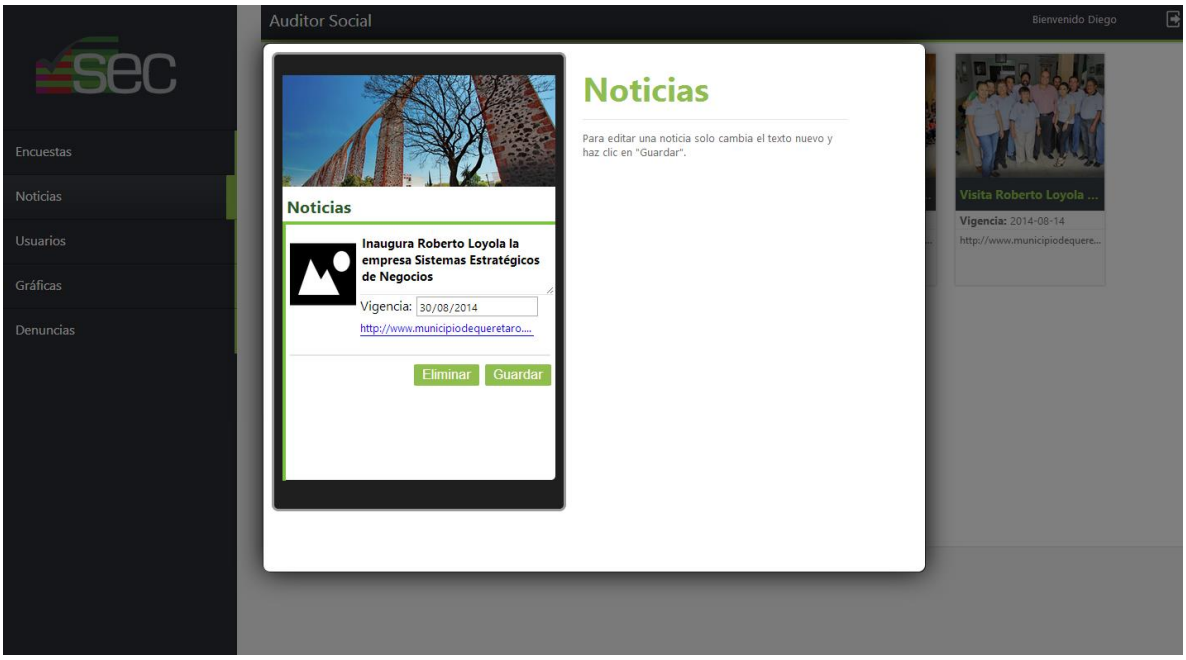


Figura 8-8 Pantalla de Graficas del sistema de administración "Auditor Social". Fuente: Elaboración Propia.

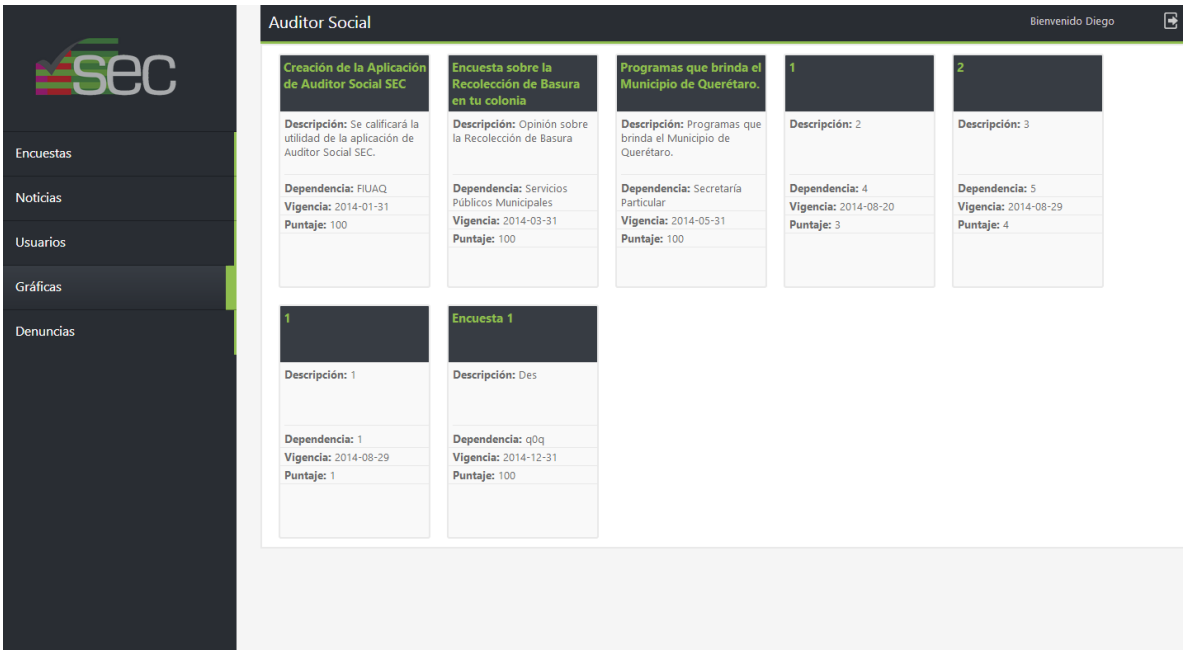
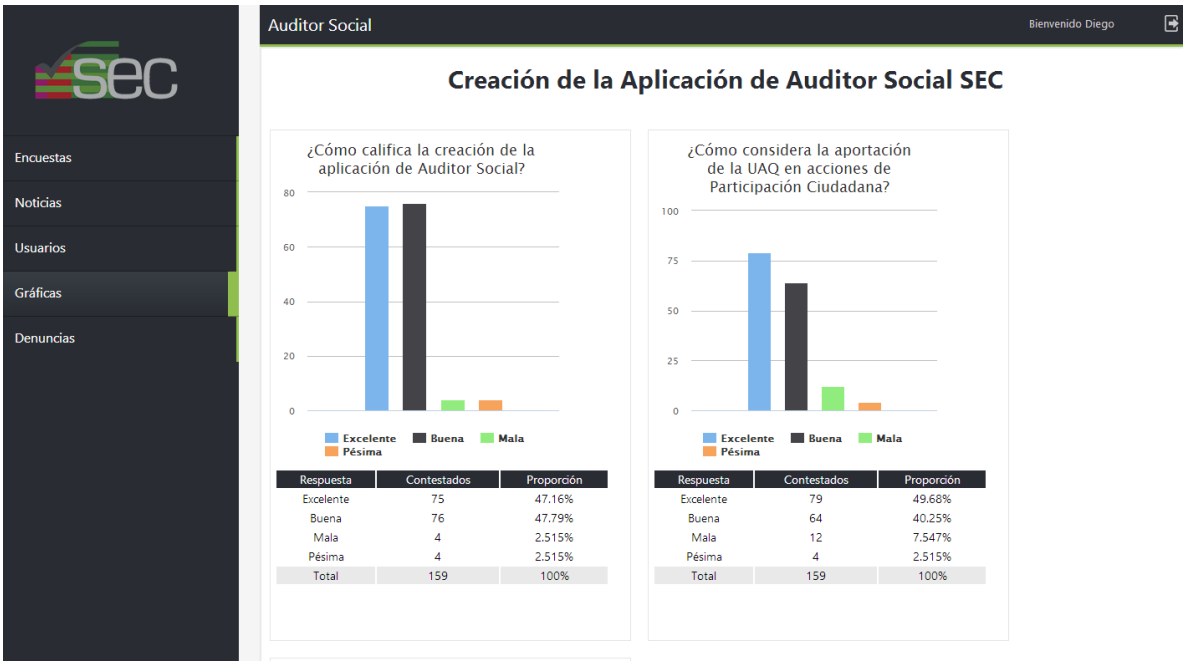


Figura 8-9 Pantalla de Ver Detalle Graficas del sistema de administración "Auditor Social". Fuente: Elaboración Propia.



El sistema fue implementado de manera exitosa y se encuentra en funciones actualmente. Uno de los retos del diseño de la interfaz fue imitar el proceso e imagen que se llevaba a cabo en los dispositivos móviles para su uso fuera más intuitivo que el del sistema anterior. Debido a la naturaleza del proyecto, algunas pantallas no pueden ser mostradas como parte de la evidencia del desarrollo.

Al finalizar el proyecto se hizo una evaluación de las fallas de la metodología y se trabajó en mejorar el proceso de desarrollo de esta. Este proyecto fue desarrollado utilizando un borrador previo de la metodología planteada en este desarrollo, ya que se encontró que tenía áreas de oportunidad en donde podía mejorar para ser mejor aplicada a un proyecto. A continuación se presenta el caso práctico 2, en el que se trabajó utilizando la metodología mejorada y es la versión más fiel a la que se presenta en el trabajo de investigación.

8.2. Caso práctico: Sistema de Punto de Venta “Homecel”

El segundo caso práctico es el del sistema de punto de venta para una distribuidora de Telcel llamada “Homecel” en el estado de San Luis Potosí. Para el desarrollo de este sistema de información se utilizó el segundo borrador de la metodología. A este último borrado se le aplicaron unos ligeros cambios que en realidad no modifican la esencia de la metodología.

Este sistema fue desarrollado para funcionar como punto de venta de mercancía, por lo tanto, ese fue su objetivo principal. Este objetivo contempla la interfaz de venta y la interfaz de inventario principalmente. Las funcionalidades extra de este sistema incluyen la creación de reportes, apartados y administración de usuarios.

El sistema fue desarrollado de manera satisfactoria y al igual que el anterior, algunas pantallas fueron omitidas por la naturaleza privada de los datos. Los datos que se presenta en algunas de las imágenes son ficticios y fueron colocados con fines de representar información verdadera.

A continuación se presentan las pantallas del sistema de información desarrollado con la metodología.

Figura 8-10 Pantalla de Inicio de Sesión del sistema de punto de venta “Homecel”. Fuente: Elaboración Propia.

Figura 8-11 Pantalla de Ventas del sistema de punto de venta “Homecel”. Fuente: Elaboración Propia.

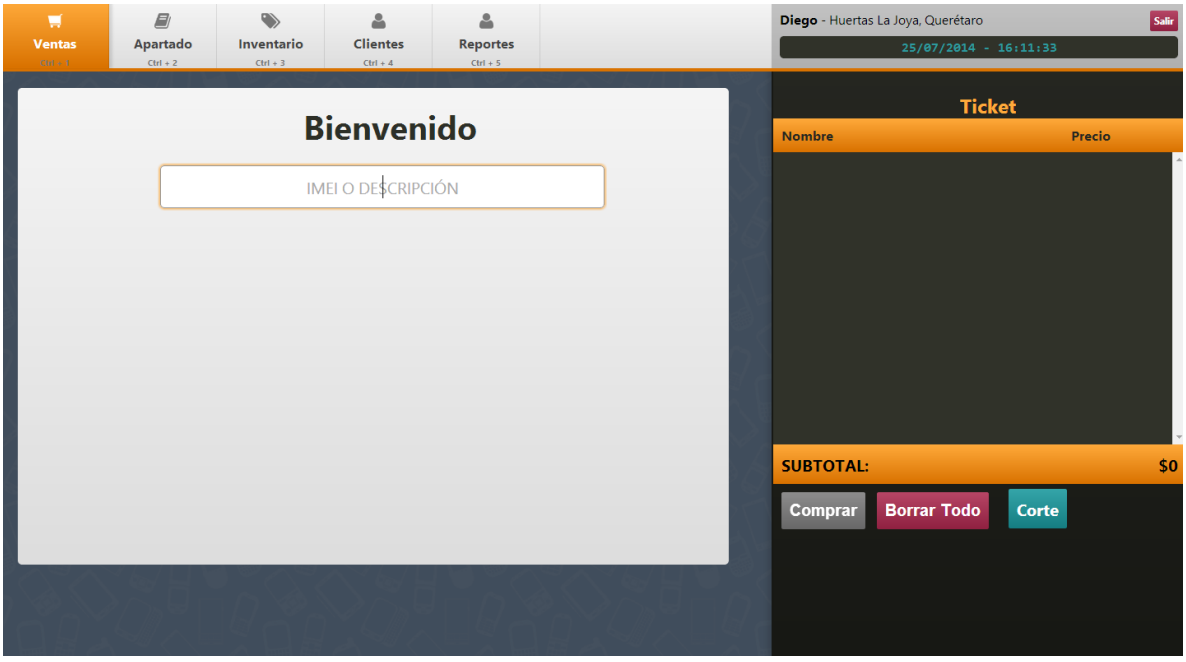


Figura 8-12 Pantalla de Ventas acción "búsqueda" del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

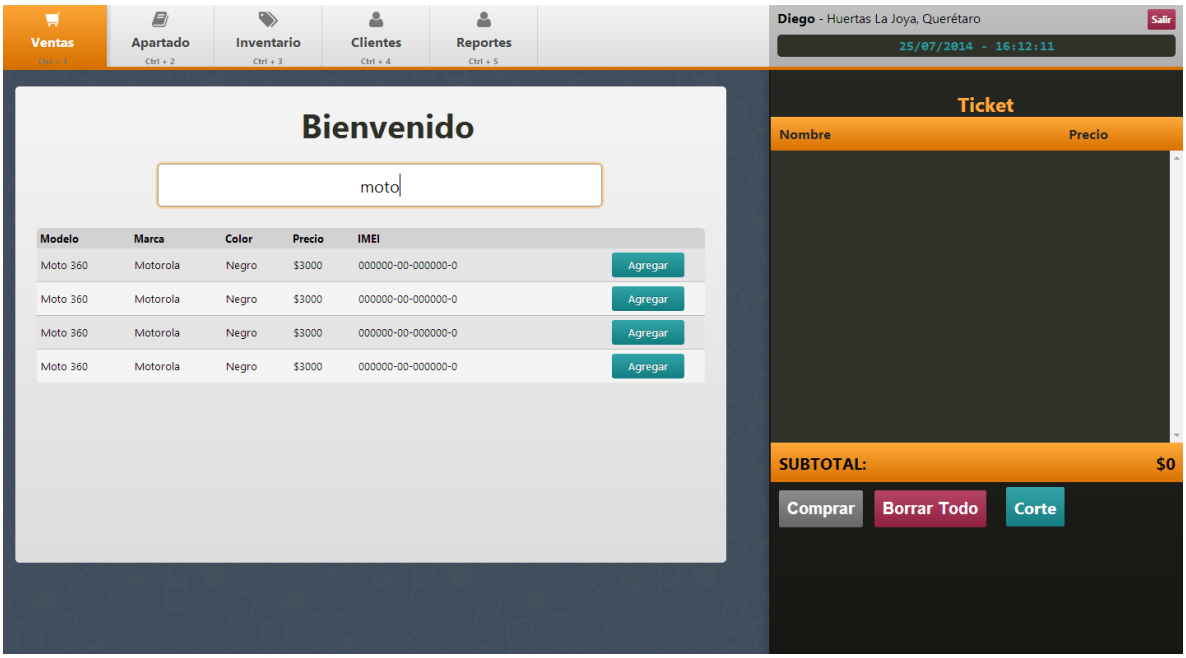


Figura 8-13 Pantalla de Ventas acción "agregar a ticket" del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

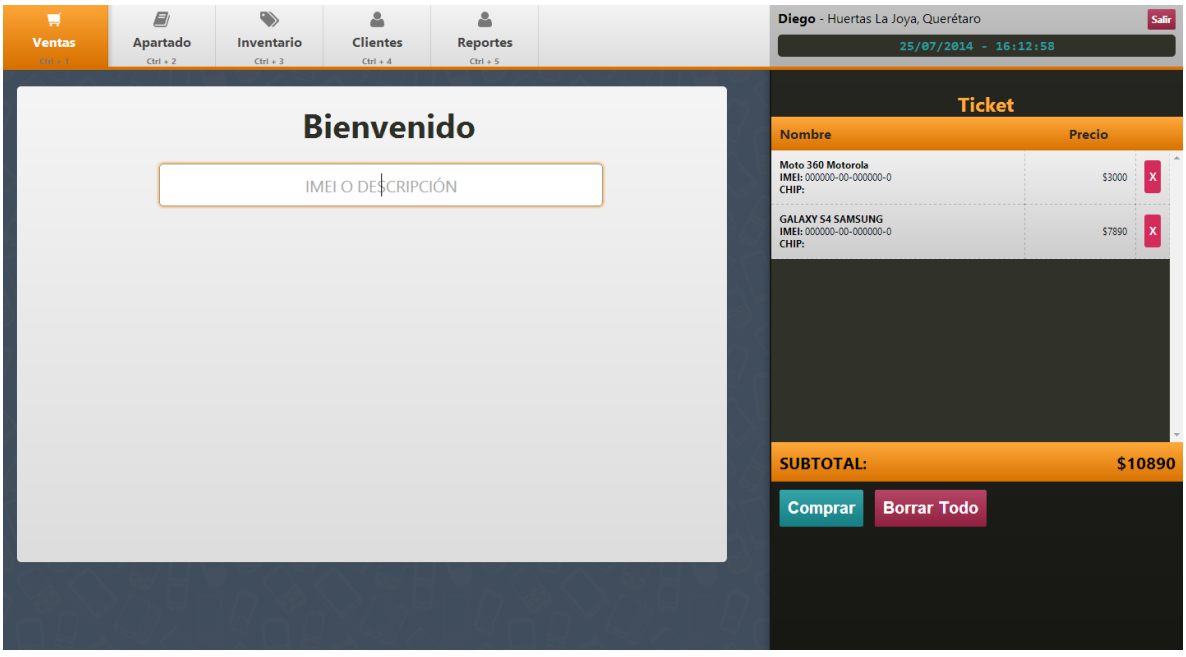


Figura 8-14 Pantalla de Ventas acción "comprar" del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

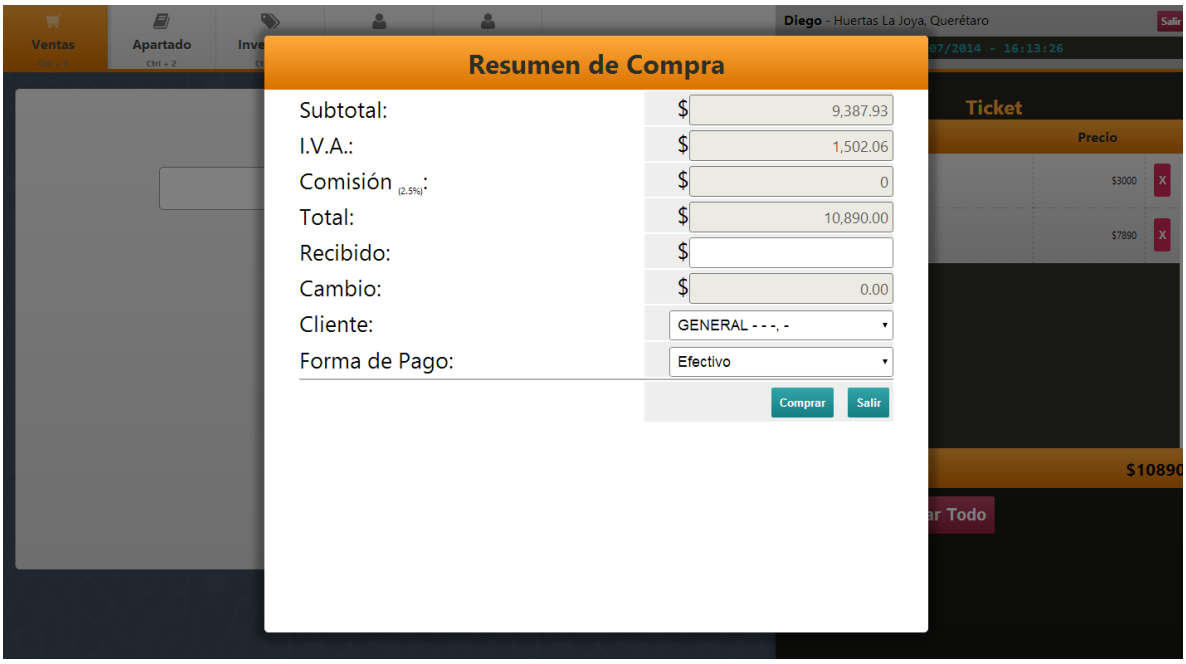


Figura 8-15 Pantalla de Agregar Apartado del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

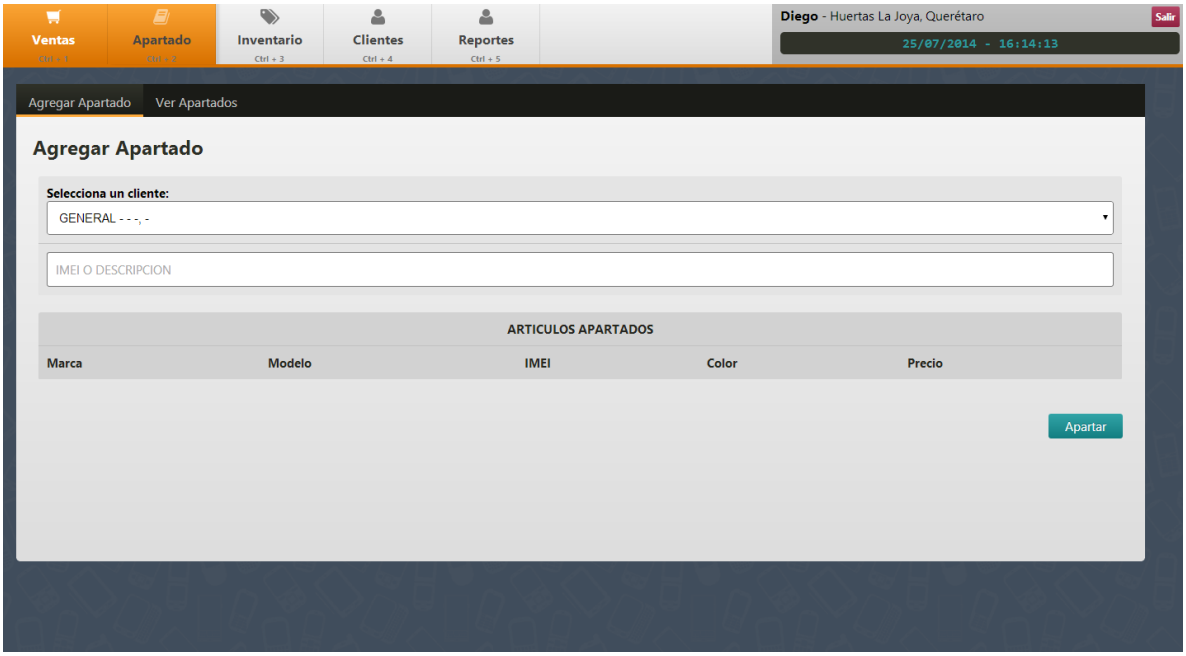


Figura 8-16 Pantalla de Ver Apartado del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

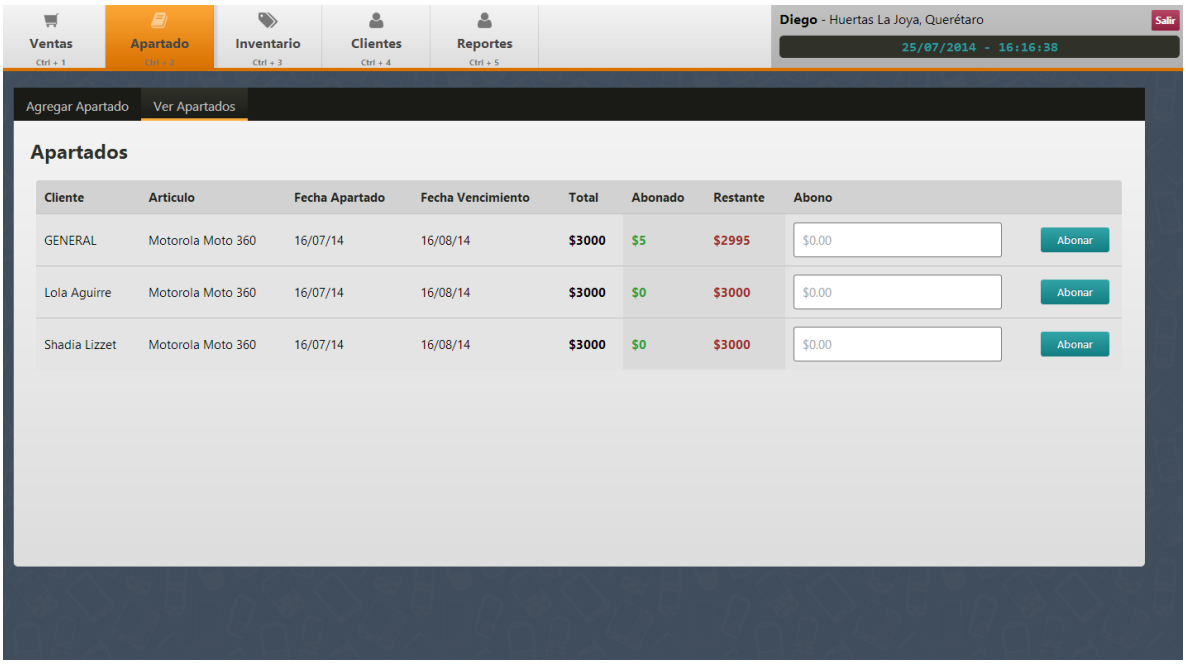


Figura 8-17 Pantalla de Agregar Inventario del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

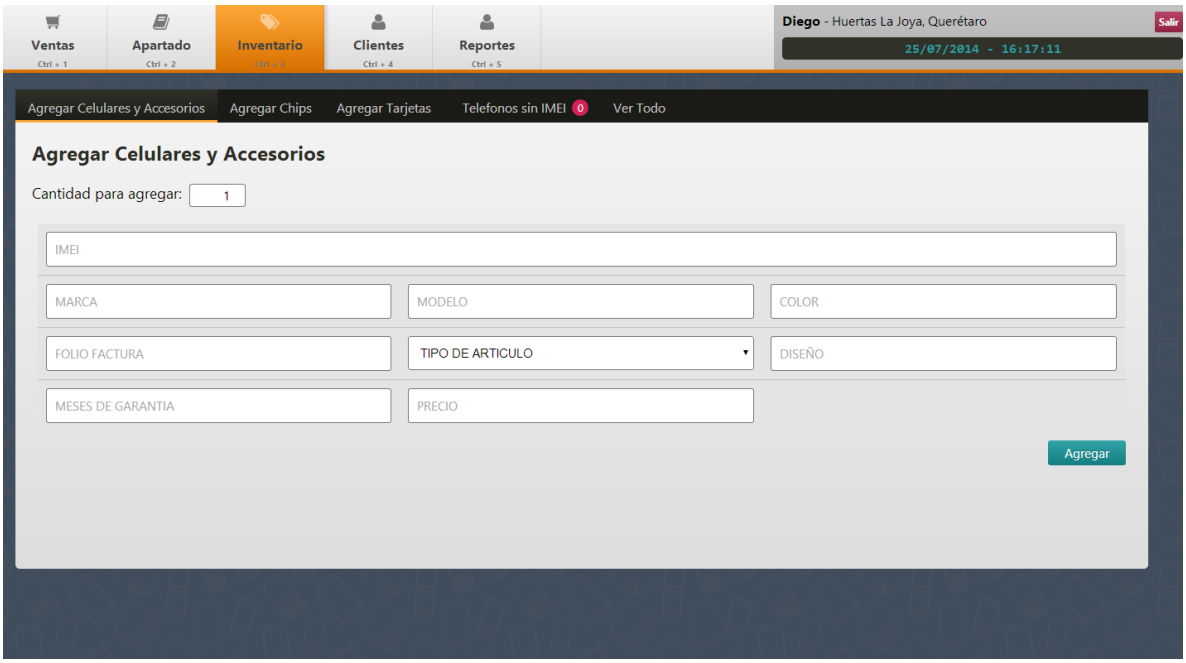


Figura 8-18 Pantalla de Ver Inventario del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

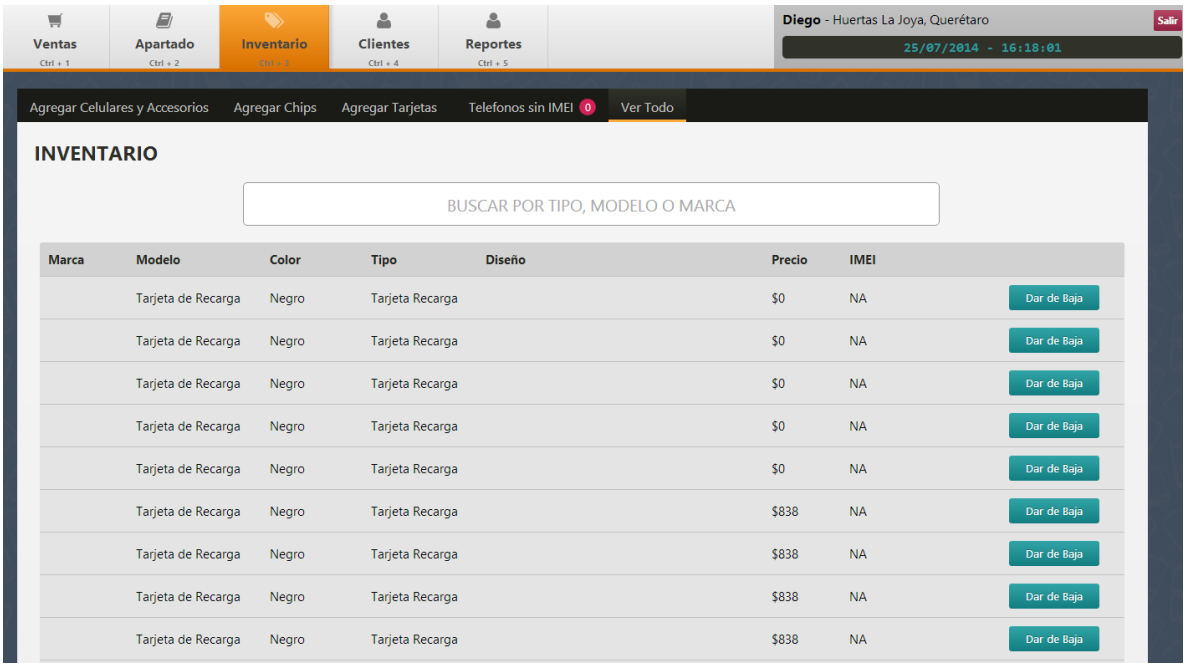


Figura 8-19 Pantalla de Agregar Cliente del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

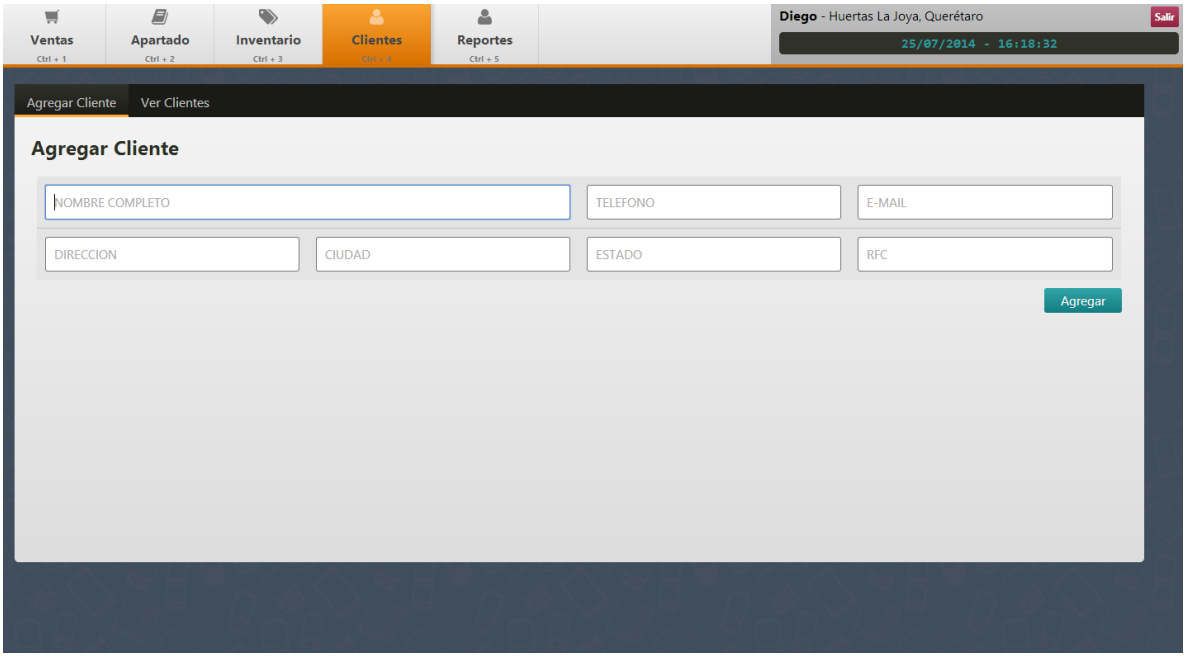


Figura 8-20 Pantalla de Agregar Cliente del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.

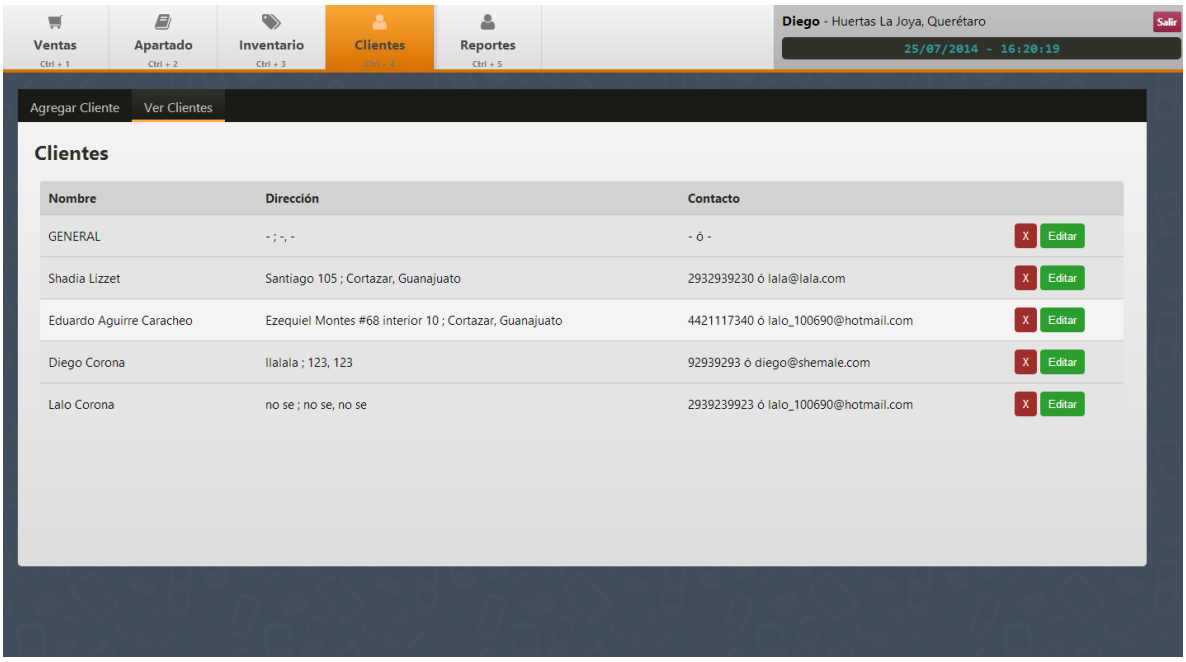
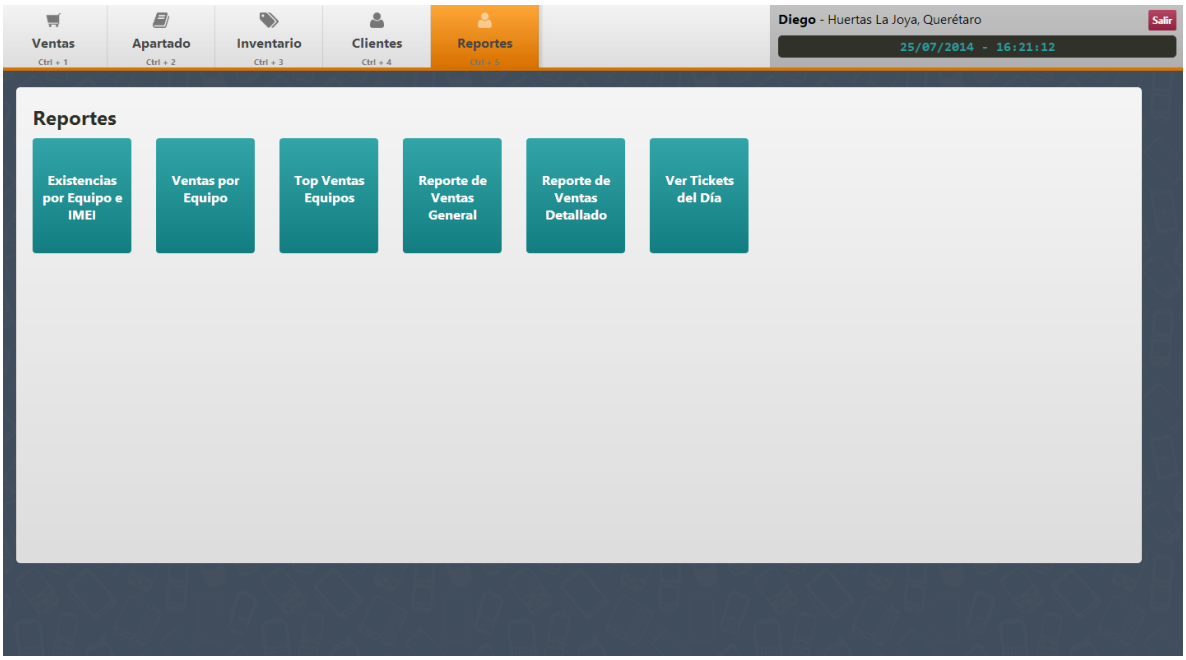


Figura 8-21 Pantalla de Agregar Cliente del sistema de punto de venta "Homecel". Fuente: Elaboración Propia.



9. CONCLUSIONES

Después de llevar a cabo la presente investigación y analizar el resultado de los casos prácticos, se llegó a las siguientes conclusiones.

El uso del internet para el desarrollo de sistemas de información es una tendencia en crecimiento y es vista como la plataforma favorita de muchos usuarios y desarrolladores.

La Ingeniería Web como disciplina aun no es tan madura como la Ingeniería en Software, sin embargo, cada vez más son los esfuerzos que buscan que esta disciplina crezca en calidad y uso.

Una de las características principales en la administración de un proyecto de desarrollo web son los cortos tiempos sobre los que se tiene que trabajar un proyecto, por lo tanto, las metodologías que son más factibles para usarse con las ágiles.

Las metodologías existentes para el desarrollo en la plataforma web están desactualizadas, y debido a que la web es una plataforma que se actualiza día con día, no son factibles para ser utilizadas como única guía durante el desarrollo de un sistema.

Muchas de las metodologías web generalizan u omiten proceso que realmente son necesarios durante el desarrollo de un proyecto.

Diseñar una metodología que fuera lo suficientemente específica y robusta para ser aplicada en cualquier proyecto pero que conservara su enfoque ligero se vuelve una tarea difícil de lograr.

La metodología diseñada en esta tesis cumple con el enfoque ligero, sin embargo específica de manera clara y estructurada el proceso de desarrollo lo suficientemente completo como para ser aplicada en cualquier proyecto web.

En los casos prácticos llevados a cabo utilizando la metodología o el borrador de la metodología se encontró que los objetivos se cumplieron dentro de los tiempos y presupuesto acordados.

Aunque esta propuesta de metodología está preparada para enfrentar el desarrollo de un proyecto, necesita atravesar un proceso de uso lo suficientemente extenso como para madurar.

10. REFERENCIAS

About: World Wide Web Foundation. (Consultado en el 2014). Obtenido de World Wide Web Foundation: https://webfoundation.org/about/sir-tim-berners-lee/

Achebe, I. (2002). Comparative Study of Web Design Methods. WISE Research Group.

Agarwal, R., Prasad, J., Tanniru, M., & Lynch, J. (2000). Risks of Rapid Application Developmnet. *Communications of the ACM*. ACM.

Balaji, S. M. (2012). Waterfall vs V-Model vs Agile: A Comparative Study on SDLC. *International Journal of Information Technology adn Buisness Management*, 26-30.

Balasubramanian, V., & Bashian, A. (1998). Document Management and Web Technologies: Alice Marries the Mad Hatter. *Communications of the ACM*, 107-114.

Boehm, B. (2000). Spiral Development: Experience, Principles, and Refinements. *Spiral Development Workshop*.

CASEMaker Totem. (2000). What is Rapid Application Development. CASEMaker Inc.

Cornford, T. S. (2013). Introduction to Information Systems. United Kingdom: University of London.

De Troyer, O., & K., L. (2000). WSDM: a user centered design method for web sites. Tilburg Universiti, Infolab.

Deshpande Y., M. S. (2002). WEB ENGINEERING. *Journal of Web Engineering*, 15.

Dudziak, T. (2000). eXtreme Programming An Overview. *Methoden und Werkzeuge der Softwareproduktion*.

Gall, D., Berntsson, R., & Johansson, J. (s.f.). Dynamic Systems Development Method and Rapid Application Development.

Ginige, A. (s.f.). Web Engineering: Methodologies for Developing Large and Maintainable Web Based Information Systems. Australia: University of Sydney.

Howcroft, D., & Carroll, J. (s.f.). A Proposed Methodology for Web Development. IS Research Centre.

Laboratorio Nacional de Calidad del Software. (Marzo de 2009). Ingeniería del Software: Metodologias y Ciclos de Vida. España.

Leiner, B., Cerf, V., & Clark, D. K. (Octubre de 2009). A Brif History of the Internet. *ACM SIGCOMM*. ACM.

Lindstrom, L., & Jeffires, R. (2003). Extreme Programming and Agile Software Development Methodologies. CRC Press LLC.

Mohammed, N., & A., G. (2010). A comparison Between Five Models OF Software Engineering. *IJCSI Internationa Journal of Computer Science Issues*.

Murugesan, S., & Ginige, A. (2005). Web Engineering: Introduction and Perspectives. Idea Group Inc.

Petersen, K., & C., W. (2009). The Waterfall Model in Large-Scale Development. *PROFES 2009*, 386-400.

Russo, N., & Graham, B. (1998). *A first step in Developing aWeb Application Design Methodology.*

Sacha, K. (2007). Software Engineering Practices: An auditor's perspective. Poland: Warsaw University of Technology.

Schwaber, K. (s.f.). SCRUM Development Process. *Advanced Development Methods*.

Serena Software. (Junio de 2007). *An Introduction to Agile Software Development.* Obtenido de Serena Software: http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf

Shwabe, D., & Moura, I. (s.f.). OOHDM-Web: An Environment for Implementation of Hypermedia Applications in the WWW. Dept. of Informatics, PUC-Rio.

Shwaber, K., & J., S. (Octubre de 2011). The Scrum Guide. The definitive Guide to Scrum: The rules of the game. *Improving the Profession of Software Development*. Scrum.org.

Standing, C. (27 de Junio de 2001). The Requirements of Methodologies For Developing Web Applications. *Global Co-Operation in the New Millenium.* Slovenia: The 9th European Conference on Information Systems.

Standing, C. (2002). Methodologies for Developing Web Applications. *Information and Software Technology 44*. Elsevier.

Suh, W. (2005). *Web Engineering: Principles and Techniques.* Hershey, Estados Unidos: Idea Group Publishing.

Voigt, B. (Enero de 2004). Dymanic System Development Method. Department of Information Technology University of Zurich.

Wirth, N. (2007). A brief History of Software Engineering. *IEEE Annals of the History of Computing*. IEEE Computer Society.

Zeinoun, I. (2005). The Rapid Application Development Process. Cambridge Technology Enterprises.