ANÁLISIS, DISEÑO, DESARROLLO, PRUEBAS Y DESPLIEGUE DE SOFTWARE, CON LOS ESTÁNDARES DE CALIDAD, PROCESO Y TECNOLOGÍAS USADAS EN PRAGMA S.A.

CARLOS EDUARDO OSPINA MORALES

CORPORACIÓN UNIVERSITARIA LASALLISTA
FACULTAD DE INGENIERÍA
INGENIERÍA INFORMÁTICA
CALDAS, ANTIOQUIA
2012

ANÁLISIS, DISEÑO, DESARROLLO, PRUEBAS Y DESPLIEGUE DE SOFTWARE, CON LOS ESTÁNDARES DE CALIDAD, PROCESO Y TECNOLOGÍAS USADAS EN PRAGMA S.A.

CARLOS EDUARDO OSPINA MORALES

Informe de práctica empresarial para aspirar al título de Ingeniero Informático

ASESOR
MAURICIO BEDOYA
INGENIERO DE SISTEMAS

CORPORACIÓN UNIVERSITARIA LASALLISTA
FACULTAD DE INGENIERÍA
INGENIERÍA INFORMÁTICA
CALDAS, ANTIOQUIA
2012

DEDICATORIA

Dedicado a mi familia, quien con su apoyo han hecho de mi sueño de ser ingeniero, una realidad.

CONTENIDO

LISTA FIGURAS

LISTA GRÁFICAS

LISTA TABLAS

GLOSARIO

ALGORITMOS: En matemáticas, ciencias de la computación y disciplinas relacionadas, un algoritmo (del latín, dixit algorithmus y éste a su vez del matemático persa Al Juarismi) es una lista bien definida, ordenada y finita de operaciones que permite hallar la solución a un problema. Dado un estado inicial y una entrada, a través de pasos sucesivos y bien definidos se llega a un estado final, obteniendo una solución. Los algoritmos son objeto de estudio de la algoritmia.

APLICACIÓN: es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar un o diversos tipos de trabajo. Esto lo diferencia principalmente de otros tipos de programas como los sistemas operativos (que hacen funcionar al ordenador), las utilidades (que realizan tareas de mantenimiento o de uso general), y los lenguajes de programación (con el cual se crean los programas informáticos).

AUTOMATIZACIÓN: es el uso de sistemas o elementos computarizados para controlar maquinarias y/o procesos industriales substituyendo a operadores humanos.

El alcance va más allá que la simple mecanización de los procesos ya que ésta provee a operadores humanos mecanismos para asistirlos en los esfuerzos físicos del trabajo, la automatización reduce ampliamente la necesidad sensorial y mental del humano. La automatización como una disciplina de la ingeniería es más amplia que un mero sistema de control, abarca la instrumentación industrial, que incluye los sensores y transmisores de campo, los sistemas de control y supervisión, los sistema de transmisión y recolección de datos y las aplicaciones de software en tiempo real para supervisar y controlar las operaciones de plantas o procesos industriales.

BACKUP: Es la copia total o parcial de información importante del disco duro, CDs, bases de datos u otro medio de almacenamiento. Esta copia de respaldo debe ser guardada en algún otro sistema de almacenamiento masivo, como ser discos duros, CDs, DVDs o cintas magnéticas (DDS, Travan, AIT, SLR, DLT y VXA). Los backups se utilizan para tener una o más copias de información considerada importante y así poder recuperarla en el caso de pérdida de la copia original.

BASES DE DATOS: es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso. En este sentido, una biblioteca puede considerarse una base de datos compuesta en su mayoría por documentos y textos impresos en papel e indexados para su consulta. En la actualidad, y debido al desarrollo tecnológico de campos como la informática y la

electrónica, la mayoría de las bases de datos están en formato digital (electrónico), que ofrece un amplio rango de soluciones al problema de almacenar datos.

CALIDAD: Calidad tiene muchas definiciones, pero la básica es aquella que dice que aquel producto o servicio que nosotros adquiramos satisfaga nuestras expectativas sobradamente. Es decir, que aquel servicio o producto funcione tal y como nosotros queramos y para realizar aquella tarea o servicio que nos tiene que realizar. Con todo y a pesar de esta definición el término "Calidad" siempre será entendido de diferente manera por cada uno de nosotros, ya que para unos la Calidad residirá en un producto y en otros en su servicio posventa de este producto, por poner un ejemplo. Lo cierto es que nunca llegaremos a definir exactamente lo que representa el término Calidad a pesar de que últimamente este término se haya puesto de moda.

CHECKLIST: Los checklist o listas de comprobación, son un elemento cotidiano que muchas personas utilizan para asegurarse de que están cumpliendo algo. En este caso, las listas de comprobación miden el progreso del cumplimiento de los objetivos del software y ver aquellas áreas en las que se necesita trabajar.

CIBERESPACIO: es una realidad virtual que se encuentra dentro de los ordenadores y redes del mundo. El ciberespacio es un tema recurrente en la ciencia ficción. El término "ciberespacio" proviene de la novela de William Gibson Neuromante, publicada en 1984, y a su vez de una obra anterior del mismo autor, Burning Chrome.

El 8 de febrero de 1996, en Davos, Suiza, John Perry Barlow escribió la Declaración de independencia del ciberespacio en la que exhortaba a los gobiernos a no ejercer soberanía sobre el ciberespacio, definido por el mismo como "el nuevo hogar de la Mente".

CMS: Un sistema de gestión de contenidos (en inglés Content Management System, abreviado CMS) es un programa que permite crear una estructura de soporte (framework) para la creación y administración de contenidos, principalmente en páginas web, por parte de los administradores, editores, participantes y demás roles.

Consiste en una interfaz que controla una o varias bases de datos donde se aloja el contenido del sitio web. El sistema permite manejar de manera independiente el contenido y el diseño. Así, es posible manejar el contenido y darle en cualquier momento un diseño distinto al sitio web sin tener que darle formato al contenido de nuevo, además de permitir la fácil y controlada publicación en el sitio a varios editores. Un ejemplo clásico es el de editores que cargan el contenido al sistema y otro de nivel superior (moderador o administrador) que permite que estos contenidos sean visibles a todo el público (los aprueba).

CPU: La unidad central de procesamiento o CPU (por el acrónimo en inglés de central processing unit), o simplemente el procesador o microprocesador, es el componente en una computadora digital que interpreta las instrucciones y procesa los datos contenidos en los programas de la computadora. Las CPU proporcionan la característica fundamental de la computadora digital (la programabilidad) y son uno de los componentes necesarios encontrados en las computadoras de cualquier tiempo, junto con el almacenamiento primario y los dispositivos de entrada/salida

CSS: Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. CSS se utiliza para dar estilo a documentos HTML y XML, separando el contenido de la presentación. Los Estilos definen la forma de mostrar los elementos HTML y XML. CSS permite a los desarrolladores Web controlar el estilo y el formato de múltiples páginas Web al mismo tiempo. Cualquier cambio en el estilo marcado para un elemento en la CSS afectará a todas las páginas vinculadas a esa CSS en las que aparezca ese elemento.

EMPOTRADO: En informática, software que reside en memoria de solo lectura y se utiliza para controlar productos y sistemas de los mercados industriales y de consumo. El software empotrado puede ejecutar funciones muy limitadas y curiosas (p. Ej.: el control de las teclas de un horno de microondas) o suministrar una función significativa y con capacidad de control (p. Ej.: funciones digitales en un automóvil, tales como control de la gasolina, indicaciones en el salpicadero, sistemas de frenado, etc.).

FICHERO: Un archivo informático o fichero es un conjunto de bits almacenado en un dispositivo periférico. Un archivo es identificado por un nombre y la descripción de la carpeta o directorio que lo contiene. Los archivos informáticos se llaman así porque son los equivalentes digitales de los archivos en tarjetas, papel o microfichas del entorno de oficina tradicional. Los archivos informáticos facilitan una manera de organizar los recursos usados para almacenar permanentemente datos en un sistema informático.

GPL: Licencia Pública General de GNU o más conocida por su nombre en inglés GNU General Public License o simplemente su acrónimo del inglés GNU GPL, es una licencia creada por la Free Software Foundation a mediados de los 80, y está orientada principalmente a proteger la libre distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es software libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios. Existen varias licencias "hermanas" de la GPL, como la licencia de documentación libre de GNU (GFDL) que cubre los artículos de la

Wikipedia, la Open Audio License, para trabajos musicales, etcétera, y otras menos restrictivas, como la MGPL, o la LGPL (Lesser General Public License, antes Library General Public License), que permiten el enlace dinámico de aplicaciones libres a aplicaciones no libres.

HARDWARE: corresponde a todas las partes físicas y tangibles de una computadora, sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; 2 sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado.

HTML: El HTML (Hyper Text Markup Language) es el lenguaje con el que se escriben las páginas web. Es un lenguaje de hipertexto, es decir, un lenguaje que permite escribir texto de forma estructurada, y que está compuesto por etiquetas, que marcan el inicio y el fin de cada elemento del documento.

Un documento hipertexto no sólo se compone de texto, puede contener imágenes, sonido, vídeos, etc., por lo que el resultado puede considerarse como un documento multimedia.

Los documentos HTML deben tener la extensión html o htm, para que puedan ser visualizados en los navegadores (programas que permiten visualizar las páginas web).

Los navegadores se encargan de interpretar el código HTML de los documentos, y de mostrar a los usuarios las páginas web resultantes del código interpretado.

INGENIERÍA: es el conjunto de conocimientos y técnicas científicas aplicadas, que se dedica a la resolución u optimización de los problemas que afectan directamente a la humanidad. En ella, el conocimiento, manejo y dominio de las matemáticas y física, obtenido mediante estudio, experiencia y práctica, se aplica con juicio para desarrollar formas eficientes de utilizar los materiales y las fuerzas de la naturaleza para beneficio de la humanidad y del ambiente. Pese a que la ingeniería como tal (transformación de la idea en realidad) está intrínsecamente ligada al ser humano, su nacimiento como campo de conocimiento específico viene ligado al comienzo de la revolución industrial, constituyendo uno de los actuales pilares en el desarrollo de las sociedades modernas.

INTERFAZ: conocida también como GUI (del inglés graphical user interface) es un tipo de interfaz de usuario que utiliza un conjunto de imágenes y objetos gráficos para representar la información y acciones disponibles en la interfaz. Habitualmente las acciones se realizan mediante manipulación directa para facilitar la interacción del usuario con la computadora. Surge como evolución de la línea de comandos de los primeros sistemas operativos y es pieza fundamental en un entorno gráfico. Como ejemplo de interfaz gráfica de usuario podemos citar el entorno de escritorio del sistema operativo Windows, el X-Window de Linux o el de Mac OS X, Aqua. En el contexto del proceso de interacción persona-ordenador, la

interfaz gráfica de usuario es el artefacto tecnológico de un sistema interactivo que posibilita, a través del uso y la representación del lenguaje visual, una interacción amigable con un sistema informático.

INTERNET: es un conjunto descentralizado de redes de comunicación interconectadas, que utilizan la familia de protocolos TCP/IP, garantizando que las redes físicas heterogéneas que la componen funcionen como una red lógica única, de alcance mundial. Sus orígenes se remontan a 1969, cuando se estableció la primera conexión de computadoras, conocida como ARPANET, entre tres universidades en California y una en Utah, Estados Unidos.

LENGUAJE DE PROGRAMACIÓN: es un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Es utilizado para controlar el comportamiento físico y lógico de una máquina. Aunque muchas veces se usan los términos 'lenguaje de programación' y 'lenguaje informático' como si fuesen sinónimos, no tiene por qué ser así, ya que los lenguajes informáticos engloban a los lenguajes de programación y a otros más, como, por ejemplo, el HTML (lenguaje para el marcado de páginas web que no es propiamente un lenguaje de programación). Un lenguaje de programación permite a uno o más programadores especificar de manera precisa sobre qué datos debe operar una computadora, cómo estos datos deben ser almacenados o transmitidos y qué acciones debe tomar bajo una variada gama de circunstancias. Todo esto, a través de un lenguaje que intenta estar relativamente próximo al lenguaje humano o natural, tal como sucede con el lenguaje Léxico. Una característica relevante de los lenguajes de programación es precisamente que más de un programador puedan tener un conjunto común de instrucciones que puedan ser comprendidas entre ellos para realizar la construcción del programa de forma colaborativa.

LICENCIAMIENTO: El licenciamiento de un software le otorga derecho legal de ejecutar y utilizar el software. Un contrato de licenciamiento controla el uso de la licencia de un software. Normalmente los contratos de licenciamiento permiten que el software sea ejecutado en un número limitado de PCs y que se realicen copias sólo con propósitos de respaldo. Microsoft tiene varios programas de licenciamiento, cada uno diseñado para diferentes necesidades.

LINQ SQL: Language Integrated Query es un lenguaje de consultas creado para facilitar la explotación de los datos sin importar el tipo de fuente de datos utilizada. Permite consultar información en tecnologías tan diferentes como ficheros XML, bases de datos relacionales o colecciones fuertemente tipadas.

LOC: Sigla de líneas de código.

OBSOLESCENCIA: es la caída en desuso de máquinas, equipos y tecnologías motivada no por un mal funcionamiento del mismo, sino por un insuficiente

desempeño de sus funciones en comparación con las nuevas máquinas, equipos y tecnologías introducidos en el mercado.

PATENTE: es un derecho negativo, otorgado por el Estado a un inventor o a su causahabiente (titular secundario). Este derecho permite al titular de la patente impedir que terceros hagan uso de la tecnología patentada, y por lo tanto el titular de la patente es el único que puede hacer uso de la tecnología que reivindica en la patente. Las patentes no son de duración indefinida si no que caducan después de un período determinado que normalmente es de veinte años. Después de la caducidad de la patente cualquier persona puede hacer uso de la tecnología de la patente sin la necesidad del consentimiento del titular de ésta.

PIRATERÍA: La piratería de software es la copia, reproducción, utilización o fabricación no autorizadas de productos de software protegidos por las leyes de copyright internacionales y de los Estados Unidos.

PROCESS DASHBOARD: Es un programa gratuito creado como iniciativa a dar soporte a equipos que manejan la metodología Personal Software Process y Team Software Process.

PROGRAMA: Conjunto unitario de instrucciones que permite a un ordenador realizar funciones diversas, como el tratamiento de textos, el diseño de gráficos, la resolución de problemas matemáticos, el manejo de bancos de datos.

REDUNDANCIA: en bases de datos se refiere a la repetición inútil de datos.

REQUERIMIENTOS: En la ingeniería de sistemas, un requerimiento (del inglés requirement: 'requisito') es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio. Se usa en un sentido formal en la ingeniería de sistemas o la ingeniería de software.

En la ingeniería clásica, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto. Establecen qué debe hacer el sistema, pero no cómo hacerlo.

La fase de captura, análisis y registro de requisitos puede estar precedida por una fase de análisis conceptual del proyecto. Esta fase puede dividirse en recolección de requisitos, análisis de consistencia e integridad, definición en términos descriptivos para los desarrolladores y un esbozo de especificación, previo al diseño completo.

ROBOT: es una entidad virtual o mecánica artificial. En la práctica, esto es por lo general un sistema electromecánico que, por su apariencia o sus movimientos, ofrece la sensación de tener un propósito propio. La palabra robot puede referirse tanto a mecanismos físicos como a sistemas virtuales de software, aunque suele aludirse a los segundos con el término de bots.

REQUISITOS FUNCIONALES: Son aquellos requisitos que hacen referencia a una funcionalidad que debe tener el sistema.

REQUISITOS NO FUNCIONALES: Son propiedades que debe tener el sistema y que a pesar que no influyen directamente sobre la funcionalidad, deben ser tenidos en cuenta, algunos requisitos no funcionales pueden ser: Tiempos de respuesta, confiabilidad, mantenimiento, requisitos de almacenamiento, capacidades de banda ancha si se trata de una aplicación web.

SCARAB: Para facilitar la administración de incidencias o requerimientos entre Pragma y sus clientes, se seleccionó como herramienta de apoyo Scarab.

Scarab es un sistema de seguimiento de incidencias o nuevos requerimientos tanto técnicos como no técnicos, permitiendo el ingreso de información a través de formularios, a los usuarios involucrados con una incidencia le llega un correo electrónico informando sobre cambios de estados, además permite a los usuarios realizar consultas personalizadas dependiendo de sus necesidades.

SEO: El posicionamiento en buscadores, posicionamiento web u Optimizador de motores de búsqueda (SEO por sus siglas en inglés, de Search Engine Optimization) es el proceso de mejorar la visibilidad de un sitio web en los diferentes buscadores, como Google, Yahoo! o Bing de manera orgánica, es decir sin pagarle dinero al buscador para tener acceso a una posición destacada en los resultados.

La tarea de optimizar la estructura de una web y el contenido de la misma, así como la utilización de diversas técnicas de linkbuilding, linkbaiting o contenidos virales con el objetivo de aparecer en las primeras posiciones de los resultados de los buscadores (cuando un usuario busca por una determinada palabra clave o keyword), es conocida como SEO, sigla en inglés que significa Search Engine Optimizer, que traduce, 'Optimizador de motores de búsqueda'.

La aplicación de técnicas SEO suele ser más intensa en sitios web con mucha competencia y lo que se pretende con su aplicación es el posicionarse por encima de los competidores por determinadas palabras clave.

Las técnicas SEO pueden ser desmedidas y afectar los resultados naturales de los grandes buscadores por lo que si incumplen las cláusulas y condiciones de uso de los mismos pueden ser consideradas, en algunos casos, como una forma de SPAM, el spamdexing.

El trabajo es amplio, ya que el posicionamiento involucra al código de programación, al diseño y a los contenidos. También nos referimos a SEO para definir las personas que realizan este tipo de trabajo.

SISTEMA DE GESTIÓN DE BASES DE DATOS: Son un tipo de software muy específico, dedicado a servir de interfaz entre la base de datos, el usuario y las aplicaciones que la utilizan.

SOFTWARE: La palabra «software» se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de una tarea específica, en contraposición a los componentes físicos del sistema (hardware). Tales componentes lógicos incluyen, entre otros, aplicaciones informáticas tales como procesador de textos, que permite al usuario realizar todas las tareas concernientes a edición de textos; software de sistema, tal como un sistema operativo, el que, básicamente, permite al resto de los programas funcionar adecuadamente, facilitando la interacción con los componentes físicos y el resto de las aplicaciones, también provee una interfaz para el usuario. No hay un consenso sobre qué máquinas pueden ser consideradas robots, pero sí existe un acuerdo general entre los expertos y el público sobre que los robots tienden a hacer parte o todo lo que sigue: moverse, hacer funcionar un brazo mecánico, sentir y manipular su entorno y mostrar un comportamiento inteligente, especialmente si ése comportamiento imita al de los humanos o a otros animales.

TECNOLOGÍA: es el conjunto de conocimientos que permiten construir objetos y máquinas para adaptar el medio y satisfacer nuestras necesidades. Es una palabra de origen griego, τεχνολογος, formada por tekne (τεχνη, "arte, técnica u oficio") y logos (λογος, "conjunto de saberes"). Aunque hay muchas tecnologías muy diferentes entre sí, es frecuente usar el término en singular para referirse a una de ellas o al conjunto de todas. Cuando se lo escribe con mayúscula, tecnología puede referirse tanto a la disciplina teórica que estudia los saberes comunes a todas las tecnologías, como a educación tecnológica, la disciplina escolar abocada a la familiarización con las tecnologías más importantes.

WEBAPPS: En la ingeniería software se denomina aplicación web a aquellas aplicaciones que los usuarios pueden utilizar accediendo a un servidor web a través de Internet o de una intranet mediante un navegador. En otras palabras, es una aplicación software que se codifica en un lenguaje soportado por los navegadores web (HTML, JavaScript, Java, asp.net, etc.) en la que se confía la ejecución al navegador. Las aplicaciones web son populares debido a lo práctico del navegador web como cliente ligero, así como a la facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software a miles de usuarios potenciales. Existen aplicaciones como los webmails, wikis, weblogs, tiendas en línea y la propia Wikipedia que son ejemplos bien conocidos de aplicaciones web.

RESUMEN

El siguiente informe de trabajo de grado se realizó con el objetivo de exponer las metodologías, estándares, herramientas y documentación con las cuales se trabajó durante el período de la práctica empresarial de la carrera Ingeniería Informática de la Corporación Universitaria Lasallista en la empresa de desarrollo de software Pragma S.A. desempeñando el papel de Ingeniero de Proyectos, cumpliendo funciones de Análisis, Diseño, Desarrollo, pruebas y despliegue de software.

Para esto se contó con la asesoría de un experto en el área de desarrollo de software, orientando sobre cómo desarrollar la práctica y cumplir los objetivos de ésta. También se contó con la asesoría de la directora de proyectos, guiando sobre las funciones y objetivos a cumplir dentro de la compañía. Y en general se contó con el asesoramiento de los compañeros de la empresa que hicieron de este proceso un ambiente muy familiar dentro del ambiente laboral, brindando apoyo y ayuda en las funciones que se realizaron durante el período de práctica.

El informe expone principalmente el cómo se trabajó en el desarrollo de los proyectos en los que se participó, con las metodologías de Pragma S.A., específicamente la metodología Personal Software Process (PSP), Team Software Process (TSP) y de la metodología del área a la cual pertenecíamos, Portales Web, dentro de la Unidad Estratégica de Negocio de Software, en el departamento de Fábrica de Software.

Se planteó cada fase de las metodologías de trabajo, haciendo un análisis de cada una de ellas, y un capítulo al final donde se exponen las conclusiones del trabajo desempeñado durante la práctica, y recomendaciones tanto para la empresa Pragma S.A. como para la Corporación universitaria Lasallista, haciendo del informe una retroalimentación a la universidad y a la empresa, realizando un paralelo sobre los conocimientos adquiridos durante el transcurso de la carrera en la universidad versus las necesidades que debe suplir en un ambiente laboral con los estudiantes de Ingeniería Informática dentro de una empresa de desarrollo de software.

Palabras Claves: Software, Análisis, diseño, desarrollo, pruebas, despliegue, ingeniería informática, ingeniero de proyectos, Personal Software Process, Team software Process, metodologías de desarrollo de software, portales Web, práctica empresarial.

ABSTRACT

The next report of degree work was conducted with the aim of exposing the methodologies, standards, tools and documentation with which worked during the period of the business practice of computer engineering of the Lasallista University Corporation career Pragma S.A. software development company to play the role of engineer projects, fulfilling functions of analysis, Design, development, testing and deployment of software.

For this was advised by an expert in the area of software development, focusing on how to develop practice and meet the objectives of this. You were also advised by the Director of projects, to be guided on the functions and objectives to fulfill the company. And in general was on the advice of the companions of the company that made this process a family atmosphere within the working environment, providing support and help in functions that were made during the period of practice.

The report mainly outlines how work in projects that are participated, with Pragma S.A. methodologies, specifically the Personal Software Process PSP, Team Software Process TSP methodology and development of the methodology of the area where they work, Web portals, within the unit, strategic business Software, in the Department of Software Factory.

Were analyzed for each phase of the work methodologies, making an analysis of each one of them, and one end where the conclusions of the work performed during practice and recommendations for the Lasallista University Corporation both Pragma SA, making the report feedback to the University and the company, are conducting a parallel on the knowledge acquired during the course of the race in the University versus the needs that must meet in a working environment with computer engineering students within a software development company.

Keywords: Software, analysis, design, development, testing, deployment, software engineering, projects, Personal Software Process, Team software Process, methodologies for software development, Web portals, business practice.

INTRODUCCIÓN

El siguiente trabajo es un Informe de práctica empresarial para aspirar al título de Ingeniero Informático en la Corporación Universitaria Lasallista, este contiene 5 capítulos, ellos son: plan de trabajo, marco teórico, metodología, conclusiones y recomendaciones. Cada uno de estos capítulos se encuentra especificado de forma clara y contiene información de todos los temas que se incluyen en el informe, en este caso, se cuenta con información de la Ingeniería Informática, especialmente al desarrollo de software y metodologías de desarrollo software, de la metodología de desarrollo de software Personal Software Process PSP y Team Software Process TSP, de las herramientas y tecnologías usadas en Pragma S.A. para el desarrollo, como Visual Studio 2008, ASP.NET, C#, SQL Server, Oracle y en general se hace referencia a todos los temas que se relacionan con el objetivo de la práctica.

El presente informe se realizó con el objetivo de exponer las actividades y experiencias obtenidas a partir del análisis, diseño, desarrollo y pruebas de software, con los estándares de calidad, proceso y tecnologías usadas en Pragma S.A., en los cuales participe como Ingeniero de Proyectos, teniendo en cuenta el objetivo de que el software sea elaborado con calidad y a la medida del cliente, Familia S.A., para satisfacer su necesidad.

La información que está contenida en este informe, ha sido obtenida de la experiencia y estudio de actividades tanto personales como técnicas que se realizaron durante el periodo de práctica en los proyectos de rediseño de los sitios web de la marca Familia S.A.: Nosotras Online, Cosas de Familia, Familia Institucional y Tena.com.

1. PLAN DE TRABAJO

1.1. TÍTULO

Análisis, diseño, desarrollo y pruebas de software, con los estándares de calidad, proceso y tecnologías usadas en Pragma S.A.

1.2. OBJETIVOS

1.2.1. Objetivo General

Complementar la formación académica y elevar el nivel de competitividad, aplicando los conocimientos y destrezas adquiridos durante la carrera, sobre prácticas de desarrollo y métodos de desarrollo de software con calidad, satisfaciendo las necesidades de los clientes, en un proceso de retroalimentación entre Universidad y Empresa, en el cual se tiene la posibilidad de conocer el ambiente interno de la organización Pragma S.A.

1.2.2. Objetivos Específico

- Aplicar en Pragma los conocimientos informáticos adquiridos durante el proceso de formación académica, presentando soluciones reales que a la vez optimicen sus recursos.

- Seguir e implementar los estándares de documentación, procesos y procedimientos que sirven como apoyo para el análisis y desarrollo de sistemas informáticos con calidad, en el caso de Pragma se utiliza la metodología Team Software Process (TSP) y Personal Software Process (PSP).

- Utilizar los estándares de arquitectura, análisis, diseño, codificación y pruebas que se utilizan en el desarrollo de software de la empresa Pragma.

- Participar activamente en las fases del proceso de desarrollo de software que involucran a los ingenieros informáticos: Análisis, Diseño, Implementación y Pruebas.

1.3. JUSTIFICACIÓN

Utilizando los conocimientos adquiridos en el transcurso de los estudios de ingeniería informática se podrá participar en desarrollo de aplicaciones para la industria, todas las actividades a desarrollar han sido previamente estudiadas y aprobadas, por lo cual los objetivos planteados deberían ser perfectamente alcanzables.

La práctica empresarial ayudará al estudiante para que pueda crear, proponer y dinamizar con propuestas y acciones, a agilizar los procesos siempre en busca de presentar soluciones reales a la empresa que optimicen sus recursos.

En el ámbito de la ingeniería informática se aplicarán los conceptos del proceso de desarrollo de software desde el levantamiento de requerimientos hasta las pruebas, para luego hacer la puesta en producción, haciendo un énfasis mayor en la etapa de desarrollo que ha sido en sí misma el pilar de los estudios llevados a cabo en la Universidad.

La práctica empresarial comprende un conjunto de actividades de formación integral, en las que se conjugan estudio y trabajo, es una experiencia organizada y supervisada, cuya finalidad primordial es hacer más apto al estudiante para su futuro profesional y para su participación en el desarrollo socio económico del país, desarrollando en él cualidades como creatividad, innovación y flexibilidad; elementos extensivos en la formación profesional.

Con la realización de la práctica empresarial se está entregando a la sociedad un nuevo ingeniero preparado social e intelectualmente para su actividad profesional, con la capacidad necesaria para hacer parte de la productividad, desarrollo y crecimiento industrial brindando nuevos conocimientos y experiencias que estimulen el mejoramiento tecnológico.

Además se está favoreciendo de la integración en equipos de trabajo multidisciplinarios, propiciando así el desarrollo de habilidades de comunicación, auto confianza e independencia. Reconociendo las fortalezas y debilidades que se han alcanzado a lo largo de la formación personal, teórica y técnica profesional.

También la proyección social de la universidad se beneficia a través de la identificación de las diferentes problemáticas del sector informático para retroalimentar el currículo y sensibilizar al estudiante frente a las necesidades de la comunidad.

En cuanto a la parte económica se beneficiara tanto la empresa Pragma como los usuarios, ya que la empresa recibe ingresos por los productos desarrollados y

ajustados a las necesidades de los usuarios. Además, los usuarios se benefician de reducir sus costos en el momento que se automatiza algunos procesos que seguramente significa un costo beneficio para ellos. Igualmente, para el país cada uno de los desarrollos de software implica que las empresas se vayan sistematizando poco a poco, lo que causa que el nivel de calidad de las empresas incremente reflejándose en el progreso tecnológico del país.

1.4. ASESOR PRÁCTICA EMPRESARIAL

Mauricio Bedoya,
Ingeniero de Sistemas,
Líder Técnico de proyectos,
Pragma S.A.
Profesor Desarrollo Web, Administración de bases de datos, Compiladores y Desarrollo para Móviles,
Corporación Universitaria Lasallista
mauricio.bedoya@gmail.com

2. MARCO TEÓRICO

2.1. SOFTWARE

2.1.1. Definición de Software

El software es un conjunto de instrucciones que nos permite ejecutar una función determinada, en un tiempo razonable y facilita una actividad. Una estructura de datos que permiten a los programas manipular adecuadamente la información y los documentos que describen la operación y el uso de programas.

De acuerdo a lo anterior, existen tres componentes que describen el software: Programas, datos y documentos.

Actualmente el software juega un papel fundamental dentro de la sociedad, apoyándola en diversos sectores y áreas, sistematizando muchas de las tareas que se realizan diariamente, hoy el software, se puede considerar con un producto y un servicio.

Como producto ofrece la producción, el manejo, la adquisición, la modificación, el despliegue, la transformación y transmisión de la información por medio del hardware, es decir, en una red de computadores donde no importa el lugar de ubicación del software, sea en una maquina central u otro dispositivo, todos van a poder acceder a él, proporcionando potencia.

Como servicio, toma la forma de provisión dando soporte lógico específico, para satisfacer las necesidades del usuario. Además sirve para ejecutar el control de la unidad central de procesamiento (CPU); los sistemas operativos; comunicación de información, y permite la creación y control de otros programas, como lo son los lenguajes de programación, en estos casos ocurre a la prestación de servicios.

Como dice Pressman[1], el software entrega el producto más importante de nuestro tiempo: información. Transforma los datos personales (por ejemplo, las transacciones financieras de un individuo) de forma que los datos sean más útiles en un contexto local; maneja información de negocios para mejorar la competitividad; proporciona una vía para las redes de información alrededor del mundo (Internet) y proporciona los medios para adquirir información en todas sus formas.

[1] Pressman, Roger, Ingeniería del software, quinta edición, McGraw Hill. 2001. Capítulo 1

2.1.2. Características del software

Hoy en día los problemas que se generaron en los inicios de la industria del software, son los mismos, son éstos los que le dan las características esenciales a un software, y son los siguientes,

• ¿Por qué tarda tanto la obtención del software terminado?

• ¿Por qué son tan altos los costos de desarrollo del software?

• ¿Por qué es imposible encontrar todos los errores en el software antes de entregarlo a los clientes?

• ¿Por qué se gastan tanto tiempo y esfuerzo en el mantenimiento de los programas existentes?

• ¿Por qué es difícil medir el progreso al desarrollar y darle mantenimiento al software?

Estas preguntas y la preocupación de la industria, han generado lo que se denomina, la ingeniería del software, donde se nos brindan las herramientas necesarias y estándar para el desarrollo.

Para entender el software (y la ingeniería del software), es importante examinar las características que lo hacen diferente de otras cosas que construye el ser humano. El software es un elemento lógico, en lugar de físico, de un sistema. Por lo tanto, el software tiene características muy diferentes a las del hardware:

2.1.2.1. El software se desarrolla no se fabrica

Esta característica parte de la necesidad de diferenciar la manufactura del hardware del desarrollo de software, en ambos la calidad se alcanza a través de un buen diseño, pero en la manufactura se pueden dar problemas de calidad difíciles de corregir, mientras en el desarrollo del software éstos son fáciles de corregir. Las dos actividades son desarrolladas por personas diferentes en cada campo, y el trabajo en ambos proyectos es diferente y se manejan o se dirigen de diferente forma.

2.1.2.2. El software sufre una curva de obsolescencia

El software no se ve afectado por las inclemencias del tiempo, o por los males ambientales que desgastan el hardware, esto hace que las tasas de errores del hardware sean altas. Ahora las tasas de errores al inicio del desarrollo de un software son altas, pero estos errores se corrigen, lo que hace que requiera añadir continuas actualizaciones. La vida útil de un producto de software sin cambios puede ser de dos a tres años. Lo que significa que un software no se desgasta sino que se deteriora, como ejemplo, cuando un componente del hardware se desgasta se sustituye con un repuesto. Pero en el software no existen repuestos. Cualquier falla del software implica un error en el diseño o el proceso mediante el cual se pasó del diseño al código máquina ejecutable. Por lo tanto, el mantenimiento del software implica de manera considerable una complejidad mayor que el del hardware.

2.1.2.3. A pesar de que la industria tiene una tendencia hacia la construcción por componentes, la mayoría del software aún se construye a la medida

Un componente de software se debe diseñar e implementar de forma que pueda utilizarse en muchos programas diferentes. Los componentes reutilizables modernos encapsulan tanto los datos como el proceso que se aplica a éstos, lo que permite al ingeniero de software crear aplicaciones nuevas a partir de partes reutilizables. Por ejemplo, las interfaces actuales con el usuario se construyen con componentes reutilizables que permiten la creación de ventanas gráficas, menús desplegables y una amplia variedad de mecanismos de interacción. Las estructuras de datos y los detalles de procesamiento requeridos para construir la interfaz están contenidos en una librería de componentes reutilizables para la construcción de la interfaz.

2.1.3. Tipos de Software[2]

En la actualidad existen siete grandes categorías del software de computadora que presentan retos continuos para los ingenieros de software.

[2] Pressman, Roger, Ingeniería del software, quinta edición, McGraw Hill. 2001. Capítulo 1.

2.1.3.1. Software de sistemas

El software de sistemas es una colección de programas escritos para servir a otros programas. Algunos programas de sistemas (como los compiladores, editores y utilerías para la administración de archivos) procesan estructuras de informaciones complejas pero determinadas. Otras aplicaciones de sistemas (por ejemplo, componentes del sistema operativo, consoladores, software de red, procesadores para telecomunicaciones) procesan datos indeterminados. En cada caso, el área de software de sistemas se caracteriza por una interacción muy intensa con el hardware de la computadora; utilización por múltiples usuarios; operación concurrente que requiere la gestión de itinerarios, de compartición de recursos, y de procesos sofisticados; estructuras de datos complejas y múltiples interfaces externas.

2.1.3.2. Software de aplicación

El software de aplicación consiste en programas independientes que resuelven una necesidad de negocios específica. Las aplicaciones en esta área procesan datos empresariales o técnicos de forma que facilitan las operaciones de negocios o la toma de decisiones técnicas o de gestión. Además del procesamiento de datos convencional, el software de aplicación se utiliza para controlarlas funciones de negocios en tiempo real (por ejemplo, el procesamiento de transacciones en los puntos de venta y el control de procesos de manufactura en tiempo real.)

2.1.3.3. Software científico y de ingeniería

El software científico y de ingeniería, que se caracterizaba por algoritmos "devoradores de números", abarca desde la astronomía hasta la vulcanología, desde el análisis de la tensión automotriz hasta la dinámica orbital de los transbordadores espaciales, y desde la biología molecular hasta la manufactura automatizada. Sin embargo, las aplicaciones modernas dentro del área científica y de ingeniería se alejan en la actualidad de los algoritmos numéricos convencionales. El diseño asistido por computadora, la simulación de sistemas y otras aplicaciones interactivas han comenzado a tomar características de software en tiempo real e incluso de software de sistemas.

2.1.3.4. Software empotrado

El software empotrado reside dentro de la memoria de sólo lectura del sistema y con él se implementan y controlan características y funciones para el usuario final y el sistema mismo. El software incrustado puede desempeñar funciones limitadas y curiosas (como el control del teclado de un horno de microondas) o proporcionar capacidades de control y funcionamiento significativas (por ejemplo, las funciones digitales de un automóvil, como el control de combustible, el despliegue de datos en el tablero, los sistemas de frenado, etcétera).

2.1.3.5. Software de línea de productos

El software de línea de productos, diseñado para proporcionar una capacidad específica y la utilización de muchos clientes diferentes, se puede enfocar en un nicho de mercado limitado (como en los productos para el control de inventarios) o dirigirse hacia los mercados masivos (por ejemplo, aplicaciones de procesadores de palabras, hojas de cálculo, gráficas por computadora, multimedia, entretenimiento, manejo de bases de datos, administración de personal y finanzas en los negocios).

2.1.3.6. Aplicaciones basadas en Web

Las "WebApps" engloban un espectro amplio de aplicaciones. En su forma más simple, las WebApps son apenas un poco más que un conjunto de archivos de hipertexto ligados que presenta información mediante texto y algunas gráficas. Sin embargo, a medida que el comercio electrónico y las aplicaciones B2B adquieren mayor importancia, las WebApps evolucionan hacia ambientes computacionales sofisticados que no sólo proporcionan características, funciones de cómputo y contenidos independientes al usuario final, sino que están integradas con bases de datos corporativas y aplicaciones de negocios.

2.1.3.7. Software de inteligencia artificial

Este software utiliza algoritmos no numéricos en la resolución de problemas complejos que es imposible abordar por medio de un análisis directo. Las aplicaciones dentro de esta área incluyen la robótica, los sistemas expertos, el

reconocimiento de patrones (imagen y voz), las redes neuronales artificiales, la comprobación de teoremas y los juegos en computadora.

Actualmente muchos ingenieros trabajan en el desarrollo de software en los diferente tipos descritos arriba, unos creando nuevos, otros corrigiéndolos, adaptándolos y mejorándolos. La función de los ingenieros de hoy en día es generar facilidad a la tarea de los futuros ingenieros. En la actualidad se están abriendo nuevos campos en el desarrollo, existen las tecnologías inalámbricas, las cuales los ingenieros deben de implementarlas en dispositivos que generen conectividad de grandes masas en grandes redes. El internet, es otro de los puntos a explotar, generando aplicaciones sencillas que les permitan a los usuarios maximizar el uso de éste, además explotarlo como una nueva forma de mercado.

Los ingenieros además deben crear código que sea auto descriptivo, es decir, crear técnicas para las generaciones futuras y para los clientes en el que el código sea de fácil entendimiento, y estén al tanto de las modificaciones que se les hace. Como se ve, la tarea primordial del ingeniero de software actual es generar aplicaciones que permita la conexión de la gente en todo el mundo y faciliten la comunicación y la distribución de dichas aplicaciones. El ingeniero debe estar atento a la evolución constante de la tecnología, para iniciar un proceso de adaptabilidad y agilización que le permita acoplarse a estos cambios tanto en las reglas de negocios como de la tecnología.

2.1.4. Metodología de Desarrollo de Software

Una metodología de desarrollo de software se puede considerar como el proceso de la ingeniería de software, proceso que permite el desarrollo racional y oportuno de la ingeniería de software. Este proceso define el marco de trabajo para un conjunto de áreas claves que se deben establecer para la entrega efectiva del software. Estas áreas claves del proceso forman la base del control de gestión de proyectos de software y establecen el contexto en el que aplican los métodos técnicos, se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.

Los métodos de la ingeniería del software indican cómo construir técnicamente el software. Los métodos abarcan una gran gama de tareas que incluyen análisis de requisitos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología, incluyen actividades de

modelado y otras técnicas descriptivas. En nuestro caso específico usamos la metodología Team Software Process (TSP.:

2.2. MODELADO DE SISTEMAS CON UML

El Lenguaje Unificado de Modelado es un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos del negocio.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

En Pragma S.A. los ingenieros de Proyectos utilizamos en la fase de desarrollo de software los diagramas de clases y los diagramas de secuencia. Mientras que los arquitectos de software hacen uso de los diagramas de componentes, de implementación y de casos de uso.

Los diagramas de casos de uso, es una aproximación al diseño centrada en el problema con los casos de uso. Son el punto de entrada para analizar los requisitos del sistema y el problema que necesitamos solucionar.

Un caso de uso se modela para todos los procesos que el sistema debe llevar a cabo. Los procesos descritos dentro del caso de uso identifican el comportamiento del sistema, éstos se examinan y amplían de manera que se puedan mostrar que objetos se interrelacionan para que ocurra este comportamiento. Los diagramas de secuencia los utilizamos para mostrar estas relaciones entre los objetos.

Conforme vamos encontrando los objetos, se agrupan por tipo y se clasifican en un diagrama de clase. Es este diagrama de clase el que se convierte en el diagrama central del análisis del diseño orientado a objetos, y el que muestra la

estructura estática del sistema. Los diagramas de componentes se usan para agrupar las clases de un componente o módulo. La distribución general del hardware del sistema se modela usando el diagrama de Implementación.

2.3. MICROSOFT VISUAL STUDIO

Visual Studio es una suite de aplicaciones creada por Microsoft para dar a los desarrolladores un ambiente completo de desarrollo de software para plataformas Windows y .NET. Visual Studio puede ser usado para escribir aplicaciones de consola, aplicaciones de escritorio, aplicaciones móviles, aplicaciones ASP.NET y servicios web ASP.NET, en Lenguajes de Programación como C++, C#, VB.NET, J# y más, permitiendo que estas se puedan intercomunicar entre estaciones de trabajo, páginas web y dispositivos móviles. Visual Studio también incluye varias herramientas de desarrollo adicionales, éstas, dependen de la versión de Visual Studio que se esté usando.

Para el caso específico de Pragma, se participo en proyectos en los cuales la suite de desarrollo era MICROSOFT VISUAL STUDIO 2008, realizando aplicaciones ASP.NET y servicios web ASP.NET, con el lenguaje de programación C#.

ASP.NET es una potente tecnología que es usada para crear aplicaciones web, manejando una base de datos. Además provee un completo modelo de servicios web que permite crearlos rápida y fácilmente.

2.3.1. Visual Studio 2008

Es una versión de Visual Studio que trabaja con la versión 3.5 de framework de .NET, sin embargo, esto no es una restricción ya que ofrece la posibilidad de trabajar con múltiples framework en el mismo entorno de desarrollo. A las mejoras de desempeño, escalabilidad y seguridad con respecto a versiones anteriores, se agregan, entre otras, las siguientes novedades:

- La posibilidad de poder enfocar sus aplicaciones al manejo de la información por medio del nuevo lenguaje de consultas integrado LINQ. Un nuevo conjunto de herramientas diseñado para reducir la complejidad del acceso a bases de datos a través de extensiones para C++ y Visual Basic, así como para Microsoft .NET Framework. Permite filtrar, enumerar, y crear proyecciones de muchos tipos y colecciones de datos utilizando toda la misma sintaxis, prescindiendo del uso de lenguajes especializados.

- Visual Studio 2008 ahora permite la creación de soluciones multiplataforma adaptadas para funcionar con las diferentes versiones de .NET Framework: 2.0 (incluido con Visual Studio 2005), 3.0 (incluido en Windows Vista) y 3.5 (incluido con Visual Studio 2008).

- .NET Framework 3.5 incluye la biblioteca ASP.NET AJAX para desarrollar aplicaciones web más eficientes, interactivas y altamente personalizadas que funcionen para todos los navegadores más populares y utilicen las últimas tecnologías y herramientas Web, incluyendo Silverlight e IntelliSense para Java Script.

2.3.2. C#

C# es un lenguaje orientado a objetos elegante y con seguridad de tipos que permite a los desarrolladores compilar diversas aplicaciones sólidas y seguras que se ejecutan en .NET Framework. Se puede utilizar C# para crear aplicaciones cliente de Windows tradicionales, servicios Web XML, componentes distribuidos, aplicaciones cliente-servidor, aplicaciones de base de datos, y mucho más. Visual C# proporciona un editor de código avanzado, cómodos diseñadores de interfaz de usuario, depurador integrado y numerosas herramientas más para facilitar el desarrollo de aplicaciones basadas en la versión 3.5 de .NET Framework.

Tiene sus raíces en C, C++ y java reuniendo las mejores características de cada uno de ellos y proporcionando algunas propias que lo hacen uno de los lenguajes de programación más utilizado en el mundo del desarrollo. C# simplifica muchas de las complejidades de C++ y proporciona características eficaces tales como tipos de valor que admiten valores NULL, enumeraciones, delegados, expresiones lambda y acceso directo a memoria, que no se encuentran en Java.

Como lenguaje orientado a objetos, C# admite los conceptos de encapsulación, herencia y polimorfismo. Además de estos principios básicos orientados a objetos, C# facilita el desarrollo de componentes de software a través de varias construcciones de lenguaje innovadoras, entre las que se incluyen las siguientes:

- Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios en línea de documentación XML.
- Language-Integrated Query (LINQ) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

2.3.3. .NET Framework

.NET Framework es un componente integral de Windows que admite la compilación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .NET Framework está enfocado a cumplir los objetivos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet o ejecutar de forma remota.

- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.

- Ofrecer un entorno de ejecución de código que promueva la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.

- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan scripts o intérpretes de comandos.

- Ofrecer al programador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.

- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework. El motor en tiempo de ejecución se puede considerar como un agente que administra el código en tiempo de ejecución y proporciona servicios centrales, como la administración de memoria, la administración de subprocesos y la comunicación remota, al tiempo que aplica una seguridad estricta a los tipos y otras formas de especificación del código que promueven su seguridad y solidez. De hecho, el concepto de administración de código es un principio básico del motor en tiempo de ejecución. El código destinado al motor en tiempo de ejecución se denomina código administrado, a diferencia del resto de código, que se conoce como código no administrado. La biblioteca de clases, el otro componente principal de .NET Framework, es una completa colección orientada a objetos de tipos reutilizables que se pueden

emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML.

2.4. MICROSOFT SQL SERVER 2008

Es un sistema para la gestión de base de datos relacionales, permite almacenar información, realizar consultas y construir algunos procedimientos para obtener la información deseada.

Fue construido por Microsoft. Este motor de base de datos usa el lenguaje TSQL el cual es el principal medio de programación y administración.

SQL SERVER 2008 cuenta con características como: Seguridad, Confiabilidad, Accesibilidad, escalabilidad, además permite reducir costos. Esta poderosa herramienta ofrece soluciones muy importantes para resolver necesidades. A parte de las características ya mencionadas, SQL Server 2008 facilita:

- El uso de las herramientas de desarrollo .NET, al ser de la familia de Microsoft, la integración de aplicaciones con bases de datos en SQL server es realmente fácil.
- Contiene una comprensión integrada que permite comprimir las bases de datos para ofrecer mayos escalabilidad, sobre todo en bases de datos crecientes. Esta comprensión se realiza tanto a nivel de fila como a nivel de página, reduciendo la memoria requerida, ya que los datos están comprimidos mientras están en memoria.
- Permite mejorar el rendimiento debido a que reduce los bloqueos que se presentan en las transacciones.
- Ofrece mejores posibilidades de reflejos de bases de datos.
- Permite utilizar comandos LINQ directamente en las tablas, con LINQ SQL.

2.5. ORACLE

Oracle es una herramienta cliente/servidor para la gestión de base de datos relacional. Es considerado como uno de los sistemas de base de datos más completos, vendido a nivel mundial, aunque su potencia y elevado precio hace que la mayoría de empresas que lo utilizan son muy grandes y multinacionales, por lo general.

Entre sus características se distinguen principalmente:

- Soporte de transacciones
- Estabilidad
- Escalabilidad y
- Soporte multiplataforma

Para desarrollar en Oracle, como se mencionó antes, tecnología cliente servidor, se debería tener instalado la herramienta servidor, por ejemplo, Oracle 10g, y posteriormente podríamos ejecutar comandos sobre la base de datos desde otros equipos con herramientas de desarrollo como Oracle Developer.

El lenguaje utilizado para desarrollar en Oracle es PL/SQL, lenguaje de 5ª generación, potente a la hora de tratar y gestionar la base de datos.

En Pragma se utiliza el Oracle Developer, que permite compilar y ejecutar paquetes de procedimientos que se comparten, de modo que si alguien quiere realizar un cambio, pueda hacerlo y volver compilar y ejecutar los paquetes. Aunque en ocasiones parece bastante engorroso y poco fiable este proceso, ya que si al tiempo están ejecutando cambios en la base de datos, solo queda el último ejecutado y es posible que la versión no sea la correcta. La principal ventaja de esta herramienta es que es bastante intuitiva.

2.6. EPISERVER CMS

EPiServer CMS es la plataforma CMS comercial de más rápido crecimiento en el mundo, es un rápido, flexible y robusto sistema de Manejo de Contenido Web. EPiServer CMS es basado en la tecnología Microsoft .NET 3.5. Por lo que para alcanzar un alto potencial en el desarrollo de sitios web con el CMS es necesario tener suficientes conocimientos en desarrollo sobre .NET.

EPiServer CMS 6, ha realizado grandes avances tecnológicos, creando bases para la creación de sitios web muy grandes, más atractivos y potentes.

Hablar de EPiServer, es hablar de los web forms de ASP.NET, la forma clásica de construcción de los sitios web con ASP.NET, de los cuales se complementa.

EPiServer CMS, cuenta principalmente con dos funciones, desde las cuales podremos manejar y entender la herramienta, estas son:

Modo Edición: En este modo es donde creamos toda la estructura del sitio, publicamos nuevas páginas, actualizamos las viejas o porque no las eliminamos.

Modo Administración: Este modo nos permite administrar los idiomas del sitio web, crear tipos de páginas, crear propiedades a los tipos de páginas, darle nombre a nuestro sistema, entre muchas otras características como lo son generar Reportes.

Estos modos son desde el cual al final de un proyecto un cliente administra su sitio, pero toda la lógica del sitio, todas las reglas de negocio, toda la personalización de las funcionalidades que cubren las necesidades del cliente son realizadas en un proyecto ASP.NET, que utiliza todo el framework de EPiServer en conjunto con el de .NET.

Es el Desarrollador quien define las plantillas o vistas que van a utilizar los diferentes tipos de página, que de igual forma, es él quien los crea según necesite y con las propiedades que requiere, es quien desarrolla todas y cada una de las funcionalidades que van a tener las páginas, como lo son los controles web, pequeñas funcionalidades que se repiten en varias partes del sitio, todo esto utilizando las librerías que presenta EPiServer para el manejo rápido y eficaz de los datos que almacena del contenido del sitio web en una Base De datos SQL Server (bien podría ser Oracle) y utilizando las librerías de .NET permitiendo así una potencial integración que permite desarrollar grandes sitios web, con una gran rapidez y eficiencia.

En términos generales, EPiServer CMS, se puede dividir en un número de partes. La primera parte es la plataforma EPiServer CMS o el núcleo. Este contiene la

funcionalidad básica del sistema. La plataforma, es quien soporta el manejo de versiones, las pre visualizaciones, el flujo del sitio, los derechos sobre las páginas, etc. Estas son funciones disponibles cuando se trabaja en modo edición. Es esta la plataforma que está disponible en diferentes versiones y se desarrolla continuamente. Esta actualización no es automática. La actualización es pagada por las organizaciones que deseen actualizarla y lo hacen manualmente.

Sobre la plataforma esta la solución personalizada, que es lo que hace diferente el sitio desarrollado de otros sitios. Acá es donde entran partners certificados, es decir, desarrolladores entrenados en la herramienta, para crear la personalización de la solución en cooperación con el cliente. Así se puede dividir ligeramente en términos simples, la parte de personalización en 4 partes.

- La primera, las compañías tiene un perfil grafico, es decir, un diseño gráfico de su sitio web, el cuál es guardado en plantillas de formato CSS. Estos contienen los fondos, colores, imágenes etc., que son usados en un sitio web.
- La segunda parte, es un numero diferente de funciones que los visitantes pueden usar en el sitio web. Funcionalidades como un foro, enviar un email, o un link para imprimir la página. Cada una de estas funciones normalmente son desarrolladas sobre un tipo de página.
- La tercera parte consiste en los derechos para editores y visitantes de las diferentes páginas.
- La cuarta página es la información del sitio web, que es guardada en una base de datos. Cualquier imagen y documento en una página es guardada fuera del EPiServer CMS.

En estos 4 puntos se puede diferenciar los diferentes roles con los que se trabajan en EPiServer. En los primeros dos, está el rol del desarrollador, en el tercero el Administrador del sitio y en el cuarto el Editor del sitio.

La principal labor desarrollada en Pragma trabajando con EPiServer fue la segunda y parte de la primera, la tercera y cuarta. De la primera porque desde la agencia de diseño, es donde sale todo el HTML listo para montar al sitio web, pero que hay que pasar al proyecto en ASPX y hacerlo coincidir perfectamente con el HTML o corte, como lo denominan los diseñadores, y de las otras dos partes ya que para poder probar las funcionalidades se debe de crear contenido de prueba y tener los permisos correspondientes para hacerlo, además de que se deben de crear los manuales de uso tanto para el administrador como para el editor de los sitios web.

De acuerdo a todo lo anterior se pueden destacar importantes características en EPiServer CMS:

- El perfil gráfico de las compañías sobre el sitio web. Solo es necesario implementar un documento CSS, con lo que se puede hacer cambios en solo un archivo y ver los cambios en entre 10 o 10000 páginas. EPiServer soporta el uso de varios temas gráficos o diferentes colores en los menús, con propiedades dinámicas.
- Page Templates y Page Types (Plantillas y tipos de páginas). Cuando se planea la estructura del sitio web, se puede determinar diferentes funcionalidades para los usuarios, un page template es creado para cada una de esas funciones. Usando estas page templates, los editores pueden crear contenido y publicar páginas en el sitio web. EPiServer CMS permite el uso tanto de page templates y page types. Los page templates son los archivos .aspx en visual studio, que son lo que será mostrado a los visitantes y contienen la lógica requerida para ejecutar una función específica. Los page types, son los formularios en los cuales los editores pueden ingresar información en EPiServer CMS. Cada Page Type es mapeada a una page template para que esta sostenga la funcionalidad de la página. El trabajo en la Edición y Administración en el CMS es definido primordialmente por los page types.
- Contenido Centralizado en Base de datos. Toda la información creada por los editores en el CMS, es guardada en una base de datos, sin embargo, los archivos de una página no son guardados en base de datos, en su lugar, estos son guardados en una locación central. La página contiene un enlace al lugar donde está el archivo.
- Derechos de acceso. EPiServer CMS utiliza la misma autorización del modelo de Windows. Los permisos son divididos en dos partes. Una parte es la de administradores y editores. La otra es la de los permisos a los visitantes. Permitiendo que algunos empleados puedan editar información específica en algunas páginas del sitio. Por ejemplo, pueden haber diferentes departamentos que pueden trabajar en el mismo sitio web, sin acceder uno en la información del otro. Para los usuarios se les puede permitir ver algunas páginas, por ejemplo puede ser que sean diferentes versiones para los visitantes y para los empleados, como ejemplo de un sitio web tipo intranet y extranet.
- El Explorador Web. EPiServer CMS trabaja sobre varios exploradores, entre ellos los más importantes, Internet Explorer y Firefox. Aunque el HTML es diseñado para mostrar la información idéntica sin importar el hardware y el software usado, este puede lucir un poco diferente en cada explorador. Por eso hay maneras diferentes de manejar el HTML para que la interpretación sea igual en todos los exploradores.

3. METODOLOGÍA

Pragma S.A., es una empresa desarrolladora de software a la medida del cliente, debido a su gran tendencia de crecimiento se ha dedicado a fortalecer cada uno de los estándares que maneja al interior de la empresa y se ha preocupado por alcanzar certificaciones que den cuenta de su calidad, por eso actualmente se encuentra certificando al personal en la metodología de desarrollo Personal Software Process (PSP), para todas las personas que tienen que ver directamente en el desarrollo del software, como lo son los arquitectos, líderes técnicos e ingenieros de proyectos y en Team Software Process, para las personas las cuales sus principales funciones son administrativas, como los directores, gerentes, ejecutivos de cuenta y las personas de calidad, basando así sus servicios en las mejores prácticas de la industria.

Pragma atiende clientes en diversos sectores de la economía, como lo son Imusa, Comfama, Bayer Latinoamérica, cámara de comercio de Medellín, Banco Santander, Bancoomeva, Bancolombia, Isagen, Protección y Familia. Los servicios que presta Pragma S.A. son: Portales Web, integración y automatización de servicios con Service Oriented Architecture (SOA) y Business Process Management (BPM), Control Estratégico del Riesgo (CERO), Fábrica de Software haciendo Desarrollo de soluciones de software, Diagnóstico y Afinación de Aplicaciones de Software, Mantenimiento Correctivo y Evolutivo de Software y Outsourcing de Personal Especializado en Desarrollo de Software. Los servicios prestados son soportados en las siguientes tecnologías: Bases de datos en Oracle, MS/SQL Server, lenguajes de programación como C#, java, ASP.NET, PHP, PL-SQL.

Los proyectos en los que se ha tenido la oportunidad de trabajar son: Nosotras Online (Familia S.A.), Cosas de Familia (Familia S.A.) y Familia Institucional (Familia S.A), todos estos proyectos están abordados como portales dentro de la compañía, por tal motivo sigue un proceso estándar definido por Pragma S.A para realizar los portales, apoyándose este proceso en el uso de las metodologías PSP y TSP, para su ejecución desde la planeación y despliegue en producción. La metodología TSP es utilizada desde la administración mientras que la PSP es utilizada por todo el equipo que participa en el Desarrollo del sistema, como lo somos los ingenieros de proyectos. Se explicará a continuación como es trabajada la metodología TSP-PSP en el proceso de portales, partiendo de los proyectos mencionados anteriormente.

3.1. PERSONAL SOFTWARE PROCESS PSP[3]

PSP o Personal Software Process se puede definir como un conjunto de procesos cuya filosofía principal radica en la importancia del individuo que realmente hace el trabajo, los ingenieros. Con los procesos PSP cada ingeniero es el componente principal de un trabajo con alta calidad. Se puede resumir que PSP es una cultura donde cada uno está velando por la calidad de sus entregas, productos, diseños, en general del resultado de su conocimiento aplicado. Cada persona gestiona, califica y aprueba su propio producto.

La metodología PSP se centra en las prácticas individuales de trabajo de los ingenieros. El principio entonces de PSP es que para producir sistemas de software de calidad, cada ingeniero que trabaja en el sistema debe hacer un trabajo de calidad.

PSP muestra cómo planear y realizar seguimiento al trabajo, a usar un proceso definido y medido, a establecer metas medibles y hacer seguimiento del rendimiento contra las metas.

PSP muestra a los ingenieros como administrar la calidad desde el comienzo del trabajo, cómo analizar los resultados de cada trabajo y como utilizarlos para mejorar el proceso continuamente.

PSP se basa en los siguientes principios de planeación y calidad:

- Cada ingeniero es diferente, para ser más efectivo, los ingenieros deben planear su trabajo y ellos deben basar sus planes sobre su historial personal.
- Para mejorar constantemente su desempeño, los ingenieros deben usar personalmente un proceso bien definido y medible.
- Para producir productos de calidad, los ingenieros deben sentirse personalmente responsables por la calidad de sus productos.
- El costo de encontrar y corregir un defecto en etapas tempranas del proceso, es menor, en comparación con etapas posteriores.
- Es más eficiente prevenir los defectos que buscarlos y corregirlos.
- El camino correcto es siempre la manera más rápida y barata para hacer el trabajo.

Con PSP, los ingenieros antes de comprometerse y empezar a hacer su trabajo deben de planearlo y deben usar un proceso definido para planear el trabajo. Para

[3] Humphrey, Watts S., The Personal Software ProcessSM (PSPSM), Unlimited distribution subject to the copyright, Carnegie Mellon Software Engineering Institute, 2000. Capítulos 1 y 3.

hacer una mejor planeación y comprender su desempeño personal, en todos los pasos del proceso los ingenieros deben contabilizar su tiempo gastado, los defectos introducidos y removidos en cada una de ellas y el tamaño de los productos entregados. Así los ingenieros se enfocan en la calidad desde que empiezan su trabajo y finalmente deben analizar los resultados de cada trabajo y usar sus conclusiones para mejorar su proceso personal.

3.2. TEAM SOFTWARE PROCESS TSP[4]

TSP o Team Software Process se puede definir como un conjunto de procesos que sincronizan el trabajo de un grupo de personas que aplican los procesos PSP. Cuando hacemos parte de un equipo PSP compartimos responsabilidad, balanceamos cargas de trabajo y empujamos juntos hacia el logro de los resultados esperados.

TSP Y PSP definen un conjunto de prácticas de proyecto que han demostrado producir resultados altamente deseables en el desempeño del proceso en términos de calidad del producto entregado, del cronograma, y del costo. Una parte fundamental de trabajar con los equipos TSP, es la recolección y uso de medidas detalladas de tamaños, defectos, esfuerzo, tiempo y reproceso. Estas medidas aportan valiosa información que sirven para mejorar el desempeño y estado del equipo con el proyecto, y también sirve como base para prever aspectos importantes del cumplimiento del proyecto.

[4] Tamura, Shurei, Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models, Software Engineering Measurement and Analysis, Unlimited distribution subject to the copyright, Software Engineering Institute, 2000. Capitulo 1.

3.3. PROCESOS PSP Y TSP APLICADO AL PROCESO PRAGMA

En Pragma se entiende que desarrollar un portal web es diferente a cualquier otro desarrollo de software, por lo que se ha construido una metodología específica para el desarrollo de portales. Esta metodología define 5 grandes etapas, estas son: Análisis, Diseño, Construcción, Pruebas y Despliegue, que no escapan de las fases tradicionales de la Ingeniería Software. La diferencia entonces recae en los procesos que se siguen en cada una de estas etapas que combinados con los procesos y fases de PSP-TSP hacen de estos un completo proceso. Procesos medibles y con responsables directos. Estas 5 fases, también comprenden las fases de TSP y PSP, las cuales son: Requerimientos, Diseño de Alto nivel, Implementación, Pruebas de sistema y Post-mortem, que a su vez se dividen en pequeños procesos que se llevan a cabo en cada una de las fases, estos son:

- Requerimientos: Planeación, Levantamiento de Requerimientos e Inspección de requerimientos.

- Diseño de Alto nivel: Desarrollo del diseño de alto nivel e Inspección del diseño de alto nivel.

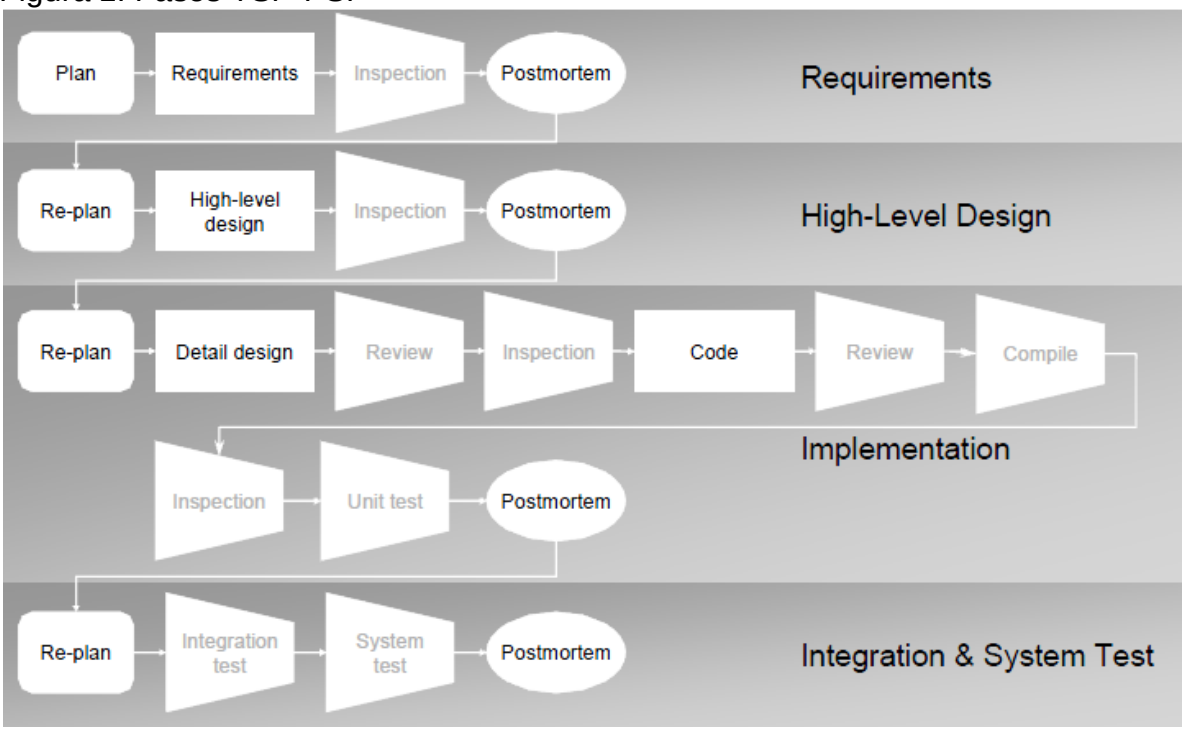
- Implementación (PSP): Diseño detallado, revisión del diseño detallado, Inspección del diseño detallado, codificación, revisión de la codificación, inspección de la codificación y pruebas unitarias.

- Pruebas de integración, pruebas de sistema.

- Post-mortem, donde se hace la evaluación final de los objetivos, metas, riesgos, cronograma y procesos definidos en la planeación, con respecto a la realidad vivida.

En la siguiente figura se puede observar cómo se distribuyen en las diversas fases los procesos:

Figura 1. Fases TSP-PSP



Fuente: PSP, A Self Improvement Process for Software Engineers.

Uno de los objetivos principales en un proyecto es la plena satisfacción del cliente, por ende en esta metodología uno de los principales actores en el proceso es el cliente, vinculándolo de forma intensiva al proyecto, donde plasme todo su conocimiento acerca de su negocio, y se refleje en el resultado final del proyecto.

3.3.1. Levantamiento y Análisis de Requisitos

La primera etapa entonces, el levantamiento de requisitos y su análisis acertado, son tareas fundamentales dentro del desarrollo del portal, por lo que es de gran importancia que se ejecuten de manera controlada y repetible. El objetivo entonces es especificar los requisitos funcionales, no funcionales y de información precisa del cliente para suplir sus necesidades, entregando como resultado el esquema visual del portal (Wireframes), Línea Gráfica del portal, el Documento de Alcance Detallado DAD y el Cronograma del proyecto.

Los esquemas visuales (wireframes) son elaborados por un arquitecto de información, en los cuales diseña la estructura del sitio a partir del conocimiento y la información que le brinda el cliente para cubrir sus necesidades, indicando como iría en cada tipo de página que forma parte del mapa total del proyecto. Este mapa debe considerar todas las funcionalidades del portal que el cliente desea,

especificadas hasta el mínimo detalle con el fin de garantizar 100% de la funcionalidad. Además de los esquemas visuales, en paralelo, se identifican los roles y perfiles de los usuarios del portal, para garantizar que su interacción sea eficiente y ajustada a los lineamientos establecido.

Luego, basados en los wireframes que diseñan los arquitectos de información, los diseñadores gráficos diseñan la línea gráfica o perfil gráfico del sitio, basados también en la información brindada por el cliente.

Con esta documentación, el equipo de desarrollo del portal cuenta con un alcance de proyecto claramente definido, que le sirve como guía para la planificación, asignación, ejecución y revisión de todas las tareas concernientes al proyecto, basándose en la metodología PSP y TSP.

3.3.1.1 Planeación

La planeación, es el primer proceso de la metodología PSP-TSP, tiene como resultado definir un plan organizado del proyecto, donde se pueda hacer un fácil seguimiento. Esta planeación se realiza en 4 o 5 días, donde a lo largo de éstos, se realizan 9 juntas, en las que se definen: Objetivos, Roles, Diseño Conceptual del software, la estrategia de desarrollo a seguir, las tareas a realizar, los tamaños estimados de las tareas, plan de calidad y el cronograma del proyecto.

- **Objetivos**: los objetivos, se basan en la necesidad que el cliente quiere satisfacer y los que la empresa quiere lograr con este proyecto. Uno de los primeros objetivos que se tuvieron en estos proyectos de familia por parte de la compañía fue el de Implementar TSP en la organización, debido a que fueron los primeros proyectos de la compañía que se realizaban siguiendo esta metodología.

- **Roles**: En TSP, cada uno de los integrantes del grupo toma un rol TSP en el equipo, estos roles se encargan de diferentes aspectos a controlar durante el desarrollo del proyecto, existen diversos roles, como lo son el Administrador de Planeación, el de soporte, el de implementación, el de diseño, el de procesos, el de interfaz con el cliente, el de pruebas, y el de Administrador de Calidad, que fue el asignado a Carlos Eduardo Ospina Morales, cada uno de estos roles tienen funciones definidas a través de guías que presenta TSP. Por motivos de confidencialidad estas guías no deben ser públicas, por lo que no se exponen. Pero cabe destacar los principales roles, el administrador de calidad, que debe velar por que se cumpla el plan de calidad definido en la planeación y el administrador de

planeación velando por que se cumplan las fechas determinadas de entrega.

Tabla 1. Definición de Roles Equipo RO, proyectos Familia S.A.

Team Member Roles	Requirements	Design	Implementation	Integration and Test
Team Leader	MB			
	SH			
Customer Interface Manager	RY			
	CO			
Design Manager	LV			
	RY			
Implementation Manager	SH			
	RY			
Planning Manager	SH			
	LV			
Process Manager	LV			
	SH			
Quality Manager	CO			
	SH			
Support Manager	CO			
	LV			
Test Manager	RY			
	CO			

Fuente: Documentación Pragma S.A.

- **Diseño Conceptual**: Para poder realizar un plan estimado, se debe saber cómo va a ser diseñado y construido el software, por esto desde esta primera fase los ingenieros definen como sería lo más cercano posible el diseño del software, basados en su experiencia. El diseño conceptual entonces es a grandes rasgos, como piensan los ingenieros será el diseño del sistema. Este diseño a medida que se avanza en los procesos se irá mejorando.

- **Estrategia de Desarrollo a seguir**: La estrategia de desarrollo a seguir, en Portales, la estrategia de desarrollo es iterativa, por lo que simplemente se define cómo se va a dividir el proyecto para su desarrollo. Nosotras Online por ejemplo se dividió en 2 fases, la primera con 7 iteraciones y la segunda con 4. Esto con el fin de mostrarle avances al cliente por cada iteración.

Tabla 2. Estrategia de Desarrollo Equipo RO, Proyectos Familia S.A.

CDF Requerimientos	Completar requerimientos	PP	22	44
HLD CDF	Crear Diseño Alto Nivel	PP	4	8
CDF I1	Construcción	LOC	1170	234
NOL Requerimientos	Completar requerimientos	PP	28	56
HLD NOL	Crear Diseño Alto Nivel	PP	4	8
NOL I2	Construcción	LOC	240	48
NOL I3	Construcción	LOC	830	166
NOL I4	Construcción	LOC	300	60
NOL I5	Construcción	LOC	270	54
NOL I6	Construcción	LOC	620	124
CDF I2	Construcción	LOC	4270	854
CDF I3	Construcción	LOC	4665	933
CDF I4	Construcción	LOC	4020	804
FI Requerimientos	Completar requerimientos	PP	39	78
HLD FI	Crear Diseño Alto Nivel	PP	4	8
FI I1	Construcción	LOC	1530	306
FI I2	Construcción	LOC	770	154
FI I3	Construcción	LOC	380	76
FI I4	Construcción	LOC	400	80
FI I5	Construcción	LOC	735	147
FI I6	Construcción	LOC	280	56
FI I7	Construcción	LOC	350	70

Fuente: Documentación Pragma S.A.

- **Tareas a Realizar**: Luego de definir todo lo anterior, se cuenta con suficiente información para dividir cada iteración por productos a realizar. Como ejemplo en una primera planeación de Nosotras Online, Cosas de familia y Familia Institucional, se realizo un plan donde se encontraron alrededor de 200 productos a realizar, que a su vez se dividen en pequeños pedazos denominados tareas, que alcanzaron las 2000.

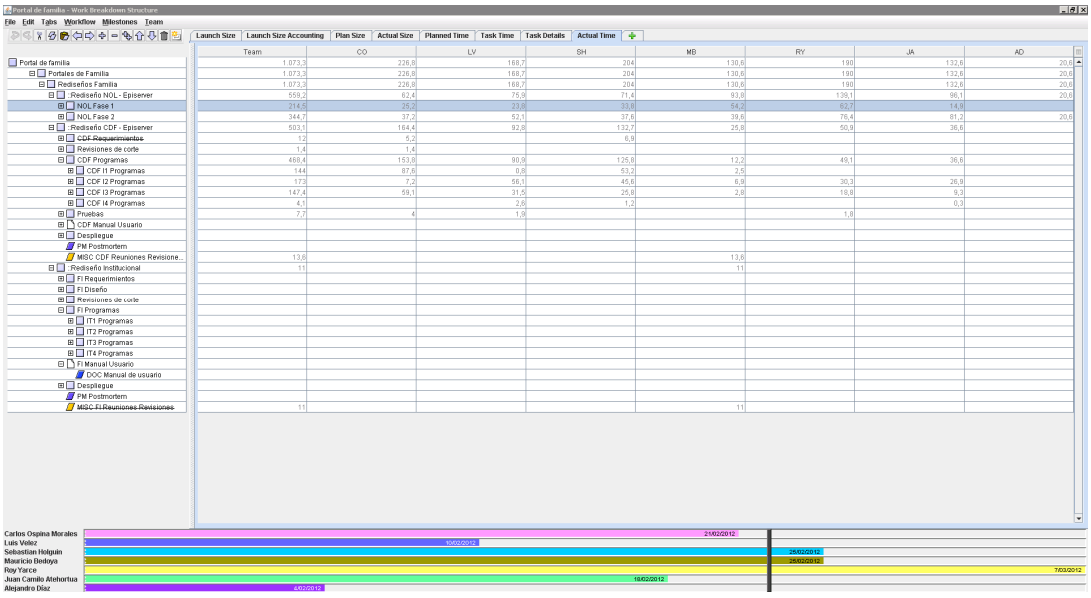
- **Tamaños Estimados**: Para cada producto a desarrollar se estima su tamaño, este tamaño puede ser en cualquier medida, pero por lo general se estima en líneas de código, con esta estimación más el promedio de líneas de código que se toma basados en la experiencia del equipo, ya se puede empezar a ver un cronograma cercano del proyecto. Para el plan inicial de Cosas de familia por ejemplo se estimaron alrededor de 12000 líneas de código. Existen diversos métodos para calcular estos tamaños de los productos, los cuales los brinda PSP y TSP, que por motivos de Confidencialidad y derechos no se pueden mencionar. Para los productos que no son programas, como los requerimientos, casos de prueba y manuales, los tamaños se miden en páginas.

- **Plan de Calidad**: El plan de calidad consiste en diversas metas a las que quiere apuntar el equipo con el proyecto, se definen por ejemplo métricas como número de defectos por cada mil líneas de código, número de defectos a encontrar, número de defectos a entregar al cliente, y

velocidades de desarrollo. Existen diversos métodos para calcular estas métricas, los cuales los brinda PSP y TSP, que por motivos de Confidencialidad y derechos no se pueden mencionar.

- **Cronograma del proyecto**: con los productos a desarrollar, con sus tamaños, con las métricas definidas y con la estrategia a seguir, existen herramientas en las que se ingresan estos datos y se da un cronograma completo a seguir por parte del equipo en el desarrollo del proyecto. En Pragma se utiliza el programa Process DashBoard, gratuito, que permite todas las funcionalidades requeridas para hacerle seguimiento a un equipo PSP-TSP. Este cronograma, será la línea base, a partir de la cual se empezara a mover el proyecto, es decir, la fecha planeada de terminación se moverá en el transcurso del desarrollo por diferentes motivos, pero la referencia y la meta siempre será la definida en el plan. Acá ya cada producto tiene una fecha estimada de entrega y cada miembro del equipo tiene su plan definido a seguir. Cada tarea aporta un porcentaje al proyecto, y cada semana se debe llegar a un porcentaje establecido, todos como equipo de ahora en adelante, luchan para que ese porcentaje se cumpla y no se cuelgue la fecha de entrega del proyecto. Este porcentaje es denominado Valor Ganado.

Figura 2. Cronograma Proyectos Familia S.A.



Fuente: Process Dashboard, Pragma S.A.

Basados en este cronograma final y las necesidades del cliente el equipo al final expone este cronograma a la empresa y al cliente para negociar la fecha de entrega, el equipo no solo debe definir una solo fecha, debe proponer varias estrategias para la entrega del proyecto basados en la necesidad de recursos, tiempo y costo, para dar la mejor opción, esto se hace el último día de la planeación y donde se espera haya una aceptación por todas las partes relacionadas, equipo, empresa y cliente.

Luego de realizar la planeación y ésta ser aceptada por la junta directiva de la empresa, el cliente y el equipo, éste último está listo para empezar todo el proceso de desarrollo del proyecto que inicia con el levantamiento de requisitos, que empezó con los arquitectos de información plasmando lo que quiere el cliente, la línea gráfica definida por los diseñadores y el plan TSP-PSP.

En este punto, todas las actividades que realizan los ingenieros, relacionadas con el proyecto, y que hayan sido planeadas, se deberán contabilizar en tiempo y tamaño, en el programa manejado por la empresa para tal fin, Process DashBoard. Si una actividad no estaba planeada y es necesaria esta se agrega al proyecto. Así semanalmente en una Junta Semanal realizada por el equipo, se expondrá el proceso y avance del equipo con respecto al plan del proyecto, y se tomaran medidas para enderezar el camino, si es necesario, para cumplir las metas y objetivos propuestos.

3.3.1.2. Documento de Alcance Detallado

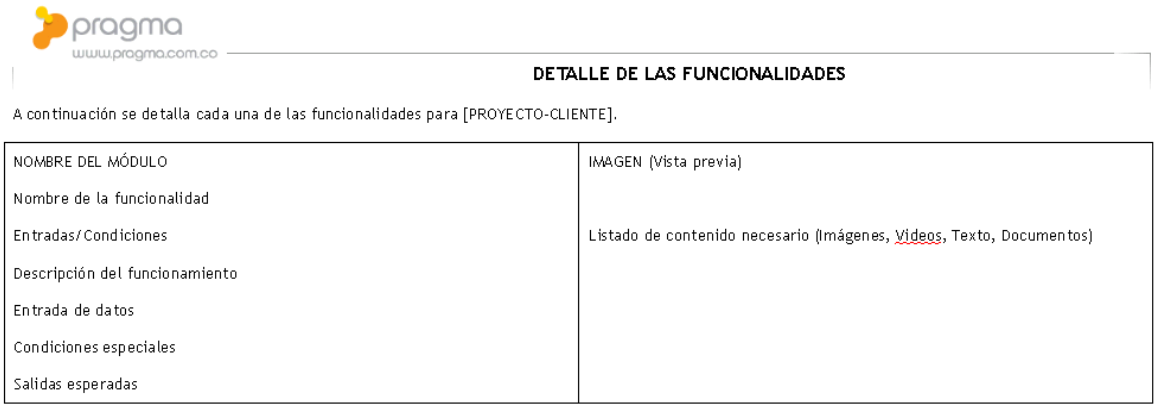
Para finalizar entonces, la etapa de levantamiento de requisitos y su análisis, los ingenieros de proyectos, elaboran el DAD. Este sigue la metodología TSP-PSP, donde un productor Realiza el Documento, lo revisa y lo pasa a un inspector, para que lo valide.

Este documento tiene como insumos los wireframes y la línea gráfica del sitio. Este documento detalla el alcance de cada funcionalidad del sitio, documentando entradas, procesamiento y salidas de la funcionalidad. También como será su visualización por parte del cliente final, sus restricciones e insumos necesarios.

Luego de ser finalizado por el ingeniero Productor, otro ingeniero realiza la inspección de este DAD, su objetivo es revisar con las guías dadas por TSP y PSP, y las propias de Pragma, que este documento cumpla con todos los requisitos para estar terminado. Si el inspector encuentra defectos al documento se lo hace saber al productor y éste debe de realizar las correcciones pertinentes o mostrar sus razones por las cuales cree que no es un defecto. Todo este proceso debe quedar registrado en las diferentes plantillas que se manejan para

estos procesos de PSP y TSP, además en las bitácoras de tiempo y defectos de cada ingeniero.

Figura 3. Formato Documento de Alcance Detallado



pragma
www.pragma.com.co

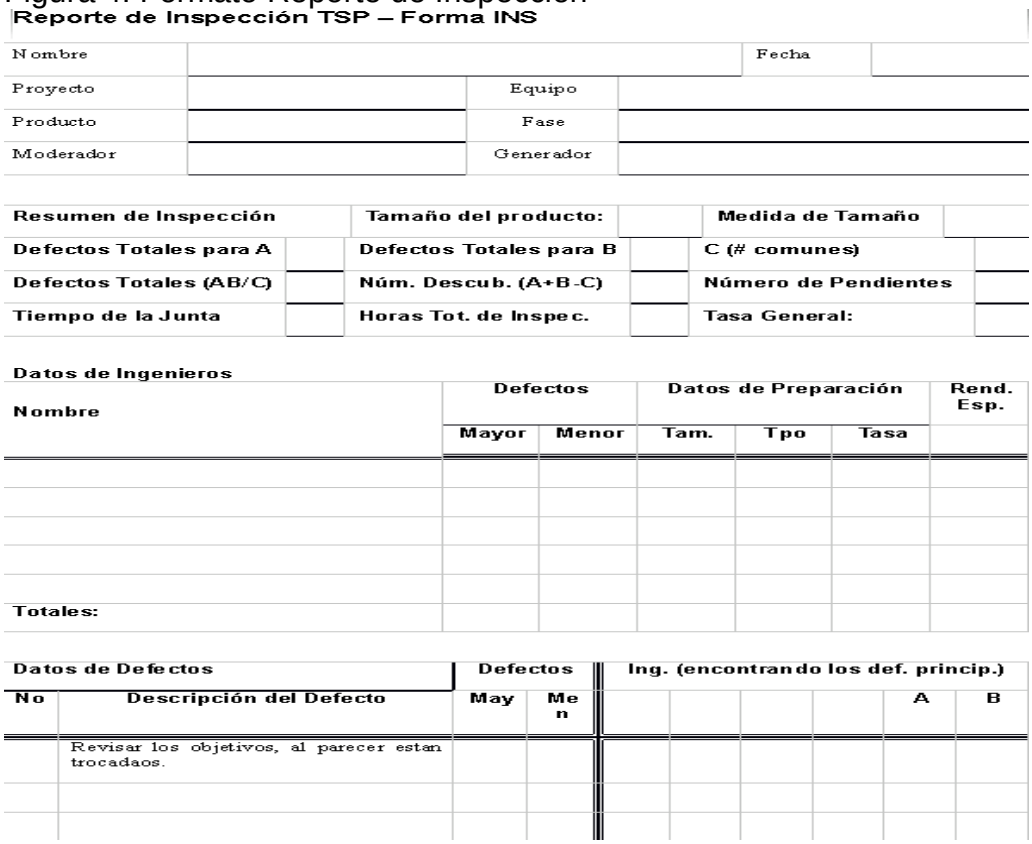
DETALLE DE LAS FUNCIONALIDADES

A continuación se detalla cada una de las funcionalidades para [PROYECTO-CLIENTE].

NOMBRE DEL MÓDULO	IMAGEN (Vista previa)
Nombre de la funcionalidad	
Entradas/Condiciones	Listado de contenido necesario (Imágenes, Videos, Texto, Documentos)
Descripción del funcionamiento	
Entrada de datos	
Condiciones especiales	
Salidas esperadas	

Fuente: Documentación Pragma S.A.

Figura 4. Formato Reporte de Inspección



Reporte de Inspección TSP – Forma INS

Nombre		Fecha	
Proyecto		Equipo	
Producto		Fase	
Moderador		Generador	

Resumen de Inspección		**Tamaño del producto:**		**Medida de Tamaño**	
Defectos Totales para A		**Defectos Totales para B**		**C (# comunes)**	
Defectos Totales (AB/C)		**Núm. Descub. (A+B-C)**		**Número de Pendientes**	
Tiempo de la Junta		**Horas Tot. de Inspec.**		**Tasa General:**	

Datos de Ingenieros

Nombre	**Defectos**		**Datos de Preparación**			**Rend. Esp.**
	Mayor	**Menor**	**Tam.**	**Tpo**	**Tasa**	
Totales:						

Datos de Defectos		**Defectos**		**Ing. (encontrando los def. princip.)**					
No	**Descripción del Defecto**	**May**	**Men**					**A**	**B**
	Revisar los objetivos, al parecer estan trocadaos.								

Fuente: Documentación Pragma S.A.

Figura 5. Process Dashboard

Fuente: Pragma S.A.

Todas estas etapas permiten y acuden a la interacción, supervisión, aprobación y retroalimentación del cliente, facilitando así la comunicación y el control del proyecto.

El DAD, los wireframes y la línea gráfica, son la base para las siguientes etapas del desarrollo del sistema, se requiere por lo tanto que su especificación sea lo más detallada posible, delimitando al máximo las funciones del sistema.

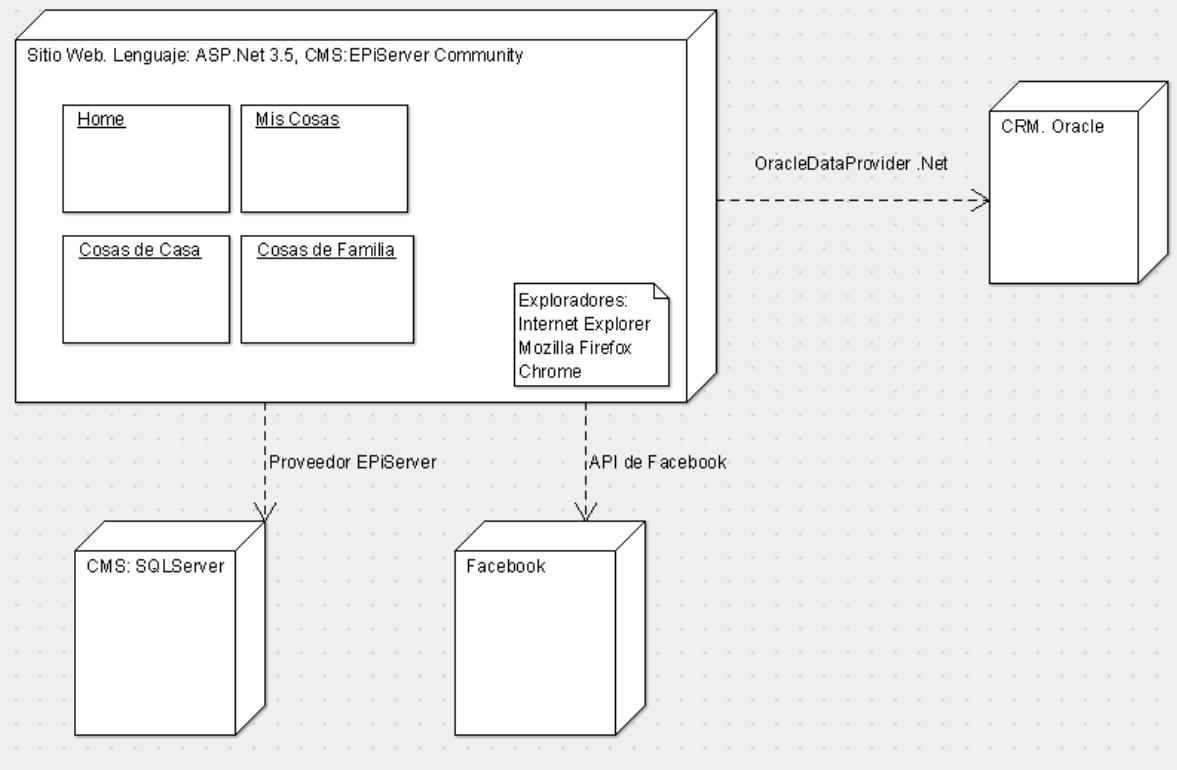
3.3.2. Diseño

En TSP-PSP, esta etapa es considerada el Diseño de Alto Nivel. Esta actividad es realizada principalmente por el Arquitecto de Software designado para el equipo. Este se encarga de definir la arquitectura general del sistema, realizando tareas como:

- Especificación de la infraestructura tecnológica, que constituirá el software.
- Definición de ambientes de software a utilizar. En los proyectos de familia por ejemplo, todos se basan en EPiServer CMS, Visual Studio .NET con lenguaje de programación C#, Base de datos en Microsoft SQL Server para el CMS y Base de datos Oracle, ésta última es la del Software para la administración de la relación con los clientes CRM.
- Descomponer el sistema en funciones principales que se puedan considerar como subsistemas, que serán evaluados con el cumplimiento de los requerimientos, generando varias alternativas. Estos subsistemas se consideran módulos que se utilizaran de los diversos software, por ejemplo módulos como Mail y Community de EPiServer
- Crear un diagrama de componentes, donde se especifican todos los que intervendrán en el desarrollo del portal a partir de la tecnología y módulos a utilizar, definiendo su estructura y tipos.
- Establecer los requisitos, normas y estándares originados como consecuencia de la adopción de una determinada solución de arquitectura o infraestructura, que serán aplicables tanto en este proceso como en la Construcción.

Un diseño de alto nivel podría lucir así, este fue el primero de Cosas de Familia:

Figura 6. Diseño de Alto Nivel Proyecto Cosas de Familia



Fuente: Documentación Pragma S.A.

Luego de que el Diseño de alto nivel es desarrollado por el arquitecto, éste lo revisa, y lo pasa para que otro ingeniero del equipo inspeccione su trabajo. Este proceso sigue el mismo del de requerimientos. El inspector sigue unas guías y checklist para inspeccionar el Diseño y el productor corrige los defectos si los tiene, ahora el Diseño de alto nivel ahora debe ser aprobado por el cliente.

3.3.3. Construcción

Antes de iniciar la construcción se debe contar con los siguientes criterios de entrada: Wireframes, Línea gráfica, DAD, Diseño de Alto nivel y la estrategia de desarrollo a seguir, si estos no se cumplen, o no se tiene claro de que trata la tarea, o se encuentra que alguno de estos documentos de entrada no está bien

realizados, la tarea no se realiza y se expone el caso ante el equipo para encontrarle solución.

La construcción de los portales se divide en dos actividades de construcción diferentes, una la desarrollada por parte de la Agencia de publicidad de la empresa, que se encarga de realizar el corte de la línea gráfica en temas gráficos en HTML, optimizando el corte de acuerdo a les mejores prácticas de Search Engine Optimization SEO, para el posicionamiento en buscadores. Corte que luego es entregado a los ingenieros de desarrollo para que continúen con la segunda parte de la construcción.

La otra parte es la implementación de todas las tareas del portal planeadas por los ingenieros de desarrollo. Ésta como se definió en la planeación es por iteraciones y sigue el cronograma acordado.

La implementación sigue también todo el proceso PSP-TSP, proceso que por parte de Pragma se dividió en 2 procesos, uno denominado Código funcional y otro Código Visual, éstos a la vez se dividen en otras fases, el funcional en 8: Diseño Detallado DDL, Revisión del DDL, Desarrollo de casos de prueba TD, Inspección del DDL, Codificación CODE, Revisión de CODE, Inspección de CODE y Pruebas Unitarias UT; y el Visual en 4: Codificación Visual, Revisión de la codificación Visual, Inspección de la codificación visual y Pruebas unitarias de la codificación Visual.

El primer proceso es el estándar de PSP-TSP, y el segundo proceso, surgió luego de identificar una problemática que se venía presentando durante el desarrollo de los proyectos, ésta se presentó, debido a que los productos que se desarrollaban no se podían compilar y ejecutar hasta UT, siguiendo los estándares PSP-TSP, por lo que no se podía observar como estaba quedando la parte visual y al llegar a la última fase UT, se apreciaba que ésta estaba mal, generando así la mayoría de defectos y afectando considerablemente las estadísticas de calidad. Así, se opto por partir el desarrollo en dos partes, esto gracias a la personalización y a la facilidad que tiene el uso de la metodología, una en la que se codificara toda la parte lógica del producto, independiente de que se viera bien o no, y la otra en la que luego de hacer la implementación lógica, se implementara la parte visual, incorporando todo el corte HTML y de estilos CSS realizado por la agencia.

Como se puede notar, la codificación visual retoma las 4 últimas fases de la codificación lógica, no incluye el diseño, debido a que sería el mismo de la codificación lógica. La diferencia entre estos dos procesos radica en que mientras en el primero nos encargamos de desarrollar toda la lógica de negocio que cumpla con el requerimiento funcional, en el otro nos encargamos de desarrollar toda la parte visual, HTML y Css, para que el portal luzca como el corte que ha sido desarrollado por la agencia, cumpliendo así con un requerimiento no funcional del software.

Partiendo entonces de las 8 fases de la codificación lógica, el objetivo principal de la construcción es plasmar un diseño detallado en código fuente, pasando por diferentes fases que lo depuran, en esta secuencia:

3.3.3.1. Diseño Detallado DDL

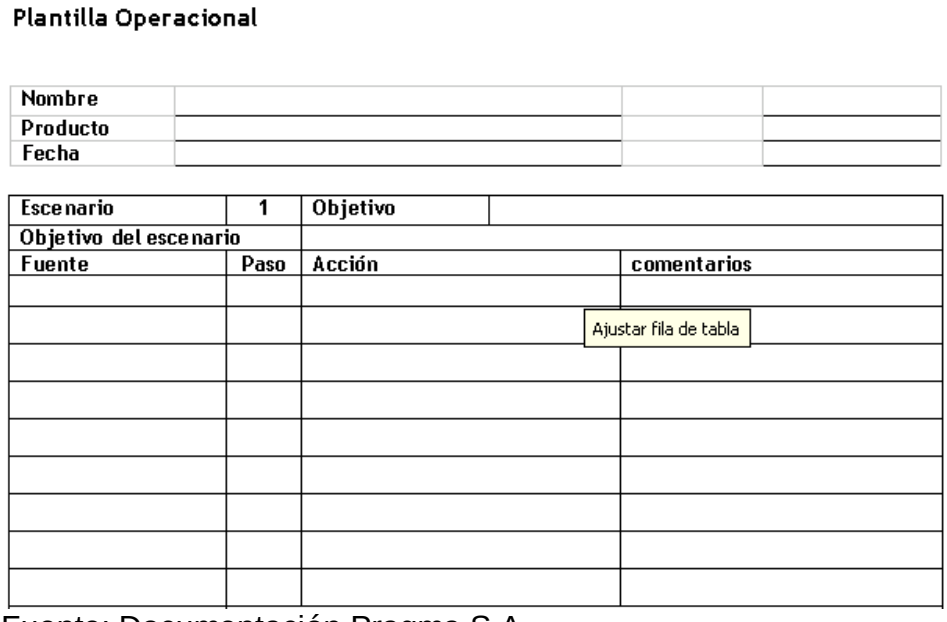
Antes de realizar el diseño, lo primero es revisar los requisitos del módulo o tarea a desarrollar, a manera de contextualización y entendimiento de lo que se debe hacer. Si los requerimientos (DAD) no son entendidos se debe hacer saber a quién los realizo para que revise el tema y lo complemente. Luego que la tarea es completamente entendible, el paso a seguir es completar tres plantillas:

- **Plantilla de especificación operacional:** En esta plantilla se define como los usuarios interactúan con el sistema.

- **Plantilla de especificación funcional**: Especifica todos los llamados y retornos de los métodos públicos identificados para los objetos y clases.

- **Plantilla de especificación Lógica:** Específica en seudocódigo la lógica interna de la tarea.

Figura 7. Plantilla de Especificación Operacional



Fuente: Documentación Pragma S.A.

Figura 8. Plantilla de Especificación Funcional

Plantilla de Especificación Funcional.

Nombre		**Fecha**	
Proyecto		**Equipo**	
Producto			

Nombre Clase	
Clase de herencia	

Propiedades

Declaración	**Descripción**

Ajustar fila de tabla

Métodos

Declaración	**Descripción**

Fuente: Documentación Pragma S.A.

Figura 9. Plantilla de Especificación Lógica

Plantilla de Especificación Lógica

Nombre		**Fecha**	
Proyecto		**Equipo**	
Producto			

CLASE

Función fin funcion
funcion CalcularVarianza() fin funcion

CLASE

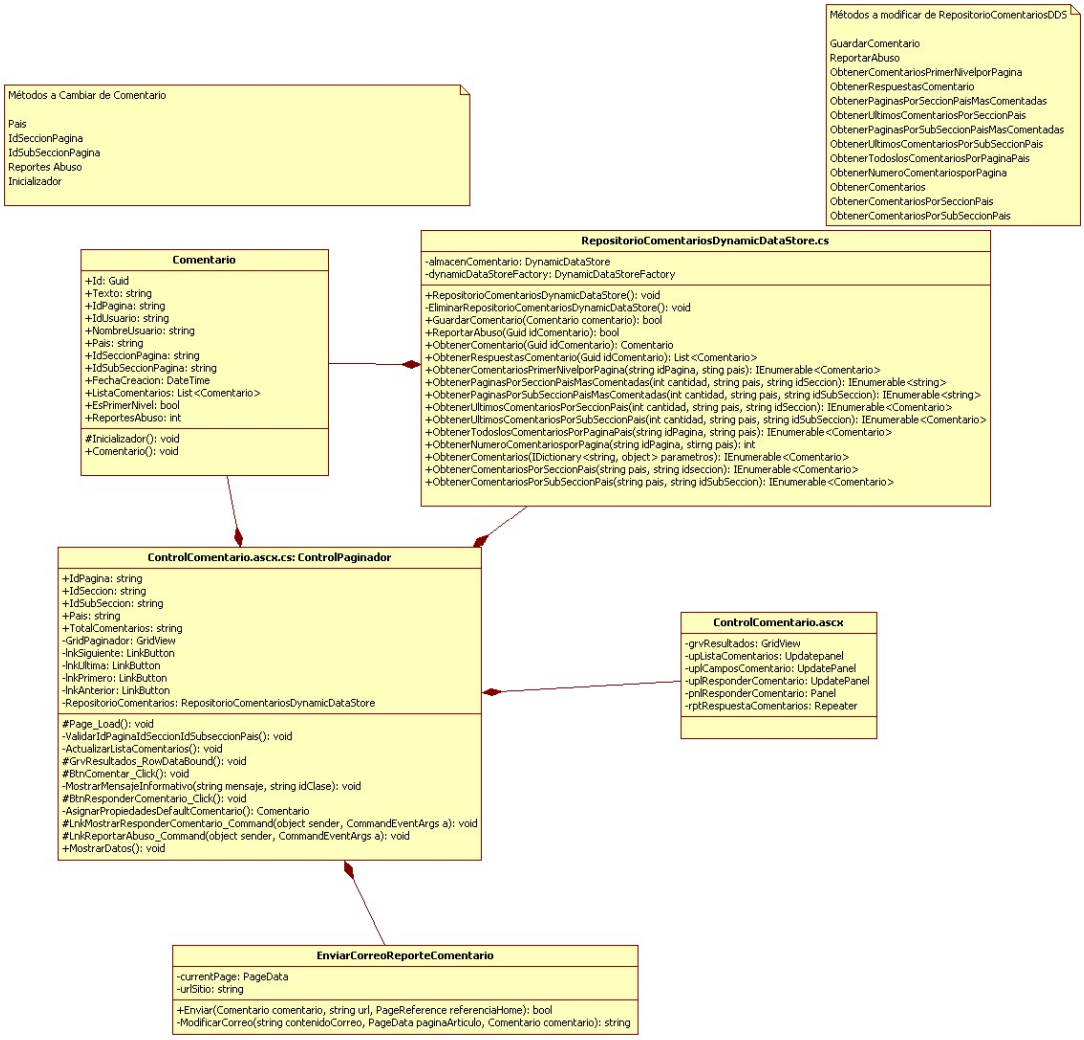
funcion fin funcion

Fuente: Documentación Pragma S.A.

Cuando se ha realizado la primera plantilla podemos:

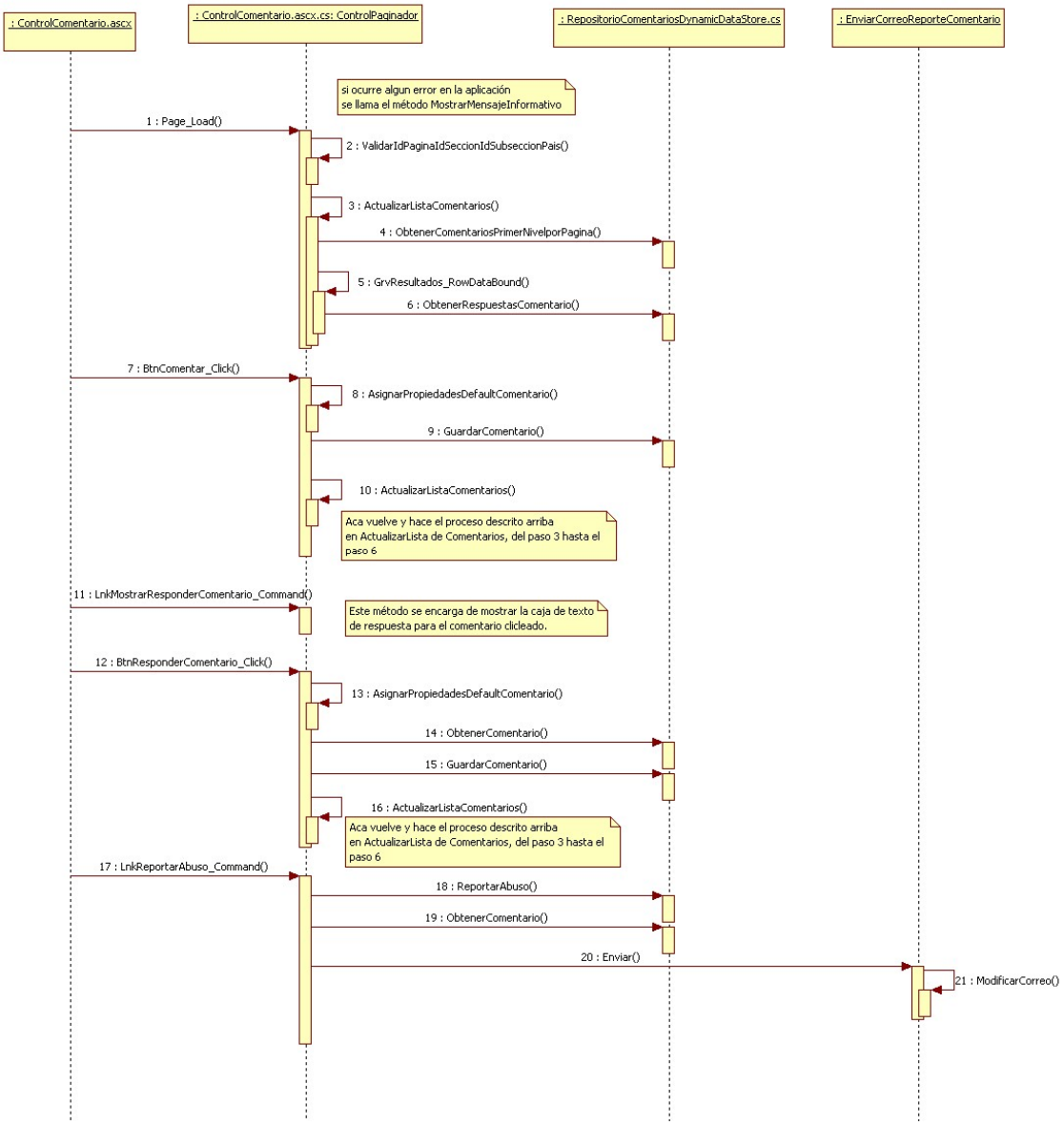
- Identificar los objetos que intervendrán durante el desarrollo.
- Establecer Propiedades, Métodos e Interfaces del objeto.
- Modelar estructura de datos.
- Definir las relaciones de los objetos entre ellos y el sistema.
- Dibujar los diagramas que se consideren necesarios hasta el nivel de detalle requerido para el desarrollo (Diagrama de secuencia, diagrama de clases, Modelo de datos), con respecto en el estándar UML.

Figura 10. Diagrama de clases Módulo Comentarios proyecto Cosas de Familia



Fuente: Documentación Pragma S.A.

Figura 11. Diagrama de secuencia Módulo Comentarios proyecto Cosas de Familia



Fuente: Documentación Pragma S.A.

A partir de estos diagramas es que podemos realizar las 2 plantillas siguientes y así terminar con el diseño Detallado.

3.3.3.2. Revisión del diseño Detallado

Para todas las revisiones e inspecciones que se hacen en todas estas fases, se definen antes de iniciar el proyecto los checklis que se utilizaran. Los checklist de revisiones son personales, es decir, cada miembro del equipo define sus propios checklist, de manera que ataque las falencias que considera tiene en dichas fases, basándose en su experiencia personal y en el histórico de defectos que ha conseguido. Los checklist de inspección son de común acuerdo entre todos los miembros del equipo, de manera que todos estén de acuerdo en lo que se inspeccionará.

El principal objetivo de estas revisiones e inspecciones es filtrar la mayoría de defectos antes de que el producto llegue al cliente, también cumplir con el plan de calidad que se defina en la planeación del proyecto, principalmente con los porcentajes de inyección y reparación de defectos antes de llegar a las pruebas unitarias. Es importante que en las revisiones personales se encuentren la mayoría de defectos, no es prudente pasar a inspecciones un producto que contenga muchos defectos, puesto que se puede considerar el producto de muy baja calidad, y quizá el administrador de calidad vea la posibilidad de que el módulo o tarea se vuelva a realizar.

Además es importante que el tiempo de las revisiones personales sea la mitad del tiempo invertido en el desarrollo de la fase que se está revisando, esta es otra métrica importante para el plan de calidad. Otra de las métricas de calidad importante es que los promedios de LOC por hora se estén cumpliendo desde el diseño hasta las inspecciones de Código, estos promedios son definidos en el plan de calidad.

Así pues, la revisión de diseño detallado se hace a nivel personal por cada miembro del equipo, demorándose no menos de la mitad del tiempo que se ha demorado para realizar el diseño. Se deben revisar todas las plantillas y los diagramas UML, si se encuentran defectos, éstos deben ser ingresados a la bitácora de defectos del miembro y contabilizar el tiempo que se demora corrigiéndolos. Luego de realizar la revisión se pasan los entregables al inspector asignado a la tarea.

3.3.3.3. Desarrollo de Casos de Pruebas Unitarias

Esta es la fase más corta en tiempo gastado. El objetivo de esta fase es escribir lo más detallado posible los diferentes casos de prueba correctos o incorrectos que afecten el funcionamiento del módulo o tarea basándonos en el diseño realizado.

Estos casos de prueba tienen mucho que ver con la plantilla de especificación operacional, solo que acá vamos un poco más allá y definimos las entradas, las salidas que esperamos sucedan en cada caso de prueba y las condiciones que debe cumplir el sistema para que el caso de prueba sea exitoso, y se es más específico en cuanto a información a ingresar al sistema para procesarla. Por ejemplo si se trata de un módulo de inicio de sesión, en la plantilla operacional definimos todos los pasos para que el usuario pueda iniciar sesión, pero en los casos de prueba detallamos la información con la que un usuario puede o no quedar haber iniciado sesión, por ejemplo, usuario: Carlos, contraseña: Carlos, condición: el usuario debe estar registrado en el sitio.

Se pueden definir cuantos casos de prueba como se consideren necesarios para probar la funcionalidad de la tarea o modulo. Estos casos de prueba serán los utilizados en la última de las fases de desarrollo, las pruebas unitarias.

Figura 12. Plantilla Casos de prueba unitarias.

Nombre			
Producto			
Fecha			

Escenario	1	Objetivo	
Objetivo del escenario			
Fuente	Paso	Acción	comentarios
Condiciones			
Resultados esperados			
Resultados actuales			

Ajustar fila de tabla

Fuente: Documentación Pragma S.A.

3.3.3.4. Inspección de Diseño detallado

Las inspecciones son guiadas por el Administrador de Calidad, éste, se encarga de juntar al productor de la tarea con el inspector, en esta junta el productor se encarga de explicarle al inspector cómo funciona el módulo o tarea, de manera que pueda entender lo que ha desarrollado, luego si la revisión del diseño detallado cumple con la mitad de tiempo gastado en diseño, el administrador de calidad da luz verde para que el inspector realice de manera personal la inspección.

El inspector sigue el checklist definido para la inspección de diseño, y en una plantilla denominada Reporte de inspección, consigna todos los defectos con su respectiva descripción y comentarios encontrados al diseño. La plantilla luce así:

Figura 13. Registro de Defectos.

Datos de Defectos		Defectos	
No	Descripción del Defecto	May	Men

Fuente: Documentación Pragma S.A.

Cuando un módulo o tarea es de vital importancia dentro del sistema, se puede considerar tener más de 1 miembro del equipo inspeccionando el producto.

Luego de que el inspector o los inspectores han realizado sus respectivas inspecciones se vuelven a reunir con el administrador de calidad y el productor, para informarles sobre los defectos que han encontrado, se discuten y se llega a un acuerdo con el productor para que este se comprometa para una fecha tener los defectos corregidos, en esta fecha el inspector revisa que efectivamente se han corregido los defectos, y puede continuar entonces el productor a la siguiente fase.

El tiempo gastado en estas inspecciones también es proporcional al tiempo gastado en la fase que se está inspeccionando. Estos porcentajes son definidos en la planeación del proyecto, en los proyectos de familia el tiempo gastado en las inspecciones era poco menos que el gastado en las revisiones personales.

3.3.3.5. Codificación

En esta fase el objetivo principal es plasmar el diseño elaborado para la tarea, en el lenguaje de codificación utilizado para desarrollar el sistema. En estos proyectos de familia, el lenguaje de codificación es C#, y dependiendo de si hay que realizar

acceso a bases de datos del CRM de familia o del EPiServer CMS, se desarrolla código SQL.

Uno de los logros importantes en esta fase es que el tiempo gastado de codificación sea poco menos que el tiempo gastado en el diseño detallado, así se estará cumpliendo con otra de las métricas de calidad, la cual significa que estamos hacienda muy buenos diseños detallados, cumpliendo con el requerimiento desde el inicio, sin necesidad realizar cambios.

Para la codificación, el equipo en la planeación define también un estándar de codificación a seguir, esto con el fin de que todos desarrollen de igual manera y hagan igual manejo de los diferentes ambientes de desarrollo utilizados. Esto evitara que en el momento de las inspecciones haya discordia con respecto al tratamiento que se le deben hacer a ciertos temas, como:

- Sintaxis
- Definición de clases
- Documentación del código
- Acceso a base de datos.
- Manipulación de excepciones
- Manipulación de variables y sus magnitudes
- Comparaciones lógicas
- Formato de llamados y retornos
- Formato en el nombramiento de variables, métodos, clases y propiedades.
- Formato en el nombramiento de elementos HTML o de ASPX.
- Buenas prácticas de desarrollo.

En esta fase pueden encontrarse defectos de las fases anteriores, éstos se guardan en la bitácora de defectos y se deben corregir para poder continuar. Por ejemplo si en la codificación nos damos cuenta que debemos cambiar algo del diseño sea porque no estaba diseñado o no fue tenido en cuenta desde los requerimientos, éstos deben de ingresarse a la bitácora de defectos, definiendo un tipo, la fase en la que se inyecto, su respectiva descripción y tiempo requerido para su corrección.

Una vez se ha terminado de plasmar todo el diseño en código, y de corregir todos los defectos hasta este momento encontrados, se puede continuar a la siguiente fase.

3.3.3.6. Revisión de la Codificación

Como se ha indicado antes la revisión es personal, y el checklist a seguir es el definido por cada miembro del equipo. El objetivo principal acá entonces es seguir el checklist de revisión de codificación personal, si se encuentran defectos corregirlos e ingresarlos a la bitácora de defectos. Es importante destacar que el código no puede ser ejecutado o testeado hasta las pruebas unitarias, por lo que se debe ser plenamente cuidadoso en la codificación para evitar encontrar muchos defectos en las pruebas unitarias.

El tiempo gastado en esta revisión al igual que en la revisión de diseño debe ser igual o superior a la mitad de tiempo consumido en la codificación. Luego de revisar el código desarrollado se continúa a la siguiente fase.

3.3.3.7. Inspección de la codificación.

La inspección de la codificación sigue el mismo proceso de la inspección de diseño detallado, solo que lo que se revisa es la codificación del módulo desarrollado por el productor, siguiendo el checklist de la inspección de código definido por el equipo.

Nuevamente los defectos encontrados se reportan en la plantilla destinada para esto, se socializan con el productor y este se compromete a tenerlos corregidos a una fecha establecida.

El tiempo gastado en esta inspección es proporcional al tiempo gastado en la etapa de codificación, poco menos del tiempo gastado en la revisión.

Luego de terminar con la etapa de inspección de la codificación se pasa a la última fase del proceso de desarrollo, pero antes se debe de consignar en la herramienta PSP, el Dashboard, donde se guardan los tiempos y defectos, el tamaño final del módulo o tarea en líneas de código LOC.

3.3.3.8. Pruebas Unitarias.

Todos los defectos que pudiesen haber tenido el módulo o la tarea, deberían haberse encontrado en este punto, por eso el objetivo de las pruebas unitarias es asegurarse que permanecen pocos si es que existen.

Las pruebas deben ser realizadas cuidadosamente, puesto que según estándares manejados por los creadores de la metodología PSP-TSP, tomaría entre 5 y 40 horas encontrar un defecto en pruebas posteriores.

En general si se hacen de manera apropiada las pruebas pueden encontrar los defectos restantes, de encontrarse debería corregirse e ingresar los tiempos de corrección, tipos y fases a los que pertenecen, y estar atentos a nuevos errores que puedan surgir después de corregir los encontrados.

Los casos de prueba deberían poder evaluar por ejemplo todos los casos de un case, o todas las rutas posibles que pueda tomar una condición. También debería probar las condiciones de error, fechas, límites mínimos, máximos, ceros, nulidad entre otras cosas.

Al final todos los casos de prueba deben pasar como exitosos para que pueda considerarse la tarea como terminada y de calidad.

3.3.3.9. Codificación Visual, Revisión de la Codificación Visual, Inspección de la Codificación Visual, Pruebas Unitarias.

Estas fases que comprenden la parte visual del código lógico desarrollado en las anteriores fases, siguen el mismo proceso de la codificación, revisión de la codificación, inspección de la codificación y pruebas unitarias de la codificación lógica. La principal diferencia radica en que en estas 4 fases se puede ejecutar el módulo o programa para chequear la parte visual debido al alto manejo de estilos, controles asp y HTML que se hace en los sitios web desarrollados. También cambia el checklist a seguir en las revisiones e inspecciones, puesto que son más orientados a la interfaz de usuario. Las pruebas unitarias a seguir son las mismas planeadas para la codificación lógica, pero teniendo en cuenta como se altera la interfaz.

De encontrarse defectos funcionales en estas fases, se consideran defectos encontrados en la fase de UT de la codificación lógica y su tiempo de corrección se cuenta en esa fase, para tener un histórico más congruente con la realidad de la tarea.

Existen tareas en las que no es necesario desarrollar codificación visual por lo que estas fases no se llegan a realizar.

3.3.4. Pruebas

Las pruebas finales desarrolladas dentro de Pragma son divididas en 3:

3.3.4.1. Pruebas de sistema

Las pruebas de sistema son elaboradas por el equipo de desarrollo y sus funciones son:

- Probar el sistema con respecto a estrés, verificar que es utilizable y que funciona de manera apropiada bajo todas las condiciones normales y anormales.
- Asegurar que son entregados los componentes y correcciones de alta calidad.
- Identificar y devolver componentes y correcciones defectuosos a sus productores.
- Mientras que pueden encontrarse ocasionalmente defectos, las pruebas de sistema tiene como objetivo las pruebas de facilidad de uso y estrés del sistema, no la de encontrar y corregir defectos.

Para la realización de las pruebas de sistema también se sigue un proceso, en el cual primero se realiza una planeación en el que se definen:

- Condiciones a ser probadas y la estrategia de pruebas.
- Herramientas de apoyo para elaborar las pruebas.
- Estimar y planear el esfuerzo de desarrollo de las pruebas de sistema.
- Numero de ciclos de pruebas de sistema y el calendario de trabajo.

Luego de que esta planeación es realizada, se revisa para asegurar que todas las condiciones son probadas en condiciones normales y de estrés, además de asegurar de que todas las capacidades de instalación, operación, recuperación y desempeño del sistema son probadas con respecto a corrección y facilidad de uso.

Cuando se ha determinado que el plan de pruebas está correcto, este se coloca en marcha, cada caso de prueba debe quedar registrado, si se encuentran defectos estos también deben registrarse y pedir revisión al responsable del componente, las pruebas deben suspenderse hasta que hayan sido revisados y corregidos los componentes defectuosos. A continuación se debe revisa que todos los casos de prueba se han corrido, que los que fallaron se corrigieron y se probaron nuevamente.

Para finalizar se entrega el sistema a las siguientes pruebas, actualizándolo con respecto a cualquier cambio realizado durante las pruebas.

3.3.4.2. Pruebas de Certificación

Una vez se ha terminado el desarrollo del portal y se han realizado las pruebas de sistema por parte del equipo, se entrega éste al departamento de calidad de Pragma, éste departamento se encarga de realizar su propia planeación y creación de casos de prueba con base al DAD desarrollado desde la planeación del proyecto.

El equipo de calidad se encarga de estudiar minuciosamente cada funcionalidad del portal y de realizar casos de prueba para cada una de ellas, pruebas de lógica, verificando cada trayectoria que pudiera tener una funcionalidad; pruebas de error, probando todas las condiciones de error; pruebas de seguridad, revisando secciones que realizan principalmente acceso de bases de datos del cliente, por ejemplo en estos proyectos de familia, donde se hacían acceso al CRM, como en el login, en el registro, en el contacto, entre otros; pruebas de desempeño, evaluando tiempos de salida y tiempo de respuesta; pruebas de compatibilidad, primordialmente las que tiene que ver con los diferentes exploradores para los que se certifica el portal; y pruebas de interfaz, revisando el comportamiento apropiado de cada interfaz, debido a la alta importancia que tienen los portales en este tema.

Una vez se entrega el sistema se encargan de ejecutar sus pruebas y de reportar cualquier anomalía a través de una herramienta utilizada por Pragma para el reporte de éstos llamada SCARAB, allí hacen los reportes de aprobación y de defectos, asignándolos a los productores de cada módulo. Una vez el productor es informado de este defecto procede a corregirlo y en SCARAB realiza el reporte para que pueda ser revisado nuevamente por el equipo de calidad. Una vez son aprobados todos los casos de prueba el departamento de calidad emite un documento en el que certifica que el portal está completamente listo para continuar con las siguientes pruebas.

3.3.4.3. Pruebas del Cliente

Estas pruebas son las últimas, y son desarrolladas por el cliente, en este caso Familia, sus pruebas las realizan de manera independiente, y se encargan principalmente de verificar que lo que se les ha vendido sea lo que se les va a entregar. Si hay defectos encontrados de parte del cliente, se informa del defecto al departamento de calidad y éste se lo asigna a través de Scarab al productor del módulo encargado, éste lo revisa, y de ser necesario lo corrige y lo vuelve a entregar a pruebas; de no ser un defecto, debido a que así estaba en el DAD o en la propuesta comercial, se toma como un cambio, se informa al cliente, y si este lo considera necesario se estima el cambio comercialmente y en desarrollo y se realiza. Cuando el cliente da su visto bueno, se procede a la última fase del proceso, el despliegue.

3.3.5. Despliegue

Durante esta tarea se realizan diferentes actividades

- Instalación y configuración en el servidor de producción.
- Elaboración de Manuales de administración y de usuario.
- Capacitaciones a administradores y editores del sitio.

Generando para estas actividades los siguientes entregables, que especifican que hacer por parte del cliente para instalar su sitio:

- Documento de instalación, configuración, integración y migración: Este documento incluye los insumos necesarios para los procesos de instalación, configuración, integración y migración del sistema, así como también la coordinación de tiempos y actividades.
- Manual de usuario: Documento en el cual se describe el uso del sitio por parte del usuario final
- Manual de administración: Documento técnico y de administración del sitio.
- Binarios: Instaladores de software base, Archivos de base de datos, Archivos Ejecutables y de configuración.

En este punto Pragma busca que los clientes sean independientes para la instalación en ambientes productivos, por lo que lo capacita en dicha instalación y aunque hace acompañamiento, solicita que sea directamente el cliente quien se haga responsable de ésta actividad.

También se realizan capacitaciones de administración funcional del portal si estás fueron contratadas.

Así finalmente se entrega el sitio al cliente y este es puesto en producción.

3.3.6. Post-mortem

El post-mortem es la última fase de un proyecto TSP-PSP, en esta fase el objetivo es encontrar mejoras en el proceso desarrollado para el cumplimiento del proyecto. Es una reunión donde participa todo el equipo dando sus puntos de vista acerca de que no se puede repetir, que se debe repetir y que se debe mejorar.

En el post-mortem también se revisa todas las métricas finales del equipo, y se toman como base para la planeación y realización de otros proyectos.

Finalmente se produce un documento final que se presenta a la gerencia, donde se evalúa el cumplimiento de los objetivos y metas planteados al inicio, el manejo de los riesgos a lo largo del proyecto, el cumplimiento en costo y calidad del proyecto y el cumplimiento del cronograma inicial.

3.4. ANÁLISIS Y SEGUIMIENTO DEL PROYECTO Y EQUIPO PSP-TSP

Como se mencionó anteriormente, la metodología PSP-TSP, define un conjunto de prácticas de proyecto que han demostrado producir resultados altamente deseables en el desempeño del proceso en términos de calidad del producto entregado, del cronograma, y del costo, para hacer cumplimiento de esto se hace continuamente análisis y seguimiento del proyecto y del equipo, por parte de un coach TSP-PSP.

El coach no es más que una persona capacitada en todo el manejo de equipos PSP-TSP, que ayuda al equipo desde su conocimiento en esta metodología a entender y mejorar su desempeño. El coach está presente desde el inicio del proyecto hasta el final, guiando al equipo en las decisiones a tomar para que estas sean las mejores, beneficiando al equipo la empresa y al cliente. Su rol, no es de jefe, es de acompañamiento, es un rol intermedio y de comunicación entre la gerencia y el equipo de desarrollo, sin inclinarse para ninguno de los dos lados. Otro de sus objetivos, es velar porque se siga la metodología al pie de la letra y de resolver cualquier duda que surja con esta, de mejorarla según el equipo vea la necesidad o de adaptarla a los casos en los que PSP-TSP no tiene información.

Conjunto con el coach, el líder del equipo, en nuestro caso el director de proyectos, y el equipo de desarrollo, se hace entonces un seguimiento y análisis semanal al desarrollo y desempeño de la metodología y del equipo en los proyectos. Este análisis semanal se hace en la junta semanal, está junta sigue un orden del día y no debe demorar más de 1 a 2 horas.

En la junta semanal, se informa acerca de comentarios por parte de la gerencia que puedan afectar el proyecto, se revisa el cumplimiento de los objetivos, metas y riesgos planteados desde la planeación, y se definen acciones preventivas o correctivas que se deben hacer para lograrlos.

Se revisa también el cumplimiento de las labores definidas para cada rol del equipo, éstos dan un estado general de sus funciones, dando prioridad a focos de alerta que pudiesen tener, para así entre todo el equipo definir como atacarlos para superarlos. Los principales roles son los de calidad y planeación que deben generar un reporte más completo acerca del estado general del proyecto. Por ejemplo el de calidad, expone el análisis de cómo va el cumplimiento del plan de calidad basado en las métricas de calidad que le arroja el Dashboard, haciendo énfasis en aquellas que necesitan prioridad urgente. Un ejemplo de esto basado en los proyectos que se siguieron, es una de las principales debilidades que hemos tenido durante el desarrollo es el Diseño detallado que se elabora en la fase de construcción. La debilidad consiste, en que no gastamos el tiempo

requerido en la revisión personal, la hacemos muy rápido, y todos los defectos que dejamos de encontrar en la revisión, nos los encuentran en la inspección, por lo que es alarmante la cantidad de defectos encontrados en las inspecciones y el porcentaje de defectos encontrados y corregidos en las revisiones es realmente bajo. Por esto se tomaron medidas en las que el administrador de calidad se encargara sin excepción alguna que todas las revisiones cumplan a cabalidad su tiempo y de revisar que los checklist personales se estén actualizando continuamente, y se vio reflejado el resultado, al encontrar mejoras en éstas métricas.

Luego de revisar los focos en el desarrollo de las funciones de los roles, se pasa a revisar el estado del proyecto con respecto al cronograma. Esta revisión consiste en que cada miembro del equipo informe acerca de su porcentaje de avance actual en su cronograma con respecto al planeado. En esta revisión si un miembro del equipo se atrasó informa a que se debió y cuál es su plan para ponerse al día. Cuando cada uno presenta su reporte también alerta sobre si puede o no cumplir con el porcentaje de avance planeado para la próxima semana, y cuanto porcentaje realmente cree pueda avanzar. Informa además si necesita algún tipo de ayuda y quien se la puede brindar, o si necesita re balancear su carga de trabajo debido a que su asignación perfectamente la puede efectuar otro miembro del equipo sin tener tantos problemas como él.

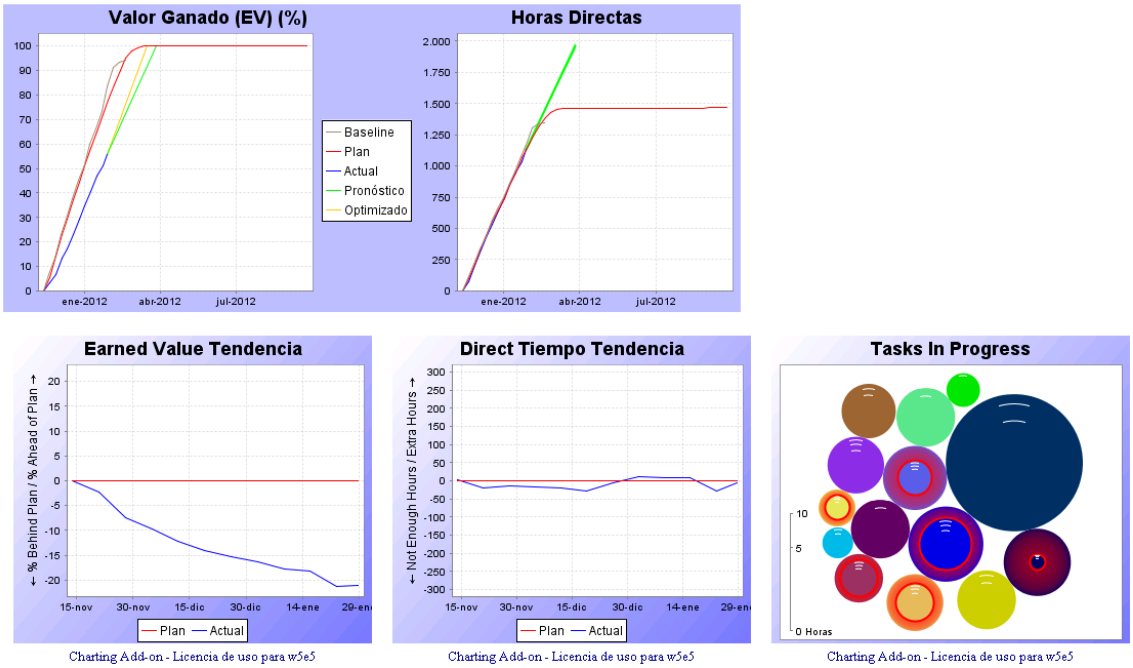
Luego de revisar el reporte de cada uno, se reporta el avance en porcentaje completo del proyecto, es decir, del equipo con respecto a la planeación, y se toman medidas de ser el caso en que se esté incumpliendo con el plan.

Al finalizar entonces se crea un acta de la junta, donde se establecen los temas tratados, si quedaron pendientes, se asignan los responsables y estos se revisan en la próxima junta.

A lo largo de la junta el coach sirve como guía en la resolución de dudas, o de inconvenientes que puedan surgir, tanto con la metodología como del proyecto y da ideas para mejorar el proceso y cumplir con los objetivos, metas y mitigar los riesgos. El coach queda con los datos del equipo y se encarga de analizarlos, y presentar comentarios en la próxima junta.

En la siguiente foto se muestran las principales gráficas que se revisan del proyecto: El valor ganado, que es el porcentaje de avance del grupo del proyecto con respecto a lo planeado; Las horas directas trabajadas, que corresponden al tiempo total consumido actualmente; Tendencia del valor ganado a lo largo de todo el proyecto, Tendencia de las horas directas de trabajo y Las actividades que actualmente se encuentran Abiertas.

Gráfica 1. Valor Ganado, Horas Directas, Valor Ganado Tendencia, tareas en Progreso.



Fuente: Documentación Pragma S.A. – Process Dashboard

Así pues se le hace el seguimiento necesario al proyecto para que este se pueda cumplir, y no llevarse sorpresas al final, adelantando posibles problemas dándoles manejo preventivo y de ser el caso correctivo tratando siempre de estar lo más cerca posible a la línea base inicial del cronograma planeado para el proyecto.

4. CONCLUSIONES

La práctica empresarial cumplió el objetivo general de ccomplementar la formación académica y elevar el nivel de competitividad, aplicando los conocimientos y destrezas adquiridos durante la carrera, sobre prácticas de desarrollo y métodos de desarrollo de software con calidad. Se logró certificar el curso Personal Software Process dictado por parte de la Universidad Carnegie Mellon Software Engineerin Institute para implementar la metodología en Pragma. Además se implementó la metodología TSP para equipos certificados con PSP. Esto permitió poner a prueba todos los conocimientos adquiridos en la universidad, que fueron pilar fundamental para cumplir las funciones adquiridas en el desarrollo de la práctica con estas metodologías.

La implementación de la metodología TSP-PSP en Pragma S.A., fue el principio de la búsqueda de una solución de los problemas de calidad, tiempo y costo que afectan a una empresa desarrolladora de software. Por ahora, estos problemas no se pueden considerar solucionados completamente, puesto que los proyectos en los que se han participado trabajando con esta metodología fueron los proyectos pilotos en incorporarla al proceso de desarrollo de software de Pragma, y donde lo más importante es aprender a trabajar con esta metodología.

Aunque en tiempo no se muestra mejora por ahora, si hay una notable mejora en el costo y la calidad, en temas como el manejo del proyecto, en el proceso de desarrollo del software y principalmente en la calidad del software. Respecto a la productividad del equipo, al principio decae, primordialmente mientras se toman como hábitos las técnicas y procesos de TSP-PSP.

Las mejoras en el manejo del proyecto, se deben a la manera como TSP realiza el seguimiento y manejo al proyecto desde el momento de la planeación, puesto que es el equipo de desarrollo, es decir, los ingenieros de proyectos, quienes hacen el plan de todo el proyecto y se comprometen a cumplirlo, es decir, no son las personas comerciales, o directores los que realizan este plan, como se hace normalmente, donde llegan luego donde los ingenieros y estos se dan cuenta que lo que han planeado estas personas está totalmente desfasado de la realidad. El equipo entonces es el que se hace responsable del plan y genera una cantidad de compromisos y planes que se deben ir cumpliendo a medida que se desarrolla el proyecto, siempre contando con ayuda, acompañamiento y guías del coach y del director del proyecto.

Además el manejo se hace fácil porque a medida que se avanza en el proyecto, con la información valiosa que se va obteniendo en cuanto a desempeño y estado, sirve para ser una base para predecir aspectos importantes para completar el proyecto, haciendo modificaciones el plan inicial de manera que se pueda cumplir con los objetivos, metas y fechas propuestas.

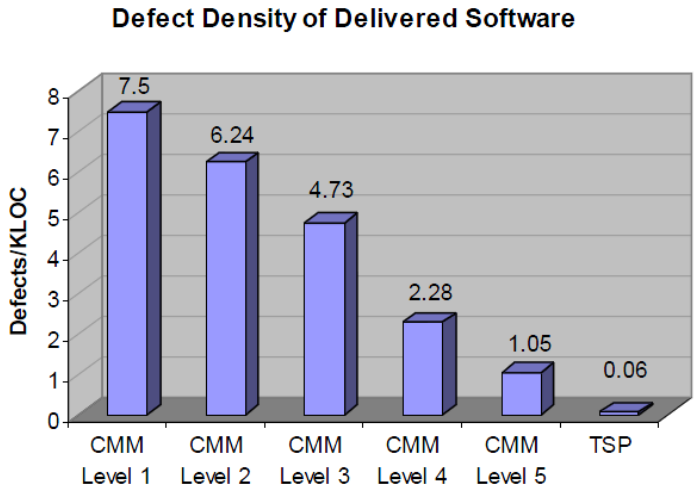
Las mejoras en el proceso de desarrollo de software son las más vistosas por los ingenieros de proyectos, puesto que se nota la diferencia de trabajar con una metodología rigurosa, a trabajar solo por cumplir de una manera desorganizada. Con la metodología PSP la dedicación del ingeniero es a la excelencia, a realizar el trabajo con calidad desde el inicio y hacerlo de manera organizada, siguiendo estrictamente el proceso a través de las fases, en tiempo y calidad.

Atado a las mejoras en el proceso vienen entonces las mejoras en la calidad del producto y del costo, como se pudo apreciar durante todo el informe de práctica, esta metodología TSP-PSP apunta a eso, la calidad como principio fundamental y el costo como principio implícito. Y se ve reflejado en el proceso, con todas las revisiones personales, inspecciones y pruebas que se llevan a cabo al software, una cantidad notable de filtros, para que al final el cliente no encuentre más de 1 solo defecto por mil líneas de código, que es la meta a la que apunta la metodología, según estudios que han realizado sus creadores, el Software Engineering Institute. Así se está ahorrando mucho tiempo de correcciones a defectos que encuentra el cliente, reflejado en dinero.

A futuro cercano, la realización de estos primeros proyectos servirá para que las planeaciones de otros proyectos sean más acertadas, y cercanas a la realidad, mejorándolos en tiempo, calidad y costo.

En la siguiente gráfica se presenta los resultados de trabajar con TSP según el Software Engineering Institute, comparado con Capability Maturity Model Integration CMMI, uno de las metodologías de manejo y desarrollo de software más conocidas en la industria.

Gráfica 2. Resultados TSP. Defectos por cada Mil Líneas de código.



Fuente: Software Engineering Institute, Technical Report 2003-014

Con PSP produciremos productos de calidad superior, obteniendo el mejor desempeño personal de los ingenieros, requiriendo que estas personas sean comprometidas con el liderazgo, los procesos y la ayuda.

Con TSP, los ingenieros trabajaremos juntos, disponibles a trabajar juntos efectivamente en un ambiente laboral de grupo y sabiendo como producir consistentemente productos de calidad.

Trabajar con una metodología como PSP-TSP, es hacer una transformación de la cultura organizacional de una empresa, a eso es lo que apunta PRAGMA S.A, una transformación a una cultura organizacional donde la calidad sea el principio fundamental y atado a este, el tiempo y costo de los proyectos desarrollados, donde la calidad sea muy alta, a tiempo y costo bajo. Meta que lleva tiempo, dinero y habilidades en la mejoras del proceso, que se darán con la continuidad en el trabajo con estas metodologías.

5. RECOMENDACIONES

Basado en la experiencia adquirida y las ofertas laborales actuales en la industria de desarrollo de software, tengo sólo una recomendación para la Corporación Universitaria Lasallista, la cual consiste en brindar a los estudiantes la posibilidad de un curso en el que se enseñe una metodología específica de desarrollo de software con calidad, como lo es la Personal Software Process.

En México por ejemplo, en la universidad nacional autónoma de México, los estudiantes de Ciencias de Computación tienen dentro de su Pensum la oportunidad de ver el Curso de PSP. Esto es un plus importante para los estudiantes que recién salen graduados de la universidad, puesto ya tienen conocimiento y manejo, principalmente de una metodología de desarrollo con calidad.

Muchas de las ofertas actuales en el mercado laboral de desarrollo de software valoran que los aspirantes a estos puestos relacionados con el desarrollo, tengan conocimiento y manejo de éstas metodologías, puesto que la mayoría de empresas del rubro compiten cada vez más duro por entrar a un negocio mundial donde lo importante es hacer los productos con mucha calidad, en tiempos y costos bajos.

Actualmente el Servicio Nacional de Aprendizaje SENA, imparte estos cursos en convenio con Federación Colombiana de la Industria de Software y Tecnologías Relacionadas FEDESOFT, por lo que el acercamiento a esta institución puede ser un buen punto de partida para que se imparta por medio de convenios cursos de estos tipos en la Corporación Universitaria Lasallista.

En cuanto a recomendaciones a PRAGMA, también hay una, está es relacionada con el proceso de comunicación y flujo de trabajo que se lleva a cabo internamente entre específicamente 2 departamentos de la compañía, la Agencia de Publicidad y la fábrica de software, puesto que la agencia genera muchos de los insumos que se requieren en la fábrica de software para el desarrollo de los portales web, como lo es el corte HTML de la línea gráfica definida por el cliente. Esta comunicación no se hace de forma clara, y a veces toca intervenir directamente entre directores para manejar asuntos muy pequeños, la recomendación entonces es que la compañía defina un proceso claro y estándar para la comunicación entre estos dos departamentos, donde su cooperación mutua es muy importante para el desarrollo de los portales web que realiza la compañía.

También hay otra recomendación para PRAGMA. Con la metodología PSP y TSP se pretende mejorar cada día más los procesos internos de la compañía,

pero esta mejora en muchos temas, en muchas ocasiones no se hace, como lo expuse por ejemplo en la recomendación anterior. Estas metodologías son claras al informar que para que sean 100% efectivas, deben estar los procesos claramente definidos, en PRAGMA hay unos definidos pero otros no, por lo que deben mejorarse y en la compañía muchas veces, se predica pero no se aplica, lo que hace difícil que la mejora sea continua. La recomendación entonces radica, en que basados en la información y en las experiencias que se han obtenido hasta ahora implementando estas metodologías, se mejoren los procesos de forma clara y contundente donde se encuentren falencias, con el fin de alcanzar la meta definida por parte de la compañía, ser una empresa de clase mundial.

BIBLIOGRAFÍA

EPISERVER. Editor's Manual for EPiServer CMS 6.0 Rev C, Sweden, 2011. 134p.

HUMPHREY, Watts S., PSP A Self Improvement Process for Software Engineers, 6 Edition, Addison Wesley, 2011.

HUMPHREY, Watts S., The Personal Software ProcessSM (PSPSM), Unlimited distribution subject to the copyright, Carnegie Mellon Software Engineering Institute, 2000. Capítulos 1 y 3.

MICROSOFT. Introducción a Visual Studio [en línea]. http://msdn.microsoft.com [Citado el 20 Enero de 2012].

PRESSMAN, Roger, Ingeniería del software, quinta edición, McGraw Hill. 2001. Capítulo 1.

TAMURA, Shurei, Integrating CMMI and TSP/PSP: Using TSP Data to Create Process Performance Models, Software Engineering Measurement and Analysis, Unlimited distribution subject to the copyright, Software Engineering Institute, 2000. Capitulo 1.

WIKIPEDIA, Calidad [en línea], http://es.wikipedia.org/wiki/Calidad [Citado el 28 Enero de 2012].

WIKIPEDIA, CMS [en línea], http://es.wikipedia.org/wiki/Cms [Citado el 28 Enero de 2012].

WIKIPEDIA, HTML [en línea], http://es.wikipedia.org/wiki/Html [Citado el 30 Enero de 2012].

WIKIPEDIA, Optimizador de motores de búsqueda [en línea], http://es.wikipedia.org/wiki/Posicionamiento_en_buscadores [Citado el 30 Enero de 2012].