Universidad de las Ciencias Informáticas

Facultad 3

Título: Sistema para la integración de métodos de estimación de proyectos de software.

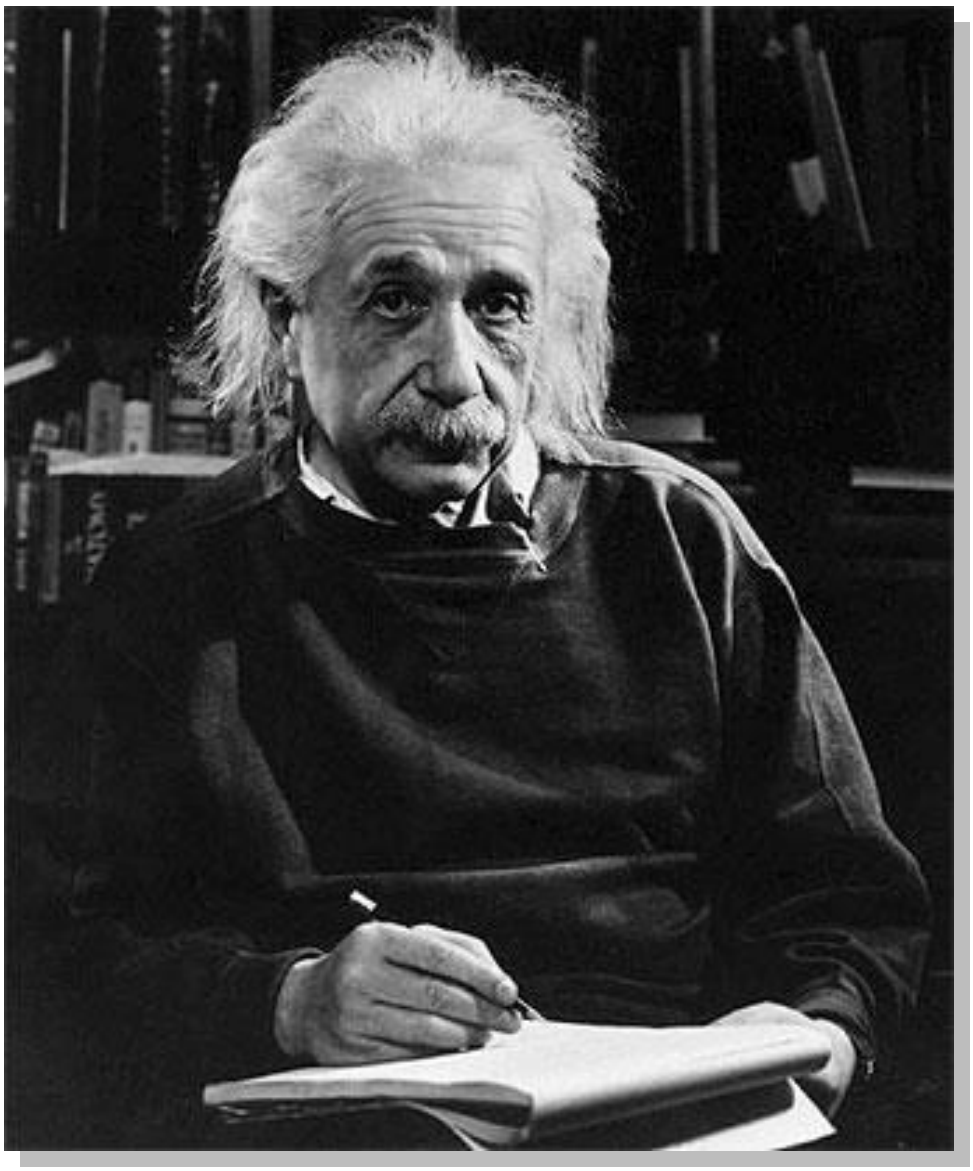
Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autor: Alian Villa Ochoa

Tutor: Ing. Mark Guzman Novik

Cotutor: Ing. Adrián Avila Atencio

La Habana, Junio de 2016



La imaginación es más importante que el conocimiento. El conocimiento es limitado. La imaginación rodea el mundo.

Albert Einstein

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo al Centro de Gobierno Electrónico CEGEL de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Alian Villa Ochoa

_____

Mark Guzman Novik

_____

AGRADECIMIENTOS

Quisiera empezar este apartado, agradeciendo a las personas más importantes de mi vida, ellos son mis padres, a mi mamá por ser la madre que ha sido durante toda la vida, por darme el apoyo y el amor que cada madre le debería dar a sus hijos, por estar presente en los momentos más difíciles. Gracias mami.

A mi papá, por ser mi guía, mi ejemplo a seguir, mi consejero, mi amigo, mi hermano, mi padre, por darme la respuesta que necesito en el momento justo y en el lugar indicado, por inculcarme que con el esfuerzo, el sacrificio y el respeto, todo en la vida se logra, sin importar cuán grande sea el obstáculo. Gracias papi.

A mi hermanita hermosa, por ser tan buena hermana y estar presente en toda ocasión. Te quiero mucho.

A mi familia en general, a mis tíos y mis primos, que de una forma u otra, me han apoyado durante el transcurso de la carrera.

A mi "hermano"de toda la carrera, Juan Carlos López, por darme tantos dolores de cabeza y yo a él, por obligarme a estudiar para las pruebas. Por estar presente en todo momento. Gracias mi hermano.

A mi amigo que lo quiero como un hermano, Adrián Avila, por estar presente en cada momento que he necesitado una mano amiga, gracias mi hermanito.

A mi tutor y amigo Mark Guzman, por darme los consejos y por su gran ayuda con la tesis.

A mi gran amigo Walter, por estar siempre presente en el momento que lo necesité durante toda la carrera. Gracias mi hermanito.

A mis compañeros de aula, los que están y los que han quedado en el camino, que de una forma u otra han aportado conocimientos y sentimientos en mí, que me han ayudado a ser una mejor persona.

A mi novia Yusnavi, por haber sido mi apoyo incondicional durante este tiempo, por enseñarme que el amor existe y que tiene mucho para darme. A ti que en tan poco tiempo te has convertido en alguien tan importante en mi vida. Te quiero mi vida.

A mis compañeros del baloncesto, por permitirme vivir momentos de triunfos y fracasos, de victorias y frustraciones, que me han permitido tomar decisiones correctas en los momentos correctos. Gracias a todos.

DEDICATORIA

Quiero dedicarle este trabajo, a mis padres, por haber mostrado plena confianza en mí y en el sueño de tener un hijo ingeniero, que en este momento se ha cumplido su sueño y el mío.

RESUMEN

La Universidad de las Ciencias Informáticas (UCI) es un centro de estudios y de producción de software que vincula el concepto estudio-trabajo logrando una mejor preparación de sus estudiantes, con el objetivo de formar profesionales preparados e integrales. Las soluciones informáticas desarrolladas en la UCI dirigidas a diversos sectores de la sociedad tienen el reto de insertarse no solo en el mercado nacional sino en el internacional; haciendo de la planificación de proyectos de software una tarea que se consolida con la práctica. Un aspecto importante a destacar en los proyectos es la estimación de software debido a que es la encargada de decir si el mismo es viable o no y para esto se cuenta en la universidad con el Método de Estimación UCI. La presente investigación surge a partir de la inexistencia de una herramienta que contenga múltiples métodos de estimación. El objetivo del presente trabajo es desarrollar un sistema que integre métodos de estimación para determinar tiempo y esfuerzo, aspectos fundamentales de un proyecto de desarrollo de software exitoso. Para esto se propone un estudio de los aspectos generales del proceso de desarrollo de software y se describe el proceso de software de la solución en cuestión.

PALABRAS CLAVE

Estimación, Métodos de estimación, Proceso de desarrollo de software, Proyecto.

ÍNDICE

SISTEMA PARA LA INTEGRACIÓN DE MÉTODOS DE ESTIMACIÓN DE PROYECTOS DE SOFTWARE

INTRODUCCIÓN

En los últimos años, las Tecnologías de la Información y las Comunicaciones (TIC) han evolucionado considerablemente, convirtiéndose en una valiosa herramienta para la gestión y transformación de la información. Su introducción en diferentes sectores de la sociedad, ha permitido que el trabajo sea mucho más rápido y eficiente.

Cuba no está ajena a estos cambios que hoy en día tienen un alto peso para el desarrollo de las tecnologías y la sociedad. Por tal motivo, aprovecha las ventajas que brindan y se ha llevado a cabo un proceso de informatización de la sociedad, en aras de fomentar el uso ordenado y masivo de estas tecnologías. En los últimos años se ha percibido un gran cambio en cuanto al crecimiento del uso de tecnologías en ramas con gran impacto social como la educación, la salud y la ciencia.

Como parte de todo ese proceso de informatización, se crea la Universidad de las Ciencias Informáticas (UCI), la cual tiene como misión: servir de soporte a la industria cubana del software, la producción de sistemas y servicios informáticos a partir de la vinculación estudio-trabajo como modelo de formación (UCI, 2012). En la UCI, se crearon varios centros de producción, en los que se llevan a cabo proyectos investigativos y de desarrollo de software a partir de la ejecución de los proyectos.

La UCI, desde sus inicios en la producción de software hasta la actualidad, ha mostrado varios inconvenientes relacionados con el proceso de estimación del esfuerzo y la duración del desarrollo en proyectos concebida en la planificación.

La estimación es una de las actividades de la planificación durante el desarrollo de software. Su objetivo es conocer en etapas tempranas y de manera aproximada, el costo, la duración y los recursos necesarios para el desarrollo de los proyectos.

La estimación de proyectos de software no es una ciencia exacta ya que existen numerosas variables humanas, técnicas, del entorno y políticas, que intervienen en su proceso y pueden afectar los resultados finales, pero una combinación de buenos datos históricos y técnicas sistemáticas puede mejorar la precisión de la estimación. Esta actividad no debe llevarse a cabo de forma descuidada puesto que es la base de todas las actividades de planificación de un proyecto (Pressman R., 2002).

SISTEMA PARA LA INTEGRACIÓN DE MÉTODOS DE ESTIMACIÓN DE PROYECTOS DE SOFTWARE

Existen dos grandes grupos de tipos de métodos de estimación de proyectos de software, los métodos empíricos, que se basan en la experiencia y los principales son el juicio de experto y el método por analogía. Los métodos paramétricos para la estimación de proyectos de software, es el otro grupo, que se basa en el uso de ecuaciones matemáticas asociadas en el que escenarios alternativos son definidos mediante la variación de los valores asumidos en un grupo de parámetros. Los líderes de proyectos de software usan modelos paramétricos de software o herramientas paramétricas de estimación para generar estimaciones de la duración, coste y necesidades de personal de un proyecto. De las ventajas de utilizar métodos de estimación paramétricos, es que su uso no depende de que en la empresa u organización, exista un experto en el tema para realizar métodos de estimación empíricos, así con la utilización de cálculos de fórmulas matemáticas, quien realice las estimaciones no debe ser un experto en el tema, ya que se guía por los pasos que brindan cada método de estimación.

Mediante una entrevista realizada a varios líderes de proyectos de la UCI, se identificaron los métodos de estimación que utilizan (Figura 1) además de varias deficiencias.

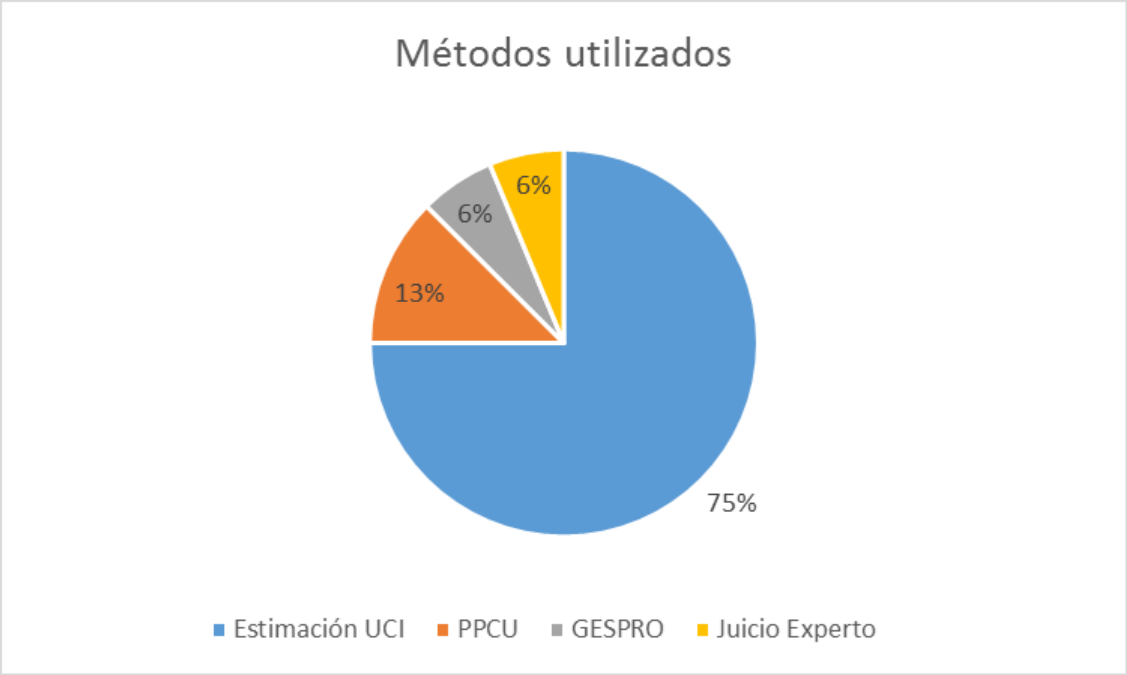


Figura 1: Métodos utilizados en la UCI.

Dentro de ellas se destacan las siguientes:

- Se basan en la utilización de un único método de estimación (el método de estimación UCI, que su uso se encuentra dentro de las políticas de la universidad) así como una única herramienta para estimar (herramienta de estimación UCI, que en su mayoría utilizan una Plantilla Excel).
- El proceso de estimación se hace manual en múltiples ocasiones.
- El déficit de otras herramientas que ejecuten otros métodos de estimación.
- La necesidad de obtener varios resultados de estimación para comparar resultados.

Hasta la actualidad, no se ha logrado determinar un método de estimación que se comporte mejor que el resto de los métodos en todas las circunstancias, por lo que podría resultar útil verificar la estimación con el método Estimación UCI, a partir de otros métodos para comprobar cuán alejado o cercano se encuentran de los resultados reales brindados.

Todas estas deficiencias y otras influyen en que la estimación de software se realice con problemas, ya que con la utilización de los métodos que no están informatizados, se pueden arrastrar errores a la hora de realizar cálculos matemáticos y esto puede llevar a que los resultados sean alejados a las necesidades de los proyectos, causando la incertidumbre por parte del cliente, así como la del grupo de desarrollo. Además, en los proyectos no siempre existe algún experto para realizarle el método juicio de experto a la estimación. En el caso de la utilización del Método Estimación UCI (Hojas de cálculos), los resultados se almacenan en carpetas de ficheros, lo cual constituye una desventaja a la hora de realizar la búsqueda de proyectos similares para poder comparar resultados.

A partir de la problemática planteada, se define el siguiente **problema a resolver:** ¿Cómo desarrollar una herramienta que integre múltiples métodos de estimación paramétricos?

El **objeto de estudio** de esta investigación es el proceso desarrollo de software.

Se determina como **campo de acción** el proceso de desarrollo de software de gestión.

SISTEMA PARA LA INTEGRACIÓN DE MÉTODOS DE ESTIMACIÓN DE PROYECTOS DE SOFTWARE

Con la finalidad de darle solución al problema propuesto se identifica el siguiente **objetivo general:** Desarrollar un sistema que integre múltiples métodos de estimación paramétricos que sirva de apoyo al proceso de estimación de software.

A partir de los que se desglosan los siguientes **objetivos específicos:**

1. Establecer el marco teórico acerca del proceso de desarrollo de software.
2. Determinar la Metodología de desarrollo de software a utilizar.
3. Construir los artefactos que permitan el desarrollo del sistema.
4. Probar la validez del resultado.

Como **posible resultado** a entregar:

Sistema que integre los métodos de estimación paramétrica Estimación UCI, Puntos de Función y Puntos por Casos de Uso.

Para el desarrollo de este trabajo se utilizan los siguientes **Métodos Científicos:**

Métodos Teóricos:

El método **Analítico-Sintético**, permite hacer un estudio y análisis de la bibliografía existente relacionada con la investigación y a partir de ahí es posible sintetizar, resumir y comprender el proceso de estimación relacionado con el esfuerzo y tiempo.

Métodos Empíricos:

Se utiliza la técnica de investigación **encuesta**, donde se visitan una gran cantidad de proyectos pertenecientes a la UCI, realizándole la misma a los Líderes de Proyecto donde se comprueba qué métodos para la estimación utilizan, además de las deficiencias que encuentran mediante el proceso de estimación.

El presente trabajo está estructurado en 3 capítulos, los cuales están distribuidos de la siguiente manera:

- ✓ **Capítulo 1: Fundamentación teórica.** Este capítulo se establece un modelo teórico sobre el proceso de desarrollo de software. Además se hace un estudio sobre las diferentes herramientas, tecnologías y metodologías para el desarrollo de la investigación que se utilizan en la actualidad.

- ✓ **Capítulo 2: Planificación y diseño del sistema.** Este capítulo contiene la propuesta de solución a través de un conjunto de características identificadas. Se selecciona la metodología de desarrollo de software y las herramientas y tecnologías a utilizar. Se definen las principales particularidades que posee la solución informática a desarrollar mediante el uso de artefactos provenientes de las fases de planificación y diseño.

- ✓ **Capítulo 3: Implementación y pruebas del sistema.** Este capítulo contiene una serie de artefactos propuestos para las fases de implementación y pruebas así como los mecanismos de implementación utilizados. Para garantizar la calidad del producto final se hacen una serie de pruebas de unidad y aceptación.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

En este capítulo se enuncian los conceptos fundamentales sobre el proceso de desarrollo de software, los cuales permitirán un mejor entendimiento del presente trabajo y una base de cómo se desarrollará la solución.

1.1 Proceso desarrollo de software

Son muchos los conceptos y definiciones sobre el proceso de desarrollo de software, dentro de las que se destacan las siguientes.

La norma internacional ISO-9001 (ISO, 2015) define un proceso como "*una actividad que utiliza recursos, y que se gestiona con el fin de permitir que los elementos de entrada se transformen en resultados*".

Roger Pressman define el software como "*el producto que los ingenieros de software construyen y después mantienen en el largo plazo*" (Pressman R., 2002).

"*El software, es el conjunto de programas de cómputo, procedimientos, reglas, documentación y datos asociados que forma parte de las operaciones de un sistema de computación*"(Castro, 2011).

Según Sommerville (2005), para muchas personas los software son solo programas de computadora, sin embargo, comenta que además, son todos aquellos documentos asociados a la configuración de datos que se necesitan para hacer que estos programas operen de manera adecuada. Estos productos de software se desarrollan para algún cliente en particular o para un mercado en general. Para el diseño y desarrollo de proyectos de software se aplican metodologías, modelos y técnicas que permiten resolver los problemas. En los años 50 no existían metodologías de desarrollo, el desarrollo estaba a cargo de los propios programadores. De ahí la importancia de contar con analistas y diseñadores que permitieran un análisis adecuado de las necesidades que se deberían de implementar.

Jacobson y otros autores lo definen como el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software (Jacobson, Booch, & Rumbaugh, 2000).

Los autores antes mencionados, coinciden en que el proceso de desarrollo de software es la descripción de una secuencia de actividades que se deben seguir por un equipo de trabajadores para generar un conjunto coherente de productos, donde se aplican metodologías de desarrollo, modelos y técnicas que permiten resolver uno o varios problemas.

El proceso de desarrollo de software describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de este programa es definir las distintas fases intermedias que se requieren para validar el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

El proceso de desarrollo de un software consta de los siguientes procedimientos:

- Definición de objetivos: definir el resultado del proyecto y su papel en la estrategia global.
- Análisis de los requisitos y su viabilidad: recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.
- Diseño general: requisitos generales de la arquitectura de la aplicación.
- Diseño en detalle: definición precisa de cada subconjunto de la aplicación.
- Programación (programación e implementación): es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.
- Prueba de unidad: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.
- Integración: para garantizar que los diferentes módulos se integren con la aplicación.
- Prueba beta (o validación), para garantizar que el software cumple con las especificaciones originales.
- Documentación: sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.
- Implementación: es cuando se realiza el despliegue del software y se le comienza a dar uso por parte de los clientes.
- Mantenimiento: para todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo) (Benchmark, 2016).

Como antes se mencionaba, los procedimientos o métodos durante el proceso de desarrollo de software, brindan una guía base para obtener una solución informática, independientemente del tipo de software que se desee obtener, ya sea un software de sistema, basado en la Web, de gestión, etc.

En el siguiente epígrafe se muestran algunos de los múltiples tipos de software que existen en la actualidad, así como el tipo de software que se seleccionará para la solución.

1.2 Tipos de software

El software puede aplicarse a numerosas situaciones del mundo real. En primer lugar, a todos aquellos problemas para los que se haya establecido un conjunto específico de acciones que lleven a su resolución (Aguilera, 2015).

Algunas veces es difícil definir categorías genéricas para las aplicaciones del software que sean significativas. Acorde aumenta la complejidad del software, es más difícil establecer comportamientos nítidamente separados (Pressman, 2002).

Las siguientes áreas del software indican la amplitud de las aplicaciones potenciales.

- **Software de sistema**

  Pressman define un software de sistema como un conjunto de programas que han sido escritos para servir a otros programas. Algunos programas de sistemas procesan estructuras de informaciones complejas pero determinadas. Otras aplicaciones de sistemas procesan datos en gran medida indeterminados. En cualquier caso, el área del software de sistemas se caracteriza por una fuerte interacción con el hardware de la computadora; una gran utilización por múltiples usuarios; una operación concurrente que requiere una planificación, una compartición de recursos y una sofisticada gestión de procesos; unas estructuras de datos complejas y múltiples interfaces externas (Pressman, 2002).

  Aguilera propone que un software de sistema, está formado por todos aquellos programas cuya finalidad es servir al desarrollo o al funcionamiento de otros programas. Estos programas son muy variados: editores, compiladores, sistemas operativos, entornos gráficos, programas de telecomunicaciones, etc. pero se caracterizan por estar muy próximos al hardware, por ser utilizados concurrentemente por numerosos usuarios y por tratarse de programas de amplia difusión, no estando

diseñados normalmente a medida. Esto permite un mayor esfuerzo en su diseño y optimización, pero también les obliga a ser muy fiables, cumpliendo estrictamente las especificaciones para las que fueron creados. Un ejemplo de este tipo de software son los sistemas operativos, como Windows y Unix (Aguilera, 2015).

Los autores estudiados, coinciden en que un software de sistema consiste en un software que sirve para controlar e interactuar con el sistema operativo, proporcionando control sobre el hardware y dando soporte a otros programas.

- **Software de tiempo real**

El software de tiempo real es el que coordina/analiza/controla sucesos del mundo real conforme ocurren, se denomina de tiempo real. Entre los elementos del software de tiempo real se incluyen: un componente de adquisición de datos que recolecta y da formato a la información recibida del entorno externo, un componente de análisis que transforma la información según lo requiera la aplicación, un componente de control/salida que responda al entorno externo, y un componente de monitorización que coordina todos los demás componentes, de forma que pueda mantenerse la repuesta en tiempo real (típicamente en el rango de un milisegundo a un segundo) (Pressman, 2002).

Otra definición es que son todos aquellos programas que miden, analizan y controlan los sucesos del mundo real a medida que ocurren, debiendo reaccionar de forma correcta a los estímulos de entrada en un tiempo máximo prefijado. Deben, por tanto, cumplir unos requisitos temporales muy estrictos y, dado que los procesos que controlan pueden ser potencialmente peligrosos, tienen que ser fiables y tolerantes a fallos. Por otro lado, no suelen ser muy complejos y precisan de poca interacción con el usuario. Un sistema de tiempo real es aquel en el que para que las operaciones computacionales estén correctas no depende solo de que la lógica e implementación de los programas computacionales sean correctos, sino también en el tiempo en el que dicha operación entregó su resultado. Si las restricciones de tiempo no son respetadas el sistema se dice que ha fallado (Aguilera, 2015).

Un software de tiempo real, según los autores estudiados, es aquel que interacciona con su entorno físico y responde a los estímulos del entorno dentro de un plazo de tiempo determinado. No basta con que las acciones del sistema sean correctas, sino que, además, tienen que ejecutarse dentro de un intervalo de tiempo determinado y precisan poca interacción con el usuario.

- **Software de gestión**

  El proceso de la información comercial constituye la mayor de las áreas de aplicación del software. Los <<sistemas>> discretos (Ejemplo: nóminas, cuentas de haberes-débitos, inventarios, etc.) han evolucionado hacia el software de sistemas de información de gestión que accede a una o más bases de datos que contienen información comercial. Las aplicaciones en esta área reestructuran los datos existentes para facilitar las operaciones comerciales o gestionar la toma de decisiones. Además de las tareas convencionales de procesamientos de datos, las aplicaciones de software de gestión también realizan cálculo interactivo (Ejemplo: el procesamiento de transacciones en puntos de ventas) (Pressman, 2002).

  El procesamiento de información de gestión constituye, casi desde los inicios de la informática la mayor de las áreas de aplicación de los ordenadores. Estos programas utilizan grandes cantidades de información almacenadas en bases de datos con objeto de facilitar las transacciones comerciales o la toma de decisiones. Además de las tareas convencionales de procesamiento de datos, en las que el tiempo de procesamiento no es crítico y los errores pueden ser corregidos a posteriori, incluyen programas interactivos que sirven de soporte a transacciones comerciales (Aguilera, 2015).

  Los autores coinciden en que el procesamiento de la información de gestión constituye la mayor área de aplicaciones de software. Además, que almacenan grandes cantidades de información almacenadas en bases de datos para facilitar las transacciones comerciales o la toma de decisiones y realizan cálculos iterativos.

Por otro lado, se tienen las aplicaciones de escritorio, que son programas que se instalan en el ordenador y sirven para realizar diferentes tareas como gestión de pedidos, gestión de incidencias, contabilidad, almacenamiento de datos, comunicación interna y externa, gestión de personal, gestión de empresas, etc.

Para ello las aplicaciones pueden trabajar sobre diferentes ámbitos y plataformas. Su ámbito de trabajo puede ser:

- **Local:** sólo en el ordenador instalado.
- **Red interna:** entre diversos ordenadores de la red o con un servidor.
- **Internet:** sobre una base de datos externa (Neosoft Sistemas SL, 2016).

Las aplicaciones basadas en Web son otro tipo de software, que son herramientas informáticas accesibles desde un navegador Web y con acceso a Internet o Intranet.

Las aplicaciones Web pueden realizar las mismas funciones que las señaladas en las aplicaciones de escritorio, además, cuentan con la ventaja de no tener que ser instaladas en ninguna computadora. Esto facilita su trabajo multiplataforma y su acceso distribuido de la información.

Una de las grandes ventajas es que no es necesario invertir en grandes máquinas, aunque la aplicación sea muy potente, ya que puede estar instalada en servidores en la red (Neosoft Sistemas SL, 2016).

En el próximo epígrafe, se ilustran las principales diferencias entre las aplicaciones basadas en la Web y las aplicaciones de escritorio, atendiendo a varios criterios de importancia para el uso de aplicaciones en la actualidad.

1.3 Aplicaciones Web y de escritorio

Las aplicaciones Web y aplicaciones de escritorio cumplen con un grupo de características, las cuales se muestran en la siguiente tabla.

Característica	Aplicación Web	Aplicación de escritorio
Personalización, actualización y soporte	Es suficiente con realizar los cambios en el servidor Web	Hay que realizarlos en cada estación de trabajo (PC) donde se tenga la aplicación
Accesibilidad y cobertura	Cualquier lugar con acceso a Internet o Intranet	Sólo en el computador donde se haya instalado previamente el software
Capacidad de usuarios concurrentes	Alta debido a la arquitectura de clientes ligeros que la pueden usar	Baja ya que la forma de diseño es centrada en un único usuario local
Portabilidad	El sistema puede ser usado con	Solo funciona en el sistema

	cualquier navegador Web	operativo para el cual fue creado
Infraestructura y movilidad	Solo se tiene que conectar a la Internet o Intranet	Está restringido a la ubicación de la computadora local
Seguridad eléctrica y lógica	Es responsabilidad del proveedor de servicio	Es responsabilidad del administrador de la compañía y de cada usuario que usa el sistema localmente.

Tabla 1: Tabla comparativa de las aplicaciones Web y de escritorio (Development, 2016)

Por otro lado, se tiene que las aplicaciones de escritorio poseen rapidez y agilidad: Si la aplicación está instalada en el mismo ordenador donde se ejecuta, necesariamente esta, será muy rápida. El acceso a datos locales y el mayor aprovechamiento de microprocesador de la PC hacen que la velocidad de la aplicación solo dependa del ordenador.

Las aplicaciones Web poseen *accesibilidad desde cualquier dispositivo*: Solo se necesita un navegador Web y una conexión a Internet o Intranet en dependencia del entorno para trabajar con una aplicación en la Web; s*eguridad*: Los datos son gestionados por una empresa distinta a la que trabaja con la aplicación. El usuario se despreocupa totalmente de copias de seguridad, integridad de datos, mantenimiento, etc.; *menor mantenimiento del equipo*: El ahorro en ampliar el equipo desde el que se trabaja se disminuye drásticamente, ya que básicamente solo es necesario tener instalado un navegador Web; *actualizaciones automáticas*: Cuando se realiza un cambio en la aplicación se recibirá automáticamente. No se necesita realizar ninguna acción para tener las últimas novedades introducidas en la aplicación; *multiplataforma*: se puede usar cualquier sistema operativo (Development, 2016).

1.3.1 Uso de aplicaciones Web

Después de haber realizado un estudio a distintos tipos de software, se decide realizar una aplicación Web.

Para darle desarrollo a este punto, se parte de la siguiente cuestión. ¿Aplicación Web o de escritorio?

La red mundial (World Wide Web –www--) se ha popularizado tanto en los últimos años que se ha convertido en la interfaz de usuario de facto para los productos de software y ha obligado al uso de

tecnologías nuevas. Las bases de datos son cada vez más sofisticadas y soportan desde un usuario en computadoras de bolsillo a miles de usuarios en computadoras centrales. Poco a poco, el desarrollo hecho a medida se va abandonando y los negocios compran productos de software muy probados, genéricos y con una buena base instalada de clientes (Albinagorta, 2015).

Una tendencia actual es el uso de las aplicaciones Web, que se han convertido en una herramienta de alto peso para el desarrollo de las empresas. Estas aplicaciones presentan una serie de ventajas y beneficios con respecto al software de escritorio, con lo cual se logra aprovechar y acoplar los recursos de una empresa de una forma mucho más práctica que el software tradicional.

Entre los beneficios que las aplicaciones desarrolladas para la Web tienen respecto a las aplicaciones de escritorio se encuentran:

- El trabajo a distancia se realiza con mayor facilidad.
- Para trabajar en la aplicación Web solo se necesita una computadora con un buen navegador Web y conexión a Internet o Intranet.
- Las aplicaciones Web no necesitan conocimientos previos de informática. Con una aplicación Web tendrá total disponibilidad en cuanto a hora y lugar, podrá trabajar en ella en cualquier momento y en cualquier lugar del mundo siempre que tenga conexión a Internet o Intranet.
- Las aplicaciones Web le permiten centralizar todas las áreas de trabajo (InternetYa, 2016).

Los autores estudiados, plantean que las aplicaciones Web, le facilitan el trabajo a todas aquellas personas que sean usuarios del sistema, ya que pueden acceder al mismo desde cualquier lugar que tengan acceso a Internet o Intranet. No necesariamente deben estar en su puesto de trabajo, ya que al acceder al servidor Web, mediante un navegador, pueden realizar su trabajo. Otras de las ventajas que proponen, son que la información del trabajo, no está distribuida en distintas computadoras, sino que se encuentran en el servidor a donde acceden. Por estas razones anteriormente planteadas, se desarrollará una aplicación Web.

Para agilizar el desarrollo de una aplicación Web en la actualidad, como resultado de esfuerzo de muchas personas, se crean los marcos de trabajo (o framework), que le “facilitan la vida” a los programadores, ya que estos contienen grupos de funcionalidades y una arquitectura previa.

1.4 Marcos de trabajo

Una de las tendencias actuales para el desarrollo de aplicaciones Web es el uso de marcos de trabajo o framework. En la actualidad, hay muchas empresas que brindan sus servicios para el desarrollo de una Web, pero no todas son eficientes. El uso de un framework ayuda mucho con la seguridad, reducir tiempo y tener un código optimizado, ordenado y entendible.

Según lo define Larman es un conjunto cohesivo de interfaces y clases que colaboran para proporcionar los servicios de la parte central e invariable de un subsistema lógico, contiene clases concretas y especialmente abstractas, que definen las interfaces a las que ajustarse, así como interacciones de objetos en las que participar, tiene clases abstractas que podrían contener tanto métodos abstractos como concretos (Larman, 2004).

Clifton define un marco de trabajo en el desarrollo de software como una estructura de soporte definida en la cual otro proyecto puede ser organizado y desarrollado (Codebox). Marc Clifton, en su artículo “What Is A Framework” publicado en el sitio Web del Proyecto Código (Code Project), menciona varios elementos categóricos de los marcos de trabajo. En aras de que los mismos facilitan el trabajo con complejas tecnologías, unifican componentes/objetos haciéndolos más útiles, promueven la implementación consistente y flexible de aplicaciones. Además, los mismos pueden ser fácilmente depurados y probados (Clifton, 2003).

A modo de resumen un marco de trabajo es un conjunto de bloques de software prefabricado que los desarrolladores pueden emplear, extender o personalizar a favor de desarrollar soluciones informáticas específicas. Con ellos los desarrolladores no tienen que preocuparse por implementar aplicaciones desde el inicio una y otra vez. Son colecciones de componentes/objetos que garantizan que ambos: diseño y código fuente puedan ser reutilizados.

Los marcos de trabajo para las aplicaciones Web, permiten el desarrollo de sitios Web dinámicos, Web services (servicios Web) y aplicaciones Web. El propósito de este tipo de marcos de trabajo es permitir a los desarrolladores el desarrollo de aplicaciones Web y centrarse en los aspectos interesantes, aliviando la típica tarea repetitiva asociada con patrones comunes de desarrollo Web. La mayoría de los marcos de trabajo de aplicaciones Web proporcionan los tipos de funcionalidad básica común, tales como sistemas de templates (plantillas), manejo de sesiones de usuario, interfaces comunes con el almacenamiento en

base de datos de contenido *cacheado*, y persistencia de datos. Normalmente, los framework de aplicación Web además promueven la reutilización y conectividad de los componentes, así como la reutilización de código y la implementación de bibliotecas para el acceso a base de datos.

Los mejores framework son especialmente buenos para organizar proyectos de gran magnitud, y a su vez tratando de mantenerse fuera del camino, sin imponerse por sobre el proyecto.

El más conocido patrón de diseño de aplicaciones Web es la arquitectura Model-View-Controller (MVC: Modelo-Vista-Controlador).

Hay una amplia gama de marcos de trabajo para aplicaciones Web que son distribuidos bajo licencia Open Source (Código abierto) que utilizan MVC, marcos de trabajo como Ruby on Rails, CodeIgniter, Django, CakePHP, Zend Framework, Symfony y Yii, que son de los framework más populares. Estos marcos de trabajo están basados en lenguajes de programación para el desarrollo Web como PHP[1], Ruby, Python, entre otros.

1.4.1 Modelo Vista Controlador

El uso del patrón arquitectónico MVC es una de las tendencias del desarrollo de aplicaciones Web. El MVC es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de los eventos y las comunicaciones. Para ello MVC propone la construcción de tres componentes distintos que son el *modelo*, la *vista*, y el *controlador*. Por un lado, se define componentes para la representación de la información, y por otro lado para interacción del usuario. Este patrón de arquitectura de software se basa en las ideas de reutilización de código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento (Burbeck, 1997) (ReensKuag, et al., 2009).

El MVC divide una aplicación interactiva en tres componentes. El modelo contiene la información central y los datos. Las vistas despliegan información al usuario. Los controladores capturan la entrada del usuario (Buschmann et al., 1996).

[1] PHP – Acrónimo de Hypertext Preprocessor

En la figura 2 se muestra su funcionamiento:

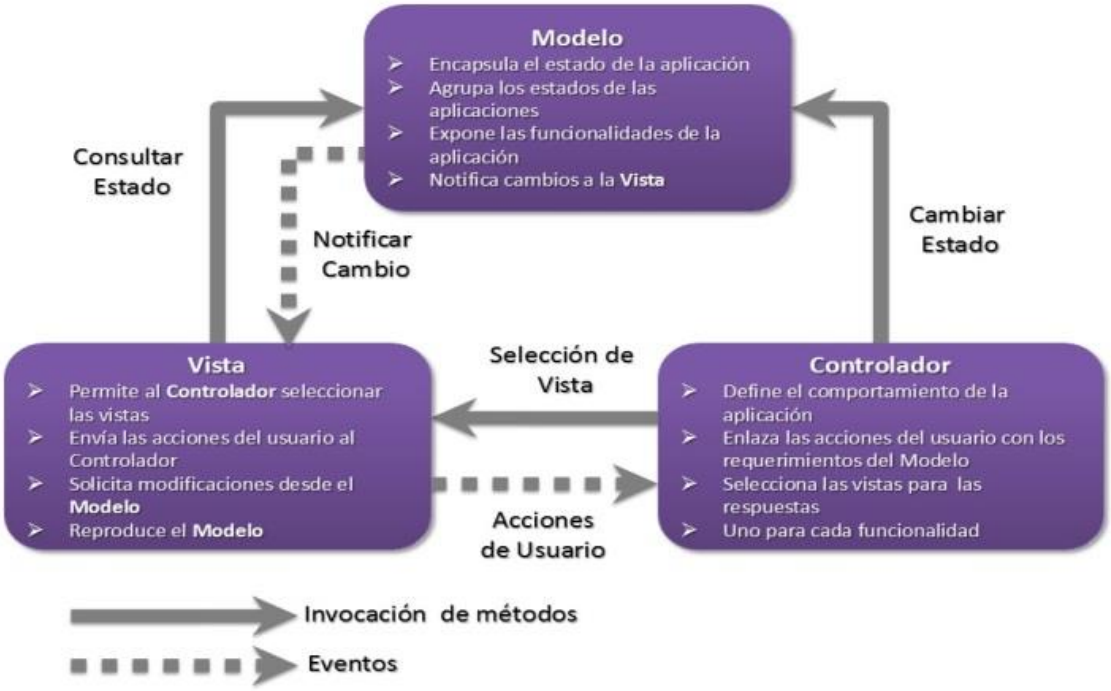


Figura 2: Patrón arquitectónico MVC.

Aunque originalmente MVC fue desarrollado para aplicaciones de escritorio, ha sido ampliamente adaptado como arquitectura para diseñar e implementar aplicaciones Web en los principales lenguajes de programación. Se han desarrollado multitud de marcos de trabajo, que implementan este patrón; estos marcos de trabajo se diferencian básicamente en la interpretación de cómo las funciones MVC se dividen entre cliente y servidor (Leff, et al., 2001).

Se usa MVC en el desarrollo Web porque este patrón brinda una separación total entre lógica de negocio y presentación. A esto se le pueden aplicar opciones como el multilenguaje, distintos diseños de presentación, etc., sin alterar la lógica de negocio. La separación de capas como presentación, lógica de negocio y acceso a datos, es fundamental para el desarrollo de arquitecturas consistentes, reutilizables y fácil de mantener, lo que al final resulta en un ahorro de tiempo en desarrollo en posteriores proyectos.

Al existir la separación de vistas, controladores y modelos es más sencillo realizar labores de mejora como:

- Agregar nuevas vistas.
- Modificar los objetos de negocios bien sea para mejorar el performance o para migrar a otra tecnología.
- Las labores de mantenimiento también se simplifican y se reduce el tiempo necesario para ellas. Las correcciones solo se deben hacer en un solo lugar y no en varios como sucedería si tuviésemos una mezcla de presentación e implementación de la lógica del negocio.
- Las vistas también son susceptibles de modificación sin necesidad de provocar que todo el sistema se paralice (Alvarez, 2014).

El uso de MVC en las aplicaciones Web, brindan ventajas para los desarrolladores, así como el uso de metodologías de desarrollo de software que guíen el ciclo de vida del software desde su inicio hasta el fin.

1.5 Metodologías de desarrollo de software

Para llevar a cabo el proceso de desarrollo de software, ya sea una aplicación de escritorio o Web, es necesario el uso de una guía de desarrollo bien definida, donde entran en escena las metodologías de desarrollo de software.

Según Piattini (1996), no hay un consenso entre los autores sobre el concepto de metodología, y por lo tanto no existe una definición universalmente aceptada. Sí hay un acuerdo en considerar a la metodología como "*un conjunto de pasos y procedimientos que deben seguirse para el desarrollo del software*" (Piattini, 1996).

Rumbaugh plantea que "*una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo"* (Rumbaugh, 2000).

De acuerdo a los diferentes criterios estudiados, se considera a la metodología de software como el conjunto de actividades que se llevan a cabo para la organización y el desarrollo del software.

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas.

Otra de las filosofías de las metodologías son las ágiles, las cuales dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo, pero manteniendo una alta calidad. Las metodologías ágiles están revolucionando la manera de producir software, generando un amplio debate entre sus seguidores y quienes por escepticismo o convencimiento no las ven como alternativa para las metodologías tradicionales.

Las metodologías ágiles y las tradicionales tienen diferencias, las cuales se muestran en la siguiente tabla:

Aspecto	Robusta o tradicional	Ágiles
Requisitos	Requieren los requisitos detallados desde el inicio del proyecto. Los requisitos no pueden cambiar.	Los requisitos son muy cambiantes. La verdad es que en software los requisitos cambian continuamente, y se requiere de un feedback (retroalimentación) sobre un resultado obtenido para determinar si es lo requerido o no.
Requisitos (funcionalidades innecesarias)	Debido a la recolección inicial de requisitos es frecuente que se soliciten funcionalidades	El enfoque continuo en el valor para el negocio no permite que se incluyan funcionalidades innecesarias.

	innecesarias.	
Cambios	Hacer un cambio al alcance requiere de un proceso formal de control de cambios.	El cambio es bienvenido en cualquier momento del proyecto.
Tiempo	Existe un compromiso respecto al tiempo de entrega del proyecto (no siempre se cumple esta meta).	Existe incertidumbre respecto al tiempo de entrega de todo el producto. Lo cierto es que máximo cada 2 meses (máximo un mes en Scrum) hay entrega de producto de valor para el cliente.
Costo	El costo del proyecto es definido para el proyecto.	Existe incertidumbre respecto al costo del proyecto. Se invierte en las funcionalidades que más valor le dan al cliente y cíclicamente se avanza hasta que se logre, ya sea: • el producto deseado • se acabe el presupuesto
Documentación	Atención exhaustiva a la documentación.	Solo se genera la documentación que genera valor al cliente y al proyecto.
El cliente	El cliente apoya el desarrollo del producto mediante la participación	Involucración directa del cliente en el desarrollo del producto.

	en reuniones.	El cliente es parte del equipo.
Iteraciones	Pocas iteraciones que generan gran volumen de información y software para construcción del producto.	Utilización de múltiples iteraciones de desarrollo para aprender y evolucionar el producto.
Riesgos	Los riesgos son asumidos por el proveedor.	Voluntad del cliente para compartir la responsabilidad en las decisiones y riesgos.
Se valora más	El proceso.	El individuo y las interacciones de los mismos.
La planeación	Requieren un plan detallado desde el inicio del proyecto.	Se va planeando a medida que se avanza en el proyecto. Planeación gradual y constante.
El éxito del proyecto	Es dado por el seguimiento del plan.	Es dado por la entrega continua de valor y funcionalidad al cliente.
Elaboración de entregables	Se generan entregables que requieren mucho tiempo de elaboración.	Se centran en hacer entregables en tiempos cortos con alta calidad inmersa.
La retroalimentación del cliente	Es conocida al final, pudiendo generar insatisfacción.	Es constante a lo largo del proyecto.
Participación del equipo	Empodera al Gerente de proyecto para el éxito del	Empodera al equipo para trabajar de

	mismo, este decide si participa de este poder o no al equipo o no.	forma creativa e innovadora.
Proceso(Plantillas)	Innumerables plantillas y artefactos para cumplir con el proceso.	Pocas plantillas y artefactos (solo los estrictamente necesarios para construir el producto).
Roles	Muchos roles para ejecutar el proyecto.	Pocos roles.
Arquitectura	Es un ejercicio que se realiza al inicio o en una etapa del proyecto.	Es un ejercicio constante durante el proyecto.

Tabla 2: Comparación metodologías ágiles y metodologías tradicionales.

Hoy en día, con el auge de la tecnología y con el objetivo de agilizar y automatizar los procesos en el desarrollo de software, existe la necesidad de implantar metodologías de desarrollo de software que ayuden a entregar un producto de calidad en tiempo y costo estimados, las metodologías ágiles de desarrollo de software han despertado interés gracias a que proponen simplicidad y velocidad para crear sistemas. Las metodologías tradicionales no se adaptan a las nuevas necesidades o expectativas que tienen los usuarios hoy en día, en parte que los métodos usados no son flexibles ante la posibilidad de la exigencia de nuevos requerimientos. Estos cambios generalmente implican altos costos, demanda de tiempo y la reestructuración total del proyecto que se esté llevando; en contraparte, los métodos ágiles permiten un desarrollo iterativo y adaptable que permite la integración de nuevas funcionalidades a lo largo del desarrollo del proyecto; para que tanto el cliente como el desarrollador queden satisfechos porque el producto final tiene una calidad adecuada.

Algunas de las metodologías de desarrollo de software más populares del tipo robustas, se encuentran RUP, PSP, MSF y Métrica 3, y dentro de las ágiles, ejemplos de ellas son XP, Scrum, Crystal Methodology, entre otras.

La tendencia actual es a seguir los principios del Agile Development o desarrollo ágil. Esta metodología mezcla un desarrollo basado en pruebas con un número bajo de diagramas y documentos. Existe y seguirá existiendo siempre ese paradigma para escoger entre hacerlo bien y hacerlo rápido. La mezcla de las dos características es un arte que no se ha podido lograr todavía, pero la tendencia es hacia ese objetivo (Deccach, 2015).

En la actualidad, los proyectos de desarrollo de software se rigen por el uso de las metodologías ágiles, realizando las funcionalidades por iteraciones, proponiendo al cliente (que también forma parte del equipo de desarrollo) pequeñas partes del producto en cada iteración.

1.6 Conclusiones del capítulo

- El uso de las aplicaciones Web, se ha convertido en una herramienta de alto peso para el desarrollo de las empresas.
- La utilización de marcos de trabajo para el desarrollo de aplicaciones Web, ayuda con la seguridad, reducir tiempo y tener un código optimizado, ordenado y entendible.
- El desarrollo de software actualmente está basado en el uso de metodologías de desarrollo ágil.

CAPÍTULO 2: PLANIFICACIÓN Y DISEÑO DEL SISTEMA

En el presente capítulo se describe la propuesta de solución para la situación problemática que dio origen a la presente investigación, a través de la identificación de los requisitos funcionales y características del sistema. Se selecciona la metodología de desarrollo de software además de las herramientas y tecnologías a utilizar mediante el proceso de desarrollo de software a realizar. Muestra los artefactos correspondientes a cada fase de la metodología seleccionada: las historias de usuario que reflejan las funcionalidades del sistema en cuestión, el plan de iteraciones en la fase de planificación y las tarjetas CRC[2] en la fase de diseño.

2.1 Métodos de estimación de proyectos de software

Los métodos de estimación de proyectos de software, están divididos en dos grandes grupos, como se mencionaba en la introducción del trabajo.

Los métodos de estimación de proyectos de software, que se tratan en el presente trabajo, son los métodos de estimación paramétricos: Estimación UCI, Puntos de Función y Puntos por Casos de Uso.

- ✓ **Método Estimación UCI**

El desarrollo del Método de Estimación parte de la necesidad en la UCI de estimar el tamaño y tiempo requerido para desarrollar un producto software. Dado que la misma no cuenta con una base histórica, se tuvo en cuenta los datos dispersos de algunos proyectos y la evaluación de algunos factores que, según criterio de expertos, pueden influir en las estimaciones del proyecto (Brito, 2012).

Para la elaboración del Método de Estimación UCI se realizó un estudio de otros métodos de la literatura y algunas buenas prácticas de los proyectos. Entre los factores definidos de complejidad se encuentra el Factor Cliente (FC), este permite determinar un nivel de incertidumbre asociado a la madurez que tiene el cliente en materia de asimilación de proyectos informáticos utilizándose para determinar la cantidad de unidades de desarrollo que se dejan como reserva. Otro fue el Factor de Valor Agregado (VA) que determina algunas acciones que pueden representar esfuerzo adicional al desarrollo de software, y se

[2] *Class, Responsabilities and Collaboration traducido al español como Clases Responsabilidades y Colaboración*

incluye en todas las etapas, fundamentalmente en la de Análisis y Diseño. También está el Factor de Ambiente (FA) el cual es interno, se determina a través de analizar las condiciones de la estructura de producción en la que se desarrollará el proyecto, no influye en el costo final, pero si en las valoraciones para la realización del proyecto y se utiliza como indicador para saber si la estructura de producción ha creado condiciones para cumplir los compromisos aplicándose en el precio de negociación y previo a la contratación. Por último el Factor Complejidad Técnica (CT) asociado a la complejidad de la tecnología y requerimientos no funcionales del sistema, influye directamente en el esfuerzo y fundamentalmente en la etapa de implementación (Brito, 2012).

Posteriormente para conseguir la eficacia requerida del producto se hizo uso de las métricas de tiempo, esfuerzo, recursos, costos y tamaño, logrando así una evaluación de la herramienta para conseguir alta calidad.

A raíz del Método de Estimación UCI surgió un segundo método de estimación, el cual resulta un complemento del primero con el objetivo de ser más preciso con el nombre de Método de Estimación Post Arquitectura. Para este método se tuvo en cuenta la definición de los requisitos del software y se tomó como punto de partida los resultados del Método de Estimación UCI. La única diferencia en cuanto a este último radica en que solo se contemplaron los Factores de Valor Agregado y Complejidad Técnica, siendo estos los que influyen en las estimaciones de este segundo método.

- ✓ **Método Puntos de Función**

El método de Puntos de Función se basa principalmente en la identificación de los componentes del sistema informático en términos de transacciones y grupos de datos lógicos que son relevantes para el usuario en su negocio. A cada uno de estos componentes se les asigna un número de puntos por función basándose en el tipo de componente y su complejidad; y la sumatoria de esto brinda los puntos de función sin ajustar. El ajuste es un paso final basándose en las características generales de todo el sistema informático que se está contando (Brito, 2012).

Los objetivos de calcular Puntos de Función son: medir lo que el usuario pide y lo que el usuario recibe; medir atributos independientemente de la tecnología utilizada en la implantación del sistema; proporcionar

una métrica de tamaño que dé soporte al análisis de la calidad y la productividad; proporcionar un medio para la estimación del software; y proporcionar un factor de normalización para la comparación de distintos software.

El análisis de los Puntos de Función se desarrolla considerando cinco parámetros básicos externos del sistema (Giraldo, 2002):

- Entrada (EI, External Input).
- Salida (EO, External Output).
- Consultas (EQ, External Query).
- Ficheros Lógicos Internos (ILF, Internal Logic File).
- Ficheros Lógicos Externos (EIF, External Interface File).

✓ **Método Puntos por Casos de Uso**

Puntos por Casos de Uso es un método que estima el esfuerzo de desarrollo de un producto de software a partir de los Casos de Uso y algunos factores de complejidad técnica y ambiente que influyen en el desarrollo. Fue propuesto originalmente por Gustav Karner y posteriormente refinado por muchos otros autores. Este método exige la existencia de un modelo de casos de uso, por lo que se deberá comenzar a aplicar, una vez que se tenga algún entendimiento del dominio del problema o cuando se estén realizando las labores de arquitectura y dimensionamiento del tamaño del sistema.

El método utiliza los actores y casos de uso identificados para calcular el esfuerzo que costará desarrollarlos. A los casos de uso se les asigna una complejidad basada en transacciones, que son pares de pasos acción-usuario->respuesta-sistema de los escenarios de los casos de uso. A los actores se les asigna una complejidad basada en el tipo de actor, es decir, si son interfaces con usuarios o si son interfaces con otros sistemas. También se utilizan factores de entorno y de complejidad técnica para afinar el resultado (Hernández, 2002).

Los métodos paramétricos, cumplen con ciertas características que son similares hasta un punto en el proceso de estimación.

Los métodos de estimación paramétricos, tiene como características, que utilizan fórmulas matemáticas asociadas en el que escenarios alternativos son definidos mediante la variación de los valores asumidos en un grupo de parámetros. A partir de las características que poseen los proyectos, los métodos de estimación brindan un conjunto de factores que según el método de estimación pueden tener diferentes clasificaciones. En el método Estimación UCI, los factores que se utilizan son: factor cliente y factor de complejidad técnica. El método Puntos de Función, utiliza los factores del valor del ajuste. Y el método Puntos por Casos de Uso utiliza los factores del entorno y los factores técnicos. A partir de la importancia que se les dé a estos factores en el proyecto, se calcula el ajuste, que de su resultado se obtiene el tamaño. Luego de obtener el tamaño del proyecto, los métodos de estimación emplean una fórmula para calcular el esfuerzo y el tiempo, única para cada uno de los métodos.

2.2 Metodología de desarrollo de software

El desarrollo de todo software debe estar guiado por una metodología de desarrollo. De esta depende, en gran medida, que el software tenga la calidad requerida. Existen dos grupos de metodologías: ágiles y tradicionales. No existe una metodología universal para cada tipo de proyecto. Se define una metodología según las características del equipo de desarrollo, el dominio de aplicación, tipo de contrato, complejidad y envergadura del proyecto. Debido a la necesidad de desarrollar la propuesta de solución en un breve período de tiempo, garantizando la flexibilidad necesaria en cuanto a la variación de los requisitos, no existiendo un contrato tradicional, siendo el cliente parte del equipo de desarrollo, permitiendo reducir la generación de artefactos; se hace necesario optar por un enfoque ágil de desarrollo de software en lugar de un enfoque tradicional. Por lo cual se decide utilizar la metodología XP, la cual cuenta con las siguientes características que se ajustan a las necesidades del proyecto:

- Es un proyecto a corto plazo.
- El funcionamiento del software es más importante que la documentación exhaustiva que se pueda generar.
- Se realizará una programación organizada.
- Se basa en la simplicidad, la comunicación y el reciclado del código.
- Mantiene una fuerte comunicación con el cliente.

- Genera poca documentación.

Esta característica se basada en caso de proyectos que tengan un plazo corto de entrega, lo cual constituye una ventaja.

- Existe una menor taza de errores
- Satisfacción del programador
- Solución de errores de programas
- Versiones nuevas

Implementa una forma de trabajo donde se adapta fácilmente a las circunstancias.

2.3 Requisitos funcionales

Se pretende realizar un sistema que permita integrar métodos de estimación de software para determinar el tiempo y esfuerzo en los proyectos productivos. A partir de datos generales y particulares en dependencia del método a emplear, se desea obtener la estimación y generar reportes, así como tener un histórico de los resultados de las estimaciones de los proyectos.

Después del estudio realizado sobre los métodos de estimación el autor decide que es necesario informatizar los siguientes requisitos funcionales (RF).

No.	Actividades importantes
1	Estimar mediante el método Puntos de Función
2	Determinar factor valor de ajuste
3	Adicionar función de dato
4	Eliminar función de dato
5	Modificar función de dato

6	Mostrar función de dato
7	Adicionar función transaccional
8	Eliminar función transaccional
9	Modificar función transaccional
10	Mostrar función transaccional
11	Determinar cantidad de trabajadores
12	Estimar mediante el Método de Estimación UCI
13	Determinar factor cliente
14	Determinar factor de complejidad técnica
15	Adicionar paquetes funcionales
16	Eliminar paquetes funcionales
17	Modificar paquetes funcionales
18	Mostrar paquetes funcionales
19	Determinar factor ambiente
20	Determinar factor valor agregado
21	Estimar mediante el método Puntos por Casos de Uso
22	Determinar los puntos por casos de uso sin ajustar

23	Determinar factores técnicos
24	Ajustar los puntos por casos de uso desajustados
25	Adicionar proyecto
26	Modificar proyecto
27	Mostrar proyecto
28	Mostrar resultado de la estimación
29	Exportar estimación seleccionada
30	Ver estimación
31	Adicionar datos necesarios para el tamaño
32	Eliminar datos necesarios para el tamaño
33	Modificar datos necesarios para el tamaño
34	Mostrar datos necesarios para el tamaño
35	Adicionar datos necesarios para el esfuerzo
36	Modificar datos necesarios para el esfuerzo
37	Mostrar datos necesarios para el esfuerzo
38	Adicionar datos necesarios para la duración
39	Modificar datos necesarios para la duración

40	Mostrar datos necesarios para la duración
41	Buscar proyecto
42	Registrar usuario
43	Autenticar usuario

Tabla 3: Requisitos funcionales.

2.4 Características del sistema

Las características que deben de tener los sistemas, o también conocido como requisitos no funcionales (RNF), son las propiedades o cualidades que un producto debe poseer. Debe pensarse en ellos como las características que lo hacen atractivo, usable, rápido o confiable. No son parte de la razón fundamental del producto, pero sí son necesarios para hacer funcionar el producto de la manera deseada (Pressman R., 2002).

Para que el funcionamiento del sistema sea correcto, el autor definió las siguientes características que debe poseer el sistema:

Característica del sistema(CS)	Descripción
Interfaz externa	**CS_1.** Sistema sencillo, fácil de usar y con funcionalidades explícitas. **CS_2.** En el momento que ocurra un error con los datos de entrada a la aplicación le será informado al usuario de manera detallada.
Usabilidad	**CS_3.** Sistema orientado a personas con conocimientos de la estimación de proyectos.

Rendimiento	**CS_4.** El sistema debe tener un tiempo de respuesta menor de 20 segundos ante cualquier solicitud del usuario.
Soporte	**CS_5.** Consta con la documentación necesaria para el aprendizaje sobre el uso de la aplicación.
Seguridad	**CS_6.** Establecer procedimiento para las salvas periódicas de la información en otros dispositivos. Además de la seguridad propia que brinda el framework Symfony2.
Hardware	**CS_7.** Para el cliente: requerimiento mínimo del hardware un procesador Pentium Celeron a 1GHz y 256MB de memoria RAM **CS_8.** Para el servidor: como requerimientos mínimos un procesador Pentium Dual Core a 2GHz de velocidad de procesamiento y 1GB de memoria RAM.
Software	**CS_9.** Se necesita tener instalado en su computadora, cualquier navegador Web con soporte HTML5. Se recomienda utilizar Mozilla Firefox la versión 3.5 (Mota, 2009) o superior. **CS_10.** Como servidor de datos debe estar instalado el gestor de base de datos PostgreSQL 9.3.2 o superior.

Tabla 4: Características del sistema

2.5 Selección de tecnologías

Luego del estudio de las características que debe poseer el sistema se definen los lenguajes y tecnologías que se ajustan a los requisitos que se identificaron.

2.5.1 Lenguajes programación

- **Lenguaje del lado del servidor**

PHP 5.4.12 es un acrónimo de Hypertext Preprocessor. Es un lenguaje de programación interpretado de alto nivel, especialmente pensado para desarrollos Web, el cual puede ser embebido en páginas HTML. Es usado principalmente en interpretación del lado del servidor. Tiene capacidad de conexión con la mayoría de los motores de base de datos que se utilizan en la actualidad, destaca su conectividad con PostgreSQL. Es un lenguaje completamente orientado al desarrollo de aplicaciones Web dinámicas con acceso a información almacenada en una Base de Datos (PHP, 2016).

Lo mejor de utilizar PHP es su extrema simplicidad para el principiante, pero a su vez ofrece muchas características avanzadas para los programadores profesionales (PHP, 2016).

Este lenguaje tiene características importantes como su utilidad para diferentes plataformas, de código abierto y muy popular, especialmente adecuado para desarrollo Web. Contiene un gran número de funciones previamente definidas como, por ejemplo, funciones para el trabajo con fechas, arreglos y cadenas de texto. Otra de sus ventajas es el manejo de excepciones. Es libre, por lo que se presenta como una alternativa de fácil acceso para todos (PHP, 2016).

Se selecciona en lenguaje de programación del lado del servidor PHP en su versión 5.4.12 por presentar ventajas como ser un lenguaje totalmente libre y abierto, tiene una curva de aprendizaje baja. Los entornos de desarrollo son de rápida y fácil configuración. Fácil despliegue: paquetes totalmente autoinstalables que integran PHP. Fácil acceso a bases de datos. Comunidad grande. Además, se tiene conocimiento previo del lenguaje, ya que el autor lo estudió mediante el transcurso de la carrera.

- **Lenguajes del lado del cliente**

    - **HTML5** (HyperText Markup Language) es la quinta versión del lenguaje de programación “básico” de la World Wide Web, el HTML. HTML5 provee básicamente tres características: estructura, estilo y funcionalidad. Nunca fue declarado oficialmente, pero incluso cuando algunas APIs (Interfaz de Programación de Aplicaciones) y la especificación de CSS3 (en inglés Cascading Style Sheets) por completo no son parte del mismo, HTML5 es considerado el producto de la combinación de HTML, CSS y Javascript. Estas tecnologías son altamente dependientes y actúan como una sola unidad organizada bajo la

especificación de HTML5. HTML está a cargo de la estructura, CSS presenta esa estructura y su contenido en la pantalla y Javascript hace el resto (Gauchat, 2012).

- **CSS3**, es un lenguaje que trabaja junto con HTML para proveer estilos visuales a los elementos del documento, como tamaño, color, fondo, bordes, entre otros. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas (Gauchat, 2012).

- **JavaScript** es un lenguaje de programación orientado a objetos. Se usa principalmente del lado del cliente, implementado como parte del navegador Web, permitiendo mejoras significativas en las interfaces de usuario, puesto que es muy eficiente para la realización de validaciones y el desarrollo de Web dinámicas. JavaScript permite crear contenido HTML dinámico y aplicaciones Web del lado del cliente interactivas. La sintaxis JavaScript se basa en los lenguajes de programación C, C + + y Java, lo que hace que sea familiar y fácil de aprender para los programadores experimentados. Al mismo tiempo, JavaScript es un lenguaje de programación interpretado, proporcionando un entorno de programación flexible (Flanagan, 2007).

2.5.2 Marcos de trabajo

1. **Symfony 2.8.0** es una versión de Symfony, el popular marco de trabajo para desarrollar aplicaciones PHP. Su arquitectura interna está completamente desacoplada, lo que permite reemplazar o eliminar fácilmente aquellas partes que no encajan en el proyecto. Symfony2 ha sido ideado para exprimir al límite todas las nuevas características de lenguaje de programación PHP y por eso es uno de los marcos de trabajo PHP con mejor rendimiento. Este marco de trabajo proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación Web compleja. Automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El concepto de Symfony2 es “no reinventar la rueda", por lo que reutiliza conceptos y desarrollos exitosos de terceros y los integra como bibliotecas para que puedan ser utilizados (Eguiluz, 2012).

   Algunas de sus características fundamentales se resumen a continuación:

- **Versátil** porque está basado en componentes.
- **Útil** porque soluciona los retos de la programación Web.
- **Buenas prácticas** porque toma ideas de otros marcos de trabajo y las aplica.
- **Flexible** pues permite usar cualquier archivo de configuración.
- **Rendimiento** pues exige usar como mínimo PHP 5.3, diferentes archivos de configuración que al final se ejecutan a PHP. Doctrine 2.0, es la versión seleccionada, ya que es la que viene por defecto con Symfony2.8 además del motor de plantilla que utiliza el marco de trabajo llamado Twig en su versión 1.7.

Se selecciona Symfony2 en su versión 2.8 como marco de trabajo PHP porque: es fácil de instalar y configurar en cualquier plataforma, libera a los desarrolladores de la tarea de crear funcionalidades menores y, en ocasiones, aburridas de implementar. Las aplicaciones desarrolladas con Symfony2 son compatibles con la mayoría de las plataformas, bibliotecas e infraestructuras que existen. Se adaptan a entornos de negocio en cambio permanente, requiriendo menos esfuerzo para su mantenimiento y es fácil de extender. Como se trata de un proyecto de código abierto, se pueden agregar nuevas funciones desarrolladas por programadores externos. Promueve el uso de buenas prácticas de programación y genera código fácilmente comprensible por el desarrollador. Si alguna vez se encuentran dificultades, se cuenta con la colaboración de una comunidad de cientos de miles de programadores y con la seguridad de que cualquier posible defecto será corregido en versiones posteriores. Se utiliza la versión de Symfony 2.8 porque la misma tiene soporte hasta 5 años, además este marco de trabajo facilita el trabajo, ya que los métodos de estimación a desarrollar se colocarán por *bundles*, lo cual permite que, en versiones posteriores del sistema a realizar, se puedan agregar otros métodos.

2. **ExtJS 4.1** es el marco de trabajo de JavaScript para la creación de aplicaciones Web multiplataforma. ExtJS aprovecha las funciones de HTML5 en los navegadores modernos, mientras que se mantiene la compatibilidad y funcionalidad para los navegadores antiguos. Originalmente fue construida como una extensión de la biblioteca YUI por Jack Slocum, en la actualidad puede usarse como extensión para las bibliotecas jQuery. Desde la versión 1.1 puede ejecutarse como una aplicación independiente. ExtJS cuenta con cientos de *widgets* de interfaz de

usuario de alto rendimiento que están meticulosamente diseñados para adaptarse a las necesidades de las más sencillas, así como las aplicaciones Web más complejas (Sencha, 2016).

Se selecciona ExtJS4.1 por el fácil acceso a diversidad de funciones que posee. Permite crear aplicaciones complejas utilizando componentes predefinidos, así como un manejador de plantillas de diseño (*layouts)*, gracias a esto provee una experiencia consistente sobre cualquier navegador, evitando el tedioso problema de validar que el código escrito funcione bien en cada uno (Firefox, IE, Safari, etc.). Además, la ventana flotante que provee ExtJS es excelente por la forma en la que funciona. Al moverla o redimensionarla solo se dibujan los bordes haciendo que el movimiento sea fluido. Usar un motor de *render* como ExtJS permite tener además estos beneficios:

- ✓ Existe un balance entre Cliente – Servidor. La carga de procesamiento se distribuye, permitiendo que el servidor, al tener menor carga, pueda manejar más clientes al mismo tiempo.
- ✓ Comunicación asíncrona. En este tipo de aplicación el motor de *render* puede comunicarse con el servidor sin necesidad de estar sujeta a un clic o una acción del usuario, dándole la libertad de cargar información sin que el cliente se dé cuenta.

- ✓ Eficiencia de la red. El tráfico de red puede disminuir al permitir que la aplicación elija qué información desea transmitir al servidor y viceversa, sin embargo, la aplicación que haga uso de la pre-carga de datos puede que revierta este beneficio por el incremento del tráfico (Sencha, 2016).

3. **Twig** es un motor y lenguaje de plantillas para PHP, rápido, seguro y flexible que permite separar el código PHP del HTML permitiendo una amplia gama de posibilidades y por sobre todo, un extraordinario orden para el código del proyecto. La sintaxis de Twig se ha diseñado para que las plantillas sean concisas, muy fáciles de leer y de escribir. Twig ha sido creado por los desarrolladores del framework Symfony (TWIG, 2012).
4. **Twitter Bootstrap 3.3.6** es un marco de trabajo de código abierto para la creación, por lo general, de diseños (layouts) para aplicaciones Web de una forma rápida, sencilla y limpia. Simplifica el

proceso de creación de diseños Web, utilizando algunas de las técnicas más modernas de los navegadores para ofrecer estilos de tipografías, formularios, botones, tablas, entre otros. Ofrece una serie de plantillas CSS y ficheros JavaScript que permiten integrar el marco de trabajo de forma sencilla y potente en proyectos Webs. Entre sus características destaca unas interfaces de gran usabilidad, integración de la biblioteca jQuery para diferentes efectos. Bootstrap utiliza propiedades incluidas en HTML5. Es un marco de trabajo ligero que se integra de forma limpia en el proyecto actual (Bootstrap, 2012).

2.5.3 Entorno de desarrollo integrado

PhpStorm9.0.2 es un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) PHP ligero e inteligente centrado en la productividad del desarrollador que entiende profundamente su código, proporciona la finalización de código inteligente, una navegación rápida y la comprobación de errores en la marcha. Siempre está lista para ayudar a dar forma a su código, correr las pruebas de unidad o proporcione depuración visual (CUBRID, 2012).

PhpStorm permite crear aplicaciones Web con PHP 5 y además existen bibliotecas de Symfony 2 que se pueden adicionar a dicho IDE para facilitar la tarea de los programadores mediante autocompletamiento de código.

Se elige este IDE por tener la característica de ser código abierto, además existen bibliotecas de Symfony 2 que se pueden adicionar a dicho IDE para facilitar la tarea de los programadores mediante autocompletamiento de código.

2.5.4 Sistema gestor de base de datos

PostgreSQL 9.3.2 es un sistema gestor de bases de datos objeto-relacional de código abierto y multiplataforma, compatible con una gran parte del estándar SQL y ofrece muchas características modernas como, consultas complejas, las claves externas, disparadores, vistas, integridad transaccional y control de concurrencia multiversión. PostgreSQL puede ser extendido por el usuario en muchas maneras, por ejemplo, mediante la adición de nuevos tipos de datos, funciones, operadores, funciones de agregado y otros. Debido a su licencia liberal, PostgreSQL puede ser utilizado, modificado y distribuido por

cualquiera de forma gratuita para cualquier propósito, ya sea privado, comercial o académico. Funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez al sistema (PostgreSQL, 2013).

Es seleccionado PostgresSLQ 9.3.2 como SGBD y PgAdmin III como interfaz gráfica. A continuación, se muestran algunas de las razones para su elección:

1. Código fuente libre y de alta calidad.
2. Requerimientos de administración y mantenimiento relativamente bajos con respecto el resto de bases de datos comerciales.
3. Fiabilidad y estabilidad legendarias.
4. Rendimiento excelente.
5. Diseñada para entornos con altos volúmenes de tráfico/transacciones.
6. Extensible
7. Multiplataforma.

Herramientas gráficas y de línea de comandos para diseñar bases de datos y administrarlas (PostgreSQL, 2013).

PgAdmin III es una aplicación gráfica para administrar el gestor de bases de datos PostgreSQL, siendo la más completa y popular con licencia Open Source. Está escrita en C++ usando la librería gráfica multiplataforma *wxWidgets*[3], lo que permite que se pueda usar en Linux, Solaris, Mac OS y Windows. PgAdmin III está diseñado para responder a las necesidades de todos los usuarios, desde escribir consultas SQL simples hasta desarrollar bases de datos complejas. La interfaz gráfica soporta todas las características de PostgreSQL y facilita enormemente la administración. La aplicación también incluye un editor SQL con resaltado de sintaxis, un editor de código de la parte del servidor, un agente para lanzar scripts programados y mucho más (pgAdmin, 2009).

[3] wxWidgets es una biblioteca de C + + que permite a los desarrolladores crear aplicaciones para Windows, OS X, Linux y UNIX en arquitecturas de 32 bits y 64 bits, así como varias plataformas móviles como Windows Mobile, iPhone SDK y GTK.

2.5.5 Servidor Web

Apache 2.4 es un servidor Web HTTP de código abierto para sistemas operativos modernos, incluyendo Unix (BSD, GNU/Linux, etc.) y Microsoft Windows. Es de fácil configuración y los usuarios pueden utilizar las funcionalidades y servicios que ofrece debido a su estructura en módulos, otras de las ventajas que tiene es que es muy popular por lo que es fácil de conseguir ayuda y soporte acerca del mismo. La arquitectura que utiliza es cliente/servidor. El protocolo que utiliza para la transferencia de datos es HTTP, proporciona contenidos al cliente Web o navegador como páginas estáticas y dinámicas. Además, es flexible, multiplataforma, gratuito, puede interpretar varios lenguajes como PHP el cual es utilizado en la aplicación (Apache, 2012).

2.6 Arquitectura del sistema

Para el desarrollo del sistema se propone la arquitectura en capas, la cual brinda un cierto aislamiento entre las distintas capas de la aplicación, de forma tal que cualquier cambio o modificación que ocurra en una de ellas no afecte al resto. Permite el desarrollo en paralelo, o sea que se puede ir trabajando en diferentes capas de forma separada, posibilitando agilizar el desarrollo. Las capas que conforman la arquitectura son: la capa de presentación, la capa de negocio, la capa de acceso a datos y la capa de datos. También se utiliza el patrón arquitectónico Modelo Vista Controlador (MVC), este patrón viene integrado al marco de trabajo con que se desarrolla la aplicación. En la figura 3 se muestra la distribución de los elementos que componen cada una de las capas.

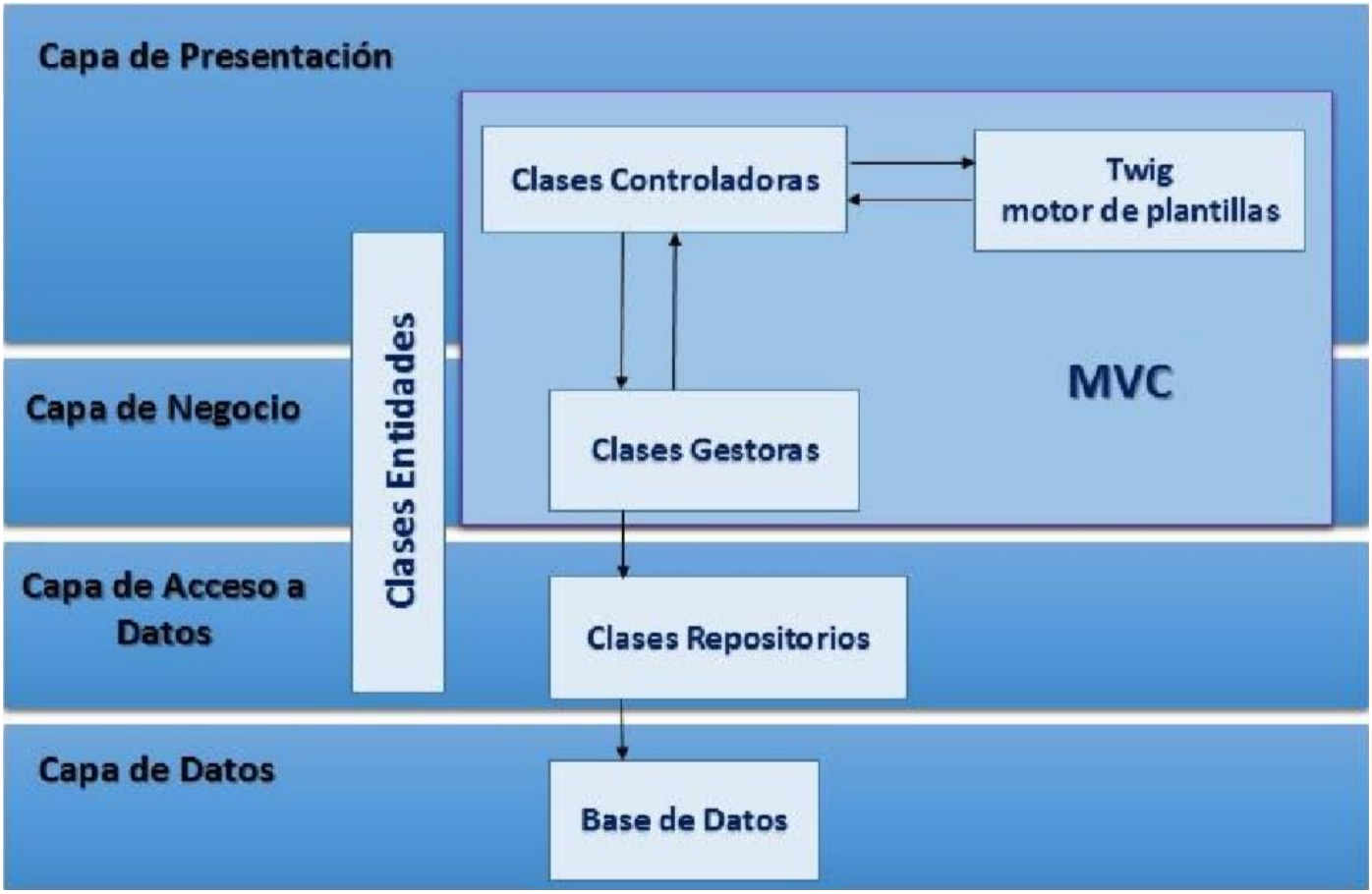


Figura 3: Vista general de la arquitectura del sistema.

- ✓ **Capa de Presentación**

Esta capa es la que se encarga de mostrar la información al usuario, está dividida en dos subcapas: cliente y servidor. En la subcapa cliente se ubica todo lo relacionado con las vistas de la aplicación, mostrando a través de un navegador Web los datos que se procesarán en el servidor. Para esto se utilizan los lenguajes del lado del cliente HTML5, CSS3 y JavaScript, empleando las facilidades que brindan los marcos de trabajo Twitter Bootstrap y ExtJS. En la subcapa servidor se encuentra todo lo relacionado con la lógica de control y la construcción de las páginas.

- ✓ **Capa de Negocio**

En esta capa se encuentran situados los componentes de negocio que en el sistema lo constituyen las clases gestoras, las cuales son las encargadas de manipular toda la lógica del negocio. Esta recibe una petición del nivel superior y brinda la posibilidad de invocar a las clases repositorio para gestionar datos procesados.

- ✓ **Capa de Acceso a datos**

La función fundamental de esta capa es servir de conector entre la capa de negocio y el gestor de base de datos, en este caso PostgreSQL. En esta capa está presente el marco de trabajo de persistencia de datos Doctrine, el cual permite mediante su lenguaje propio de consultas DQL la independencia de la aplicación respecto al gestor de base de datos. Incluye los repositorios que son los encargados de realizar las consultas para obtener los datos de las peticiones de la capa de negocio. Son parte también de esta capa las entidades del dominio, las cuales pueden ser accedidas además por la capa de presentación y la capa de negocio.

- ✓ **Capa de datos**

Es la capa donde se almacena los datos, en ella está ubicado el servidor de base de datos PostgreSQL que contiene un conjunto de esquemas, tablas, secuencias, procedimientos almacenados y otros, que permiten persistir y manejar el almacenamiento de la información con la que trabaja el sistema.

2.7 Planificación

El ciclo de vida del proyecto según la metodología seleccionada comienza en esta fase. En ella el usuario define las historias de usuario (HU), asignándole a cada HU una prioridad y los programadores se encargan de realizar las estimaciones correspondientes. Además, el equipo de desarrollo se familiariza con cada una de las herramientas y tecnologías que utiliza en el desarrollo del sistema y también explora diferentes posibilidades de conformar la arquitectura.

2.7.1 Historias de usuario

Las funcionalidades del sistema son modelados en XP como HU. Estas deben ser redactadas por el cliente, aunque los desarrolladores pueden brindar también su ayuda en su confección. El contenido que ellas abarcan debe ser concreto y sencillo. A continuación, se muestran las HU necesarias para el desarrollo del sistema.

No	Nombre de la Historia de usuario	Prioridad	Estimación
1	Estimar mediante el método Puntos de Función	Muy alta	3 semanas
2	Estimar mediante el método Puntos por Casos de Uso	Muy alta	3 semanas
3	Estimar mediante el Método de Estimación UCI	Muy alta	3 semanas
4	Gestionar proyecto	Alta	1 semana
5	Generar reportes	Media	0.5 semana
6	Exportar a extensión .pdf	Media	0.5 semana
7	Realizar búsquedas	Baja	0.5 semana
8	Registrar usuario	Media	0.5 semana
9	Autenticar usuario	Media	0.5 semana

Tabla 5: Historias de usuario

La HU que se muestra a continuación es la de “Estimar mediante el método Puntos por Casos de Uso” con una prioridad muy alta para el cliente, en lo adelante los elementos de los artefactos que se muestren como ejemplos tienen como punto de partida esta HU. El resto pueden ser consultadas en los anexos del documento.

Historia de Usuario	
Número: 02	**Usuario:** aochoa
Nombre historia: Estimar mediante el método Puntos por Casos de Uso	
Prioridad en negocio: Muy Alta	**Riesgo en desarrollo:** Alto
Puntos estimados: 3 semana	**Iteración asignada:** Segunda Iteración
Programador responsable: Alian Villa Ochoa	
Descripción: La funcionalidad estimar mediante el método puntos por casos de uso debe permitir al usuario ejecutar una estimación mediante el método Puntos por casos de uso. Inicialmente se selecciona el método "Puntos Por Casos de Uso" y se muestra una tabla donde se permite seleccionar el proyecto a estimar y posteriormente se muestran las operaciones que se pueden realizar utilizando el método seleccionado, como son calcular el esfuerzo y duración del proyecto. **Esfuerzo** Cuando se selecciona en el botón "Esfuerzo", se muestra una ventana con un formulario con los campos necesarios para calcular el tamaño del proyecto seleccionado. **Duración** Cuando se selecciona en el botón "Duración", se muestra una ventana con un formulario con los campos necesarios para calcular el tamaño del proyecto seleccionado.	
Observaciones: 1. Se validan los datos insertados mostrando un mensaje de error o si los datos son incluidos incorrectamente el sistema mostrará un mensaje de error.	

Tabla 6: Historia de usuario: Gestionar proyecto

2.7.2 Iteraciones

Las iteraciones tienen como objetivo que el cliente decida cuáles HU implementar según la prioridad. Al estar definidas las HU y la estimación por puntos de la duración de cada una de ellas se decide desarrollar el sistema en seis iteraciones. A continuación, se expone el plan de las iteraciones:

Iteración No.	Descripción
Iteración 1	Se desarrolla la HU número uno, la cual permitirá conocer el esfuerzo y duración de un proyecto haciendo uso del método Puntos de Función. Además, se corrigen los errores o las no conformidades en la HU implementadas en esta iteración.
Iteración 2	Se desarrolla la HU dos, la cual permitirá conocer el esfuerzo y duración de un proyecto de software haciendo uso del método Puntos por Casos de Uso. Además, se corrigen los errores o las no conformidades en la HU implementadas en esta iteración.
Iteración 3	Se desarrolla la HU tres la cual permite saber el esfuerzo y duración de un proyecto de software a través del Método de Estimación UCI. Además, se corrigen los errores o las no conformidades en la HU implementadas en esta iteración.
Iteración 4	Se desarrollan las HU número cuatro, cinco y seis las cuales permitirán gestionar un proyecto de software, generar reportes y exportar a extensión .pdf las estimaciones. Además, se corrigen los errores o las no conformidades en la HU implementadas en esta iteración.
Iteración 5	Se desarrollan las HU número siete, ocho y nueve, donde se muestran los

	resultados de las estimaciones, búsquedas, se podrán registrar y autenticar los usuarios en el sistema. Además, se corrigen los errores o las no conformidades en la HU implementadas en esta iteración.

Tabla 7: Plan de iteraciones

Para dar cumplimiento al Plan de iteraciones se procede a la realización del Plan de duración de las iteraciones, el cual muestra el tiempo total de las iteraciones a partir de la estimación realizada por cada HU que se implementa en esta. A continuación, se muestra el plan de duración de las iteraciones:

Iteraciones No	Historia de usuario	Estimación de la historia	Tiempo total de las iteraciones
Iteración 1	Estimar mediante el método Puntos de Función	3 semanas	12.5 semanas
Iteración 2	Estimar mediante el método Puntos por Casos de Uso	3 semanas	
Iteración 3	Estimar mediante el Método de Estimación UCI	3 semanas	
Iteración 4	Gestionar proyecto Generar reportes Exportar a extensiones .pdf	2 semanas	
Iteración 5	Realizar búsquedas	1.5 semanas	

	Registrar usuario Autenticar usuario		

Tabla 8: Plan de duración de las iteraciones

2.8 Fase de diseño

En esta etapa se procede al diseño del sistema a través de las tarjetas CRC, donde se ilustran las clases y la información necesaria a almacenar en cada una de ellas. El objetivo es obtener un diseño elegante y fácil de comprender por parte de los programadores.

2.8.1 Tarjetas CRC

Se definieron 30 tarjetas CRC para conseguir un diseño fácil de entender e implementar. Estas representan el nombre de la clase con sus responsabilidades u objetivos que debe cumplir y las clases que colaboran con cada responsabilidad. A continuación, se muestra un ejemplo de la tarjeta CRC "ProyectoController", el resto de las tarjetas pueden ser consultadas en el artefacto "Plantilla modelo de diseño".

Tarjeta CRC	
Clase: EstimacionPuntosCasosUso	
Responsabilidades	**Colaboraciones**
Obtener tamaño Calcular tamaño Obtener esfuerzo Calcular esfuerzo Obtener total de hombres Calcular total de hombres Obtener costo Calcular costo	Estimación Factor Entorno Factor Técnico Tiempo Pactado Cliente Roles Estimados

Tabla 9: Tarjeta CRC: ProyectoController

La clase “EstimacionPuntosCasosUso”, es la encargada de obtener los datos y realizar los cálculos necesarios para obtener el esfuerzo y la duración, en conjunto con sus clases colaboradoras, que son las encargadas de almacenar los datos que el usuario insertará en el sistema.

2.9 Conclusiones del capítulo

- Después de realizado el análisis del sistema en términos de solución quedó definida la estructura del sistema, proporcionando una comprensión detallada de las HU, así como de la arquitectura.
- La propuesta de arquitectura de la aplicación se sustenta en un conjunto de componentes reutilizables que tiene como base el patrón arquitectónico MVC, lo que conforma un sistema robusto y flexible a cambios.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBAS DEL SISTEMA

En el presente capítulo se planifican y ejecutan las tareas de programación y se muestran los estándares del código utilizados en la fase de implementación de metodología de desarrollo seleccionada. Además, se describe la validación mediante las pruebas de software, con el objetivo de lograr la aceptación del cliente.

3.1 Implementación

En esta fase se implementan las historias de usuario, las que se fragmentan en tareas de programación para ayudar a organizar la implementación del sistema. Estas tareas son descritas y usadas por los programadores como guía para el desarrollo del sistema.

3.1.1 Tareas de ingeniería

Las tareas de ingeniería describen la funcionalidad específica de la HU a implementar, son dinámicas y flexibles, pueden ser cambiadas por otras más generales o más específicas según las necesidades existentes. Cada HU puede contener una o más tareas de programación en dependencia de la complejidad de la funcionalidad a desarrollar. Las tareas podrán ser comprobadas a través de los casos de prueba.

A continuación, se relacionan las 29 tareas a desarrollar para la implementación del sistema:

HU	Tareas de ingeniería
Estimar mediante el método Puntos de Función	• Gestionar función de dato • Gestionar función transaccional • Determinar los puntos de función no ajustados • Determinar el valor del factor de ajuste • Determinar los puntos función ajustados • Determinar esfuerzo del proyecto

	• Determinar el tiempo empleado en el proyecto • Determinar el tamaño
Estimar mediante el método Puntos Por Casos de Uso	• Determinar los puntos por casos de uso sin ajustar • Determinar factores técnicos • Determinar los factores del entorno • Ajustar los puntos por casos de uso desajustados
Estimar mediante el Método de Estimación UCI	• Determinar factor cliente • Determinar factor de valor agregado • Determinar factor ambiente • Determinar factor de complejidad técnica • Determinar el tamaño • Determinar el esfuerzo de desarrollo • Determinar el esfuerzo total • Determinar el tiempo
Gestionar proyecto	• Adicionar proyecto • Modificar proyecto • Mostrar proyectos
Generar reportes	• Mostrar reportes a través de tablas con las estimaciones realizadas a proyectos
Exportar a extensión .pdf	• Exportar proyecto consultado con la estimación seleccionada • Exportar proyecto consultado con las estimaciones que

	se le hayan realizado
Realizar búsquedas	• Realizar la búsqueda de proyectos
Registrar Usuario	• Registrar Usuario
Autenticar Usuario	• Autenticar Usuario

Tabla 10: Tareas de ingeniería

A continuación, se muestra un ejemplo de la descripción de cuatro tareas de programación correspondientes a la historia de usuario “Estimar mediante el método Puntos Por Casos de Uso”, el resto pueden ser consultadas en el artefacto “Plantilla Tarea de ingeniería”.

Tarea	
Número Tarea: 1	**Número Historia de Usuario:** 2
Nombre Tarea: Determinar los puntos por casos de uso sin ajustar	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio:7/3/2016	**Fecha Fin:** 11/3/2016
Programador Responsable: Alian Villa Ochoa	
Descripción: Los puntos por casos de uso sin ajustar se calculan sumando la dificultad de las interacciones de los actores y la complejidad de los casos de uso, sumando el total de los pesos de los actores y el total de los pesos para los casos de uso.	

Tabla11: Tarea 1 de la historia de usuario "Estimar mediante el método Puntos Por Casos de Uso"

Tarea

Número Tarea: 2	**Número Historia de Usuario:** 2
Nombre Tarea: Determinar factores técnicos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 0.5
Fecha Inicio: 14/3/2016	**Fecha Fin:** 17/3/2016
Programador Responsable: Alian Villa Ochoa	
Descripción: Los factores técnicos, a cada factor definido se le asigna un valor entre 0 y 5, dependiendo de su influencia en el proyecto. En este sentido, se asigna un valor 0 si significa que el factor es irrelevante para el proyecto, un valor 3 si es promedio y un valor 5, si significa que el factor es esencial. Luego, se realiza una multiplicación entre la influencia del factor y su peso asociado y se calculan los resultados de todos los factores.	

Tabla 12: Tarea 2 de la historia de usuario "Estimar mediante el método Puntos Por Casos de Uso"

Tarea	
Número Tarea: 3	**Número Historia de Usuario:** 2
Nombre Tarea: Determinar los factores del entorno	
Tipo de Tarea: Desarrollo	**Puntos Estimados:**0.5
Fecha Inicio: 17/3/2016	**Fecha Fin:** 19/3/2016
Programador Responsable: Alian Villa Ochoa	
Descripción: Para calcular los factores de entorno, a cada factor de entorno definido se le asigna un valor entre 0 y 5 dependiendo de su influencia en el proyecto. Asignar un valor 0 significa que el factor es irrelevante para el proyecto, un valor 3 es promedio y un valor 5 significa que el factor es	

esencial. Una vez que todos los factores de entorno tienen asignado el valor de la influencia, se procede al cálculo de los resultados de cada factor, es decir, se realiza una multiplicación entre la influencia del factor y su peso asociado.

Tabla 13: Tarea 3 de la historia de usuario "Estimar mediante el método Puntos Por Casos de Uso"

Tarea	
Número Tarea: 4	**Número Historia de Usuario:** 2
Nombre Tarea: Ajustar los puntos por casos de uso desajustados	
Tipo de Tarea: Desarrollo	**Puntos Estimados:**0.5
Fecha Inicio: 20/3/2016	**Fecha Fin:** 25/3/2016
Programador Responsable: Alian Villa Ochoa	
Descripción: Para obtener los puntos por casos de uso ajustados se utilizan los datos obtenidos en las tareas anteriores, multiplicando los puntos por casos de uso no ajustados, los factores de ajuste del entorno y técnicos.	

Tabla 13: Tarea 4 de la historia de usuario "Estimar mediante el método Puntos Por Casos de Uso"

3.1.2 Estándares de código

Un estándar de código se basa en la estructura y apariencia física de un programa con el fin de facilitar la lectura, comprensión, mantenimiento del código, reutilización a lo largo del proceso de desarrollo de un software y no en la lógica del programa. Un estándar de programación no solo busca definir la nomenclatura de las variables, objetos, métodos y funciones, sino que también tiene que ver con el orden y legibilidad del código escrito. Partiendo de lo dicho anteriormente, se definen 3 partes principales dentro de un estándar de programación:

- **Nomenclatura de las clases**

Los nombres de las clases siempre comienzan con la primera letra en mayúscula y el resto en minúscula, en caso de que sea un nombre compuesto se emplea notación UpperCamelCase, la cual define que la primera letra de cada una de las palabras es mayúscula y con solo leerlo se reconoce el propósito de la misma.

Ejemplo: EstimacionPrimaria. En este caso el nombre de la clase está compuesto por dos palabras iniciadas cada una con letra mayúscula.

- ✓ **Nomenclatura según el tipo de clases**

Clases Controladoras: Las clases que se encuentran dentro de las carpetas **Controller** después del nombre de la clase llevan la palabra: “Controller”.

Ejemplo: ProyectoController.

Entity (Entidades): Las clases que se encuentran dentro de la carpeta **Entity** el nombre que reciben es el de la tabla de la base de datos, pero siguiendo la nomenclatura de UpperCamelCase.

Ejemplo: EstimacionPostArquitectura.

- **Nomenclatura de las funcionalidades y atributos**

El nombre a emplear para las funciones y los atributos se escriben con la inicial del identificador en minúscula, en caso de que sea un nombre compuesto se empleará notación CamelCase.

Ejemplo de función: listarAction().

Ejemplo de atributo: $fechaInicio. El nombre del atributo está compuesto por el signo $ y dos palabras, la primera en minúsculas y la segunda iniciando con letra mayúscula.

- **Nomenclatura de los comentarios**

PHP puede tener dos tipos de comentarios: comentarios de implementación y comentarios de documentación. Los que se usan durante toda la implementación son los de implementación delimitados por /*...*/, y //.

3.2 Pruebas

La (IEEE, 2015) define las pruebas de software como una actividad en la que un sistema o un componente es ejecutado bajo condiciones especificadas. El objetivo es diseñar una serie de casos de pruebas que tengan una alta probabilidad de encontrar errores. Para ello se aplican las técnicas de pruebas del software, las cuales facilitan una guía sistemática para diseñar pruebas que comprueben la lógica interna de los componentes de software y verifiquen los dominios de entrada y salida del programa para descubrir errores en la funcionalidad, el comportamiento y rendimiento.

Las pruebas tienen como objetivo valorar y mejorar la calidad de los productos del trabajo generado durante el desarrollo y modificación del software. Según (Pressman, 2000) verificación *es el conjunto de actividades que aseguran que el software implemente correctamente una función específica, y la* validación *es un conjunto diferente de actividades que aseguran que el software construido corresponde y satisface los requisitos del cliente.* Se tuvieron en cuenta los siguientes niveles de pruebas:

- ✓ Nivel de Unidad
- ✓ Nivel de Aceptación

3.2.1 Nivel de Unidad

Las pruebas a nivel de unidad que se realizaron, fueron a través de los métodos de caja blanca y caja negra con el uso de las técnicas de camino básico y partición de equivalencia con valores límites respectivamente.

3.2.1.1 Método de caja blanca

La aplicación del método de caja blanca, a través de la técnica camino básico permite comprobar el funcionamiento de los códigos que se vayan implementando. El uso de estas pruebas es adecuado para

observar el correcto funcionamiento de un módulo de códigos. Su objetivo es implementar casos de prueba para las funciones principales de forma que cada caso sea independiente del resto.

Para la realización de esta prueba se escogieron los métodos PFSA(), Tamanho(), TiempoRealDesarrollo() y Esfuerzo().

A continuación se expone el procedimiento de la prueba aplicado al método PFSA(), traducido a *Puntos de Función Sin Ajustar* de la clase Componente, teniendo como objetivo calcular los puntos de función sin ajustar de una consulta externa perteneciente a un componente de función transaccional, uno de los siete pasos que se utilizan para implementar el método de estimación Puntos de Función. Ver Figura 4.

```
public function PFSA()
{
    $pfsa = 0;
    if ($this->Complejidad() == "Baja") {
        $pfsa = 3;
    } else {
        if ($this->Complejidad() == "Media") {
            $pfsa = 4;
        } else {
            if ($this->Complejidad() == "Alta") {
                $pfsa = 6;
            }
        }
    }
    return $pfsa;
}
```

Figura 4: Método PFSA()

Inicialmente se construyó el grafo de flujo correspondiente al método. Ver Figura 5.

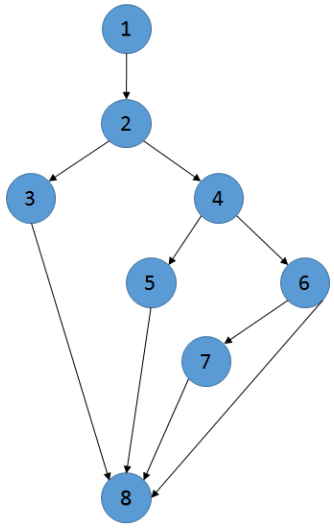


Figura 5: Grafo de flujo del método PFSA()

Luego se determina la complejidad ciclomática V (G) del grafo resultante G, para esto se hace uso de las siguientes fórmulas:

El número de regiones del grafo de flujo coincide con la complejidad ciclomática:

V (G) = A – N + 2

Donde A es el número de aristas del grafo de flujo y N es el número de nodos del mismo:

V (G) = P + 1

Donde P es el número de nodos predicados contenidos en el grafo de flujo G.

Del grafo obtenido se tiene que:

A = 10	V (G) = A - N + 2	P=3
N = 8	V (G) = 10 - 8 +2	V (G) = P+1
	V (G) = **4**	V (G) = 3+1
		V (G) = **4**

El número de caminos independientes de la estructura del programa es igual a 4 por lo que se obtiene el conjunto de caminos básicos:

Camino **1**: 1-2-3-8

Camino **2**: 1-2-4-5-8

Camino **3**: 1-2-4-6-7-8

Camino **4**: 1-2-4-6-8

Luego se define 1 caso de prueba con varios escenarios a partir de cada camino al código del método.

Camino#1: Condición de ejecución: para esta prueba es necesario que el método Complejidad() haya devuelto como tipo de complejidad del proyecto baja.
Resultados esperados: teniendo en cuenta la condición de ejecución se espera que recorra el camino#1 y devuelva 3 en la variable $pfsa.
El resultado obtenido fue correcto.

Camino #2: Condición de ejecución: para esta prueba es necesario que el método Complejidad() haya devuelto como tipo de complejidad del proyecto media.
Resultados esperados: Teniendo en cuenta la condición de ejecución se espera que recorra el camino #2 y devuelva 4 en la variable $pfsa.
El resultado obtenido fue correcto.

Camino #3: Condición de ejecución: para esta prueba es necesario que el método Complejidad() haya devuelto como tipo de complejidad del proyecto alta.
Resultados esperados: Teniendo en cuenta la condición de ejecución se espera que recorra el camino #3 y retorne 6 en la variable $pfsa.
El resultado obtenido fue correcto.

Camino #4: Condición de ejecución: para esta prueba es necesario que la variable $pfsa esté inicializada.
Resultados esperados: Teniendo en cuenta la condición de ejecución se espera que recorra el camino #4 y retorne 0 la variable $pfsa.
El resultado obtenido fue correcto.

3.2.1.2 Método de caja negra

Para la realización de las pruebas de caja negra se empleó la técnica partición de equivalencia que garantizó efectividad al examinar los valores válidos e inválidos de las entradas existentes en el software. A continuación se muestra un caso de prueba de uno de los requisitos que componen la HU “Gestionar proyecto”.

Escenario	Descripción	Descripción	Respuesta del sistema	Flujo central
EC. 1.1 Datos correctos	Permite adicionar un proyecto con sus datos correspondientes.	V		Seleccione el botón “Adicionar proyecto” en el menú superior.
		CEGEL	El sistema debe adicionar el proyecto en el panel central.	
EC. 1.2 Datos incorrectos	Adicionar un proyecto con los datos correctos.	I	El sistema no permite adicionar el proyecto porque existen campos incorrectos	
		asd123@2324.**		
EC. 1.3 Datos incompletos	Adicionar un proyecto con los datos correctos.	I	El sistema muestra un mensaje de error porque existen campos obligatorios vacíos.	

Descripción de las variables.

No	Nombre del campo	Clasificación	Valor nulo	Descripción
1	Descripción	Campo de texto	No	El campo este es obligatorio y está representado en el sistema poniendo el

				campo en color rojo y mostrando un mensaje de admite solo mayúsculas y números.

Tabla 14: Caso de prueba “Adicionar proyecto”.

Luego de aplicar las pruebas de caja negra, realizadas por el grupo de calidad del Centro de Gobierno Electrónico (CEGEL), se obtuvieron como resultado, 3 iteraciones. Estos resultados se ilustran en la figura 6.

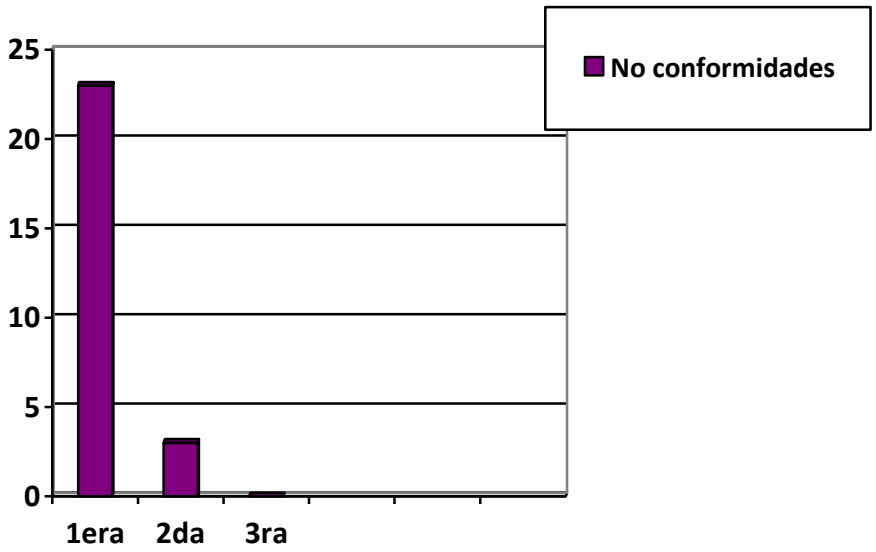


Figura 6: Resultados de las pruebas de caja negra

Las no conformidades encontradas en la primera iteración estaban relacionadas con la validación de los datos que se introducen en el sistema. En la segunda iteración de detectaron nuevas no conformidades relacionadas con el lenguaje de los mensajes que mostraba el sistema. En la tercera iteración realizada, se pudo comprobar que los errores identificados en la segunda iteración fueron corregidos y que el sistema desarrollado, funciona correctamente.

Después de concluidas las pruebas, el grupo de calidad de CEGEL, liberó el Sistema *MultiEstima,* entregándose al equipo de desarrollo el Acta de Liberación Interna de Productos de Software en el que consta que la aplicación está apta para ser utilizada.

3.2.2 Nivel de Aceptación

Según el plan de iteraciones explicado en el Capítulo 2 al finalizar una iteración se realiza una prueba de aceptación por el cliente a una o varias HU implementadas según la iteración. Para evaluar la calidad del sistema se planificaron cinco iteraciones de pruebas de aceptación, en las cuales se validó su funcionamiento.

A continuación, se muestran los casos de prueba de la historia de usuario “Estimar mediante el método Puntos Por Casos de Uso”, el resto se puede encontrar en el artefacto “Plantilla caso de prueba de aceptación”.

Caso de Prueba de Aceptación	
Código Caso de Prueba: HU#2_p1	**HU:** Estimar mediante el método Puntos Por Casos de Uso.
Descripción de la Prueba: Prueba para verificar el resultado del esfuerzo mediante el método de estimación Puntos por Casos de Uso.	
Condiciones de Ejecución: Se deben llenar los campos del formulario para poder realizar los cálculos.	
Entrada / Pasos de ejecución: El usuario selecciona la opción de esfuerzo.	
Resultado Esperado: El sistema calcula el esfuerzo y los muestra si se llenaron los campos correctamente si no debe mostrar un mensaje de error**.**	
Evaluación de la Prueba: Satisfactoria.	

Tabla 15: Caso de prueba de aceptación de la HU "Estimar mediante el método Puntos Por Casos de Uso"

Caso de Prueba de Aceptación

Código Caso de Prueba: HU#2_p2	**HU:** Estimar mediante el método Puntos Por Casos de Uso.
Descripción de la Prueba: Prueba para verificar la duración del proyecto.	
Condiciones de Ejecución: Debe llenar los campos del formulario.	
Entrada / Pasos de ejecución: El usuario selecciona la opción Duración y rellena el formulario.	
Resultado Esperado: El sistema calcula la duración si se llenaron los campos correctamente si no debe mostrar un mensaje de error**.**	
Evaluación de la Prueba: Satisfactoria.	

Tabla 16: Caso de prueba de aceptación de la HU "Estimar mediante el método Puntos Por Casos de Uso"

Al concluir los casos de prueba a la HU “Estimar mediante el método Puntos Por Casos de Uso” no se encontraron no conformidades por lo que la evaluación de la prueba fue satisfactoria.

El plan de pruebas se desarrolló conforme explica el cronograma de iteraciones (ver Tabla 7: Plan de iteraciones), donde en la primera, segunda y tercera iteraciones se liberaron las HU número uno, dos y tres respectivamente y cada una de ellas se convirtió en una prueba de aceptación con uno o varios casos de prueba. Además se corrigieron las no conformidades encontradas por el cliente a través de las pruebas de aceptación. De igual manera en la iteración cuatro se desarrollaron las HU cuatro, cinco y seis, donde sus no conformidades fueron resueltas al realizarles las pruebas de aceptación por parte del cliente. Así hasta finalizar la iteración cinco. Ver Figura 7.

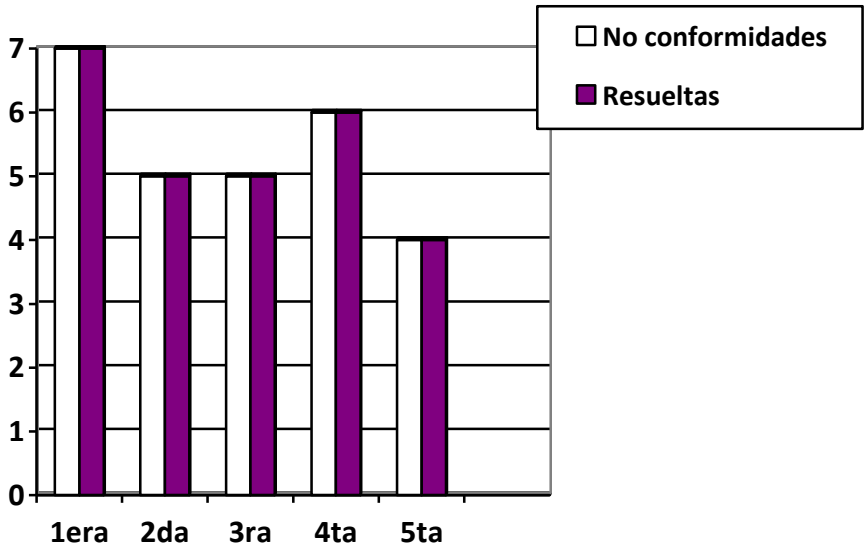


Figura 7: Resultados de las pruebas de aceptación

Las no conformidades encontradas en las diferentes iteraciones se clasifican en ortografía, validación, funcional e interfaz de usuario como se muestra en la tabla 16.

No conformidades	Ortografía	Validación	Funcional	Interfaz de usuario
1era Iteración	2	3	1	1
2da Iteración	1	3	0	1
3era Iteración	1	2	1	0
4ta Iteración	2	1	2	1
5ta Iteración	0	2	2	0

Tabla 17: Clasificación de no conformidades.

De manera general, estas cinco iteraciones realizadas en el sistema permitieron la obtención de un total de 24 no conformidades, a las cuales se les dio el tratamiento requerido para el logro de un 100% de pruebas satisfactorias.

3.3 Resultado obtenidos

Los resultados de la investigación realizada, y luego de haber elegido una metodología de desarrollo de software, lenguajes de programación, marcos de trabajo, herramientas y tecnologías, se obtuvo el sistema *MultiEstima* (Véase en la Figura 8), el cual integra múltiples métodos de estimación paramétricos, como son Puntos por Casos de Uso, Puntos de Función, y el método Estimación UCI con sus versiones de estimación primaria y post-arquitectura.

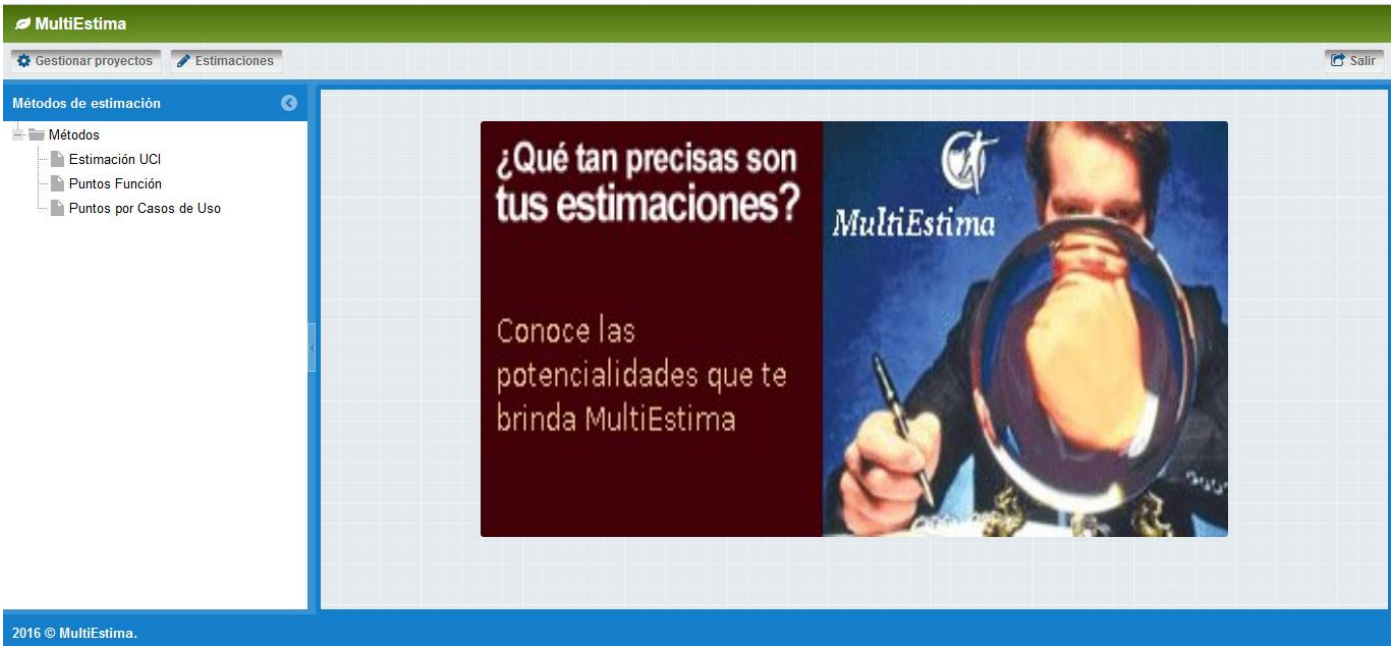


Figura 8: Sistema para la integración de métodos de estimación: *MultiEstima*

El sistema *MultiEstima*, permite al usuario adicionar o modificar el proyecto que desea estimar. Brinda la opción de al seleccionar un método de estimación, muestra un listado con los proyectos que desee estimar, al seleccionar un proyecto, el sistema muestra una interfaz donde permite al usuario, realizar operaciones sobre el proyecto seleccionado a partir del método de estimación. Luego de realizar las operaciones correspondientes al método seleccionado, el sistema muestra los resultados (Véase en la Figura 9) y se guarda la información en la base de datos para cuando el usuario desee, pueda consultar los resultados de los métodos.

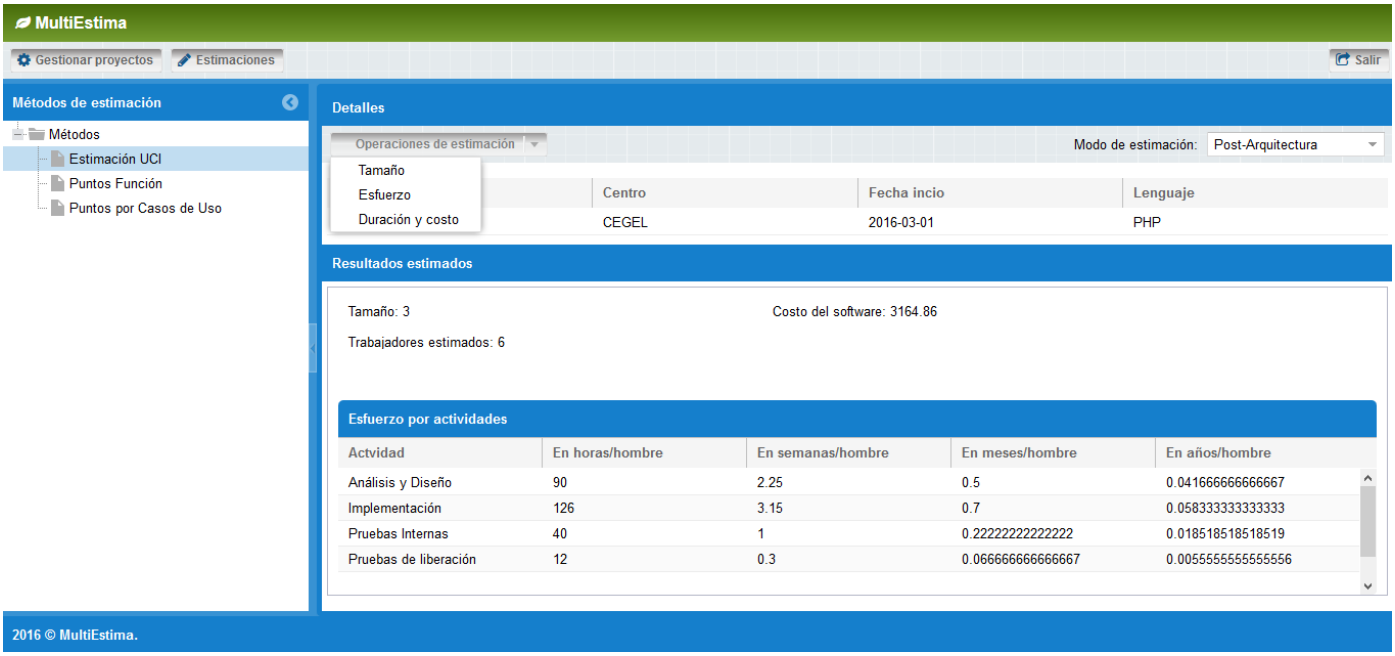
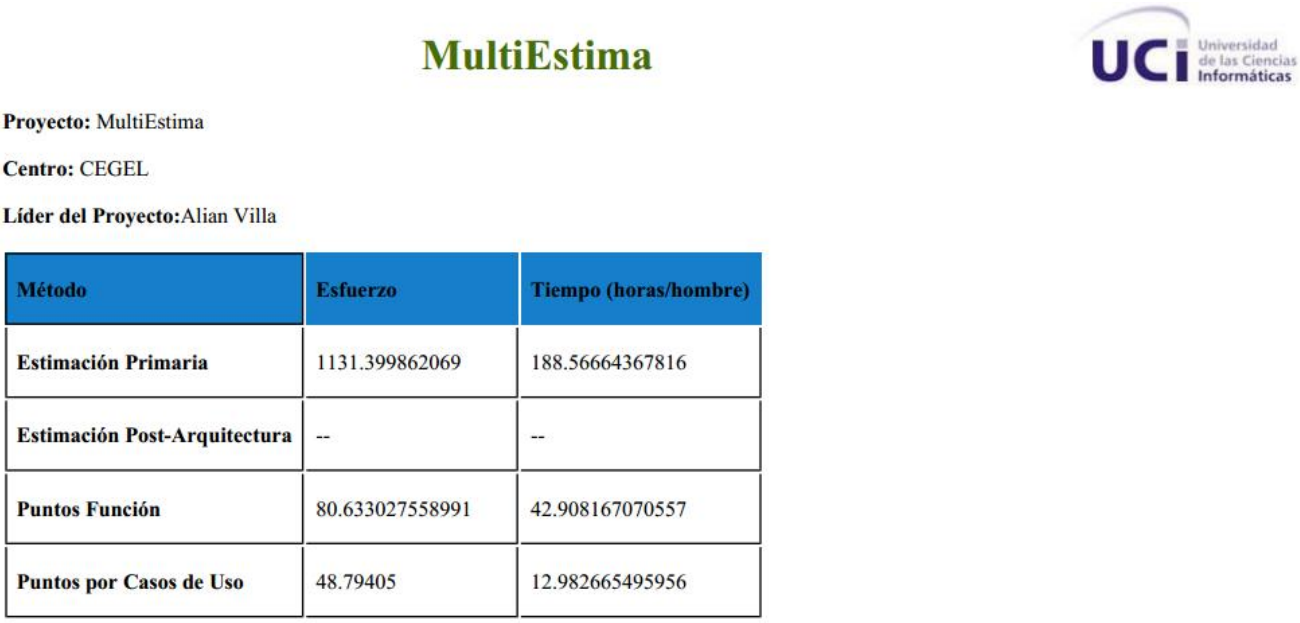


Figura 9: Resultados de las operaciones de los métodos de estimación: *MultiEstima*

El sistema permite además, realizar comparaciones de los métodos de estimación ejecutados sobre un proyecto (Véase en la Figura 10), y de tal manera, el usuario que utiliza la herramienta, puede comparar los resultados que brinda el método de Estimación UCI, con otros métodos utilizados para la estimación de esfuerzo y tiempo en el desarrollo de software.



MultiEstima

Proyecto: MultiEstima

Centro: CEGEL

Líder del Proyecto: Alian Villa

Método	Esfuerzo	Tiempo (horas/hombre)
Estimación Primaria	1131.399862069	188.56664367816
Estimación Post-Arquitectura	--	--
Puntos Función	80.633027558991	42.908167070557
Puntos por Casos de Uso	48.79405	12.982665495956

Figura 10: Comparación de los métodos de estimación: *MultiEstima*

Nota: Los resultados que aparecen en las figuras del sistema *MultiEstima,* presentes en este documento, son producto a la elección de datos al azar para realizar los cálculos de las estimaciones en forma de ejemplo.

3.4 Conclusiones del capítulo

- La utilización de los artefactos Tareas de Ingeniería, permitió durante el desarrollo del sistema, tener una programación organizada de los requisitos funcionales.
- La realización de las pruebas de software, permitieron que en la versión final del sistema, no tuvieran no conformidades y contara con la calidad requerida.

CONCLUSIONES

Durante el desarrollo del presente trabajo se logró cumplir con los objetivos trazados en la investigación, arribando a las siguientes conclusiones:

- A partir del establecimiento del marco teórico sobre el proceso de desarrollo de software, se identificaron los principales conceptos asociados al desarrollo de software que sustentan la investigación.
- La metodología utilizada, permitió que el proceso de desarrollo de software, se realizara de forma rápida y organizada.
- Los artefactos Historias de usuario, Tarjetas CRC, Tareas de ingeniería y los Casos de prueba de aceptación, generados durante las fases del ciclo de vida de la metodología, facilitaron el desarrollo del sistema.
- Mediante la realización de pruebas a nivel de unidad y a nivel de aceptación, se comprobó la validez del sistema, permitiendo que se obtuviera como resultado un producto listo para su uso.

RECOMENDACIONES

Con vistas a mejorar la solución en futuras versiones y por el tiempo limitado que el autor posee para la realización de la aplicación, se recomienda:

- Implementar otros métodos de estimación de proyectos de software.
- Continuar perfeccionando la herramienta realizada para lograr apoyar a la mejora de las estimaciones de proyectos de software en la UCI.

BIBLIOGRAFÍA

2012. *Ingeniería De Software I.* [En línea] 2012. http://es.scribd.com/doc/3062020/Capitulo-I-HERRAMIENTAS-CASE.

Aguilera, Sergio. 2015. Universidad de Belgrano, Buenos Aires, Argentina. [En línea] 3 de Junio de 2015. http://repositorio.ub.edu.ar:8080/xmlui/handle/123456789/5213.

Albinagorta, Victor. 2015. Slideshare. [En línea] 20 de Abril de 2015. http://es.slideshare.net/CoordinadorOrdoez/tendencias-actuales-del-desarrollo-de-software.

Alexandrou, Marios. 2016. infolific: technology. [En línea] 2016. http://infolific.com/technology/methodologies/crystal-methods/.

Alvarez, Miguel Angel. 2014. Desarrolladorweb.com. *Desarrolladorweb.com.* [En línea] 2014. http://www.desarrolloweb.com/manuales/.

Apache . 2012. Apache. *Apache.* [En línea] 2012. http://httpd.apache.org/.

Beck, K. 2000. *Extreme Programming Explained. Embrace Change.* s.l. : Pearson Education, 2000.

Benchmark, CCM. 2016. CCM. [En línea] Marzo de 2016. http://es.ccm.net/contents/223-ciclo-de-vida-del-software.

Blaha, Michael. 2010. *PATTERNS OF DATA MODELING.* s.l. : CRC press, 2010. ISBN 978-1-4398-1989-0.

Bootstrap. 2012. Bootstrap. *Introduccion a Bootstrap: front-end framework.* [En línea] 2012.

Brito, Dayami Rodriguez. 2012. *Método de Estimacion para proyectos de desarrollo de Software de la UCI.* La Habana : s.n., 2012.

Burbeck, Steve. 1997. *Applications Programming in Smalltalk-80(TM): How to use Model-View-Controller (MVC).* 1997.

2009. Buyto. [En línea] 2009. http://www.buyto.es/general-diseno-web/diferencias-entre-aplicaciones-web-y-aplicaciones-desktop.

Calderón, Sarah Dámaris Amaro y Rebaza., Jorge Carlos Valverde. 2007. Metodologías Ágiles. Trujillo - Perú : Universidad Nacional de Trujillo.Facultad de Ciencias Físicas y Matemáticas. Escuela de Informática : s.n., 2007.

Castro, Yohon Jairo Bravo. 2011. *Clasificación del software.* Plamira : Remington'Cat, 2011.

Code Project. What Is A Framework? **Clifton, Marc. 2003.** 2003.

Collins-Sussman, Ben, Fitzpatrick, B.W y Pilato. 2004. Version Control with Subversion. *Version Control with Subversion.* 2004.

Cornejo, J.E.G. 2012. ¿Qué es UML? [En línea] 2012. http://www.docirs.cl/uml.htm.

Cortes, Ignacio. 2014. Prezi. [En línea] 24 de Agosto de 2014. https://prezi.com/3xvgxaj3loyi/software-para-ingenieria-y-cientifico/.

2005. CoStar. [En línea] 2005. www.costar.com.

2016. CostXpert. [En línea] 2016. cost-xpert.software.informer.com.

CUBRID.org. 2012. *CUBRID:Open Source RDBMS - Seamless, Scalable, Stable and Free.* 2012.

Development, OSD - Open Systems. 2016. OSD. [En línea] 2016. http://www.osdglobal.com/faq/desarrollo-software/comparativo-web-vs-escritorio. 49C 59-05 of: 1001.

Doctrine. 2016. doctrine. [En línea] 2016. http://www.doctrine-project.org/about.html.

Drake, José M. 2008. *Programación orientada a objetos: Lenguajes, Metodologías y Herramientas.* Santander : s.n., 2008.

Eguiluz, Javier. 2012. symfony.es. *symfony.es.* [En línea] 2012. http://www.symfony.es/libro.

Escribano, G.F. 2013. EXtreme Programming / Programación Extrema. [En línea] 2013. http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf.

Estimación de Proyectos de Software. **Capuchino, Ana María Sánchez. 1998.** 1998.

Flanagan, David . 2007. *JavaScript: The Definitive Guide.* s.l. : O'Reilly Media, 2007.

Fowler, Martin. 2004. Inversion of Control Containers and the Dependency Injection pattern. *Inversion of Control Containers and the Dependency Injection pattern.* [En línea] 23 de Enero de 2004. http://martinfowler.com/articles/injection.html.

GAMMA, ERICH. 1998. *DESIGN PATTERNS CD ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE.* EE. UU : ADDISON-WESLEY, 1998. ISBN 9780201634983.

Garzás, Javier. 2011. Javier Gárzas. [En línea] 2011. javiergarzas.com.

Gauchat, Juan Diego . 2012. *El gran libro de HTML5,CSS3 y Javascript.pdf.* España : MARCOMBO, S.A., 2012. ISBN edición en formato electrónico: 978-84-267-1782-5.

Giraldo, Otoniel Pérez. 2002. *Métricas.* 2002.

Heemstra, F.J. 1992. *Software cost estimation, Information and Software Technology.* 1992.

Hernández, S.E.B. 2002. *Métricas de Estimación de Tamaño. Puntos de Caso de Uso.* 2002.

IBM. 2012. IBM developerWork. [En línea] 2012. http://www.ibm.com/developerworks/ssa/java/tutorials/j-introtojava1/.

InternetYa. 2016. InternetYa Soluciones Web. [En línea] 2016. http://www.internetya.co/ventajas-y-beneficios-de-las-aplicaciones-web/.

ISO. 2015. ISO 9000 - Quality management. [En línea] 2015. http://www.iso.org/iso/home/standards/management-standards/iso_9000.htm.

Jacobson, Ivar, Booch, Grady y Rumbaugh, James. 2000. *El Proceso Unificado de Desarrollo de Software.* s.l. : Adison Wesley, 2000.

JIMÉNEZ, JAVIER. 2011. *ALGORITMIA.* España : s.n., 2011.

jQuery. 2013. jQuery. *jQuery.* [En línea] 2013. http://jquery.com/.

Kruchten, Philippe. 1995. "Architectural Blueprints--The 4+1 View Model of Software Architecture". *IEEE Software, Institute of Electrical and Electronics Engineers.* 1995, págs. pp. 42-50.

Krutchen, Philippe. 2000. *The Rational Unified Process An Introduction.* s.l. : Addison Wesley, 2000.

2015. La Revista Informática.com. [En línea] 2015. http://www.larevistainformatica.com/ASP.htm.

Larman, Craig. 2004. *UML y Patrones.* s.l. : PEARSON, 2004. ISBN: 8420534382.

Leff, Avraham y Rayfield, James T. 2001. *IEEE Enterprise Distributed Object Computing Conference <<Web-Application Development Using the Model/View/Controller Design Pattern>>.* s.l. : IEEE, 2001.

LinuxLinks.com. 2015. LinuxLinks.com. [En línea] 2015. http://www.linuxlinks.com/article/20120525000545879/Django.html.

—. 2015. LinuxLinks.com. [En línea] 3 de Julio de 2015. http://www.linuxlinks.com/article/20120525000539219/RubyonRails.html; http://www.linuxlinks.com/article/20120525000531497/CodeIgniter.html; http://www.linuxlinks.com/article/20120525000545879/Django.html; http://www.linuxlinks.com/article/2012052500025250.

Machado, Yadira y Quintero, Mairelis. 2008. *Gestión de indicadores influyentes en la estimación de tiempo y esfuerzo en los proyectos productivos de la Universidad de las Ciencias Informáticas.* La Habana : s.n., 2008. pág. 145.

Matínez, Raúl Noriega. 2015. *El proceso de desarrollo de software.* s.l. : IT Campus Academy, 2015.

2008. MÉTRICA. VERSIÓN 3 Metodología de Planificación, Desarrollo y Mantenimiento de Sistemas de Información 2008. [En línea] 2008. http://www.csi.map.es/csi/metrica3/gespro.pdf.

Microsoft. 2016. Microsoft Solutions Framework Essentials. [En línea] 2016. https://msdn.microsoft.com/es-es/library/jj161047%28v=vs.120%29.aspx. ISBN 9780735623538.

Mota, Oscar. 2009. Maestros de la web. [En línea] 2009. http://www.maestrosdelweb.com/nueva-version-de-firefox-35/.

2013. MPI. [En línea] 2013. http://www.mpibccha pter.com/content/knowledge-plan.

Neosoft Sistemas SL. 2016. Neosoft Sistemas SL. [En línea] 2016. https://www.neosoft.es/servicios-informaticos/desarrollo-de-software/.

ONEI. 2013. *TECNOLOGÍA DE LA INFORMACIÓN Y LAS COMUNICACIONES.INDICADORES SELECCIONADOS.* 2013.

Palacio, J. 2006. El modelo Scrum. [En línea] 2006. http://www.navegapolis.net/files/s/NST-010_01.pdf..

pgAdmin. 2009. pgadmin: PostgreSQL Tools. [En línea] 2009. http://www.pgadmin.org/.

PHP. 2016. PHP. *¿Qué es php? -Manual.* [En línea] 2016. http://php.net/manual/es/intro-whatis.php.

Piattini, Mario . 1997. *Fundamentos y modelos de bases de datos.* Madrid : RA-MA, 1997. ISBN 9788478972838.

Piattini, Mario. 1996. *Análisis y Diseño Detallado de Aplicaciones Informáticas de Gestión.* Madrid : s.n., 1996.

PMBOK. 2004. *Guía de los Fundamentos de la Dirección de Proyectos (Guía del PMBOK®) Tercera Edición .* Four Campus Boulevard, Newtown Square, PA 19073-3299 EE.UU. : s.n., 2004.

PostgreSQL. 2013. PostgreSQL. *PostgreSQL.* [En línea] 2013. http://www.postgresql.org/docs/9.1/static/intro-whatis.html.

—. PostgreSQL: The world's most advanced open source database. *PostgreSQL: The world's most advanced open source database.* [En línea] [Citado el: 4 de diciembre de 2012.] http://www.postgresql.org/.

Pressman, Roger. 1998. *"Ingeniería del Software, un enfoque práctico".* 1998.

Pressman, Roger S. 2002. *Ingeniería del Software. Un enfoque práctico, sexta edición.* s.l. : Mc Graw Hill, 2002. ISBN 970-10-5473-3.

ReensKuag, Trygve y Coplien, James. 2009. *The DCI Architecture: A new Vision of Object-Oriented Programming.* 2009.

Rumbaugh, J., Ivar. 2000. *El Lenguaje Modificado de Modelado. 19.* 2000.

S.A. 2007. *G.S.I. Rational Rose Enterprise.* [En línea] 2007. http://www.rational.com.ar/herramientas/roseenterprise.html.

Sencha. 2016. Sencha. [En línea] 2016. https://www.sencha.com/products/extjs/.

Sun MicroSystems. 2014. netbeans.org. *netbeans.org.* [En línea] 2014. http://netbeans.org/index_es.html.

Tabares, John Deivis y Ramirez, Luis Fernando. 2008. *Administración de Redes de Computadores.* 2008.

Tendencias del desarrollo de software. **Daccach, José Camilo. 2015.** 2015.

TWIG. 2012. TWIG. [En línea] 2012. http://www.twig-project.org/.

UCI. 2012. Universidad de la Ciencias Informáticas. *Universidad de la Ciencias Informáticas.* [En línea] UCI, 2012. http://www.uci.cu/?q=mision.

Valdéz, José Luis Cendejas. 2014. *Implementación del modelo integral colaborativo (MDSIC) como fuente de innovación para el desarrollo ágil de software en las empresas de la zona centro - occidente en México.* Puebla, México : s.n., 2014.

2015. Visual Paradigm. [En línea] 2015. http://www.visual-paradigm.com.

1