Claudia Leticia Tapia Urbina.

PSP/TSP, SCRUM e ISO 9126 y 90003 en combinación como herramientas de trabajo para la construcción en pequeños y medianos proyectos de software.

Universidad Autónoma de Querétaro
Facultad de Ingeniería

Nombre de la tesis

PSP/TSP, SCRUM e ISO 9126 y 90003 en combinación como herramientas de trabajo para la construcción en pequeños y medianos proyectos de software.

Tesis
Que como parte de los requisitos para obtener el grado de

Maestro en Ingeniería de Calidad

Presenta

Claudia Leticia Tapia Urbina

Santiago de Querétaro Octubre del 2009

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Maestría en Ingeniería de Calidad.

PSP/TSP, SCRUM e ISO 9126 y 90003 en combinación como herramientas de trabajo para la construcción en pequeños y medianos proyectos de software.

Que como parte de los requisitos para obtener el grado de

Maestro en Ingeniería de Calidad

Presenta:
Claudia Leticia Tapia Urbina.

Dirigido por:
Ma. Del Carmen Espino Gudiño.

SINODALES

Dra. Ma. Del Carmen Espino Gudiño
Presidente

Firma

M.C. Vicente Rodríguez Hernández
Secretario

Firma

Dra. Rebeca del Rocío Peniche Vera
Vocal

Firma

M.C. Benjamín Zúñiga Becerra
Suplente

Firma

Dr. Gilberto Herrera Ruíz
Suplente

Firma

Dr. Gilberto Herrera Ruíz
Director de la Facultad

Dr. Luis Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Septiembre del 2009
México

RESUMEN

Esta investigación presenta un análisis de metodologías para el aseguramiento de la calidad en los productos y servicios de software. Con base en este análisis se determino que existe la necesidad de brindar mejores productos y servicios de ingeniería de software, como resultado se propone una combinación de metodologías y estándares que garanticen la calidad de proyectos de software en pequeñas y medianas empresas. Esta propuesta consiste en llevar a cabo actividades desde la planeación del proyecto hasta la construcción y pruebas del producto de software. Para lograr el objetivo, se seleccionaron PSP/TSP y SCRUM como herramientas de gestión y seguimiento de calidad en proyectos de software, la norma ISO 9126 para definir la calidad de los productos y la norma ISO 90003 como apoyo y/o complemento en la definición del modelo de calidad. Estas herramientas fueron seleccionadas debido a que para la mediana y pequeña empresa no representan un costo fuera de su alcance y garantizan que las actividades inherentes a la ingeniería de software tengan calidad desde un inicio y ,por ende, que los resultados sean productos y servicios de calidad. Un óptimo resultado en el desarrollo de software es lograr que los ajustes y cambios sean menores al 10%, siendo este uno de los resultados obtenidos en este proyecto. Además, con esta investigación y la aplicación de la combinación metodológica propuesta los resultados fueron satisfactorios de acuerdo a la funcionalidad, consistencia y validación del ISO 9126. Se obtuvo un ahorro del 33% en inversión de horas y se empleo el 8.8% del total de horas en la corrección de defectos y fallas. Dado que las expectativas de la ingeniaría de software son la calidad medible, mejorable y alcanzable, se requiere de una flexibilidad y control apropiados sobre el proceso de desarrollo de software. En suma, las metodologías ágiles mencionadas son altamente recomendables debido a la flexibilidad de ajuste a cambios constantes obteniendo beneficios en el incremento de la productividad y en la elaboración de estimaciones más precisas para elevar la competitividad de los equipos de trabajo.

(Palabras clave: calidad, proyecto, metodología, estándar).

SUMMARY

This research work presents an analysis of methodologies used for assuring quality in software products and services. Based on this analysis it was determined that there is a need to offer better software products and engineering services. A combination of methodologies and standards that guarantee quality software projects for small and medium-sized business is therefore proposed. This proposal consist of carrying out activities from the order to achieve this objective, we chose PSP/TSP and SCRUM as tools for the administration and follow-up of quality in software projects, the ISO 9126 standard to define product quality and the ISO 90003 standard as a support and/or complement in the definition of the quality model. These tools were selected because they guarantee the quality of the inherent activities of the software engineering right from the start, thus guaranteeing that quality products and services will be the result. An optimal result in software development is to achieve adjustments and changes below 10%; this is one of the results obtained in this project. In addition, using this research and the application of the proposed methodological combination, results were satisfactory regarding the functionality, consistency and validation of ISO 9126. A savings of 33% was obtained in hours invested and 8.8% of the total hours were used in the correction of defects and deficiencies. Give that the expectations for software engineering are measurable, improvable and reachable quality, adequate flexibility and control over the process of software development are needed. In summary, the agile methodologies mentioned are highly recommendable due to flexibility in adjustments to constant change, thus obtaining benefits in the form of increased productivity and the making of more precise estimates in order to raise the competitiveness of the work equipment.

(**Key words**: Quality, project, methodology, standard)

AGRADECIMIENTOS

Este trabajo de tesis tiene impreso el esfuerzo, apoyo y colaboración de más de dos personas y espero que si alguno no lee su nombre aquí no lo interprete más que como un descuido de mi parte, porque sin duda alguna su aportación fue de gran valor para mí.

Agradezco de infinita manera al Ingeniero Luis Guillermo Delucio por todo su apoyo, colaboración y aportación, por todas las horas que ocupo de su tiempo en apoyarme y ayudarme a sacar adelante este proyecto de tesis y la maestría. Sin tu apoyo y porras muchas cosas no serian posibles.

Agradezco a la Dr. María del Carmen Espino por compartir su conocimiento y tiempo para lograr los objetivos de este trabajo.

Agradezco al Maestro Vicente por orientarme y sembrar la semilla de la que surgió todo este trabajo.

Agradezco a la Dr. Rebeca Peniche por no quitar el dedo del renglón y empujar a que este trabajo llegara a su fin en el tiempo debido.

Agradezco a mis padres y hermanos por todo el apoyo que me brindaron, por sus porras y aliento a lo largo de los dos años que estuve en la maestría. Gracias por ayudarme a terminar todo lo que empiezo.

Gracias a los sinodales por su tiempo y por su apoyo, por participar en el cierre de este ciclo de dos años y ayudarme a ponerle el broche de oro.

Gracias a Dios, que siempre ha estado y sé que estará a mi lado.

ÍNDICE

Página

ÍNDICE DE FIGURAS

Figura Página

ÍNDICE DE GRAFICAS

Gráfica Página

ÍNDICE DE TABLAS

I. INTRODUCCIÓN

El desarrollo de la informática y la computación es relativamente joven comparado con otras áreas de la ingeniería. Más aun la ingeniería de software tiene solo unas cuantas décadas de haberse concebido como tal, por lo que en un principio, los pioneros en el área no tenían ni la más remota idea, de lo que varias décadas después se podría lograr con una computadora de un décimo de tamaño menor. De igual forma, se tuvieron que crear paradigmas y mejores prácticas para la ingeniería de software, la informática y la computación.

Pese al surgimiento de nuevas tendencias, corrientes y paradigmas, modelos, estándares y demás; las actividades básicas para el desarrollo de un producto de software han prevalecido:

- Definición del proceso de desarrollo.
- Administración del proyecto de desarrollo.
- Descripción del producto a desarrollar.
- Diseño del producto a desarrollar.
- Desarrollo del producto.
- Pruebas sobre el producto.

Entre las décadas de los 80s y de los 90s, se reconoció la falta de una disciplina que se ocupara de la forma en la que se desarrollaba un producto de software. Fue durante este tiempo cuando surgieron estándares de grandes organizaciones tales como la IEEE, el SEI, ISO y la OMG, entre los más destacados. Todos los esfuerzos apuntan a un objetivo en común: lograr que la ingeniería de software sea un ingeniería en toda la extensión de la palabra; lograr producir un producto, medible y por tanto mejorable, un producto de calidad desarrollado con calidad.

La Calidad del Software es “la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados

y con las características implícitas que se esperan de todo software desarrollado profesionalmente" (Pressman, 2002).

La calidad del producto de software se diferencia de la calidad de otros productos de fabricación industrial, ya que el software tiene ciertas características especiales: el software es un producto mental, no restringido por las leyes de la Física o por los límites de los procesos de fabricación; se desarrolla, no se fabrica; el software no se deteriora con el tiempo; el software, en la mayoría de los casos, se construye a medida, es decir, no existen moldes que den la misma forma a dos aplicaciones; el mantenimiento del software es mucho más complejo que el mantenimiento del hardware; el software con errores no se rechaza. Se asume que es inevitable que el software presente errores.

Las distintas actividades para la implantación del control de calidad en el desarrollo de software son:

1. Aplicación de metodología y técnicas de desarrollo,
2. Reutilización de procesos de revisión formales,
3. Prueba del software,
4. Ajustes a los estándares de desarrollo,
5. Control de cambios, mediciones y recopilación de información; y
6. Gestión de informes sobre el control de calidad (Pressman, 2002).

En la actualidad las organizaciones buscan ser capaces de desarrollar y, entregar software confiable, a tiempo y en presupuesto. Por esto las organizaciones adoptan una norma, estándar y/o modelo que pueda ayudarlas a conseguir su meta de calidad.

En los intentos por implantar se han creado propuestas tales como COMPETISOFT (Oktaba et al., 2009), que propone una forma de llevar proyectos de ingeniería de software basándose en la mejora (continua) de procesos, retomando las bases de CMMI, ISO/IEC 15540 solo por mencionar algunos de los elementos que la propuesta aborda. Pese a que esta propuesta fue elaborada con la intención de disminuir los costos de la implantación de modelos y estándares de calidad, deja de lado el hecho de que la implementación de modelos y estándares requiere la participación y sobre todo la concientización de los

involucrados. Sin duda alguna, la gente que está encargada de implementar la calidad es la que día a día desarrolla y trabaja en y con los productos que al final conformarán un solo producto de software, programa o aplicación.

También existe propuestas orientadas a la implementación de la calidad por medio de la implementación de procesos de prueba y estrategias robustos orientados a la mitigación de posibles errores retomando las funcionalidades implementadas en el producto de software desarrollado (Pérez, 2007). Sin embargo, estas propuestas no contemplan el hecho de que los equipos de aseguramiento de la calidad son menospreciados por los miembros del equipo que los ven como jueces injustos que emiten veredictos y calificaciones sobre su trabajo, ya que ellos, los miembros de equipo de QA, no está involucrados de forma directa con la creación de lo que evalúan (Correira, 2007).

Hay propuestas más ambiciosas que encaminan todos sus esfuerzos a proponer la implementación de CMMI en combinación con estándares de alto costo definiendo y describiendo un conjunto fundamental de procesos y productos, proporcionando guías para la selección de un ciclo de vida desde un conjunto de ciclos de vida comunes de varios modelos, describiendo la integración de procesos primarios dentro de un proyecto cohesivo. (Blakemore, 1996). Sin embargo, este esfuerzo por impregnar de calidad a la ingeniería de software, se desajusta a las necesidades de nuestro país. Ya que en el muy particular caso de México la calidad aún no es un concepto por el que las pequeñas y medianas empresas apuesten debido a los altos costos que implica. El costo que implica una certificación en CMMI sobre pasa las capacidades económicas de las llamadas PYMES mexicanas.

Dentro de todas las propuestas existentes para la inyección de calidad a los productos de software como PSP y TSP.

La implementación de PSP y TSP son de bajo costo, ya que la distribución de la información de estos procesos y sus talleres son gratuitos.

También existen propuestas como la filosofía ágil SCRUM, que es una herramienta poderosa y ayuda a los equipos de desarrollo a alcanzar sus objetivos y metas dentro del proyecto con la colaboración de cada uno de sus integrantes (Martin, 2004). Esta metodología de desarrollo en combinación con TSP y PSP puede dar como resultado una metodología de bases solidas y de inversión casi nula.

El núcleo de una empresa dedicada a la comercialización de software por grande o pequeña que esta sea es el equipo de desarrollo. Este núcleo es asistido por otros equipos que participan en las actividades de definición e implementación de entornos, documentación y descripción de las funcionalidades del producto de desarrollo. Implementando PSP y TSP se logra la integración de estos equipos para que trabajen en armonía y por un objetivo en común. Con PSP y TSP los participantes del equipo tienen posibilidades de jugar más de un rol dentro del equipo de trabajo. Lo anterior crea una empatía tanto por el trabajo de los demás integrantes como por los integrantes mismos. Esto al final conduce al equipo de trabajo a marchar en una misma dirección entendiendo las necesidades dentro del proyecto de cada uno de sus integrantes y teniendo plena conciencia de la importancia de que su trabajo sea realizado con calidad.

El objetivo de cualquier empresa o negocio es la creación de capital. Además de este objetivo universal de cualquier empresa, una empresa dedicada al desarrollo de software es conseguir un producto en plazo, costo y calidad adecuado mediante la organización y especialización de funciones (Piattini, 2003). La propuesta de este trabajo de tesis está orientada en este último objetivo, que sin lugar a dudas ayudara para alcanzar al primero. Puesto que PSP, TSP y SCRUM no representan un costo que desembolsar, lo único que queda de cuenta es la implementación de estos en combinación con un estándar, para este trabajo de tesis serán las normas ISO 9126 e ISO 90003, para asegurar la calidad del producto de software.

II. ESPECIFICACIONES DEL PROYECTO

Dado que la competencia cada día es mayor, es necesario que las empresas se preocupen en ofrecer mejores productos y servicios. La complejidad de los problemas que hoy en día buscan una solución en el software ha aumentado de manera considerable. Este crecimiento ha sobrepasado la habilidad de desarrollar y mantener con calidad garantizada el software por parte de las organizaciones que se dedican a ello.

Las organizaciones buscan ser capaces de desarrollar y, entregar software confiable, a tiempo y en presupuesto. Aunado a lo anterior se debe de considerar la importancia que tiene cada una de las aplicaciones que es desarrollada. Se debe de tomar conciencia y responsabilidad de las consecuencias que un defecto en un producto de software podría ocasionar. Algunos defectos del software pueden ocasionar serios daños y hasta perjudicar físicamente a personas. Imaginemos el riesgo que lleva usar una aplicación defectuosa en el ambiente de la medicina o en la aeronáutica.

Cada defecto representa un costo adicional. Un error identificado en fases tempranas o en la misma fase donde se produjo es mucho más barato de resolver que el mismo defecto en una fase posterior o cuando el producto ha sido puesto en producción.

Las siguientes razones son importantes para implementar un sistema de calidad desde el punto de vista de una empresa que quiere una posición en el mercado:

- Satisfacción del cliente.
- Competencia.

En la entidad de Querétaro existen medianas y pequeñas empresas dedicadas al desarrollo de software. Dentro de sus esfuerzos por obtener productos de software de calidad se han adoptado estándares y metodologías. En muchos casos estos intentos tienen fines como la creación de un proceso enorme y rígido que pocos proyectos pueden ejecutar, o procesos ligeros que se prestan a la ambigüedad y a las interpretaciones personales.

Basándose en estas observaciones, surge como una necesidad lograr un producto de y con calidad sin que esto implique un desgaste y un costo desmesurado en el intento.

2.1 Objetivo General

Proponer una estrategia para que las personas involucradas en un proyecto de desarrollo de software en empresas medianas y pequeñas formen equipos de trabajo altamente efectivos a un bajo costo con la finalidad de desarrollar productos de software de calidad.

2.2 Objetivos específicos

- Proponer una metodología de desarrollo de software apoyada en los estándares ISO 9126 y 90003 que asegure la calidad de los productos creados a lo largo de la ingeniería de software así como la del producto final.
- Proponer una metodología de bajo costo para que las empresas medianas y pequeñas puedan garantizar la calidad de sus productos.
- Aplicar la metodología propuesta a un proyecto de software para verificar su efectividad.

2.3 Hipótesis

La presente propuesta se desarrolla bajo la hipótesis de que es posible crear una estrategia de trabajo para que los equipos desarrolladores de software incrementen la calidad de sus productos.

III. REVISIÓN DE LA LITERATURA

A diferencia de otras áreas de la ingeniería donde la calidad ha incursionado (mecánica, civil, hidráulica, química, eléctrica, electrónica, etc.) y en las cuales se obtiene un producto tangible y medible; la ingeniería de software genera productos intangibles y con mediciones que, en la mayoría de los casos, caen en lo subjetivo.

El desarrollo de un producto de software depende de factores tan variables como: la experiencia del desarrollador, el tiempo, el costo y, sin lugar a duda los requerimientos del cliente. No solo en tiempo y costo sino también en funcionalidad.

El empleo de la computadora para el desarrollo de soluciones tecnológicas de software es relativamente reciente comparado con el resto de las áreas en donde la calidad ha incursionado como una poderosa herramienta de perfeccionamiento en la elaboración y desarrollo de productos.

Hoy en día existe un sin número de temáticas en la ingeniería de software y algunas de las más interesantes se mencionan a continuación.

3.1 Un acercamiento de la ingeniería de calidad a los factores humanos que afectan la fiabilidad del software en el proceso de diseño

Kazuhiro Esaki et al., (2002) en su artículo “A Quality Engineering Approach to Human Factors Affecting Software Reliability in Design Process” habla de cómo los errores humanos afectan la confiabilidad del software y plantea la posibilidad de disminuir los errores y defectos por medio de un proceso de diseño de software.

Los autores de este artículo se plantean elementos tales como el grado de entendimiento de la tecnología, el grado de entendimiento del requerimiento (en negocio) y el entendimiento de la metodología utilizada para diseñar, entre otros, como elementos relevantes en la obtención de software confiable por medio de un DOE.

3.2 Calidad en el desarrollo y mantenimiento del software

Mario Piattini et al., (2003) en su libro recopila trabajos de varios proyectos de investigación relacionados con la calidad de software en el desarrollo y el mantenimiento del mismo. Dada la naturaleza de su trabajo, su libro ofrece una visión amplia sobre los diferentes factores que deben ser tomados en cuenta a lo largo de la construcción del software para obtener como salida un producto de calidad.

A lo largo del libro el autor retoma temas que van desde la introducción al concepto de calidad hasta el planteamiento de métricas para medir y controlar la calidad en los productos de software.

3.3 Ingeniería de software: una perspectiva orientada a objetos

Eric Braude, (2003) en su libro aborda el tema de la ingeniería de software no solo como un concepto, sino que refiere al ¿Cómo? desarrollar la ingeniería de software en todas y cada una de las etapas que la comprenden dentro del paradigma orientado a objetos.

El autor proporciona información del proceso de ingeniería de software, de la organización de proyectos de ingeniería de software, del análisis de requerimientos, del diseño del producto de software, del desarrollo del producto de software y por último aborda el tema de pruebas sobre el producto de software.

3.4 Prácticas de la ingeniaría de software. ¿Es la calidad de software incompatible al tiempo del mercado? Una entrevista con Steve Cross

Steve Cross (2003) en su entrevista con el Doctor Stephen E. Cross (CEO y Director del Software Engineering Institute (SEI)) habla acerca de las razones que, desde su punto de vista, son las causas de que el software haya incrementado su costo en las últimas décadas.

Durante la entrevista el Gurú plasma la idea de que la calidad en el software está directamente relacionada a los métodos utilizados para el desarrollo de un producto de software.

También se toca el tema de las constantes demandas de nuevas funcionalidades de software que hace que éste tenga un tiempo de vida muy limitado en el mercado y en el gusto y preferencial de la gente.

3.5 Las 7 C's para crear software viviente: una perspectiva de la investigación para la Ingeniería del Software orientado a la Calidad

Mehmet A (2003) en su artículo propone lo que él define como las 7 C's para desarrollar software orientado a la calidad. Las 7 C's son: Selección y/o creación de los productos a desarrollar (Concern-oriented processes), Modelos que permiten la adecuación de determinadas partes ó productos. (Canonical models), Composable models (Capacidad de ajustar partes), Capacidad de decidir qué partes del software son verificables, certificables, controladas (Certifiable models), Para crear un producto de software, éste debe ser posible de construir (Constructible models), Capacidad de diseñar un controlador estable, solido que provea un conjunto de operaciones controlables (Closure property of models) y Las propiedades de los productos deben de ser monitoreados y debe ser posible su control (Controllable models).

Este artículo propone una forma de resolver los problemas que se presentan a los diferentes integrantes del equipo del proyecto de ingeniería de software por medio del desarrollo y/o selección de productos adecuados a lo largo del proceso de ingeniería de software y de su debido seguimiento, monitoreo y control con la finalidad de lograr un producto de calidad como resultado final.

3.6 Equipos de aseguramiento de la calidad – menospreciados, pero raramente entendidos

Edward J. Correira (2007) en este artículo aborda la problemática a la que se enfrenta el equipo de Aseguramiento de la Calidad cuando el proyecto presenta problemas de retrasos de fechas, por poner un ejemplo. Además, resulta interesante ya que plasma como una fotografía la situación actual de las áreas dedicadas al aseguramiento de la calidad, en las cuales hay necesidad de justificar el tiempo que se llevan los participantes de esta área en desarrollar tal o cual tarea. El autor del artículo plantea soluciones tales como la comunicación efectiva con los demás miembros del equipo (desarrolladores, lideres, administradores), el entendimiento del los alcances del proyecto, el entendimiento del requerimiento y el conocimiento del negocio; por solo mencionar algunos de los consejos que se plasman en el artículo.

3.7 Análisis y diseño de sistemas de información

James A. Senn (1999) se enfoca al desarrollo de software involucrando a todas las áreas definidas por la ingeniería de software pero principalmente a la de análisis del requerimiento y a la de diseño; e incluye al usuario final del sistema como parte importante durante el desarrollo del mismo.

El esfuerzo del autor está orientado a no perder de vista que el producto de software es, en la mayoría de las veces, utilizado por personas que usan la computadora y el software que reside en ella como una herramienta de trabajo y que no son expertos en el uso de la misma. Es por ello que se destaca continuamente la importancia de crear aplicaciones listas para usarse, pensando en el cliente o usuario final de las mismas.

3.8 Pruebas del Software

El Dr. Macario Polo (2006) en su curso de pruebas del software hace referencia a los distintos tipos de pruebas que existen y en qué etapa del desarrollo del software pueden ser aplicadas.

Hace una definición muy clara de cada uno de los subtipos de pruebas que conforman las pruebas de caja negra y de caja blanca. Hace también mención de cómo pueden ser usados estos tipos de pruebas y algunas de las aportaciones que cada tipo de pruebas puede traer a la ingeniería de software en donde son aplicadas.

También hace referencias puntuales a los modelos, estándares y procesos que plantean la fase de pruebas como parte de ellos, tales como CMM e ISO/IEC 12207.

3.9 La implementación del proceso de revisión de requerimientos

Ernesto Kiszkurno (2007) planteó la importancia de la implementación de revisiones de requerimientos durante el desarrollo de un producto de software como medio de detección de fallas en etapas tempranas; hecho que ayuda a reducir en gran medida el costo (el ponente dio a conocer que el 50% de los defectos de un producto son generados en la fase de requerimientos) que tiene una falla en fases finales del proyecto de software.

El autor también planteó puntos y aspectos relevantes y que no deben de ser pasados por alto durante la revisión de los requerimientos tales como la consistencia de la información, la claridad y facilidad de lectura en los requerimientos, objetivos claros, entre otras cosas.

Todo esto en conjunto con un proceso de pruebas bien llevado puede garantizar como resultado una aplicación adecuada a las necesidades del cliente final.

3.10 La lista de los 10 puntos para la reducción de defectos de software

Barry Boehm et al., (2001) hacen hincapié en el salto que debe dar la ingeniería de software de prácticas de moda pasajera a prácticas de ingeniería a través de la derivación, organización, y diseminación de datos empíricos sobre el desarrollo de software y del fenómeno que representa. En resumen, estos son los 10 puntos:

1. Buscar y corregir un problema en el software después de ser entregado es frecuentemente 100 veces más caro que buscarlo y corregirlo en las etapas de levantamiento de requerimientos y diseño.
2. Proyectos de software actuales gastan cerca del 40 al 50 porciento de sus esfuerzos en re trabajo evitable.
3. Cerca del 80 porciento del re trabajo evitable proviene del 20 porciento de los defectos.
4. Cerca del 80 porciento de los defectos proviene del 20 porciento de los módulos y cerca de la mitad de los módulos está libre de defectos.
5. Cerca del 90 porciento de los periodos de inactividad por fallas, viene de la menos del 10 porciento de los defectos.
6. Las inspecciones filtran 60 porciento de los defectos.
7. Las inspecciones basadas en perspectivas filtran un 35 porciento más defectos que las inspecciones sin dirección.
8. Prácticas de disciplina personal reducen las tasas de introducción de defectos hasta en un 75 porciento.
9. Estando todas las otras cosas iguales, cuesta 5 porciento mas por instrucción de código desarrollar productos de software altamente dependientes, que desarrollar productos de software poco ó con baja dependencia. Sin embargo, la inversión es más costosa si el proyecto cubre costos significativos de operación y mantenimiento.
10. Cerca del 40 al 50 porciento de los programas de usuario contienen defectos pocos o no triviales.

Los autores hacen una invitación en expandir la comunidad y los esfuerzos de reducción de defectos en los productos de software.

3.11 Estudio comparativo de los modelos y estándares de calidad de software

Scalone (2006) en su tesis presenta un acercamiento a las diferentes metodologías y estándares conocidos en el ámbito de la industria del software.

De su trabajo los puntos más destacables son los que hacen referencia a CMM, Modelo que hace su objetivo la madurez del proceso de software. La Madurez del Proceso de Software es aquello en el que un proceso específico es definido, administrado, medido y controlado. Modelo Bootstrap, metodología cuyo interés primario fijo en la evaluación y mejora de la capacidad de las Unidades Productoras de Software con la finalidad de mejorar el proceso de software mediante el uso de estándares reconocidos y prácticas de la ingeniería de software. PSP proceso medido y diseñado para ser usado por un ingeniero de software con la finalidad de mejorar el proceso por medio de la mejora personal. Este proceso involucra planear, medir y administrar la calidad del producto para mejorar el trabajo personal de quien lo aplica. TSP proceso enfocado en construir y guiar equipos altamente efectivos en la elaboración de productos de software con calidad; para lograr lo anterior, previamente se debe de entrenar y formar a los integrantes del equipo que deben ser preferentemente de 3 a 15 ingenieros; entre otros.

Al final de su trabajo, la autora reflexiona acerca de la importancia de la calidad en el desarrollo del software. Así mismo hace hincapié en la necesidad de inculcar el concepto de "calidad" a todo el personal involucrado en la creación y desarrollo del producto de software; evaluar y controlar los resultados de la empresa y establecer a la mejora continua como parte de la empresa.

3.12 Mejores prácticas en la administración de proyectos SCRUM y el desarrollo de software de forma ágil con XP

Martin et al., (2004) en este artículo hace una descripción de la situación organizacional que llevo a la empresa Primavera Systems a implementar la filosofía ágil SCRUM y la metodología XP como parte de su forma de administrar y llevar los proyectos de la compañía.

En este trabajo, él autor pone de manifiesto la situación de la compañía antes y después de la implantación de la combinación SCRUM y XP. Antes de la llegada de SCRUM y XP a Primavera Systems se dio comienzo por motivar al personal con cambios

en los espacios de oficina, reuniones y programas de motivación y que pese a todos estos esfuerzos los resultados del equipo seguían siendo malos, a tal grado que la compañía en la última liberación de uno de los productos que ésta ofrecía, se tomó la decisión de eliminar los bonos anuales de todo el equipo por malos resultados en la entrega. Esta acción resultó en el desanimo y desmotivación del equipo del proyecto ya que sentían que sus esfuerzos y sacrificios no eran redituados.

Con la implantación de SCRUM el equipo vio sus mejoras mes a mes, los puestos gerenciales y directivos veía la mejora; se terminaron los tiempos extras y los fines de semana invertidos en la liberación del proyecto de software. Juntos, SCRUM y XP ayudaron al equipo y a la compañía a prever sobre las acciones y acontecimientos sobre el proyecto. Así mismo, frente a los clientes de la compañía, la entrega y calidad del producto había mejorado.

3.13 El libro de conocimientos del proceso personal de software SM (PSPSM), versión 1.0

Promeroy-Huff et al., (2005) presenta uno de los libro que contiene unas de las principales temáticas y conceptos que componen las áreas de conocimiento y competencias del PSP y las 7 áreas de competencia que presenta se numeran en seguida. Dichas áreas de competencia a su vez contienes áreas de conocimiento:

Área de competencia 1: Conocimiento básico.

1.1 Definición del proceso
1.2 Elementos del proceso
1.3 Estadísticas

En esta área de competencia se definen los conceptos básicos así como los elementos que debe de ser parte del los procesos ya las habilidades necesarias que permitirán diseñar productos de calidad a tiempo y dentro del presupuesto.

Área de competencia 2: Conceptos básicos de PSP

2.1 Fidelidad del proceso

2.2 Recolección de datos

2.3 Análisis de datos

2.4 Mejora del proceso

En esta área de conocimiento se introduce el concepto “fidelidad del proceso”. En esta área de conocimiento se relaciona éste concepto al grado con el que es seguido el proceso y como este apego al proceso se ve reflejado en la calidad final del producto. Se aborda el análisis de datos y la toma de decisiones en base a estos análisis en pro de la mejora del proceso.

Área de competencia 3: Tamaño de las Medidas y estimaciones

3.1 Tamaño de medidas

3.2 Tamaño de datos

3.3 Tamaño de principios de estimación.

3.4 Proxies

3.5 El método de estimación PROBE

3.6 Combinación de estimaciones

3.7 Directrices para los tamaños de estimación

Esta área de conocimiento se presenta los objetivos de las métricas, el criterio por seleccionar una métrica y los sistemas de contabilidad de las métricas. También se da una introducción a los principios de estimación y se presenta el método PROBE.

Área de competencia 4: Haciendo y dando seguimiento a los planes del proyecto.

4.1 Principios de planeación PSP

4.2 El marco de planeación de PSP

4.3 Tamaño de software y esfuerzo

4.4 Planeación de tareas y agendas

4.5 Seguimiento de agenda con valor ganado

4.6 Planeación y seguimiento de temas

En esta área de conocimiento se presenta la relación que hay entre el tamaño del software y la cantidad de esfuerzo para producirlo. Así mismo, se introduce a los principios de planeación y al uso del método PROBE para la generación de estimaciones sobre los recursos.

Área de competencia 5: Planeando y dando seguimiento a la calidad del software

5.1 Principios de calidad PSP
5.2 Medidas de calidad
5.3 Métodos de calidad
5.4 Revisión de código PSP
5.5 Revisiones de diseño PSP
5.6 Temas de revisión

En esta área de conocimiento se presentan el marco de calidad en el que se basa PSP. Se hace una introducción a los métodos de revisión de código y diseño con la intensión de que el proceso definido sea eficaz y eficiente en la detección y reparación de errores. Se presentan también los conceptos de revisiones personales y medición de producto y proceso.

Área de competencia 6: Diseño de software

6.1 Principios de diseño de software
6.2 Estrategias de diseño
6.3 Calidad de diseño
6.4 Documentación de diseño
6.5 Templetes de diseño
6.6 Verificación de diseño

Esta área de conocimiento está enfocada en su totalidad al diseño del software. En ella se presenta lo que debe de ser documentado y los principios a seguir dentro del diseño dejando abierto el cómo debe de hacerse.

Área de competencia 7: Extensiones del proceso y ajustes.

7.1 Definiendo un proceso personal a la medida

7.2 Evolución del proceso

7.3 Aplicaciones de avances del proceso

7.4 Responsabilidades profesionales

En esta área de conocimiento se aborda la definición de proceso a la medida de las personas que los van a utilizar y a ejecutar. También se aborda la mejora del proceso por medio de la evolución del proceso: el proceso puedo ajustarse de acuerdo a las experiencias de los participantes cuando lo ejecutan.

3.14 Guía de conocimientos comunes para la certificación de probador de software versión 6.1.1

El Instituto de aseguramiento de la calidad (Quality Assurance Institute) (2006) presenta en su guía para la certificación como probador de software conceptos de calidad que abarcan desde la definición de calidad desde el punto de vista del proveedor del servicio y desde el punto de vista del cliente final hasta la definición de características de calidad como usabilidad flexibilidad, portabilidad entre otras.

Esta guía también contempla lo relacionada a la creación de planes de prueba, los roles involucrados en las pruebas de software, ambientes de pruebas y herramientas de pruebas. En esta guía no se contempla a ninguna otra área diferente a la de pruebas, sin embargo se hace notar en que para ser un buen probador de software se requiere de experiencia en el desarrollo de proyectos de software para que las actividades de pruebas agreguen valor al proyecto en el cual éstas son desarrolladas.

De la presente revisión de literatura se puede concluir que el desarrollo de trabajos alrededor del tema de la calidad del software es cada vez más extenso y variado. Sin embargo, en su mayoría los trabajos apuntan al hecho de que la calidad no es solo tema de los directivos de las empresas sino de todos y cada uno de los que participan en el

desarrollo y producción de los productos que se generan a lo largo de la ingeniería de software.

Dentro de los trabajos revisados se pudo observar que no hay ninguno que lance una propuesta que abarque la ingeniería de software en su totalidad así como la carencia de conceptos como flexibilidad y cambio continuo, conceptos que día a día son más frecuentes en el demandante mercado del software.

IV. FUNDAMENTACIÓN TEORICA.

4.1 Introducción

El desarrollo de software cuenta con estándares, normas y metodologías enfocadas en guiar en la creación de productos con ciertas características y calidad.

Una problemática que se presenta como constante en la sociedad mexicana es la dificultad de los individuos para trabajar en equipo. Hecho que se refleja en varios campos de la sociedad como el deportivo, el académico y por supuesto en el laboral. En la ingeniería de software es necesario que el trabajo en equipo se dé de forma natural y que no sea un obstáculo a salvar.

Otra problemática es el costo de implementar la calidad en la ingeniería de software. Existen modelos costosos en su implementación y que salen del alcance de las pequeñas empresas, no solo por su costo sino también porque han sido diseñadas para macro empresas. Tal es el caso de CMMI, que fue diseñado para empresas que pueden costear appraisals y certificaciones de costos elevados, los cual deja fuera a la gran mayoría de las pequeñas y medianas empresas mexicanas.

Es por ello que esta propuesta de tesis se apoya en PSP/TSP, SCRUM e ISO 9126 y 90003 con la finalidad de unificar el concepto de calidad a lo largo de la vida de un proyecto de desarrollo de software, reforzar el trabajo en equipo y explotar al máximo las capacidades, habilidades y conocimientos de los integrantes del equipo de trabajo y con ello hacer más competitivo al equipo a un costo bajo.

4.2 Ingeniería e Ingeniería de software

El Consejo de Acreditación de Ingeniería y tecnología (ABET, Accreditation Board of Engineering and Technology) define a la ingeniería como la *profesión en la que el conocimiento de las ciencias naturales y matemáticas obtenido con el estudio, la experiencia y la práctica se aplica con juicio para desarrollar formas de utilizar, de modo económico, los materiales y fuerzas de la naturaleza para beneficio de la humanidad* (Braude, 2003). La ingeniaría de software es un tipo de ingeniería y por ende tiene el mismo conjunto de responsabilidades que las demás.

El IEEE ha desarrollado una definición más completa para ingeniería del software: "(1) La aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento del software; es decir, la aplicación de ingeniería al software. (2) El estudio de enfoques en (1)" (Pressman, 2002).

Pressman (Pressman, 2002) caracteriza la Ingeniería de Software como "una tecnología multicapa", ilustrada en la Figura 1.

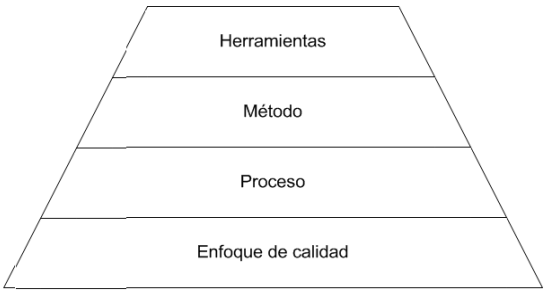


Figura 1 Capas de la Ingeniería de Software.

Dichas capas se describen de la siguiente manera:

- Cualquier disciplina de ingeniería debe descansar sobre un esfuerzo de organización de calidad (enfoque de calidad). La cultura de mejora continua de procesos conduce al desarrollo de enfoques cada vez más robustos para la ingeniería del software.
- El proceso define un marco de trabajo para un conjunto de áreas clave, las cuales forman la base del control de gestión de proyectos de software y

establecen el contexto en el cual se aplican los métodos técnicos, se producen resultados de trabajo, se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.

- Los métodos de la ingeniería de software indican cómo construir técnicamente el software partiendo desde el análisis de requisitos o requerimientos, incluyendo el diseño, la construcción, las pruebas y el mantenimiento.
- Las herramientas de la ingeniería del software proporcionan un soporte para el proceso y los métodos.

Dado lo anterior se puede concluir que Ingeniería de Software es la rama de la ingeniería que aplica los principios de la ciencia de la computación y las matemáticas para lograr soluciones a los problemas de desarrollo de software de forma económica.

4.3 Calidad de software

La definición de la palabra calidad puede resumirse como la propiedad o conjunto de propiedades inherentes a una cosa que permiten apreciarla o distinguirla con respecto a los restantes de su especie.

La Calidad del Software es “la concordancia con los requerimientos funcionales y de rendimiento explícitamente establecidos, con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente” (Pressman, 2002).

Se puede decir que los requisitos del software son la base de las medidas de calidad y que la falta de concordancia con los requisitos es una falta de calidad. Los estándares o metodologías definen un conjunto de criterios de desarrollo que guían la forma en que se aplica la Ingeniería del Software. Si no se sigue ninguna metodología siempre habrá falta de calidad. Todas las metodologías y herramientas tienen un único fin producir software de alta calidad.

A la hora de definir la calidad del software se debe diferenciar entre la calidad del Producto de software y la calidad del Proceso de desarrollo. Sin embargo, las metas que se establezcan para la calidad del producto van a determinar las metas a establecer para la calidad del proceso de desarrollo, ya que la calidad del producto va a estar en función de la calidad del proceso de desarrollo.

4.4 Aseguramiento de la calidad

El aseguramiento de calidad del Software es la revisión de los productos y documentación relacionada con el software para verificar su cobertura, corrección, confiabilidad y facilidad de mantenimiento. El aseguramiento de calidad incluye la garantía de que una aplicación de software cumple con las especificaciones y los requerimientos para su uso y desempeño deseados (Senn, 1999).

Este aseguramiento debe de comenzar a ser desarrollada desde el comienzo del desarrollo de la aplicación y no después. El aseguramiento de la calidad del software engloba un enfoque de gestión de calidad, métodos y herramientas, revisiones técnicas formales aplicables en el proceso de software, una estrategia de prueba, el control de la documentación del software y de los cambios realizados, procedimientos para ajustarse a los estándares de desarrollo del software y mecanismos de medición y de generación de informes.

Las revisiones del software son un "filtro" para el proceso de Ingeniería del Software y deben de ser aplicadas en varios momentos del desarrollo del software y sirven para detectar errores y defectos que pueden ser eliminados antes de que lleguen a ser fallas dentro del producto final.

4.5 Herramienta de software

Una herramienta es cualquier dispositivo que mejora el desempeño de una tarea cuando se emplea de forma adecuada. El uso de herramientas puede mejorar la efectividad y eficiencia con la que son desarrolladas las aplicaciones del software. Al mismo tiempo,

las herramientas de software beneficia la calidad del sistema bajo desarrollo. Las herramientas se clasifican en herramientas de alto nivel, que hacen referencia a las tareas de análisis y diseño, y herramientas de bajo nivel, que son las que apoyan a la conversión de los diseños en código (Senn, 1999)

4.6 Modelo de proceso de software

El modelo de proceso de software se define como una representación simplificada de un proceso de software. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real.

Los modelos “genéricos” no son descripciones definitivas de procesos de software; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software.

La tabla 1 muestra una comparativa entre los modelos más conocidos en el desarrollo de proyectos de software. En los siguientes párrafos se abordaran las características principales de cada modelo en la tabla.

El modelo en cascada se caracteriza por que una fase no comienza hasta que termine la fase anterior, en las fases se incluye las correcciones a todo lo que haya sido detectado en la o las fases previas, y después de cada fase se obtienen como resultado documentos que deben ser aprobados por el usuario

El modelo en cascada consta de las siguientes fases (Figura 2):

- Definición de los requisitos.
- Diseño de software.
- Implementación y pruebas unitarias.
- Integración y pruebas del sistema.
- Operación y mantenimiento.

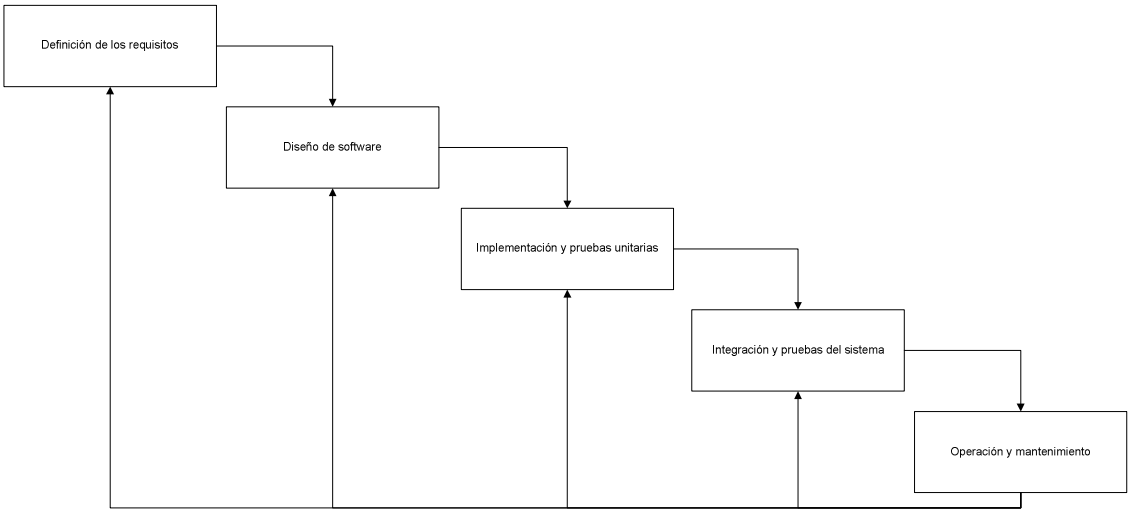


Figura 2 Modelo de proceso de desarrollo en cascada.

En la práctica es difícil que este modelo se ejecute de forma lineal; por lo general involucra varias iteraciones e interacción entre las distintas fases de desarrollo. Este modelo tiene como principal contra los siguientes puntos:

- Las iteraciones son costosas e implican re trabajo debido a la producción y aprobación de documentos.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados, olvidados o corregidos de forma paliativa.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos.
- Es difícil responder a cambios en los requisitos.

En el modelo de desarrollo en espiral el ciclo de desarrollo se representa como una espiral, en lugar de una serie de actividades sucesivas con retrospectiva de una actividad a otra.

Cada ciclo de desarrollo se divide en cuatro fases:

- Definición de objetivos: Se definen los objetivos. Se definen las restricciones del proceso y del producto. Se realiza un diseño detallado del plan administrativo. Se identifican los riesgos y se elaboran estrategias alternativas dependiendo de estos.

- Evaluación y reducción de riesgos: Se realiza un análisis detallado de cada riesgo identificado. Pueden desarrollarse prototipos para disminuir el riesgo de requisitos dudosos. Se llevan a cabo los pasos para reducir los riesgos.
- Desarrollo y validación: Se escoge el modelo de desarrollo después de la evaluación del riesgo. El modelo que se utilizará (cascada, sistemas formales, evolutivo, etc.) depende del riesgo identificado para esa fase.
- Planificación: Se determina si continuar con otro ciclo. Se planea la siguiente fase del proyecto.

Este modelo toma en consideración explícitamente el riesgo, esta es una actividad importante en la administración del proyecto. El ciclo de vida inicia con la definición de los objetivos. Después se identifican los riesgos. Se evalúan los riesgos y se desarrolla un poco el sistema. Por último se planifica la siguiente fase.

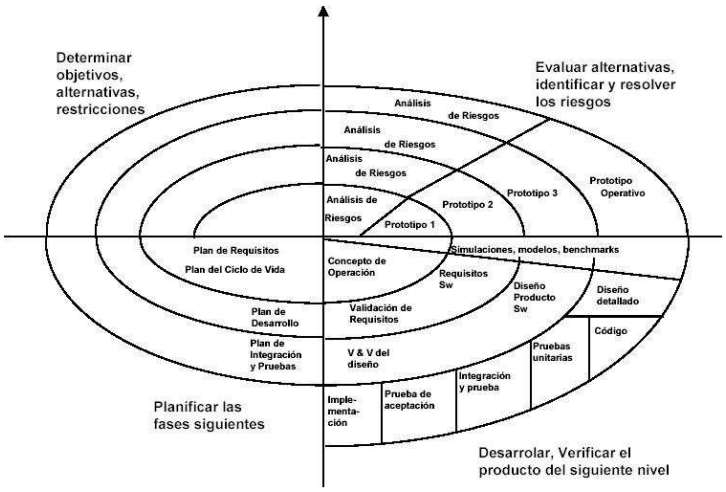


Figura 3 Modelo de proceso de desarrollo en espiral.

El modelo iterativo se propone como una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.

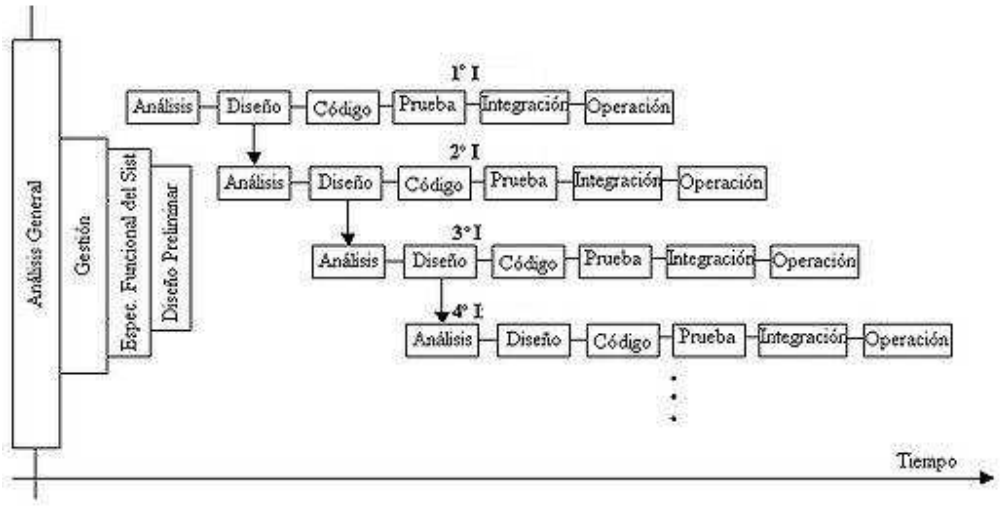


Figura 4 Modelo de proceso de desarrollo iterativo.

Entre las ventajas del modelo iterativo se encuentran:

- Los clientes no esperan hasta el fin del desarrollo para utilizar el sistema. Pueden empezar a usarlo desde la primera iteración.
- Los clientes pueden aclarar los requisitos que no tengan claros conforme ven las entregas del sistema.
- Se disminuye el riesgo de fracaso de todo el proyecto, ya que se puede distribuir en cada iteración.
- Las partes más importantes del sistema son entregadas primero, por lo cual se realizan más pruebas en estos módulos y se disminuye el riesgo de fallos.

4.7 Metodología

Un proceso de software detallado y completo suele denominarse “Metodología”. Las metodologías se basan en una combinación de los modelos de proceso genéricos. Adicionalmente una metodología debería definir con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas, guías de adaptación de la metodología al proyecto, guías para uso de herramientas de apoyo, etc.

	Cascada	Espiral	Iterativa
Definición del proceso.	Requerida	Requerida	Requerida
Producto Final	Determinado durante la planeación	Determinado durante la planeación	Fijado durante el proyecto
Costo del proyecto	Determinado durante la planeación	Parcialmente variable	Fijado durante el proyecto
Fecha de finalización	Determinado durante la planeación	Parcialmente variable	Fijado durante el proyecto
Sensibilidad al ambiente	Planeado solamente	Planeado primariamente	Al final de cada iteración
Flexibilidad del equipo, creatividad.	Parecido a un recetario	Parecido a un recetario	Parecido a un recetario
Transferencia de conocimientos	Entrenamiento previo al proyecto	Entrenamiento previo al proyecto	Entrenamiento previo al proyecto
Probabilidad de éxito.	Bajo	Medio-bajo	Medio

Tabla 1Comparativa de modelos de proceso de software.

La comparación y/o clasificación de metodologías no es una tarea sencilla debido a la diversidad de propuestas y diferencias en el grado de detalle, información disponible y alcance de cada una de ellas. Si se toma como criterio las notaciones utilizadas para especificar artefactos producidos en actividades de análisis y diseño, es posible clasificar las metodologías en dos grupos: Metodologías Estructuradas y Metodologías Orientadas a Objetos. Pero si se considera su filosofía de desarrollo, aquellas metodologías con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Metodologías Pesadas. Las denominadas Metodologías Ágiles, están más orientadas a la generación de código con ciclos muy cortos de desarrollo, se dirigen a equipos de desarrollo pequeños, hacen especial hincapié en aspectos humanos asociados al trabajo en equipo e involucran activamente al cliente en el proceso.

4.8 Proceso de desarrollo de software

El proceso de desarrollo de software se define como el orden en el que se realizan las actividades de desarrollo (Braude, 2003).

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas.

A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos:

- Especificación de software: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
- Diseño e Implementación: Se diseña y construye el software de acuerdo a la especificación.
- Validación: El software debe validarse, para asegurar que cumpla con lo que quiere el cliente.
- Evolución: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Otra manera de determinar los elementos del proceso de desarrollo de software es establecer las relaciones entre elementos que permitan responder Quién debe hacer Qué, Cuándo y Cómo debe hacerlo. Así las interrogantes se responden de la siguiente forma:

- Quién: Las personas participantes en el proyecto de desarrollo desempeñando uno o más roles específicos.
- Qué: Un artefacto es producido por un rol en una de sus actividades. Las herramientas apoyan la elaboración de artefactos soportando ciertas notaciones.
- Cómo y Cuándo: Las actividades son una serie de pasos que lleva a cabo un rol durante el proceso de desarrollo. El avance del proyecto está controlado mediante hitos que establecen un determinado estado de terminación de ciertos artefactos.

4.9 Proceso ágil

La ingeniería de software es una disciplina joven que busca fundamentar sus procedimientos en principios de ingeniería probados. La comunidad de ingeniería de software inspirada en Deming (Deming, 1986) y Juran (Juran, 1988) ha determinado que se requiere una buena capacidad de desarrollo del proceso para obtener un producto de alta calidad. Estándares de proceso como el ISO 9000 y la integración de modelos de madurez de la capacidad (CMMI) se desarrollaron para alcanzar resultados más previsibles mediante guías para incorporar procedimientos probados en los procesos. Las compañías que han adoptado los estándares ISO 9000 y CMMI han logrado mejoras sustanciales.

El proceso de adopción por las empresas, de los estándares mencionados, inspiró la aparición de varias técnicas que refinan a las normas en la última década del siglo XX y en los primeros años del siglo XXI. Primero surgió PSP (Watts S., 1995) que proporciona los elementos necesarios para el trabajo disciplinado de un desarrollador individual; en seguida apareció TSP (Watts S., 1999) que agrega los elementos necesarios para planificar y controlar el trabajo de un equipo de desarrolladores (Laurie, 2000). Dicha forma de trabajo es una práctica sugerida en los métodos ágiles.

Los métodos ágiles son una reacción a las formas tradicionales de desarrollar software y el reconocimiento de la necesidad de una alternativa a los procesos pesados de desarrollo de software, guiados por documentación. En la implementación mediante los métodos tradicionales, el trabajo comienza con la recopilación y documentación de un conjunto completo de requerimientos, seguida por el diseño de alto nivel, desarrollo e inspección.

El proceso ágil usa la documentación, pero no cientos de páginas que nunca se mantienen y rara vez se usan. Se planifica, pero se reconocen los límites de la planificación. Las prácticas adoptadas en los métodos ágiles se resumen en el manifiesto ágil (Beck, 2001) que establece:

“Se promueven mejores formas de desarrollo de software mediante práctica, ayudando a que otros las realicen. En las prácticas promovidas se valoran:

- ✓ Más a los individuos y su interacción, que al proceso y las herramientas.
- ✓ Más la obtención de software corriendo en una plataforma, que completar la documentación.
- ✓ Más la colaboración del cliente, que la negociación del contrato.
- ✓ Más la respuesta al cambio, que seguir un plan.

Damos mayor relevancia a los primeros elementos, pero reconocemos el valor de los elementos enunciados después.”

4.10 PSP

El PSP es un proceso de software definido y medido diseñado para ser usado por un ingeniero de software, como su nombre lo indica es personal (Figura 5). El PSP fue desarrollado por Watts Humphrey y tiene como objetivo guiar la planeación y desarrollo de los módulos de software o pequeños programas. Es una tecnología de SEI que trae disciplina a las prácticas de los ingenieros de software, mejorando la calidad del producto y reduciendo el tiempo del ciclo de desarrollo del software.

Figura 5 PSP.

El PSP está basado en principios de mejora de procesos, a diferencia de CMM, que se enfoca en la mejora organizacional.

Para fomentar el mejoramiento a nivel personal, PSP ofrece la administración y control del proceso de forma personal (Humphrey, 2000). Con PSP los ingenieros

desarrollan software usando una propuesta estructurada y disciplinada. Los ingenieros se ocupan de:

- Seguir un proceso definido,
- Planificar, medir y seguir su trabajo,
- Administrar la calidad del producto y
- Aplicar aspectos cuantitativos para mejorar los procesos de trabajo personales.

El proceso de PSP consiste de un conjunto de métodos, formularios y scripts que muestran cómo planificar, medir y administrar el trabajo. El PSP está diseñado para ser usado en cualquier lenguaje de programación o metodología de diseño, y puede ser usado en varios aspectos del trabajo de software, no solo en el desarrollo como tal. PSP utiliza un proceso CMM de nivel 5 para calcular el costo individual que se aplica en la mayoría de las tareas de desarrollo de software como lo son la definición de requerimientos, el diseño de la arquitectura, la documentación, las pruebas del sistema, el mantenimiento del sistema, etc.

PSP requiere de la participación de todos los niveles de la organización. Una estrategia efectiva es primero involucrar a los principales Ejecutivos y Gerentes para luego entrenar a los ingenieros. El PSP instruye a los ingenieros en la administrar la calidad de sus proyectos, en la mejora de las estimaciones y planificaciones; y en la reducción de los defectos de sus productos.

El diseño de PSP está basado en los siguientes principios de planificación y calidad:

- Todo ingeniero es diferente y para ser más efectivo, cada uno de ellos debe planificar su trabajo y debe basar sus planes en sus datos personales.
- Para mejorar su desempeño, los ingenieros deben usar procesos bien definidos y medidos.
- Para producir productos de calidad, los ingenieros deben ser responsables de la calidad de sus productos.
- Es menos costoso encontrar y arreglar los defectos de un proceso.
- Es más eficiente prevenir defectos que encontrarlos y arreglarlos.

El proceso de PSP tiene 7 fases o procesos:

1. PSP0: proceso base, registro de tiempos, registro de errores, estándar de tipo de errores.[Proceso personal de arranque]
2. PSP0.1: estándar de codificación, medición de tamaño, propuesta de mejoramiento del proceso (PIP).[Proceso personal de arranque]
3. PSP1: estimación del tiempo, reporte de pruebas.[Proceso personal de administración]
4. PSP1.1: planeación de actividades, planeación de tiempos.[Proceso personal de administración]
5. PSP2: revisión de codificación, revisión del diseño.[Proceso personal de calidad]
6. PSP2.1: formatos de diseño.[Proceso personal de calidad]
7. PSP3: desarrollo en ciclos.[Proceso cíclico]

4.11 TSP

El TSP fue desarrollado por Watts Humphrey en 1996. El objetivo era suministrar un proceso operacional que ayude a los Ingenieros hacer trabajos de calidad. El principal motivador para el desarrollo de TSP fue la convicción de que los equipos de ingenieros puedan hacer el trabajo de manera extraordinaria, pero solo si ellos son formados y entrenados. El objetivo del TSP es construir y guiar a los equipos. El desarrollo de sistemas es una actividad en equipo, y la efectividad del equipo determina la calidad de la ingeniería. En Ingeniería, los equipos de desarrollo tienen múltiples especialidades y todos los miembros trabajan en vista de un objetivo en común.

Los objetivos de TSP son:

- Ayudar a los equipos de ingeniería de software a elaborar
- Productos de calidad dentro de los costos y tiempos establecidos,
- Tener equipos rápidos y confiables; y
- Optimizar el performance del equipo durante todo el proyecto.

Para el uso de TSP, los desarrolladores de software deben ser entrenados primero en PSP. Al usar PSP los desarrolladores siguen un proceso personal definido y medido,

planifican el trabajo antes de hacerlo, reúnen datos acerca del tiempo, tamaño los defectos que insertan en el producto; y después utilizan estos datos para administrar el trabajo del personal y de esta forma asegurar la calidad de los productos que se desarrollan (Humphrey, 2000).

TSP es una manera de guiar a los integrantes del equipo en la utilización de métodos de trabajo en equipos efectivos. Los miembros del equipo deben tener roles, los cuales proveen un sentido de liderazgo y pertenencia. Los roles ayudan a los miembros del equipo a realizar sus trabajos, prevenir conflictos y establecer un grado de control respecto de su ambiente de trabajo. El sentido de control es un requerimiento fundamental para que los miembros estén motivados. La interdependencia es un elemento importante del equipo de trabajo. Esto significa que cada miembro del equipo depende del desempeño de los otros miembros.

La interdependencia mejora el desempeño individual debido a que los miembros pueden ayudarse.

Para ser efectivos, los equipos deben ser capaces de trabajar como unidades cohesivas. Los equipos efectivos tienen las siguientes características:

- El objetivo del equipo es importante, definido, visible y realista;
- Los recursos del equipo son adecuados al trabajo,
- Los miembros del equipo son motivados para alcanzar el objetivo del equipo,
- Los miembros del equipo cooperan entre sí y
- Los miembros del equipo son disciplinados en su trabajo.

El TSP está diseñado para establecer las condiciones que caracterizan a los equipos efectivos. Los principios para la construcción de un equipo utilizados en TSP para establecer estas condiciones son:

- Los miembros del equipo establecen objetivos en común y roles definidos,
- El equipo desarrolla una estrategia,

- Los miembros del equipo definen un proceso en común para su trabajo,
- Todos los miembros del equipo participan en la producción del planeamiento, y cada miembro conoce su rol en ese planeamiento,
- El equipo negocia el plan con la dirección,
- La dirección revisa y acepta el plan negociado y
- Los miembros del equipo se comunican de manera frecuente.

La formación de equipos efectivos requiere que los miembros entiendan qué y cómo hacer el trabajo; y que sus planes son alcanzables. Para hacer un trabajo disciplinado es necesario contar con “procesos operacionales” que definan cómo es realizado el trabajo. El proceso operacional es semejante a un script y es diseñado para ser usado por los miembros del equipo.

El TSP provee un proceso operacional definido que guía tanto a ingenieros como directivos en los pasos para la construcción de un equipo. Con un proceso definido y un plan que sigue ese proceso, los integrantes del equipo son eficientes. TSP provee los procesos operacionales necesarios para formar los equipos, establecer un ambiente de trabajo efectivo y guiar a los equipos en la realización del trabajo.

TSP es una serie de métodos que pueden ayudar a los equipos a desarrollar sistemas. PSP provee la disciplina ingenieril necesaria para utilizar un proceso definido, planificado y medido (Figura 6).

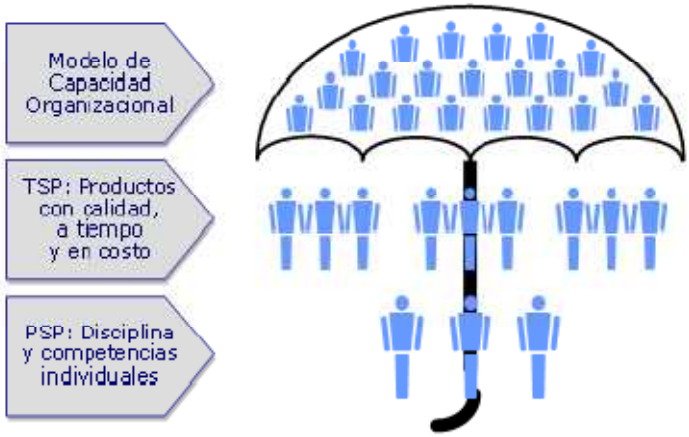


Figura 6 PSP / TSP.

4.12 ISO 9126

La norma ISO/IEC 9126 está enfocada a la calidad de Producto y consta de las siguientes partes:

Parte 1: Modelo de Calidad

Parte 2: Métricas externas

Parte 3: Métricas internas

Parte 4: Calidad en el uso de métricas

La especificación y la evaluación de la calidad de producto de software se pueden conseguir definiendo características de calidad apropiadas, tomando en cuenta el objetivo de uso del producto de software.

El modelo de calidad establecido en la primera parte del estándar, ISO 9126-1, clasifica la calidad del software en un conjunto estructurado de características y subcaracterísticas de la siguiente manera:

Funcionalidad - Un conjunto de atributos que se relacionan con la existencia de un conjunto de funciones y sus propiedades específicas. Las funciones son aquellas que satisfacen lo indicado o implica necesidades.

Idoneidad

Exactitud

Interoperabilidad

Seguridad

Cumplimiento de normas.

Fiabilidad - Un conjunto de atributos relacionados con la capacidad del software de mantener su nivel de prestación bajo condiciones establecidas durante un período de tiempo establecido.

Madurez

Recuperabilidad

Tolerancia a fallos

Conformidad de Fiabilidad

Usabilidad - Un conjunto de atributos relacionados con el esfuerzo necesitado para el uso, y en la valoración individual de tal uso, por un establecido o implicado conjunto de usuarios.

Aprendizaje

Comprensión

Operatividad

Atractividad

Conformidad de Usabilidad

Eficiencia - Conjunto de atributos relacionados con el vínculo entre el nivel de desempeño del software y la cantidad de recursos necesitados bajo condiciones establecidas.

Comportamiento en el tiempo

Comportamiento de recursos

Conformidad de Eficiencia

Mantenibilidad - Conjunto de atributos relacionados con la facilidad de extender, modificar o corregir errores en un sistema software.

Estabilidad

Facilidad de análisis

Facilidad de cambio

Facilidad de pruebas

Conformidad de facilidad de mantenimiento

Portabilidad - Conjunto de atributos relacionados con la capacidad de un sistema software para ser transferido desde una plataforma a otra.

Capacidad de instalación

Capacidad de reemplazamiento

Adaptabilidad

Co-Existencia

Conformidad de Portabilidad

La subcaracterística Conformidad no está listada arriba ya que se aplica a todas las características.

Cada subcaracterística está dividida en atributos. Un atributo es una entidad la cual puede ser verificada o medida en el producto software. Los atributos no están definidos en el estándar, ya que varían entre diferentes productos software.

Un producto software está definido en un sentido amplio como: modelos, diseños, código fuente, etc.

El estándar provee elementos para la definición de un modelo de calidad para el producto software. El estándar define dos tipos de métricas: las métricas internas y las métricas externas:

- Métricas internas son aquellas que no dependen de la ejecución del software.
- Métricas externas son aquellas aplicables al software en ejecución.

Idealmente, la calidad interna determina la calidad externa y esta a su vez la calidad en el uso. La calidad en las métricas de uso están sólo disponibles cuando el producto final es usado en condiciones reales.

ISO 9126 distingue entre fallos y no conformidad, siendo un fallo el no cumplimiento de los requisitos previos, mientras que la no conformidad afecta a los requisitos especificados. Una distinción similar es hecha entre la validación y la verificación.

4.13 ISO 90003

La norma ISO / IEC 90003:2004 es un checklist para la implementación de la norma ISO 9001 – 2000 de calidad en la ingeniería de software. Este checklist usa un esquema de clasificación de evidencia física de procedimientos, planes y auditorias. Este checklist define todos los puntos necesarios para cumplir con las mejores prácticas en la ingeniería de software.

Este checklist especifica lo que se requiere para alcanzar la conformidad con el estándar definido proporcionando una lista de evidencias que ayudará cualquier organización desarrolladora de software.

4.14 SCRUM.

SCRUM es una forma de auto-gestión de los equipos de programadores. Un grupo de programadores deciden cómo hacer sus tareas y cuánto van a tardar en ello. SCRUM ayuda a que trabajen todos juntos, en la misma dirección, con un objetivo claro. SCRUM es un proceso de desarrollo de software iterativo y creciente utilizado comúnmente en entornos basados en el desarrollo ágil de software.

SCRUM es un modelo de referencia que define un conjunto de prácticas y roles, y que puede tomarse como punto de partida para definir el proceso de desarrollo que se

ejecutará durante un proyecto. Los roles principales en SCRUM son el SCRUMMaster, que mantiene los procesos y trabaja de forma similar al director de proyecto, el ProductOwner, que representa a los stakeholders (clientes externos o internos), y el Team que incluye a los desarrolladores.

Durante cada sprint, un periodo entre 15 y 30 días (la magnitud es definida por el equipo), el equipo crea un incremento de software potencialmente entregable (utilizable). El conjunto de características que forma parte de cada sprint viene del Product Backlog, que es un conjunto de requisitos de alto nivel priorizados que definen el trabajo a realizar. Los elementos del Product Backlog que forman parte del sprint se determinan durante la reunión de Sprint Planning. Durante esta reunión, el Product Owner identifica los elementos del Product Backlog que quiere ver completados y los hace del conocimiento del equipo. Entonces, el equipo determina la cantidad de ese trabajo que puede comprometerse a completar durante el siguiente sprint (Schwaber, 2004). Durante el sprint, nadie puede cambiar el Sprint Backlog, lo que significa que los requisitos están congelados durante el sprint.

Existen varias implementaciones de sistemas para gestionar el proceso de SCRUM, que van desde notas amarillas "post-it" y pizarras hasta paquetes de software. Una de las mayores ventajas de SCRUM es que es muy fácil de aprender, y requiere muy poco esfuerzo para comenzarse a utilizar.

Aunque surgió como modelo para el desarrollo de productos tecnológicos, también se emplea en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad; situaciones frecuentes en el desarrollo de determinados sistemas de software.

V. METODOLÓGICA Y PROCESOS DEFINIDOS

5.1 Introducción

Una vez realizada la revisión de la literatura relacionada con el tema de la tesis, este trabajo sigue los pasos a continuación descritos (Figura 7):

Como parte de este trabajo de tesis y para probar que los equipos efectivos para un pequeño proyecto de desarrollo de software incrementan la calidad de sus productos apoyándose en el uso de PSP / TSP, SCRUM y el ISO 9126 y 90003 como herramientas de trabajo, se optó por desarrollar una aplicación de software a la medida enfocada al manejo de inventarios de una PYME dedicada a la comercialización de acero.

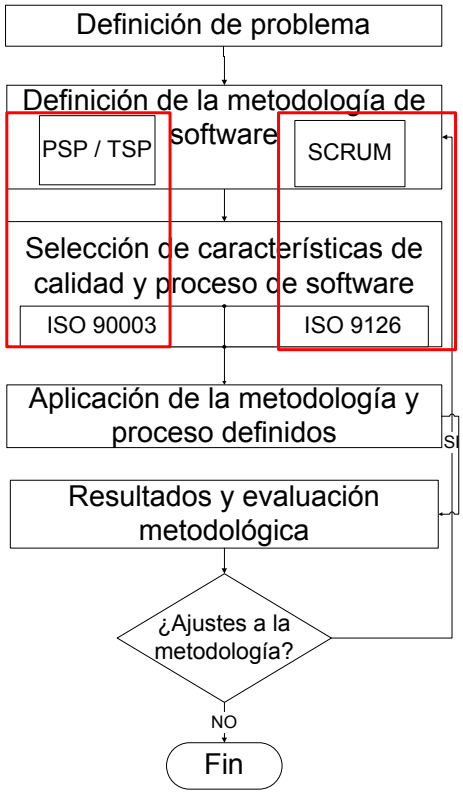


Figura 7 Flujo de trabajo para el desarrollo de la tesis.

5.2 Definición del problema

En esta actividad se toma una situación de la vida real que requiera y que pueda ser solucionada por medio del desarrollo de una aplicación de software. La palabra *problema* hace referencia a una necesidad detectada en una empresa o negocio que pueda ser solucionada por medio de la creación de una aplicación de software.

5.3 Definición de la metodología del software

La siguiente actividad consiste en seleccionar las características y puntos importantes tanto de PSP/TSP como de SCRUM. De esta selección se obtuvo como resultado el esqueleto, por así llamarlo, de la metodología: se definió el qué se debía hacer durante la ejecución de la ingeniería de software en cada una de sus etapas.

Al combinar PSP/TSP y SCRUM se obtiene una metodología que permite tener previsibilidad al identificar riesgos y ubicar controles, además de asegurar el seguimiento de un proceso propuesto debido a la autogestión que SCRUM proporciona y las bases sólidas en el desarrollo de productos elaborados con calidad y, sobre todo, en los plazos establecidos; lo anterior debido al entrenamiento en PSP/TSP que recibe el equipo de ingenieros que participan en el proyecto.

Para desarrollar esta propuesta se tomaron los puntos considerados medulares y comunes tanto de PSP/TSP y SCRUM para dar paso a una amalgama metodológica que permita implementar calidad desde las fases tempranas de un proyecto de software hasta sus últimas etapas de liberación y cierre. Esta amalgama se resume en la tabla 2.

Además de considerar las proposiciones de trabajos previos, existen cinco expectativas importantes de la ingeniería de software contemporánea de Humphrey que se deben especificar. La primera es decidir qué medidas y/o características de calidad se aplicarán al proyecto y al producto. La segunda es reunir los datos de todos los proyectos para formar una base que permita hacer estimaciones en los proyectos futuros. La tercera establece que todos los integrantes del equipo deben disponer con libertad y facilidad de

todos los requerimientos, diseños, códigos y materiales. La cuarta indica que todos los miembros del equipo deben seguir el proceso, (Braude, 2003). Y, la quinta es medir que el producto cumpla con la calidad predeterminada y por ende con la meta, ver Figura 8.

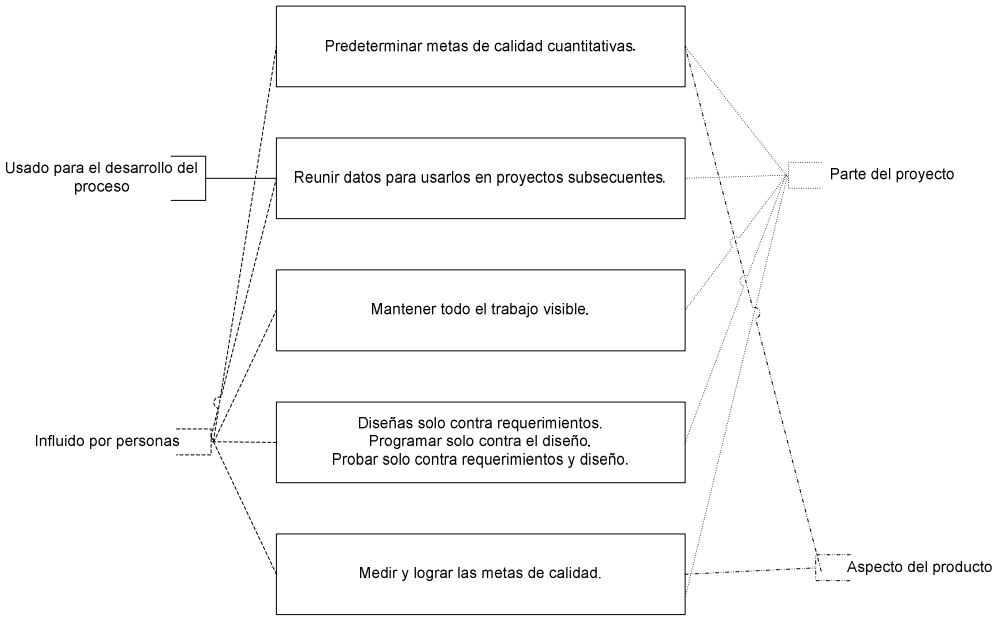


Figura 8 Expectativas de la ingeniería de software.

Dado que el proceso de desarrollo de sistemas es complicado y complejo, se requiere de una flexibilidad y control apropiado. La evolución favorece a las metodologías que operan bajo cambios constantes y tienen una optimización para adaptarse a ellos (Schwaber, 1997).

Después de establecer las cinco expectativas importantes, se propone hacer una división de tres grandes actividades retomado la metodología SCRUM: el “pre-juego”, el “juego” y el “post-juego” (Figura 9) (Schwaber, 1997).

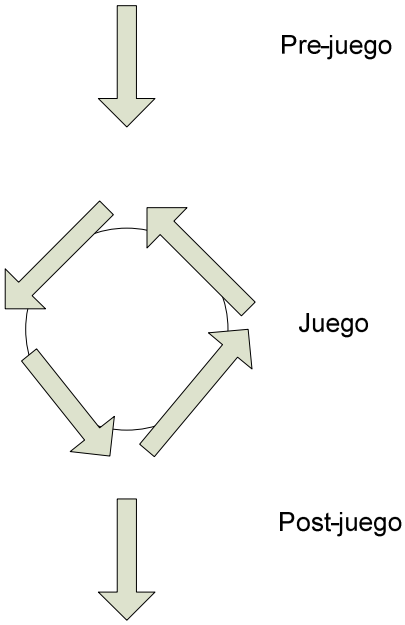


Figura 9 Presentación grafica de la propuesta metodológica.

La primer etapa de “Pre-juego” contempla actividades propias de la planeación del proyecto incluyendo la definición de objetivos, roles, equipos, estrategias y planes, el primer acercamiento a los requerimientos y a un diseño de alto nivel que incluye la arquitectura del futuro sistema a muy grosso modo.

La segunda etapa del “Juego” incluye actividades propias del desarrollo del producto: el análisis y administración de requerimientos, diseño de bajo nivel, desarrollo, revisiones/inspecciones de los productos creados, correcciones y ajustes a los productos creados, concluir la creación de los productos, aplicar las pruebas al producto (de integración, de sistema y todas las establecidas en las estrategias y planes). Todas estas actividades deben de ejecutarse de acuerdo a los Sprints definidos en el “Pre-juego” y las juntas de seguimiento de los Sprints deben de ejecutarse de acuerdo a las reglas del SCRUM (periodicidad, duración, participantes, ubicación).

Y, la tercer etapa del “Post-juego” incluye actividades como la revisión del trabajo del equipo para asegurar el cumplimiento de los objetivos del proyecto, del equipo y el cumplimiento de los requisitos de calidad y tiempo del proyecto, la preparación de la entrega del producto y la entrega del mismo así como la recopilación de las impresiones del

cliente. Como puede apreciarse, la metodología plasmada en este trabajo de tesis es iterativa.

Las actividades básicas requeridas para la construcción del software han permanecido estables a pesar del advenimiento de nuevas tendencias. El desarrollo de software en el mundo real, por lo general depende de una demandante lista de características así como de estrictas fechas de entrega determinadas por el cliente y/o el mercado, (Braude, 2003).

De forma gráfica, en la figura 12 puede apreciarse la estructura de TSP. Para definir la propuesta metodológica se retomaron actividades de TSP y de SCRUM (Figura 13), además de sugerir una definición de 3 roles básicos (Tabla 2 y Tabla 3), retomados de las definiciones de roles de PSP/TSP (Figura 10) y SCRUM (Figura 11), para la ejecución y desarrollo de las actividades de la metodología (Figura 14). Estos roles fueron complementados con otros auxiliares para dar soporte a cada una de las actividades de la metodología propuesta (Tabla 4).

Es necesario destacar que ninguno de los roles definidos tiene implícito el papel de jefe. Fue definido de esa forma con la intención de que el trabajo en equipo se dé de forma natural. Es decir, cada uno de los integrantes tiene la responsabilidad de convertirse en un miembro activo del equipo que debe de contribuir al proyecto para que éste avance en un ambiente de trabajo armonioso. Lo anterior es en gran parte contribución de PSP.

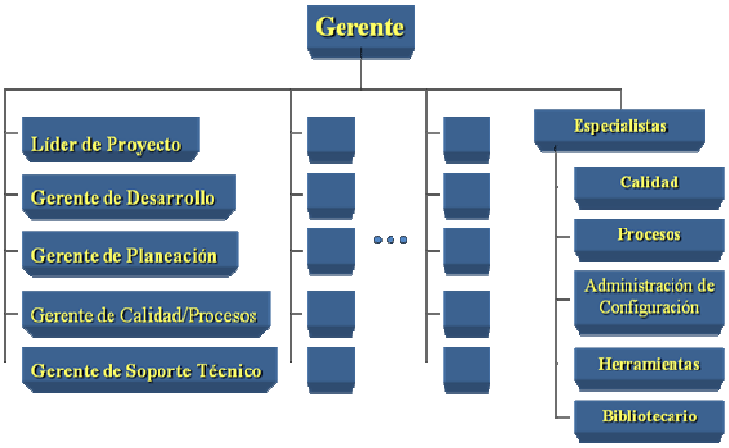


Figura 10 Roles PSP/TSP.

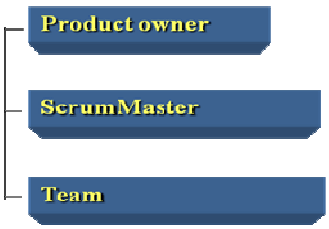


Figura 11 Roles SCRUM.

ROLES		
SCRUM	**PSP/TSP**	**Metodología**
Producto Owner	Líder	Propietario del producto
	Administrador de la interfaz con el cliente	
SCRUM Master	Administrador del proceso	Facilitador de procesos
Equipo	Miembros del equipo	Equipo implementador
Stakeholders		Involucrados con el proyecto

Tabla 2 Empalmamiento de roles para la definición de los roles básicos de la metodología.

ROLES	
Metodología	**Actividades principales del rol**
Propietario del producto	Dar seguimientos a las metas y objetivos del equipo y del proyecto. Dar seguimiento y encabezar el desarrollo y evolución de los requerimientos del producto. Definir las prioridades en el desarrollo de los requerimientos. Asegurar que el equipo trabaja de acuerdo a los objetivos del proyecto. Entender las necesidades y requerimientos del cliente y asegurar que estos sean plasmados en el o los productos creados por el equipo.
Facilitador de procesos	Eliminar los distractores y obstáculos que se presenten para el equipo durante el proyecto. Auxiliar al equipo en el buen entendimiento y ejecución de los procesos. Auxiliar en la solución de problemas.
Equipo implementador	Cada miembro del equipo es responsable de desarrollar las actividades y productos que le sean asignados de acuerdo a los procesos y características definidas.

Tabla 3 Definición de actividades principales de roles básicos.

ROLES AUXILIARES	
Rol	**Responsabilidad**
Diseñador	Revisión de los elementos del Sprintlog. Creación y revisión de los diseños. Revisión de prototipos.
Desarrollador	Creación y revisión del código. Revisión de los elementos del Sprintlog. Creación y revisión de prototipos.
Administrador	Auxiliar en la creación de los planes, estimaciones y estrategias.
Gestor de calidad	Creación y revisión de estrategias de pruebas. Revisión de documentación de requerimientos, diseños y prototipos. Revisión de los elementos del Sprintlog.
Integrador	Auxiliar en la creación de planes y estrategias de integración. Revisión de planes y estrategias de pruebas. Revisión de los elementos del Sprintlog

Tabla 4 Roles auxiliares.

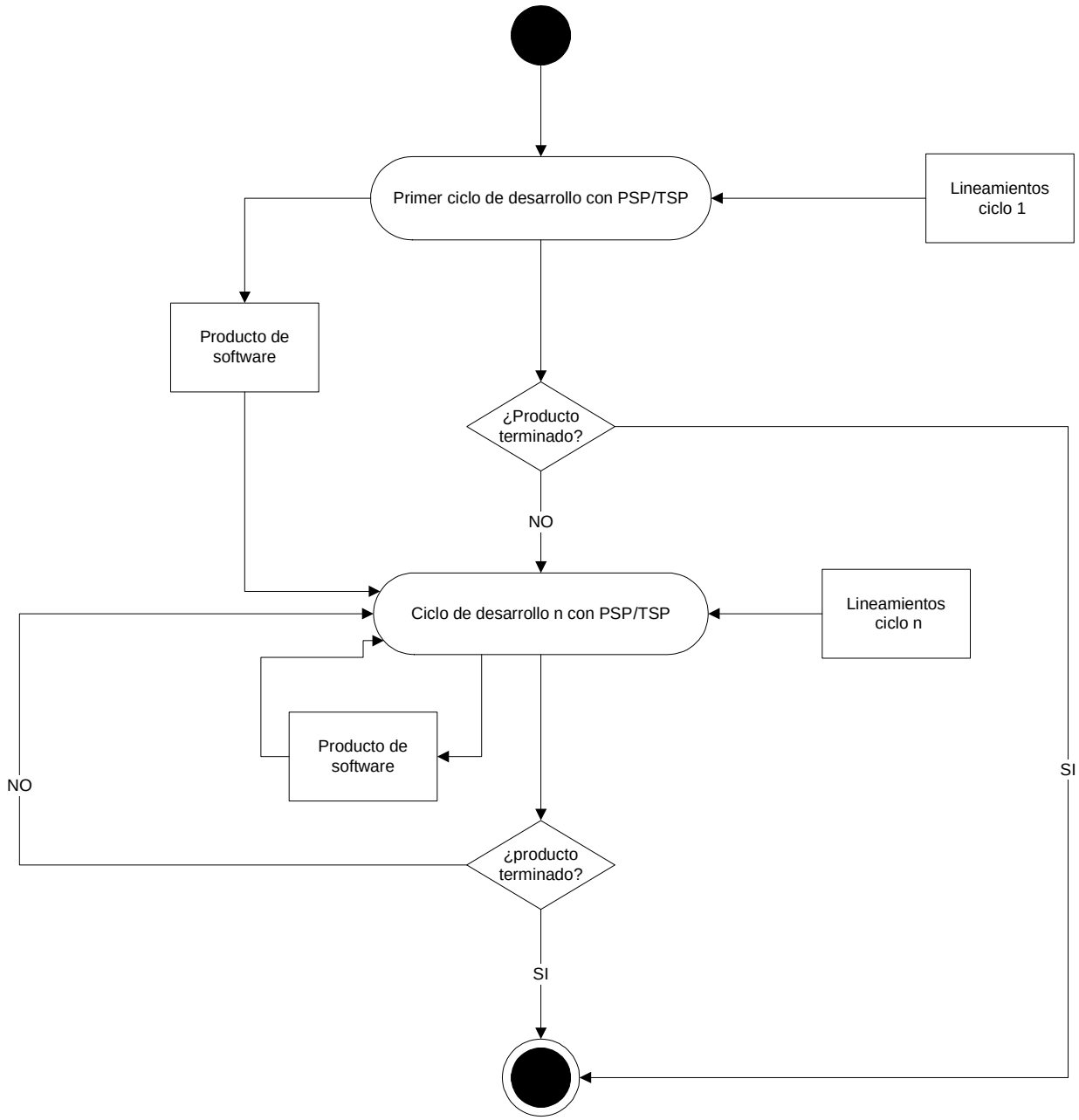


Figura 12. Estructura de TSP.

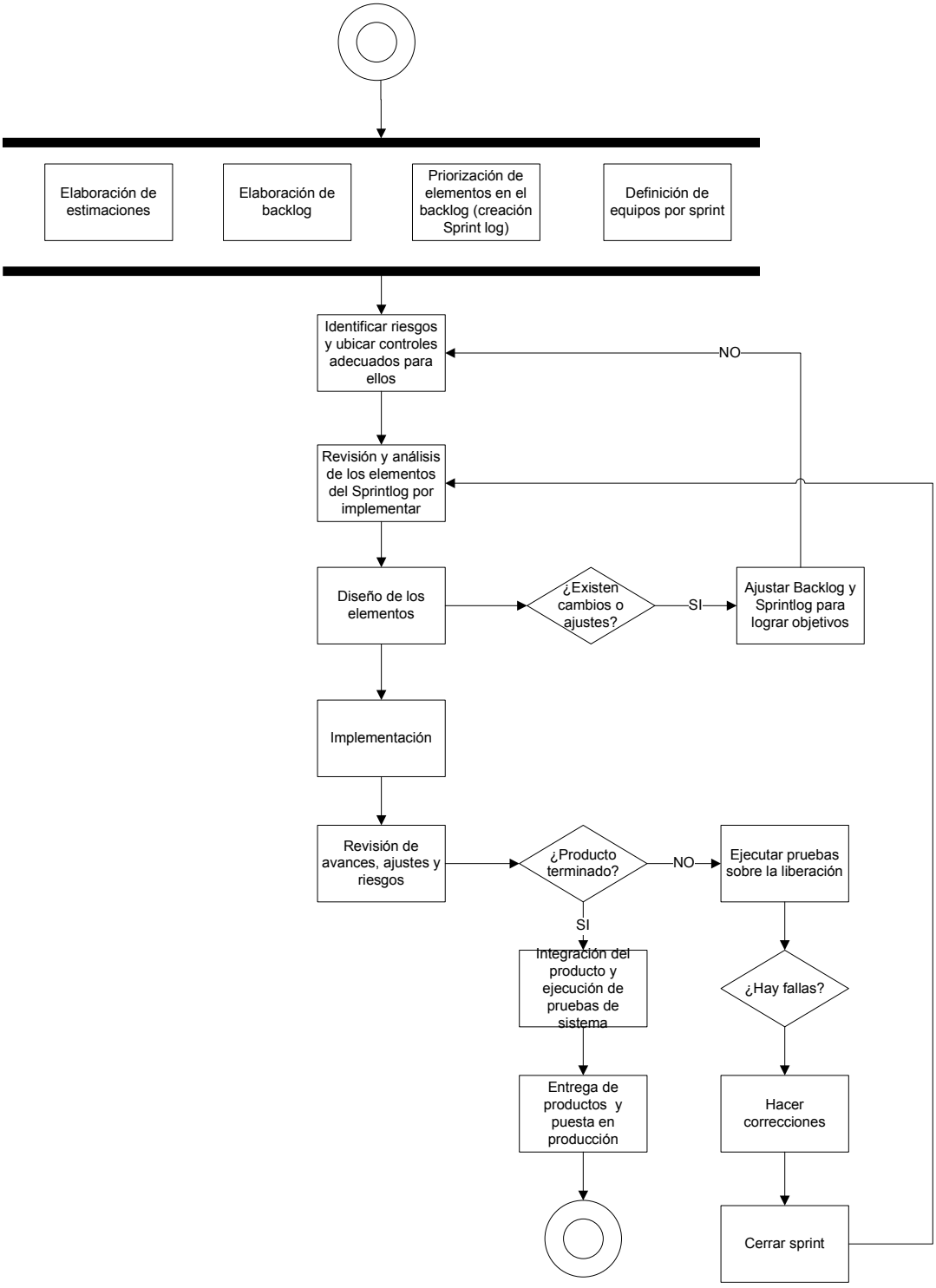


Figura 13 Estructura de SCRUM.

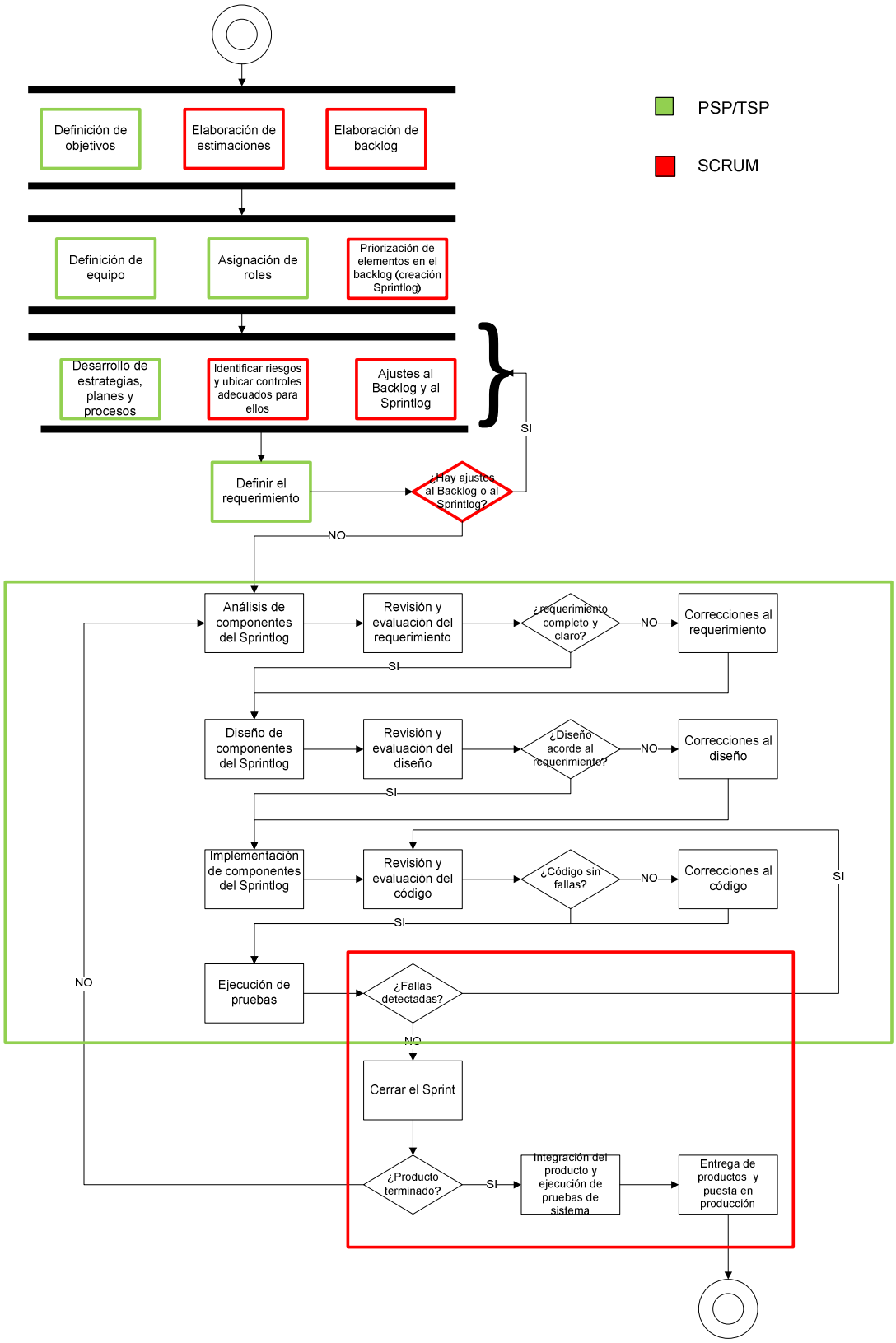


Figura 14 Estructura de la metodología propuesta

5.4 Selección de características de calidad y proceso del software

La ingeniería de software es la aplicación práctica del conocimiento científico dirigido al diseño y construcción de programas computacionales, (Bohem, 1996). Incluyendo personas, proceso, proyecto y producto, la ingeniería de software mejora la calidad de los productos aumentando la productividad y el trabajo de los ingenieros. La presente propuesta ofrece: base para la construcción de software de alta calidad, define la velocidad de producción en plazos establecidos y hace posible alcanzar una reducción de costos; haciendo todo esto es posible garantizar la calidad de los productos ofrecidos al mercado.

El producto de la ingeniería de software consiste en mucho más que un código. Incluye artefactos como planes, informes, gráficas y desempeño de roles.

Para reforzar la metodología y, sobre todo, para la implementación de calidad con base a los estándares internacionalmente reconocidos, se toman conceptos de calidad del ISO 9126 y se toma como apoyo en la definición del proceso a seguir el ISO 90003.

Para fines del experimento, del ISO 90003 se tomaron los siguientes puntos para la definición de un proceso que de soporte a la creación del producto de software:

Del requisito 4. Sistema de la gestión de la calidad, la cláusula:

4.1 Requisitos generales.

4.2.3 Control de los documentos.

Del requisito 5. 5. Responsabilidad de la dirección:

5.2 Enfoque al cliente.

5.4 Planificación.

5.5.1 Responsabilidad y autoridad.

5.5.2 Representante de la dirección

Del requisito 6. Administración de los recursos, la cláusula:

6.2.2 Competencia, toma de conciencia y formación.

6.3 Infraestructura.

Del requisito 7. Realización del producto, los subrequisitos y las cláusulas:

7.1 Planificación de la realización del producto, en

7.2 Procesos relacionados con el cliente,

7.2.1 Determinación de los requisitos relacionados con el producto,

7.2.2 Revisión de los requisitos relacionados con el producto,

7.2.3 Comunicación con el cliente.

7.3 Diseño y desarrollo,

7.3.1 Planificación del diseño y desarrollo,

7.3.2 Elementos de entrada para el diseño y desarrollo,

7.3.3 Resultados del diseño y desarrollo,

7.3.4 Revisión del diseño y desarrollo,

7.3.5 Verificación del diseño y desarrollo,

7.3.6 Validación del diseño y desarrollo, y

7.3.7 Control de los cambios del diseño y desarrollo.

Del requisito 8. Medición, análisis y mejora, los subrequisitos y las cláusulas:

8.2 Seguimiento y medición,

8.2.3 Seguimiento y medición de los procesos, y

8.2.4 Seguimiento y medición del producto,

8.3 Control del producto no conforme,

8.4 Análisis de datos.

Del ISO 9126 se tomaron los siguientes conceptos para la definición de características de calidad del producto de software:

De la Usabilidad los conceptos de Operatividad y Comprensión:

Operatividad: La capacidad de un producto de software para posibilitar al usuario para operarlo y controlarlo (ISO/IEC9126-1:2001).

Comprensión: La capacidad de un producto de software para posibilitar al usuario a entender si el software es conveniente, y como puede ser utilizado para tareas particulares y condiciones de uso (ISO/IEC9126-1:2001).

De la Funcionalidad los conceptos de Exactitud y Conformidad:

Exactitud: La capacidad de un producto de software para proporcionar los resultados correctos o acordados o cumplir con el grado necesario de precisión (ISO/IEC9126-1:2001).

Conformidad: La capacidad de un producto de software para proveer un apropiado conjunto de funciones para tareas especificas y objetivos del usuario (ISO/IEC9126-1:2001).

De la Eficiencia el concepto de Utilización de recursos:

Utilización de recursos: La capacidad de un producto de software para utilizar las cantidades apropiadas y tipos de recursos cuando éste desarrolla sus funciones bajo condiciones expresas (ISO/IEC9126-1:2001).

De la Mantenibilidad los conceptos de Cambiable y Estable:

Cambiable: La capacidad un producto de software para hacer posible la implementación de cambios específicos (ISO/IEC9126-1:2001).

Estable: La capacidad de un producto de software para evitar efectos inesperados a causa de posibles modificaciones (ISO/IEC9126-1:2001).

En conjunto, todos estos elementos, definieron el qué y el cómo para la creación de un producto de software mediante la ejecución de la ingeniería de software. Para tal fin se crearon procesos y procedimientos (Figura 15) que incluyen los conceptos medulares tanto de PSP/TSP y SCRUM (Tabla 2) así como de las normas ISO.

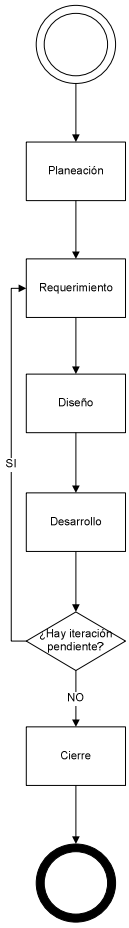


Figura 15 Secuencia de ejecución de los procesos de cada etapa de la ingeniería de software.

De la figura 15 cabe destacar que no es necesario terminar cada iteración para que se retomen las actividades de la documentación y definición de requerimientos, en la figura se presenta así para fines ilustrativos. Sin embargo es necesario hacer notar que el equipo del proyecto puede retomar las actividades desde los requerimientos hasta la construcción en cuanto sea posible y conveniente de acuerdo a la definición de modelo iterativo.

Pre juego				Juego	Post juego
Planeación		Requerimientos	Diseño	Desarrollo (Implementación)	Cierre
Objetivos	Definir los objetivos, planes y estrategias de trabajo del proyecto. Definir el equipo del proyecto y los roles de cada uno de sus integrantes.	Describir los requisitos del proyecto.	Construir y documentar el diseño que se utilizará como base para la fase de implementación.	Desarrollar el código y efectuar las pruebas unitarias.	Hacer las pruebas de integración del sistema antes de su liberación e implantación.
Actividades	Desarrollo de la lista backlog Definición de una fecha de liberación y de la funcionalidad de uno o más entregas. Selección de las entregas más apropiadas para su inmediato desarrollo. Mapeo de paquetes de producto (objetos) de los elementos del backlog dentro de las entregas seleccionadas. Definición de equipo(s) para la construcción de la nueva entrega. Valoración de riesgos y los controles apropiados de los riesgos. Revisión y el posible ajuste de elementos del backlog y paquetes. Aprobación o reselection de herramientas de desarrollo e infraestructura. Estimación de costo de la entrega, incluyendo el desarrollo, el material colateral, mercadeo, entrenamiento, y rollout. Verificación de aprobación de dirección y consolidación.	Revisar la descripción de las necesidades del producto. Definir y documentar los requisitos funcionales y no funcionales. Generar el plan de pruebas del sistema. Efectuar la inspección de documentos y productos generados.	Revisión de la asignación de los elementos del backlog. Identificar cambios necesarios para llevar a cabo la implementación de los elementos del backlog. Realizar el análisis del dominio para extender el requerimiento a construir, mejorar o actualizar los modelos de dominio para reflejar el contexto del nuevo sistema y sus requerimientos. Refinar la arquitectura del sistema para soportar el nuevo contexto y requerimientos. Identifique cualquier problema o problemas desarrollando o implementando los cambios. Junta de revisión de diseño, cada equipo presenta su acercamiento y cambios para la implementación de cada elemento del backlog. Reasignar cambios como requisito (Reassing changes as required) Distribución, revisión y ajuste de los estándares con los cuales el producto estará conformado.	Reunión con equipos para revisar la entrega de los planes. Sprints iterativos, hasta que el producto esté listo para su distribución. Planificar las tareas individuales de implementación. Codificar. Revisar el código entre colegas. Efectuar las pruebas unitarias.	Se prepara la entrega del producto completo. Integración, pruebas del sistema, documentación del usuario, material de entrenamiento preparado. Preparar el ambiente de pruebas. Realizar y documentar las pruebas. Integrar el producto y probar el sistema. Corregir los defectos. Producir la documentación del usuario. Implantación del sistema.

Tabla 5 Resumen de la metodología de desarrollo propuesta para el desarrollo del experimento.

5.5 Aplicación de la metodología y proceso definido

Como parte de los elementos a medir para la validación de que los equipos efectivos para un pequeño proyecto de desarrollo de software incrementan la calidad de sus productos apoyándose en el uso de PSP / TSP, SCRUM y los estándares ISO 9126 e ISO 90003 como herramientas de trabajo, se seleccionaron los siguientes:

- Número de defectos encontrados en las inspecciones/revisiones de los productos creados.
- Número de horas invertidas en el desarrollo de los productos.
- Número de fallas encontradas durante la ejecución de las pruebas.

Para que estos datos ayudaran a obtener la información necesaria para la evaluación del método, se decidió aplicar el control estadístico.

Es importante destacar que estos datos fueron tomados exclusivamente para evaluar el método y no a las personas que colaboraron en el experimento. Esto se recalca debido a que la propuesta tiene su alcance en la evaluación del método utilizado y no de las personas, no obstante que las personas son un aspecto fundamental y de alto impacto en la buena aplicación de los métodos y si ellas no sería posible lograr los objetivos establecidos.

VI. APLICACIÓN DE LA METODOLOGÍA Y PROCESOS DEFINIDOS.

6.1 Introducción al desarrollo experimental

La presente propuesta de aplicación metodológica pertenece a las metodologías que alientan y apoyan la flexibilidad con un alto grado de tolerancia a los cambios. Estas metodologías permiten considerar al proceso de desarrollo como imprevisible al comienzo, y los mecanismos de control son puestos en su lugar para manejar la imprevisibilidad. SCRUM utiliza mecanismos de control para manejar la imprevisibilidad y controlar el o los riesgos. La flexibilidad, la sensibilidad y la fiabilidad son los resultados (Schwaber, 1997). Por su parte, PSP/TSP son una forma de guiar a los integrantes del equipo en la utilización de métodos de trabajo en equipos efectivos.

6.2 Aplicación: Definición del problema

Como primera actividad en el experimento se definió un problema real con necesidad de solución tecnológica. El problema se tomo de una empresa centrada en la compra y venta de acero. El dueño y administrador de la empresa (de aquí en adelante definido como cliente) requiere que sus inventarios sean administrados de forma automática. En busca de una solución existente, el cliente puso a prueba herramientas como Excel, AdminPaq entre otras. Ninguna de las herramientas que el cliente probó cubrió sus expectativas y necesidades de administración de inventarios.

El cliente planteo como un problema la inexistencia de herramientas administrativas en el mercado que se ajustaran a las necesidades especificas de su negocio. A tal problema la solución dada fue la creación de una aplicación de software a la medida que automatizará el control de inventarios de su empresa.

Con el desarrollo de la aplicación el cliente podrá administrar sus inventarios permitiendo obtener:

- Reportes de productos en existencia diarios, semanales y mensuales.
- Reportes de salidas contra entradas de producto y viceversa
- Actualizar masivamente las existencias, costos y precios de venta de los productos
- Manejar las salidas de productos (cotizaciones, pedidos, facturas)
- Manejo de clientes, proveedores y productos.

La administración de inventarios con la aplicación de software a la medida permitirá manejar las entradas y salidas de productos de forma sencilla, evitara la salida indebida de productos, permitirá dar seguimiento a los pedidos de los clientes y al reabastecimiento del almacén cuando sea necesario evitando tener productos almacenados por tiempo indeterminado (y por tanto capital invertido) y sufrir carencias de productos más demandados por el mercado. Así mismo permitirá dar seguimiento puntual a los pedidos pendientes por surtir tanto del lado de los proveedores como del lado del cliente.

6.3 Aplicación: Definición metodología del software

Para dar inicio al desarrollo experimental de este trabajo de tesis se elaboraron los procesos que debían de seguirse en durante la creación de la aplicación. Para la elaboración de este proceso. Se crearon en total 3 procesos principales: el proceso de planeación de proyecto, el proceso de administración de requerimiento y por último el proceso de ingeniería de productos de software. Cabe mencionar que estos tres procesos principales tienen conceptos tanto de PSP/TSP como de SCRUM. Una de las aportaciones más importantes de SCRUM a este trabajo fue la gestión de tiempos y actividades, así como el seguimiento; ambos conceptos fueron incluidos de forma diaria en la operación. Durante las juntas de seguimiento se seguían las siguientes reglas de acuerdo a SCRUM:

- La reunión debe comenzar puntualmente, en el mismo lugar siempre a la misma hora.
- Solo los comprometidos pueden hablar
- La reunión dura 15 minutos.
- Todos los asistentes deben mantenerse de pie.
- Durante la reunión, cada miembro del equipo debe contestar a tres preguntas:

- ¿Qué has hecho desde ayer?
- ¿Qué es lo que estás planeando hacer hoy?
- ¿Has tenido algún problema que te haya impedido alcanzar tu objetivo?

De esta forma todo el equipo y tiene conocimiento del estado del proyecto, avances, retrasos, necesidades, riesgos por documentar, riesgos materializados y, sobre todo, se da la oportunidad de prever con tiempo situaciones que puedan retrasar al proyecto o que impidan que se cumplan los objetivos y/o acuerdos plasmados en planeación.

A continuación se describirán los procesos utilizados para el desarrollo de las actividades de la ingeniería de software del proyecto.

Dentro del proceso de planeación se contemplan las siguientes actividades:

- Generación de la propuesta de negocio.
- Presentación de la propuesta de negocio.
- Anuncio y presentación del inicio del nuevo proyecto.
- Conformación de equipos.
- Identificar actividades y productos que deberán de ser generados.
- Estimación de tamaño del proyecto.
- Generación del plan del proyecto.

El proceso de planeación de proyectos (Figura 11) inicia cuando se manifiesta en algún cliente potencial la necesidad de una solución tecnológica a algún problema en su negocio o empresa. Esta necesidad puede ser manifestada de forma directa del cliente al proveedor de servicios de creación de soluciones tecnológicas de software o por medio de un anuncio de licitación. Para dar solución a la necesidad tecnología se debe de elaborar una propuesta de negocio analizando las características de la solicitud del cliente, la viabilidad de la solución tanto de desarrollo como de implantación y los beneficios que traería para el negocio o empresa del cliente dicha solución.

Una vez aprobada la propuesta de negocio, es necesario dar inicio internamente con el arranque del proyecto. Es de vital importancia que ya exista un equipo de proyecto y que se le dé a conocer el arranque del proyecto desde antes de que exista un plan de trabajo. Hacer partícipe al equipo del proyecto en su totalidad desde el anuncio de la aceptación de la propuesta de negocio por parte del cliente lo fortalece internamente y los involucra desde el inicio provocando un sentido de pertenencia y propiciando mayor entusiasmo a la hora de colaborar durante el tiempo de vida del mismo.

Para dar inicio con las actividades propias del recién nacido proyecto deben de hacerse las estimaciones pertinentes, tanto de actividades como de esfuerzos y recursos. Es importante que en esta etapa de la planeación se eche mano de los recursos con mayor experiencia en el proyecto. PSP/TSP ayuda a la identificación de las capacidades de los recursos del proyecto permitiendo que sea posible que cualquiera de los elementos del equipo pueda tener más de un rol. Es decir, es recomendable que se tomen en cuenta a todos los integrantes del equipo sin necesidad de que su actividad principal sea administrador de proyectos o líder. En el caso particular de este experimento se echó mano de programadores, arquitectos, diseñadores y gestores de calidad que con su aportación ayudaron a la estimación de actividades y a la creación del plan de trabajo. Es durante esta etapa dentro de la planeación, retomando la aportación de SCRUM, en donde se crea la estructura de descomposición de trabajo (backlog) para crear subequipos responsables de las iteraciones que conforman a la totalidad del proyecto (Figura 12).

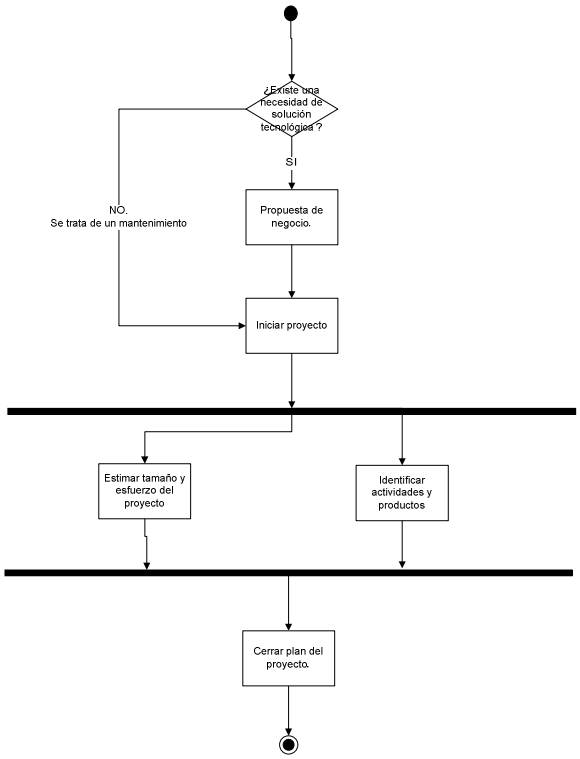


Figura 16 Diagrama de flujo del proceso de planeación de proyectos.

Cuando se han creado ya los planes y estimaciones, éstas se le dan a conocer a todo el equipo nuevamente. Dando la oportunidad de identificar ajustes y/o actualizaciones antes de arrancar con las actividades de análisis y administración de requerimientos. Una vez aprobados los planes, estimaciones y la estructura de descomposición del trabajo, se da inicio con las actividades del proceso de administración de requerimientos.

Sistema Control Total

ID	Descripción	Sprint #	1	2	3	4	5	6	7	8	9
	Esfuerzo necesario para liberar la entrega 1 en horas		62	58	59	66	72	70	42	42	16
1	Documentación de la iteración 1		20	0	0	0	0	0	0	0	0
2	Desarrollo de prototipado.		10	0	0	0	0	0	0	0	0
3	Presentación de prototipado.		3	0	0	0	0	0	0	0	0
	Planeación del proyecto	*0*									
4	Creación de vista		6	0	0	0	0	0	0	0	0
5	Desarrollo del negocio		10	0	0	0	0	0	0	0	0
6	Creación de DAO		10	0	0	0	0	0	0	0	0
7	Desarrollo del modelo de datos		2	0	0	0	0	0	0	0	0
8	Pruebas unitarias		1	0	0	0	0	0	0	0	0
	Construcción de la funcionalidad relacionada a la empresa	*1*									
9	Documentación de la iteración 2.		0	10	0	0	0	0	0	0	0
10	Creación de vista		0	6	0	0	0	0	0	0	0
11	Desarrollo del negocio		0	16	0	0	0	0	0	0	0
12	Creación de DAO		0	16	0	0	0	0	0	0	0
13	Desarrollo del modelo de datos		0	4	0	0	0	0	0	0	0
14	Pruebas unitarias		0	3	0	0	0	0	0	0	0
15	Pruebas de integración		0	3	0	0	0	0	0	0	0
	Construcción de la funcionalidad relacionada a los clientes	*2*									
16	Documentación de la iteración 2.		0	0	10	0	0	0	0	0	0
17	Creación de vista		0	0	6	0	0	0	0	0	0
18	Desarrollo del negocio		0	0	16	0	0	0	0	0	0
19	Creación de DAO		0	0	16	0	0	0	0	0	0
20	Desarrollo del modelo de datos		0	0	4	0	0	0	0	0	0
21	Pruebas unitarias		0	0	3	0	0	0	0	0	0
22	Pruebas de integración		0	0	4	0	0	0	0	0	0
	Construcción de la funcionalidad relacionada a los proveedores	*3*									
Entrega 1	**Liberación correspondiente a manejo de Clientes y Proveedores**										

Figura 17 Estructura de descomposición de trabajo o Producto Backlog.

Para dar inicio a las actividades de la administración de requerimientos (Figura 13) y tomando como antecedentes los documentos creados durante la fase anterior, se solicita al cliente la información necesaria o faltante para integrar la especificación funcional. Dicho documento contendrá el listado de los requerimientos que serán desarrollados a lo largo del ciclo de vida del desarrollo del software, así como las características de calidad que deberán de ser contempladas dentro del producto final. Cabe destacar que este documento se inicia pero puede ser que no se concluya de forma inmediata.

Dado que el ciclo de vida será iterativo puede darse el caso que solo de los requerimientos que se contemplan durante la primera iteración del proyecto sean detallados y documentados completamente para dar inicio, y que los demás vayan siendo documentados hasta contemplar el 100% en otro momento del ciclo de vida. Lo anterior quiere decir que los casos de uso de los requerimientos de la primera iteración deberán ser documentados totalmente y que los requerimientos que falten deben de tener una fecha de inicio de documentación que puede ser mediante casos de uso, mapas mentales o cualquier

otra herramienta que ayude a la documentación detallada. Los requerimientos en esta etapa deben de quedar listados, divididos y detallados en la Estructura de descomposición del trabajo a forma de Sprint backlog de acuerdo a SCRUM (Figura 14).

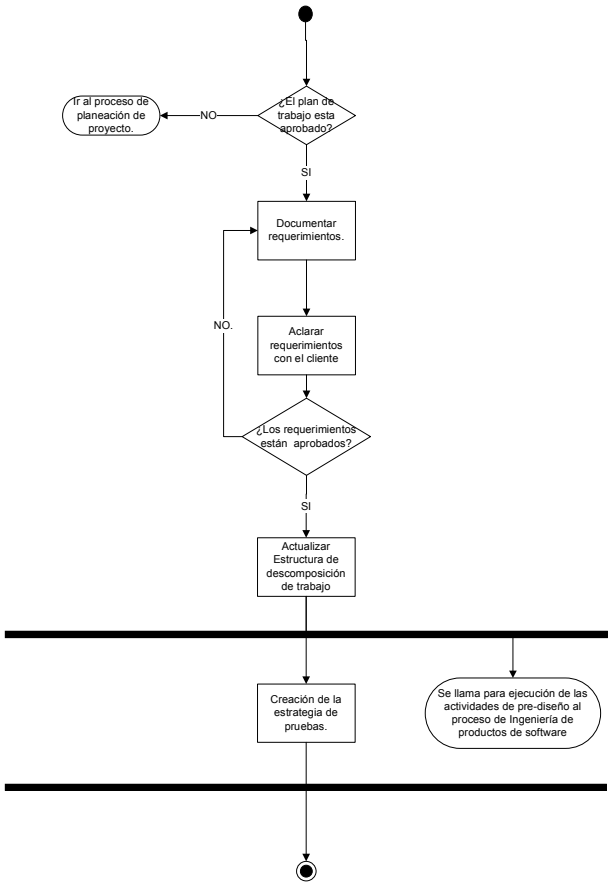


Figura 18 Diagrama de flujo del proceso de administración de requerimientos.

Una vez documentados y detallados los requerimientos necesarios para dar inicio a la primera iteración y en paralelo de la creación de la estrategia de pruebas, se está en la posibilidad de comenzar con la creación de un pre-diseño de la aplicación. Este pre-diseño y otras actividades se documentan como parte del proceso de ingeniería de productos de software. A continuación se listan las actividades más significativas del proceso de administración de requerimientos:

- Definición y listado de los requerimientos.
- Reuniones de aclaración de especificación funcional con el cliente.
- Aprobación de la especificación funcional por parte del cliente.

- Documentación, al menos de los requerimientos que serán desarrollados durante la primera iteración.
- Creación de la estrategia de pruebas.
- Actualización de la estructura de descomposición de trabajo.

SPRINT	INICIO	DURACIÓN
1	15-may-09	20

				V	L	M	X	J	V	L	M	X	J	V	L	M	X	J	V	L	M	X
				15-may	18-may	19-may	20-may	21-may	22-may	25-may	26-may	27-may	28-may	29-may	1-jun	2-jun	3-jun	4-jun	5-jun	8-jun	9-jun	10-jun
			Horas de trabajo	2	2	2	2	2	2	2	2	2	2	4	1	1	4	3	3	3	6	
PILA DEL SPRINT				ESFUERZO																		
Backlog	Tarea	Tipo	Estad	Responsal																		
Sprint0	Documentación de la iteración 1. En	Análisis	Terminada	Analista de s	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5									
Sprint0	Documentación de la iteración 1. Cli	Análisis	Terminada	Analista de s	1	1	1	1	1	1	1	1	1									
Sprint1	Documentación de la iteración 2. Pr	Análisis	Terminada	Analista de s	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5	1.5									
Sprint0	Documentación de la iteración 1. Pr	Análisis	Terminada	Analista de s	1	1	1	1	1	1	1	1	1									
Sprint0	Desarrollo de prototipado	Prototipado	Terminada	Analista de s										3								
Sprint0	Reunión del Sprint 0	Análisis	Terminada	Analista de s	stemas		.15						.15									
Sprint0	Presentación de prototipado	Reunión	Terminada	Lider										1								
Sprint1	Creación de vista - Empresa	Diseño	Terminada	Arquitecto de	software										1							
Sprint1	Desarrollo del negocio - Empresa	Codificación	Terminada	Constructor de	sistemas												3	3	3	3	3	
Sprint1	Creación de DAO - Empresa	Diseño	Terminada	Diseñador de	base de datos											1	1					
Sprint1	Desarrollo del modelo de datos - Em	Diseño	Terminada	Diseñador de	base de datos																	
Sprint1	Pruebas unitarias - Empresa	Pruebas uni	Terminada	Gestor de Calidad																	1	
Sprint1	Corrección de fallas reportadas dura	Correccione	Terminada	Constructor de	sistemas																2	
Sprint1	Reunión del Sprint 1	Reunión	Terminada	Lider												.15					.15	

Figura 19 Estructura de descomposición de trabajo detallada o sprint backlog.

El pre-diseño es la actividad que marca la entrada al proceso de ingeniería de productos (Figura 15). En esta actividad se elabora un prototipo de la interfaz de usuario al menos para la primera iteración. La finalidad del prototipado (Figura 16) es mostrarle al cliente la interfaz que tendrá el producto final y recopilar sus impresiones, observaciones y sugerencias. A partir de esta actividad pueden desarrollarse otras como:

- Generar la arquitectura del sistema.
- Diseñar la base de datos (Figura 17).
- Definición de plataforma y lenguaje de programación.
- Actualizar la estrategia de pruebas para incluir las pruebas de integración y de sistema.
- Programación del sistema
- Ejecución de las pruebas unitarias y después de la segunda iteración, las pruebas de integración.
- Ejecutar las pruebas de sistema cuando el producto ha quedado totalmente integrado y antes de liberarlo al cliente.

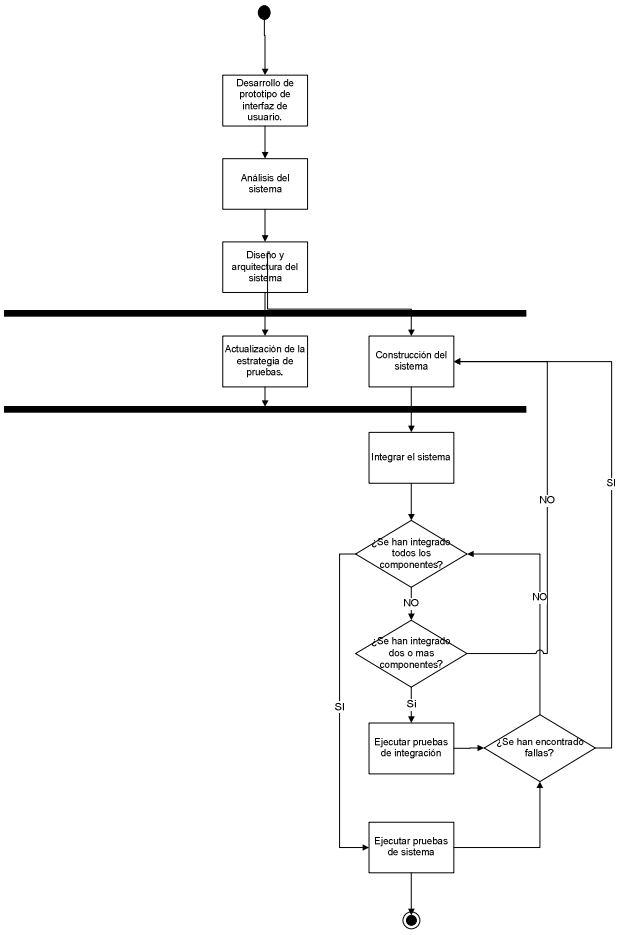


Figura 20 Diagrama de flujo del proceso de ingeniería de productos de software.

Durante la creación de los diferentes modelos se detalla el diseño y la arquitectura del sistema, se agrega mayor detalle a los documentos, modelos y diseños con el objeto de detectar cualquier necesidad, riesgo, hueco de información o imposibilidad para seguir adelante. Lo anterior con la finalidad de que cuando se construya no exista ninguna inconsistencia que provoque fallos o implementaciones erróneas del requerimiento. Aunque de antemano se sabe que la posibilidad encontrar fallas aun después de probarlas y ser liberadas a un ambiente productivo existe, todos estos esfuerzos son encaminados a minimizar esta posibilidad y detectar su presencia de forma temprana. Prevenir es mucho más barato que lamentar, en el caso de la ingeniería de software no es diferente.

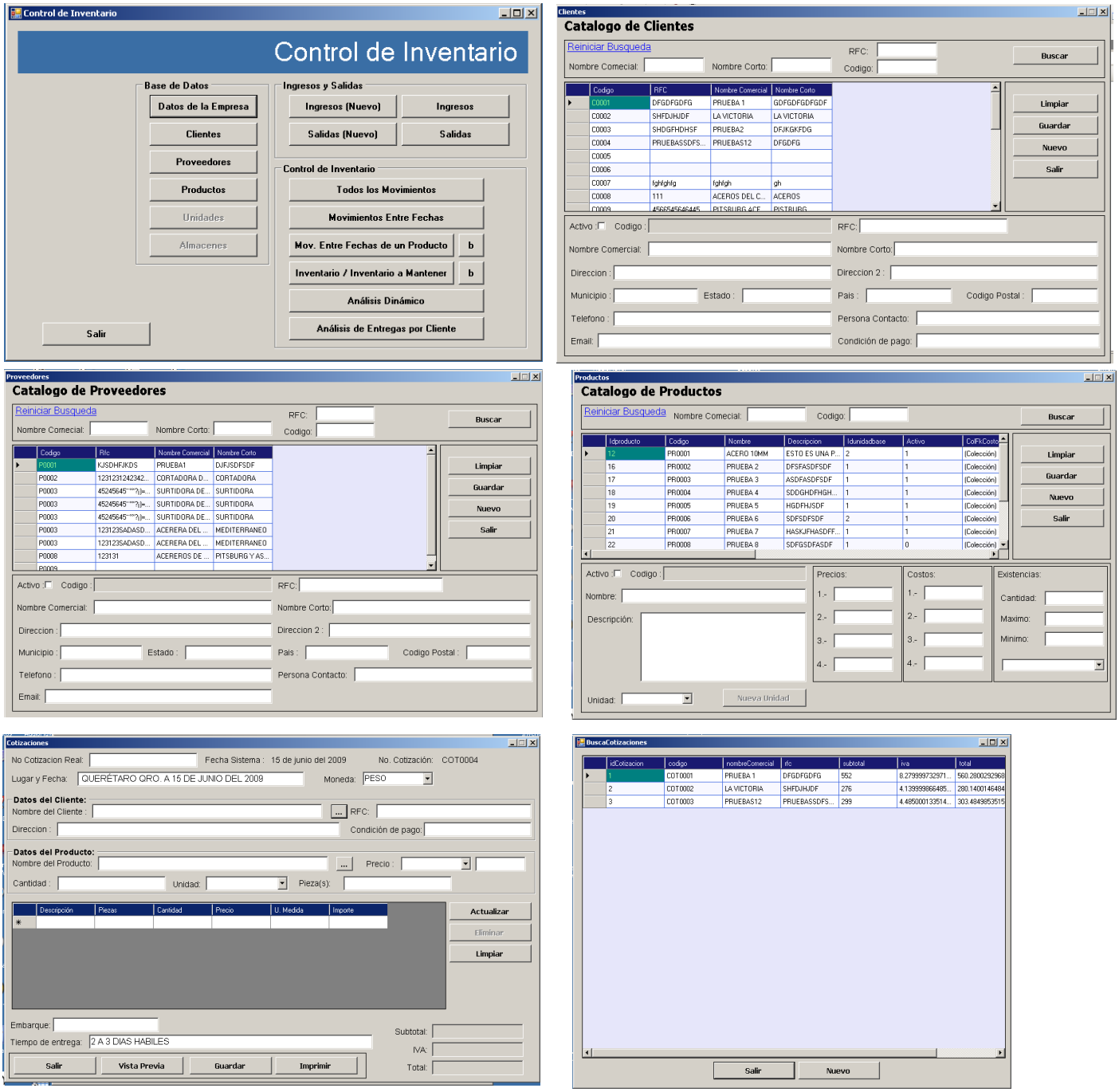


Figura 21 Pantallas de prototipado.

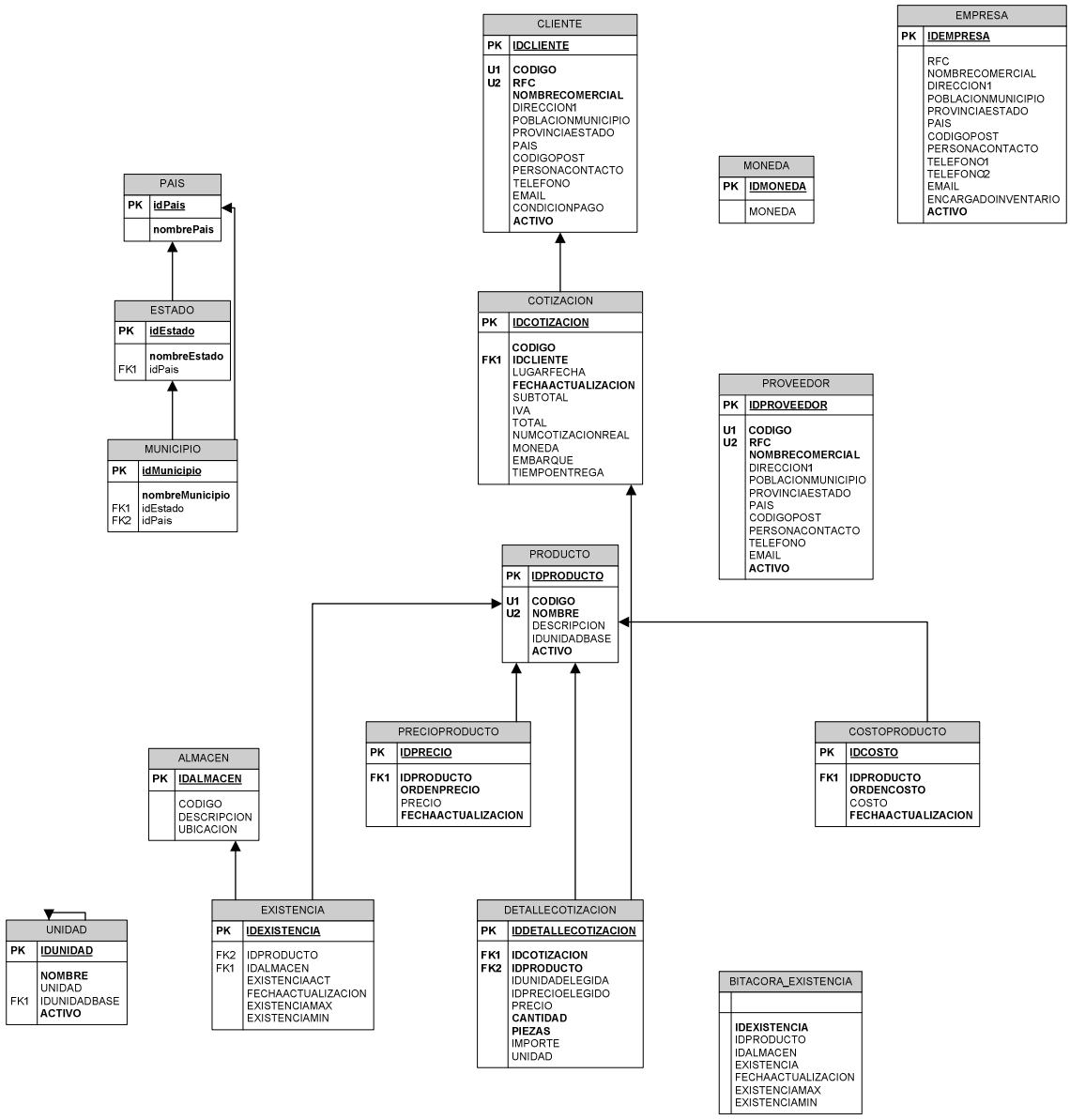
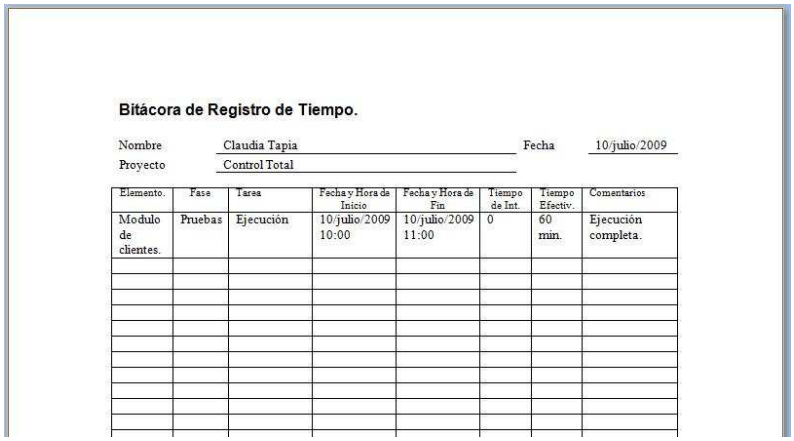


Figura 22 Diagrama de la base de datos creado como parte de las actividades de diseño y arquitectura del sistema.

A lo largo de las tareas desarrolladas se fueron registrando los tiempos de ejecución de cada actividad en un formato definido de acuerdo a PSP/TSP (Figura 18). Pese a que las reuniones de los Sprints estaban topadas a un máximo de 15 minutos por reunión cada de que se iniciaba una nueva semana de trabajo como parte de la gestión del proyecto sugerida por SCRUM, estas también fueron registradas en estos formatos. Los formatos fueron llenados por cada uno de los integrantes y se registraban cada una de las actividades que ejecutaban durante su periodo de trabajo en el proyecto.



Bitácora de Registro de Tiempo.

Nombre Claudia Tapia Fecha 10/julio/2009

Proyecto Control Total

Elemento.	Fase	Tarea	Fecha y Hora de Inicio	Fecha y Hora de Fin	Tiempo de Int.	Tiempo Efectiv.	Comentarios
Modulo de clientes.	Pruebas	Ejecución	10/julio/2009 10:00	10/julio/2009 11:00	0	60 min.	Ejecución completa.

Figura 23 Bitácora de registro de tiempo utilizado para el seguimiento del esfuerzo aplicado a las actividades.

Para los fines de este experimento fue muy importante que este registro fuera apegado a la realidad.

Dentro de las restricciones que se encontraron para la ejecución de este experimento:

- El proyecto era una actividad extra para cada uno de los participantes debido a las actividades laborales que cada uno desempeña.
- La cantidad de tiempo dedicada al proyecto era limitada en horas.
- La disposición del cliente para resolver dudas o aclarar requerimientos no era constante, es decir, ocasionalmente el cliente tiene que ausentarse del estado y las reuniones de sesión aclaratoria se aplazan de forma constante ya que la persona encargada del mostrador no tenía un contexto amplio de las solicitudes del cliente para el sistema.
- La información de avance para el cliente debe de ser en un lenguaje no técnico y de fácil entendimiento para él. Lo cual implicaba inversión de tiempo extra en las presentaciones con el cliente.

Ante estas situaciones la propuesta metodológica permitió aplicar medidas como las a continuación descritas:

- ✓ Cada integrante del equipo desarrolló su rol secundario.

- ✓ Los integrantes del equipo optaron por eliminar los desperdicios de tiempo para aprovechar al máximo el tiempo dedicado al desarrollo de proyecto.
- ✓ Antes de asistir a una reunión con el cliente se programaba una reunión de seguimiento en la cual cada uno de los integrantes presentaba de forma puntual sus dudas, necesidades y solicitudes. Con estos elementos se elaboraba un listado y se presentaba ante el cliente durante la reunión.
- ✓ Durante cada reunión con el cliente se presentaba el calendario con los avances, metas y desfases. Cada uno de estos puntos era aclarado de forma directa con él.

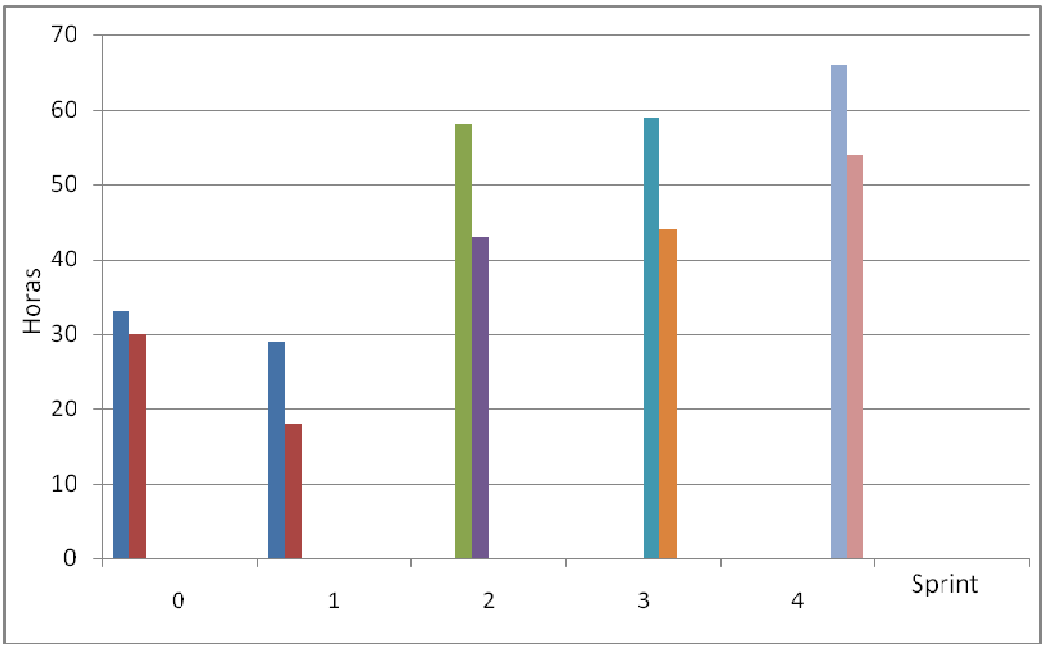
6.4 Aplicación: Selección de características de calidad y proceso de software

Y como punto final se utiliza las normas ISO 9126 y 90003 para la definición de las características y normativas de calidad que fueron aplicadas. Esto consistió en aplicar la norma ISO/IEC 9126 que está enfocada a la calidad de Producto usando, por ejemplo, los conceptos de Funcionalidad, Consistencia y Validación. La especificación y la evaluación de la calidad de producto de software se consiguieron definiendo características de calidad apropiadas, tomando en cuenta el objetivo de uso del producto de software. Del ISO 90003 se seleccionaron puntos para la definición de los procesos y que requisitos debían de ser cumplidos durante su ejecución (ver detalles en la sección 5.4).

VII. RESULTADOS Y EVALUACIÓN METODOLOGÍCA

7.1. Análisis de la información obtenida

Como resultado del seguimiento por medio de las reuniones semanales del Sprint los primeros 4 fueron cerrados satisfactoriamente. Los componentes fueron diseñados y desarrollados en su totalidad, las pruebas unitarias fueron ejecutadas y las fallas atendidas. La Grafica 1 muestra una comparativa entre los tiempos estimados y los tiempos reales de los 5 primeros Sprints (del Sprint 0 al Sprint 4).



Gráfica 1 Tiempos estimados contra tiempos reales.

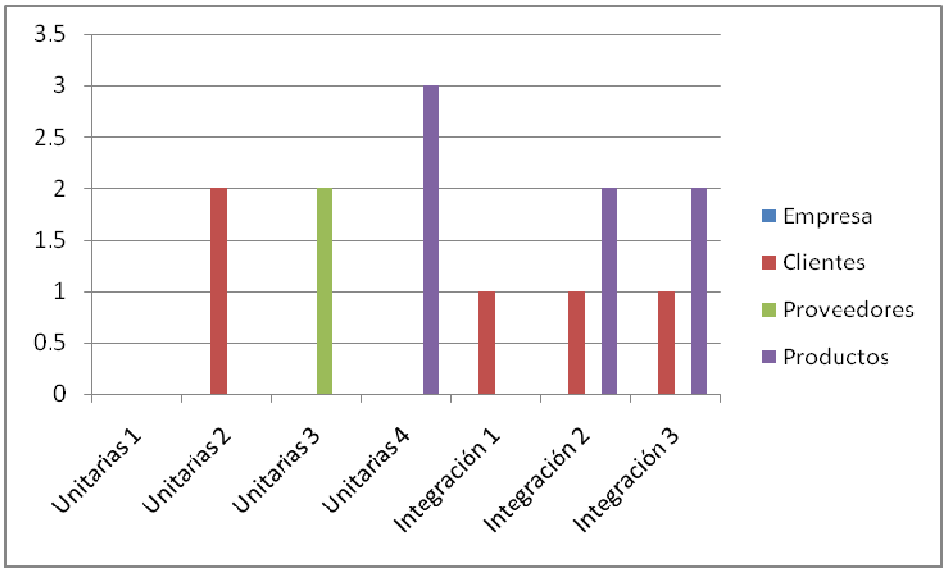
El total de horas planeadas que debían de ser invertidas entre los 5 Sprints se calculó de 245 y el total real invertido fue de 189. Este resultado se tribuye a que se contó con los insumos necesarios para la ejecución de todas las actividades descritas en la Estructura de descomposición de trabajo puesto que durante el análisis y administración de requerimientos se resolvieron todas las dudas satisfactoriamente, esto aunado con el diseño del prototipado permitió que el diseño y arquitectura del sistema se elaboraran de forma

transparente. Teniendo los diseños bien definidos fue posible el desarrollo del código sin contratiempos.

En cuanto a los defectos encontrados durante las revisiones uno a uno en los productos elaborados, la cantidad total fue de 20. Los defectos fueron corregidos en la documentación y de su corrección se resolvieron dudas, evitando que surgieran más adelante como fallas en la aplicación. El tiempo total invertido en la corrección de los defectos equivale al 2.55% del total del tiempo invertido en la duración de los 4 sprints

De acuerdo a la planeación, en el cuarto Sprint se ejecutaron un total de 4 ciclos de pruebas unitarias y 3 ciclos de pruebas de integración. A lo largo de estos ciclos de pruebas se detectaron un total de 14 fallas. De estas 14 fallas se pudieron obtener los siguientes análisis:

Las fallas se presentaron más en lo que refería al modulo de productos con un total de 7 fallas, seguido del de clientes con un total de 5 fallas y por último el de proveedores con 2 fallas (Grafica 2).



Gráfica 2 Comparativa entre las fallas obtenidas en cada modulo a lo largo de los ciclos de pruebas.

Las fallas obtenidas en estos ciclos de pruebas fueron revisadas, analizadas y clasificadas en tres rubros:

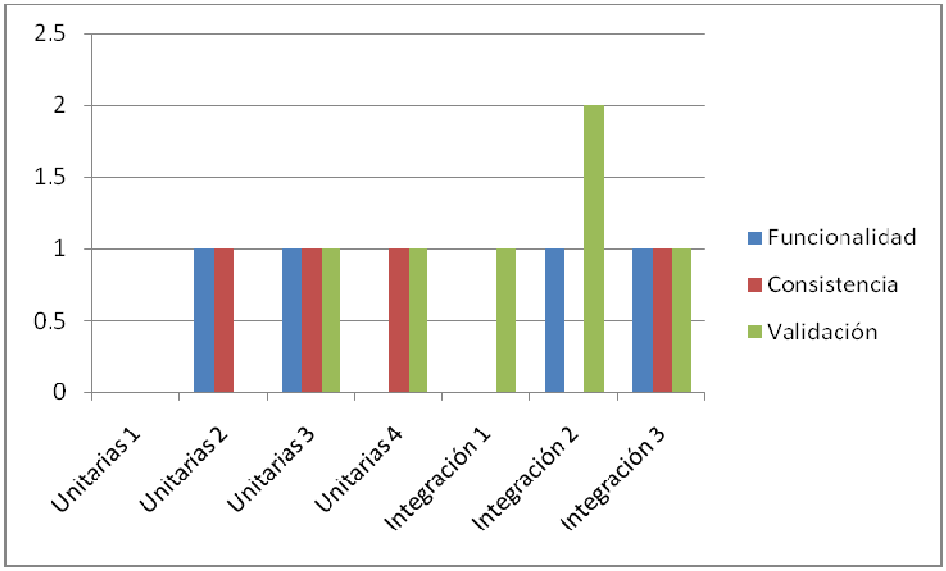
Funcionalidad: Se determinó como una falla de funcionalidad toda aquella respuesta del sistema que no cumpliera con los requisitos plasmados en los requerimientos como por ejemplo permitir el alta de un registro vació, búsquedas no exitosas aun cuando los requisitos existen, mostrar datos incompletos como resultado de una búsqueda, dejar los campos con información cuando se solicita limpiarlos.

Consistencia: Se determinó como una falla de consistencia toda aquella respuesta del sistema que imposibilitara al usuario la operación del sistema como mensajes no enmascarados para el usuario, despliegue de nombres de campos de las tablas en la interfaz de usuario, que la aplicación se pasme al solicitar la ejecución de una búsqueda.

Validación: Se determinó como una falla de validación toda aquella respuesta del sistema que permitiera hacer cosas no permitidas como por ejemplo la introducción de caracteres especiales en los campos de formato especifico como el RFC, omisión de información como la @ en los campos donde se solicitara el correo electrónico, introducción de más caracteres de los debidos.

Tomando en cuenta las clasificaciones anteriormente descritas se obtuvo la siguiente grafica (Grafica 3) que muestra se presentaron más fallas de validación que de consistencia o de funcionalidad.

El total de horas empleadas para la corrección de las fallas detectadas fue de 12. Lo cual representa un 8.17% del total del tiempo invertido en la etapa de ingeniería de productos de software y un 6.34% del total del tiempo invertido en la duración de los 4 sprints.



Gráfica 3 Comparativa entre la clasificación de las fallas detectadas durante los ciclos de pruebas.

Para complementar los resultados anteriores se agregará el factor monetario. Es decir, si a cada hora invertida en el proyecto le asignamos un costo de $60 MN y con base en que en los primeros 5 sprints se obtuvo un ahorro del 22.8% de horas al invertir solo 189 de las 245 horas planeadas, se puede estimar que el costo total neto del proyecto será de $20,736 MN. Al comparar este costo con el de otros productos existentes en el mercado puede garantizarse que es un costo bajo al considerar que este es un producto a la medida y por ende no es necesario hacerle ningún tipo de ajuste (Tabla 6). Las aplicaciones y paquetes que se ofertan en el mercado son genéricos, y si se requiere de un ajuste este implica un costo extra.

Producto	Características	Costo
AdminPaq	Actualización de 10 a 20 usuarios.	$27,240 MN
Admincontrol PLUS PRO	Paquete de 20 usuarios	$24,700 MN
Aspel-SAE	Sistema Administrativo Empresarial para 20 usuarios	$ 34,595 MN
Sistema propuesto	Paquete de n usuarios	$20,736 MN

Tabla 6 Comparativa de aplicaciones de administración de inventarios y el producto desarrollado por el proyecto.

Dado todo lo anterior, es posible dar paso a las conclusiones.

VIII. CONCLUSIONES

El presente trabajo consistió en ejecutar las actividades de planeación de proyecto, análisis y administración de requerimientos, diseño y construcción de productos de software y pruebas sobre los productos diseñados y construidos con la finalidad de demostrar que los equipos efectivos para un pequeño proyecto de desarrollo de software incrementan la calidad de sus productos apoyándose en el uso de PSP / TSP, SCRUM y el ISO 9126 y 90003 como herramientas de trabajo.

El uso de la propuesta metodológica para equipos pequeños donde se combinan primeramente puntos importantes de PSP / TSP, SCRUM y el ISO 9126 y 90003 que de acuerdo a los resultados obtenidos, se garantizó una disminución en la inversión de tiempo, esfuerzos y defectos. Así mismo, incremento la calidad de los productos y actividades desarrolladas por los miembros del proyecto, ya que desarrollan sus actividades con una visión de compromiso y calidad, generando beneficios satisfactorios.

De las deducciones significativas probadas en la experimentación es importante puntualizar que para que funcione la metodología propuesta de forma exitosa deben de tenerse en cuenta los siguientes aspectos:

- ✓ Cada uno de los integrantes del equipo debe entender el negocio del proyecto.
- ✓ Cada uno de los integrantes debe ser entrenado en PSP/TSP y SCRUM.
- ✓ Cada uno de los integrantes del equipo debe entender y asumir sus responsabilidades, obligaciones y roles dentro del equipo.
- ✓ Cada uno de los integrantes debe entender la importancia de su participación activa y proactiva dentro del proyecto y dentro del equipo.

Considerando estos puntos y toda la metodología propuesta el resultado del producto final incrementa significativamente la calidad. Además, estas metodologías ágiles son altamente recomendables debido a la flexibilidad de ajustarse a los constantes cambios. Sin embargo se debe de tener cuidado en distinguir y separar el cambio constante de los

remiendos. Es decir, tomando como base la experiencia adquirida en el campo laboral real, los pequeños cambios son solo ajustes que no deben de impactar mas allá de un 10% del trabajo desarrollado hasta el momento del ajuste.

Se minimiza hasta casi eliminar el desperdicio de tiempo en correcciones al identificar los errores y defectos desde las fases de análisis y diseño impidiendo que confusiones, malos entendimientos del requerimiento, dudas y huecos de información se lleven a la codificación y se reflejen en el producto final como una falla o una no conformidad con el requerimiento. Así mismo, se minimizan los tiempos muertos y los tiempos mal empleados por los integrantes del equipo.

Como beneficios se obtienen el incremento de la productividad, la posibilidad de hacer estimaciones más justas y elevar la competitividad del equipo. Para los integrantes del equipo es un beneficio que sus jornadas de trabajo sean realmente de ocho horas laborales, de esta forma cada uno de ellos tiene la oportunidad de planear su tiempo libre con certeza, eliminando casi por completo la incertidumbre de los tiempos extras por retrasos en el proyecto.

Otro beneficio obtenido de esta metodológica es que las revisiones e inspecciones e incluso las correcciones de defectos y fallas ocuparon menos del 10% del tiempo total del proyecto. Esto se adjudica a las situaciones siguientes:

- ✓ El seguimiento constante y ordenado que proporciona SCRUM.
- ✓ El buen entendimiento del requerimiento y/o necesidad que antecede a la solución tecnológica buscada antes de la construcción de la solución.
- ✓ El hacer las cosas bien desde el principio y sin esperar a que la inspección llegue para que los defectos y/o hallazgos sean corregidos.

Los objetivos fueron cubiertos:

- Se propuso una estrategia para que las personas involucradas en un proyecto de desarrollo de software en empresas medianas y pequeñas formen equipos

de trabajo altamente efectivos a un bajo costo y desarrollando productos de software de calidad.

- Se propuso una metodología de desarrollo de software apoyada en los estándares ISO 9126 y 90003 que asegura la calidad de los productos creados a lo largo del proceso de software así como la del producto final.
- Se propuso una metodología de bajo costo con la que las empresas medianas y pequeñas garantizan la calidad de sus productos.
- Se aplicó la metodología propuesta a un proyecto de software y se verificó su efectividad.

Por todo lo anterior se testifica que es posible crear una estrategia de trabajo para que los equipos desarrolladores de software incrementen la calidad de sus productos.

IX. APÉNDICE

Apéndice 1.

ABET Accreditation Board for Engineering and Technology

SEI Software Engineering Institute

CMMI Capability Maturity Model

PYMES Pequeñas y medianas empresas

PSP Personal Software Process

TSP Team Software Process

SCRUM Metodología ágil

Apéndice 2.

A2.1. Extracto del producto “propuesta de negocio”. Etapa de planeación.

A2.1.1. Situación actual.

Hace un año se fundó la empresa X. El área de negocio, se centra en la compra y venta de acero.

El cliente requiere de la administración de su inventario de forma automatizada. Para tal fin el cliente utilizó en un principio aplicaciones como Excel y Word, posteriormente se utilizo aplicaciones ya elaboradas (aplicaciones de caja) como ADMINPAQ. El objetivo de esta aplicación es la automatización de las labores administrativas de un negocio promedio.

Al no ajustarse el ADMINPAQ a sus necesidades especificas, el cliente ha solicitado la creación de una aplicación a la medida de su negocio.

A2.1.2. Objetivos del proyecto

Con el desarrollo de la aplicación, el cliente podrá administrar sus inventarios permitiendo obtener:

Reportes de productos en existencia diarios, semanales y mensuales.
Reportes de salidas contra entradas de producto y viceversa
Actualizar masivamente las existencias, costos y precios de venta de los productos
Manejar las salidas de productos (cotizaciones, pedidos, facturas)
Manejo de clientes, proveedores y productos.

A2.1.3. Producto o servicio

Desarrollo de software a la medida.

A2.1.4. Lista de entregables

Instalador de la aplicación.
Manual de usuario.
Capacitación del usuario final.
Reportes de seguimiento y avance del proyecto.

A2.1.5. Supuestos

El cliente proporcionará toda la información necesaria para el desarrollo de la aplicación.
El cliente solucionará las dudas que surjan durante el ciclo de desarrollo de la aplicación.
El cliente proporcionará todos los formatos y formas que utiliza para llevar a cabo su labor diaria que tenga relación con los objetivos del negocio.

A2.1.6. Restricciones

No forman parte del alcance los siguientes puntos:

Manuales y/o material de capacitación del usuario.
Instalación y/o sustitución de hardware en ambiente operativo.
Funcionalidades diferentes o extras a los registrados en este documento, a menos que sea acordado antes de la construcción y con los debidos ajustes la documentación y productos elaborados.

A2.1.7. Visión del proyecto

El proyecto deberá de contribuir a la tarea de administración de los inventarios de la empresa de forma ágil y sencilla por medio de la construcción de una aplicación de software que pueda soportar las necesidades actuales y futuras de la empresa.

A2.1.8. Beneficios del proyecto

Cuantitativos	Cualitativos
Mejora en la administración de inventarios. Mejora en el procesamiento de información. Incrementa la productividad del operador del sistema al contar con únicamente lo necesario para la administración de los inventarios. Evitara los problemas de ingreso de datos e información inútil para el negocio.	Genera reportes con información que permitirá tomar decisiones respecto a los inventarios. En base a la información de los reportes se podrá controlar de manera confiable y precisa los inventarios físicos. Genera reportes con información delos movimientos de entradas y salidas permitiendo prever problemas financieros y de almacenaje.

A2.1.9. Plan de trabajo

Etapa /Disciplina	Propósito	Fecha acordada con Cliente	
		Inicio	Fin
Planeación	1ª.Aprobación de proyecto. 2ª.Asentamiento de fechas y entregas 3ª.Definición de alcance y objetivos	15/04/09	15/06/09
Requerimientos	1ª.Analisis y documentación de requerimientos. 2ª.Diseño de prototipo 3ª.Aclaración de requerimientos y posibles ajustes a los mismos	15/05/09	20/08/09
Diseño	1ª.Diseño de alto y de bajo nivel 2ª.Definición de estrategias de prueba e integración de sistema 3ª.	15/06/09	22/08/09
Construcción	1ª. Desarrollo de funcionalidades 2ª.Verificación de funcionalidades contra requerimientos 3ª.Corrección de fallas.	20/06/09	14/10/09
Implementación	1ª.Puesta en producción de la aplicación 2ª.Liberación al cliente de entregables. 3ª.Validación de la funcionalidad por parte del cliente.	15/10/09	30/10/09
Pruebas	1ª.Verificar producto contra requerimientos. 2ª.Aprobación de producto. .	20/06/09	01/10/09

A2.1.10. Administración de la calidad de los productos

Requerimientos	Características de calidad	Criterio de aceptación	Prioridad
Reportes de productos en existencia diarios	Usabilidad	Operatividad Comprensión	7
Reportes de productos en existencia semanales	Usabilidad	Operatividad Comprensión	7
Reportes de productos en existencia mensuales	Usabilidad	Operatividad Comprensión	7
Reportes de salidas contra entradas de producto	Usabilidad	Operatividad Comprensión	7
Manejo de ajustes del inventario real contra lo que se reporta el sistema	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	7
Manejo de clientes	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	1
Manejo de proveedores	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	2
Manejo de productos	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	4
Actualizaciones masivas de existencias de productos.	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	4
Actualizaciones masivas de costos de productos	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	4
Actualizaciones	Usabilidad	Exactitud	4

masivas precios de venta	Funcionalidad	Comprensión Operatividad	
Manejo de cotizaciones	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	3
Manejo de pedidos	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	5
Manejo de facturas	Usabilidad Funcionalidad	Exactitud Comprensión Operatividad	6

A2.1.11. Mecanismo de recepción de requerimientos

La documentación que sustenta y respalda los requerimientos será entregada en las oficinas del cliente en las fechas y horarios establecidos en el presente documento.

La documentación e información serán entregadas por las tardes noches debido a la disponibilidad del cliente.

Debido a que el cliente es el que conoce a fondo el negocio, será él personalmente quien entregue la información, la documentación y aclare los requerimientos al equipo para que los requerimientos sean correctamente documentados.

A2.1.12. Mecanismo para el control de cambios

Todos los cambios se formalizan por escrito antes de ser evaluados.
Todos los cambios son evaluados por un comité de control de cambios.
En todas las ocasiones, el comité incluirá al patrocinador del proyecto.
Cuando el cambio tiene origen interno e impacta al cliente en entregables o resultados del proyecto, el comité incluirá, además, al cliente.

A2.2. Extracto del producto “especificación funcional/no funcional”. Etapa de análisis y administración de requerimientos.

A2.2.1. Introducción

Hace un año se fundó la empresa X. El área de negocio, se centra en la compra y venta de acero.

El cliente requiere de la administración de su inventario de forma automatizada. Para tal fin el cliente utilizó en un principio aplicaciones como Excel y Word, posteriormente se utilizo aplicaciones ya elaboradas (aplicaciones de caja) como ADMINPAQ. El objetivo de esta aplicación es la automatización de las labores administrativas de un negocio promedio.

Al no ajustarse el ADMINPAQ a sus necesidades especificas, el cliente ha solicitado la creación de una aplicación a la medida de su negocio.

A2.2.2. Objetivos y alcance

Objetivo Generales

Administrar de inventarios de ADMINPAQ.

Objetivos Particulares

Reportes de productos en existencia diarios, semanales y mensuales.
Reportes de salidas contra entradas de producto y viceversa
Manejo de ajustes de el inventario real contra lo que se reporta el sistema
Manejo de clientes, proveedores y productos
Actualizaciones masivas de existencias, costos y precios de venta de los productos (entradas)
Manejo de salidas de productos (cotizaciones, pedidos, facturas)

Alcance

En esta versión del sistema Control Total solo se desarrollarán los módulos relacionados con los objetivos particulares; los cuales serán descritos a detalle en apartados posteriores en este mismo documento.

A2.2.3. Requerimientos funcionales

Referencia	Descripción
01	Reportes de productos en existencia diarios
02	Reportes de productos en existencia semanales
03	Reportes de productos en existencia mensuales
04	Reportes de salidas contra entradas de producto
05	Manejo de ajustes de el inventario real contra lo que se reporta el sistema
06	Manejo de clientes
07	Manejo de proveedores
08	Manejo de productos
09	Actualizaciones masivas de existencias de productos
10	Actualizaciones masivas de costos de productos
11	Actualizaciones masivas precios de venta
12	Manejo de cotizaciones
13	Manejo de pedidos
14	Manejo de facturas

A2.2.4. Requerimientos de calidad funcional y no funcional

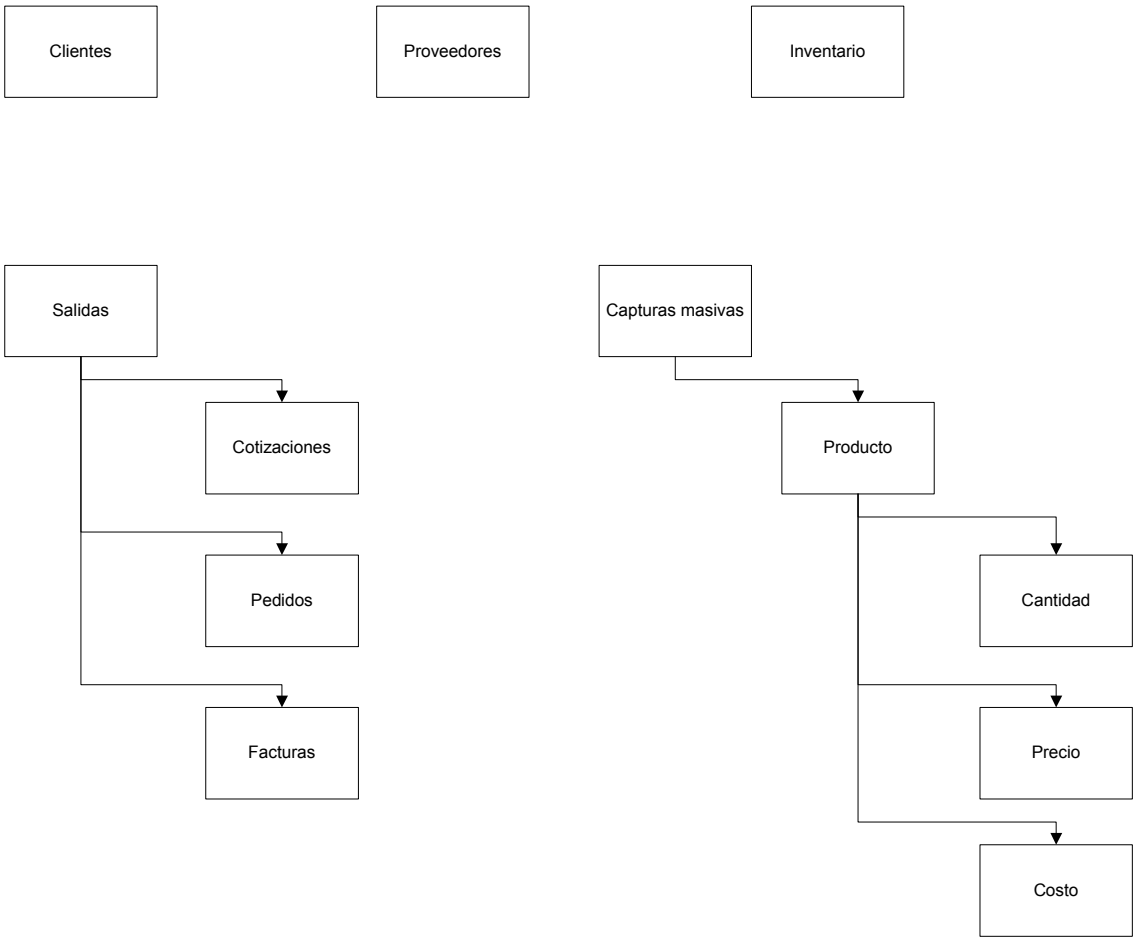
Usabilidad
Operatividad: La capacidad de un producto de software para posibilitar al usuario para operarlo y controlarlo (ISO/IEC9126-1:2001).
Comprensión: La capacidad de un producto de software para posibilitar al usuario a entender si el software es conveniente, y como puede ser utilizado para tareas particulares y condiciones de uso (ISO/IEC9126-1:2001).

Funcionalidad
Exactitud: La capacidad de un producto de software para proporcionar los resultados correctos o acordados o cumplir con el grado necesario de precisión (ISO/IEC9126-1:2001)
Conformidad: La capacidad de un producto de software para proveer un apropiado conjunto de funciones para tareas especificas y objetivos del usuario (ISO/IEC9126-1:2001)

Eficiencia
Utilización de recursos: La capacidad de un producto de software para utilizar las cantidades apropiadas y tipos de recursos cuando éste desarrolla sus funciones bajo condiciones expresas (ISO/IEC9126-1:2001).

Mantenibilidad
Cambiable: La capacidad un producto de software para hacer posible la implementación de cambios específicos (ISO/IEC9126-1:2001).
Estable: La capacidad de un producto de software para evitar efectos inesperados a causa de posibles modificaciones (ISO/IEC9126-1:2001).

A2.2.5. Modelo de dominio



A2.2.6. Modelo de Casos de Uso

Descripción

Los casos de uso serán definidos siguiendo el orden a continuación descrito:

Clientes:

La funcionalidad de clientes debe de abarcar los movimientos de alta, consulta y modificación. No existirán las bajas de clientes, solo podrán activarse o desactivarse. Cuando un cliente se encuentra en estado Activo, los datos relacionados a este cliente podrán ser utilizados para la generación de facturas y cotizaciones, si el estado del cliente es diferente de Activo, se podrá solo consultar el registro del cliente, pero no podrá ser usado para cotizaciones o facturas.

Las consultas deberán poder ejecutarse con solo ingresar el Código del cliente, el nombre del cliente o el RFC.

Toda la información del cliente se podrá modificar.

Proveedores:

La funcionalidad de proveedores debe de abarcar los movimientos de alta, consulta y modificación. No existirán las bajas de proveedores, solo podrán activarse o desactivarse. Cuando un proveedor se encuentra en estado Activo, los datos relacionados a este proveedor podrán ser utilizados para la generación de pedidos, si el estado del proveedor es diferente de Activo, se podrá solo consultar la información del proveedor, pero no podrá ser usado para pedidos.

Las consultas deberán poder ejecutarse con solo ingresar el Código del proveedor, el nombre del proveedor o el RFC del proveedor.

Toda la información del proveedor se podrá modificar.

Producto:

La funcionalidad de productos debe de abarcar los movimientos de alta, consulta y modificación. No existirán las bajas de productos, solo podrán activarse o desactivarse. Cuando un producto se encuentra en estado Activo, los datos relacionados a este producto podrán ser utilizados para la generación de pedidos, cotizaciones y facturas; si el estado del producto es diferente de Activo, se podrá solo consultar la información del producto, pero no podrá ser usado para ninguna de las funcionalidades ya descritas.

Las consultas deberán poder ejecutarse con solo ingresar el Código del producto o su nombre.

Toda la información del producto se podrá modificar.

Entradas (Actualizaciones masivas de existencias de productos, Actualizaciones masivas de costos de productos, Actualizaciones masivas precios de venta):

Las entradas serán manejadas de forma masiva y no debe de relacionarse un producto proveedor de forma única.

Salidas (Cotizaciones, Pedidos y Facturas):

La creación de facturas debe de siguiendo el flujo a continuación descrito:

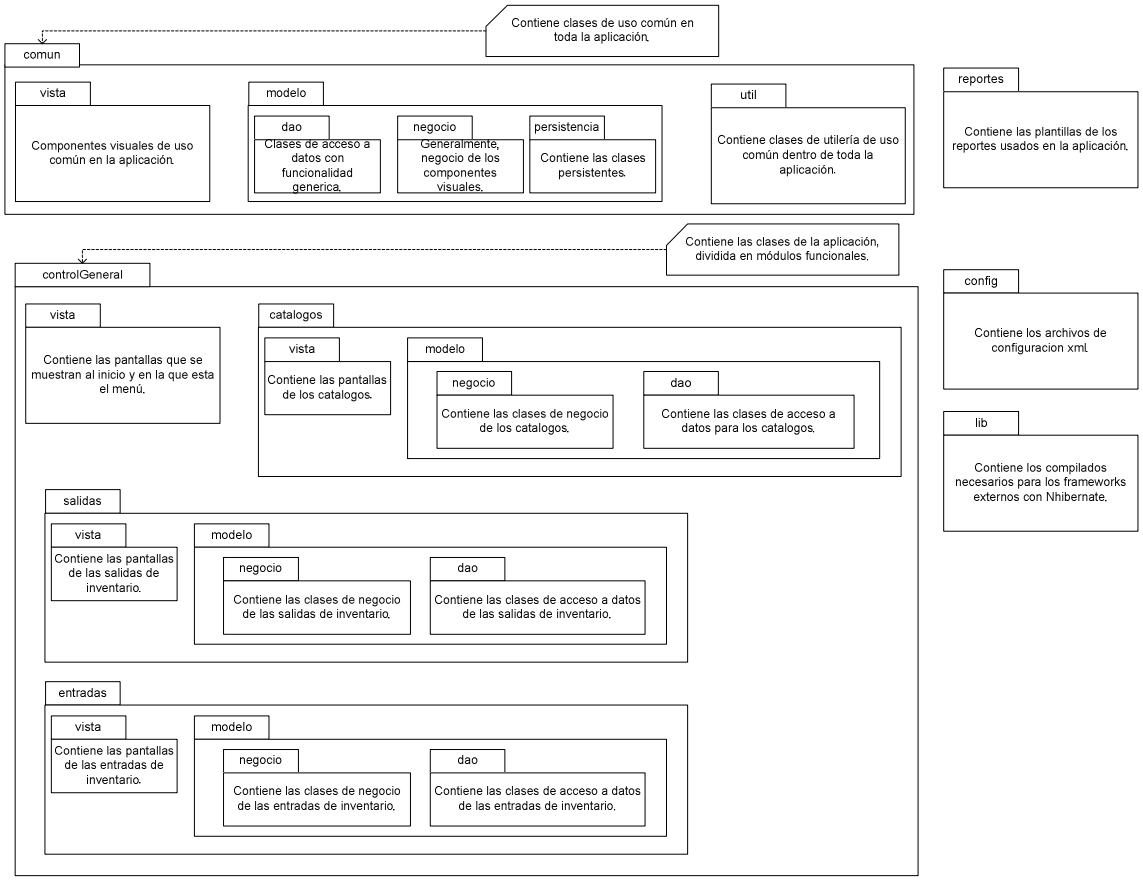
El cliente se acerca a solicitar una cotización de uno o varios productos. La cotización es elaborada por el empleado de acuerdo a los requerimientos del cliente. Cuando el cliente se decide a comprar los productos cotizados, la cotización se convierte en un pedido que debe de surtirse en el almacén. Cuando el pedido es surtido y la mercancía está siendo entregada/recibida por el cliente se convierte en una factura.

A lo largo de este flujo la información que contiene la cotización, el pedido y la factura podrá ser actualizada o modificada. Antes de imprimir una factura deberá de poder revisarse su contenido ya que las facturas están foliadas y deben de evitarse en la medida de lo posible los errores en las impresiones.

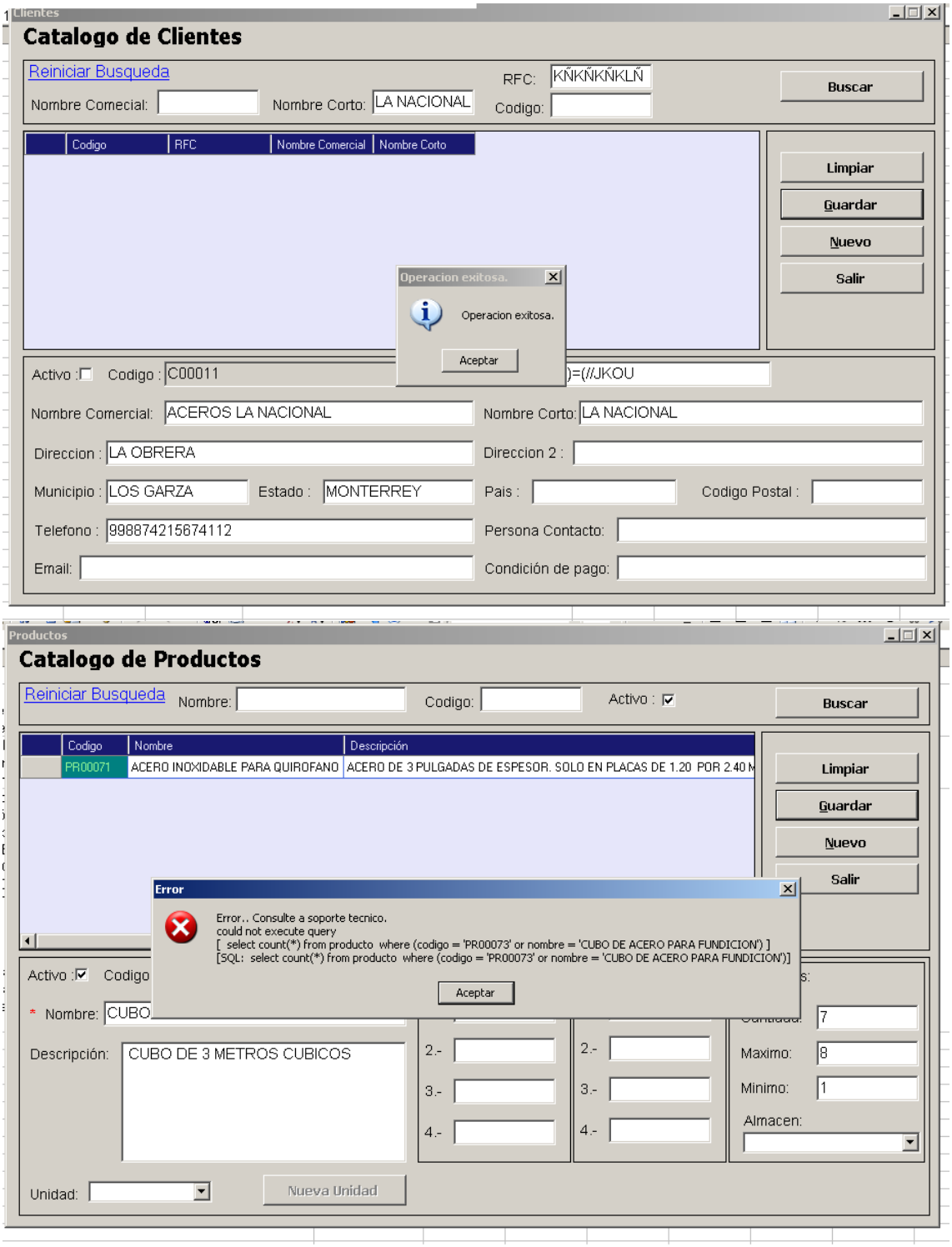
Reportes (Reportes de productos en existencia diarios, Reportes de productos en existencia semanales, Reportes de productos en existencia mensuales, Reportes de salidas contra entradas de producto, Manejo de ajustes del inventario real contra lo que se reporta el sistema):

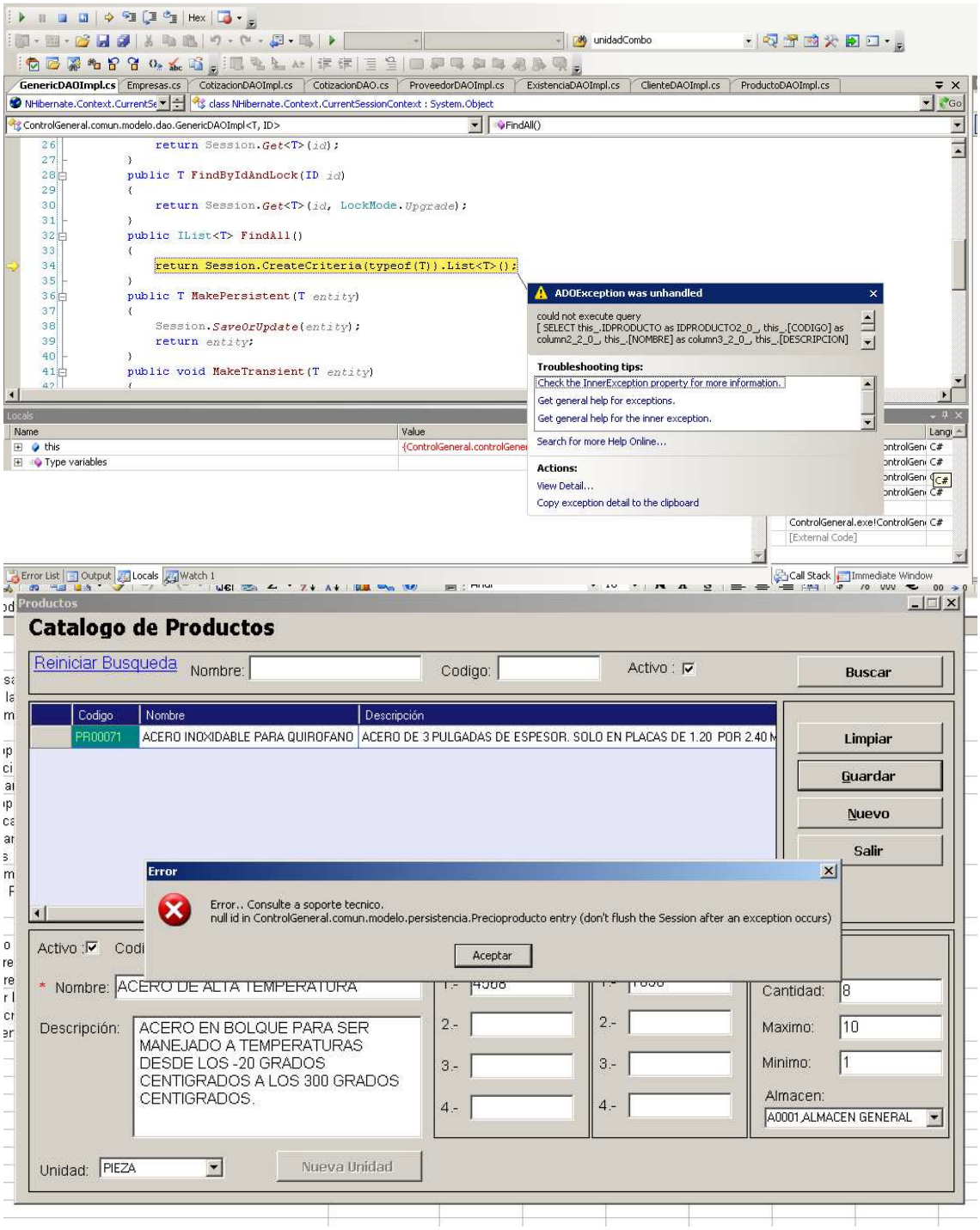
Los reportes deben de ser acumulados y deben poder estar disponibles para su consulta e impresión por un periodo determinado de tiempo.

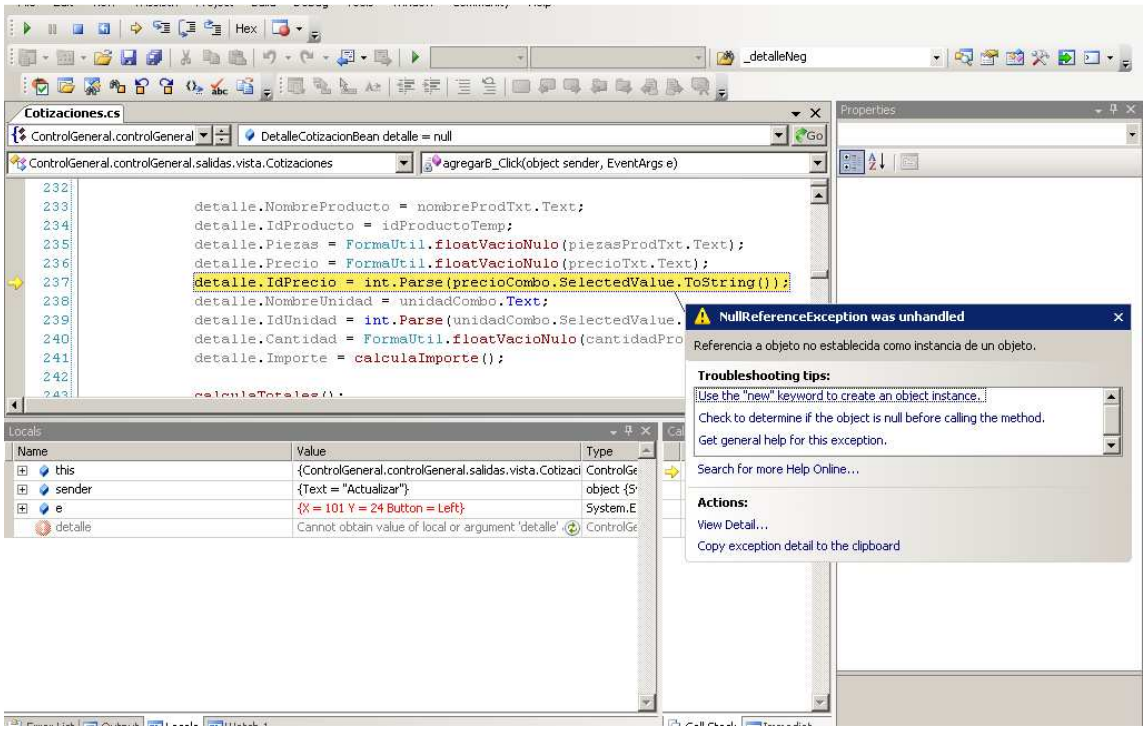
A2.3. Diagrama de paquetes. Etapa de diseño.



A2.4. Evidencia de la ejecución de pruebas. Etapa de pruebas.




GenericDAOImpl.cs
Empresas.cs
CotizacionDAOImpl.cs
CotizacionDAO.cs
ProveedorDAOImpl.cs
ExistenciaDAOImpl.cs
ClienteDAOImpl.cs
ProductoDAOImpl.cs
ControlGeneral.comun.modelo.dao.GenericDAOImpl<T, ID>
FindAll()
return Session.Get<T>(id);
public T FindByIdAndLock(ID id)
return Session.Get<T>(id, LockMode.Upgrade);
public IList<T> FindAll()
return Session.CreateCriteria(typeof(T)).List<T>();
public T MakePersistent(T entity)
Session.SaveOrUpdate(entity);
return entity;
public void MakeTransient(T entity)
ADOException was unhandled
could not execute query
[SELECT this_.IDPRODUCTO as IDPRODUCTO2_0_, this_.[CODIGO] as column2_2_0_, this_.[NOMBRE] as column3_2_0_, this_.[DESCRIPCION]
Troubleshooting tips:
Check the InnerException property for more information.
Get general help for exceptions.
Get general help for the inner exception.
Search for more Help Online...
Actions:
View Detail...
Copy exception detail to the clipboard
Locals
Name
Value
this
Type variables
Error List
Output
Locals
Watch 1
Call Stack
Immediate Window
Productos
Catalogo de Productos
Reiniciar Busqueda
Nombre:
Codigo:
Activo :
Buscar
Codigo
Nombre
Descripción
PR00071
ACERO INOXIDABLE PARA QUIROFANO
ACERO DE 3 PULGADAS DE ESPESOR. SOLO EN PLACAS DE 1.20 POR 2.40 M
Limpiar
Guardar
Nuevo
Salir
Error
Error.. Consulte a soporte tecnico.
null id in ControlGeneral.comun.modelo.persistencia.Precioproducto entry (don't flush the Session after an exception occurs)
Aceptar
Activo :
* Nombre:
ACERO DE ALTA TEMPERATURA
Descripción:
ACERO EN BOLQUE PARA SER MANEJADO A TEMPERATURAS DESDE LOS -20 GRADOS CENTIGRADOS A LOS 300 GRADOS CENTIGRADOS.
Cantidad:
8
Maximo:
10
Minimo:
1
Almacen:
A0001,ALMACEN GENERAL
Unidad:
PIEZA
Nueva Unidad


Cotizaciones.cs
ControlGeneral.controlGeneral
DetalleCotizacionBean detalle = null
ControlGeneral.controlGeneral.salidas.vista.Cotizaciones
agregarB_Click(object sender, EventArgs e)
detalle.NombreProducto = nombreProdTxt.Text;
detalle.IdProducto = idProductoTemp;
detalle.Piezas = FormaUtil.floatVacioNulo(piezasProdTxt.Text);
detalle.Precio = FormaUtil.floatVacioNulo(precioTxt.Text);
detalle.IdPrecio = int.Parse(precioCombo.SelectedValue.ToString());
detalle.NombreUnidad = unidadCombo.Text;
detalle.IdUnidad = int.Parse(unidadCombo.SelectedValue.
detalle.Cantidad = FormaUtil.floatVacioNulo(cantidadPro
detalle.Importe = calculaImporte();
NullReferenceException was unhandled
Referencia a objeto no establecida como instancia de un objeto.
Troubleshooting tips:
Use the "new" keyword to create an object instance.
Check to determine if the object is null before calling the method.
Get general help for this exception.
Search for more Help Online...
Actions:
View Detail...
Copy exception detail to the clipboard
Locals
Name
Value
Type
this
{ControlGeneral.controlGeneral.salidas.vista.Cotizaci
sender
{Text = "Actualizar"}
e
{X = 101 Y = 24 Button = Left}
detalle
Cannot obtain value of local or argument 'detalle'
Properties

X. REFERENCIAS

Aksit Mehmet (2004)**.** *The 7 C's for Creating Living Software: A Research Perspective for Quality-Oriented Software Engineering***.** Los países bajos. Universidad de Twente.

Beck, K.; Cockburn, A.; Jeffries, R.; and Highsmith, J. (2001). *Agile Manifesto***.** http://www.agilemanifesto.org.

Blakemore, Ronald Alvin (1996) *Defining software processes complaint with the capability maturity model. ISO 9000-3, and IEEE STD-1074-1991***.** Universidad Houston-Clear Lake, USA.

Boehm, Barry (2001). *Software Defect Reduction Top 10 List.* Universidad del sureste de california.

Braude Eric J. (2003). *Ingeniería de software: una perspectiva orientada a objetos*. Madrid. España. ALFAOMEGA.

CAELUM (Information & Quality Technologies). (2004). *Calidad del software*. http://www.calidaddelsoftware.com

Cai Xia (2002) *Component-based embedded software engineering: development framework, quality assurance and a generic assessment environment*. International Journal of Software Engineering and Knowledge Engineering. Vol. 12, No. 2,107-133.

Coster, Andy. (2004). *ISO 90003:2004 Checklist for Software Development.* http://www.rcglobal.com/wrcg13septrcg.htm

Cross Steve (2003). *Software Engineering Practices. Are Software Quality and Time to Market Incompatible?.* IOS Press.

Deming, W. Edward. (1986). *Out of the Crisis*. Cambridge, MA: MIT Center for Advanced Engineering.

Esaki Kazuhiro (2002). *A Quality Engineering Approach to Human Factors Affecting Software Reliability in Design Process*. Japón. Electronics and Communications. Part 3, Vol. 85, No. 3.

Feamster, Nick. (2001). *La seguridad en la Ingeniería de Software.* www.acm.org/crossroads/espanol/xrds7-4/onpatrol74.html

Febles Estrada, Isabel (2005). *Métricas para la Gestión de la Configuración de Software, un Planteamiento Formal.* Facultad de Ingeniería Industrial, CUJAE.

Fowler, Martin. (2003). *The new methodology.* http://www.programacionextrema.org/articulos/newMethodology.es.html

García, Joaquín. (2005). *CMM - CMMI*. http://www.ingenierosoftware.com/calidad/cmm-cmmi.php.

Guerini, M. (2007). *Revisión de Métodos Estadísticos Utilizados en Experimentación de Software. Proyecto de Tesis de Magister en Ingeniería del Software*. Escuela de Postgrado. ITBA.

Humphrey, Watts (2000). "*The Personal Software Process (PSP)*" (CMU/SEI-2000-TR-022). Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Humphrey, Watts S. (1989). *Managing the Software Process.* Addison-Wesley.

IEEE Std. 1028 – 1997. (1997). *IEEE Standard for Software Reviews*. IEEE Software.

ISO. (2009). *ISO Página principal*. http://www.iso.org/

ISO/IEC 90003:2004(E). (2004) *Software engineering — Guidelines for the application of ISO 9001:2000 to computer software.*

ISO/IEC 9126-1:2001(E). (2001) *Software engineering — Product quality — Part 1: Quality mode.*

ISO/IEC TR 9126-2:2003(E). (2001) *Software engineering — Product quality — Part 2: External metrics.*

ISO/IEC TR 9126-3:2003(E). (2001) *Software engineering — Product quality — Part 3: Internal metrics.*

ISO/IEC TR 9126-4:2004(E). (2001) *Software engineering — Product quality — Part 4: Quality in use metrics.*

Johnson Jim (1995). *Creating CHAOS*. American Programmer.

Juran, J. M. (1998). *Juran on Planning for Quality*. New York, New York: MacMillan.

Kernel. (2009). *Metodología PSPSM/TSPSM (Personal Software Process / Team Software Process).* México. Kernel Technologies Group, S.A. de C.V.

Kiszkurno Ernesto (2007) *La implementación del proceso de revisión de requerimientos.* Distrito Federal. México. Expo Software Gurú.

Laurie Ann Williams. (2000). *The Collaborative Software Process.* PhD dissertation of The University of Utah.

Martin, Robert C. (2004). *Best Practices in SCRUM Project Management and XP Agile Software Development.* Object Mentor, Inc. and Certified SCRUM Master TM.

Michael P. Papazoglou (2009). *Service-Oriented Computing: Research Roadmap*. ftp://ftp.cordis.europa.eu/pub/ist/docs/directorate_d/st-ds/services-research-roadmap_en.pdf.

Oktaba, Hanna et all. (2009). *COMPETISOFT. Mejora de proceso de software para pequeñas y medianas empresas y proyectos*. Madrid, España. Alfaomega.

Pérez Lamancha Beatriz (2007). *Estrategia de gestión de las pruebas funcionales en el centro de ensayos de software*. Madrid, España. REICS**.**

Piattini, Mario G et all. (2008). *Medición y estimación del software*. Madrid, España. Alfaomega.

Piattini, Mario G. et all. (2001). *Mantenimiento del software*. Madrid. España. ALFAOMEGA.

Piattini, Mario G. et all. (2003). *Calidad en el desarrollo y mantenimiento del software. Madrid*. España. ALFAOMEGA**.**

Polo Usaola, Macario (2006). *Curso de doctorado sobre Proceso software y gestión del conocimiento. Pruebas del Software.* Ciudad Real. España. Departamento de Tecnologías y Sistemas de Información, UCL.

Pomeroy-Huff, Marsha (2005). *The Personal Software ProcessSM (PSPSM) Body of Knowledge, Version 1.0 (CMU/SEI-2005-SR-003).* Pittsburgh, PA: Software Engineering Institute, Carnegie Mellon University.

Pressman, R.S. (2002). *Ingeniería del Software. Un enfoque práctico*. Madrid, España. Mc Graw Hill.

Pytel, P. (2007). *Método de Reducción de Error Humano en Experimentación de Software*. Proyecto de Tesis de Magister en Ingeniería del Software. Escuela de Postgrado. ITBA.

Quality Assurance Institute. (2006) .*Guide to the CSTE Common body of knowledge version 6.1.1*. USA. Quality Assurance Institute.

Rising, Linda et al. (2000).The *SCRUM Software Development Process for Small Teams*. Phoenix, AZ. IEEE SOFTWARE.

Scalone, Fernanda (2006). *Estudio comparativo de los modelos y estándares de calidad del software.* Universidad Tecnológica Nacional. Buenos Aires. Argentina.

Schwaber, Ken .(1997). *SCRUM Development Process*. Miami, USA. Advanced Development Methods.

Schwaber, Ken .(2004). *Agile Project Management with SCRUM*. USA. Microsoft Press.

SCRUM Methodology. (2009). *SCRUM Methodology.* http://SCRUMmethodology.com/. Fecha de consulta: Febrero 2009.

SCRUM. (2009). *SCRUM* .http://www.chuidiang.com/ood/metodologia/SCRUM.php. Fecha de consulta: Febrero 2009.

Senn, James A. (1999). *Análisis y diseño de sistemas de información*. Universidad del estado de Georgia. USA. MC Graw Hill.

Watts S. Humphrey (1999). *Introduction to Team Software Process*. Addison-Wesley.

Watts S. Humphrey. (1995). *A Discipline for Software Engineering*. Addison-Wesley.