RESUMEN

La aplicación de modelos de procesos de desarrollo de software en nuestro medio, no es un tema importante para muchas organizaciones. El resultado, pérdida de tiempo, dinero y esfuerzos en proyectos fracasados. La Cooperativa de Ahorro y Crédito Jardín Azuayo, es una institución financiera que por su crecimiento acelerado necesita ordenar y redefinir muchos de sus procesos y entre ellos está el de desarrollo de software.

Por esta razón se consideró importante, dar a conocer una breve historia de la creación de la Cooperativa, para identificar la magnitud de nuestro universo de estudio y abordar temas relacionados con los Modelos de Procesos de Desarrollo de Software y Aseguramiento de la Calidad de Software, debido a que muchos profesionales desconocen totalmente los temas mencionados. Posteriormente se realizó el levantamiento y diseño del modelo actual, realizando varias reuniones con el equipo de la Unidad de Desarrollo quienes aportaron eficazmente con su experiencia.

Definido el mismo, se realizó el análisis FODA, a través del cual se pudo identificar sus fortalezas y debilidades. Toda esta información permitió diseñar un modelo basado en la aplicación de normas y estándares de calidad tanto para procesos como para productos, como el Modelo CMMI y SQUARE respectivamente. Finalmente se definieron métricas que ayuden al modelo propuesto a mejorar continuamente.

ÍNDICE

UNIVERSIDAD DE CUENCA

María Eugenia Montero Arévalo, reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de Magister en Gerencia de Sistemas de Información. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

María Eugenia Montero Arévalo

C.I. 0102254950

María Eugenia Montero Arévalo, certifica que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autora.

María Eugenia Montero Arévalo

C.I. 0102254950

UNIVERSIDAD DE CUENCA

FACULTAD DE INGENIERÍA

MAESTRÍA EN
"GERENCIA DE SISTEMAS DE INFORMACIÓN"

DISEÑO DE LA PROPUESTA DEL MODELO DE PROCESOS PARA LA UNIDAD DE DESARROLLO DE SOFTWARE DEL "*ÁREA DE SERVICIOS INFORMÁTICOS Y REDES DE COMUNICACIÓN*" DE LA "COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO"

PROYECTO DE GRADUACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN GERENCIA DE SISTEMAS DE INFORMACIÓN

AUTOR: Ing. Sis. María Eugenia Montero
DIRECTOR: PhD cand Luis Chavarría.

Mayo 2012

CUENCA-ECUADOR

Dedicatoria

A mis padres, por su apoyo incondicional en ésta nueva etapa de mi vida profesional y por inculcarme siempre con su ejemplo deseos de superación y perseverancia. A mis hermanos y sobrinos, por ser el soporte y compañía de mi hija en mi ausencia durante estos años de estudio de la Maestría y de manera muy especial a mi hija Kamila por ser mi ángel y demostrarme su amor, con su apoyo, su responsabilidad y comprensión para que pudiera cumplir con ésta meta fijada.

Agradecimiento

A la Universidad de Cuenca, por brindarme la oportunidad de participar en la Maestría de Gerencia de Sistemas de Información II Edición, la misma que fue dictada por profesores de alto nivel profesional tanto nacional e internacional. A la Cooperativa de Ahorro y Crédito "Jardín Azuayo" por su colaboración para el desarrollo de ésta tesis. A mi Director de Tesis y amigo, PhD cand Luis Chavarría, quien a pesar de la distancia, siempre estuvo dispuesto a guiarme y asesorarme, a mi amiga Paola Montenegro, por ser compañía y apoyo durante el desarrollo de éste trabajo y a todos mis amigos(as), que siempre estuvieron pendientes, de manera especial a David Avila, Edison Tirado y Henry Bacuilima.

RESUMEN

La aplicación de modelos de procesos de desarrollo de software en nuestro medio, no es un tema importante para muchas organizaciones. El resultado, pérdida de tiempo, dinero y esfuerzos en proyectos fracasados. La Cooperativa de Ahorro y Crédito Jardín Azuayo, es una institución financiera que por su crecimiento acelerado necesita ordenar y redefinir muchos de sus procesos y entre ellos está el de desarrollo de software.

Por esta razón se consideró importante, dar a conocer una breve historia de la creación de la Cooperativa, para identificar la magnitud de nuestro universo de estudio y abordar temas relacionados con los Modelos de Procesos de Desarrollo de Software y Aseguramiento de la Calidad de Software, debido a que muchos profesionales desconocen totalmente los temas mencionados. Posteriormente se realizó el levantamiento y diseño del modelo actual, realizando varias reuniones con el equipo de la Unidad de Desarrollo quienes aportaron eficazmente con su experiencia.

Definido el mismo, se realizó el análisis FODA, a través del cual se pudo identificar sus fortalezas y debilidades. Toda esta información permitió diseñar un modelo basado en la aplicación de normas y estándares de calidad tanto para procesos como para productos, como el Modelo CMMI y SQUARE respectivamente. Finalmente se definieron métricas que ayuden al modelo propuesto a mejorar continuamente.

ABSTRACT

The aplication of software process models in our environment not is a topic importante for many organizations. The result, time lost, money and efforts in failed proyects. The Savings and Credit Co operative Jardín Azuayo is a financial institution that has grown rapidly and needs to order and redesign many its process, between them its development software process.

For this reason is important to know a brief history of how born the cooperative, for to identify the size of study universe. Others important topics are the software process models and the software quality assurance, because some don't know about it. Subsequently conducted the survey and design of the current model, with several meetings with the team of Development Unit who provided their expertise effectively.

Defined therein, SWOT analysis was performed, through which it could identify its strengths and weaknesses. All this information allowed the design a model based on the application of standards and quality standards for both processes and products, such as the CMMI Model and SQUARE, respectively. Finally, we defined metrics that help to continuously improve the proposed model.

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

CAPITULO I

CONTEXTO GENERAL DE LA COOPERATIVA DE AHORRO Y CRÉDITO "JARDÍN AZUAYO"

El presente capítulo abarca una breve reseña histórica de la creación y evolución que ha tenido la Cooperativa de Ahorro y Crédito Jardín Azuayo hasta la fecha. Se da a conocer la problemática que tiene el proceso de Desarrollo de Software dentro de la Unidad responsable de ésta actividad, así como también los objetivos y productos que se esperan conseguir con la elaboración de éste trabajo.

1 CONTEXTO GENERAL DE LA COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO

1.1 HISTORIA, PRESENTE Y FUTURO DE LA COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO

1.1.1 Breve Reseña Histórica.[9]

"El silencio no busca amplificadores; se basta consigo mismo. Con él en la conciencia y con el monte que se pasó a la otra orilla y decidió cerrar el cauce de varios ríos, para que ahoguemos la historia en un lago invasor, sentimos necesidad de revelar, antes de cualquier balance, una aproximación a la auténtica realidad que un suceso físico, por cuyas consecuencias, cambia la expresión de un pueblo".[1]

Marzo 29 de 1993, siendo las 20h30, el inmenso silencio del sector de La Josefina se vio irrumpido por un estruendoso ruido como si se tratara de una explosión, un repentino corte de energía eléctrica fue el inicio del mayor fenómeno natural nunca antes ocurrido en el Ecuador.

Esa noche el cerro Tamuga se vino abajo produciendo dos grandes deslaves, el primero taponó el sitio donde se unían los ríos Cuenca y Jadán y el segundo deslizamiento arrasó con un pequeño caserío poblado por 300 personas. El río Cuenca se represó y provocó una rápida inundación que cubrió el sector de La Josefina y San Pablo, varias casas fueron cubiertas por una enorme avalancha de piedras y lodo, mientras otras eran devoradas paulatinamente por el agua. Eran las primeras horas de tristeza, amargura y dolor de un pueblo que perdió todo, familia, casas, animales y tierras.

La montaña deslizada acumuló miles de toneladas métricas de enormes piedras y tierra, cubriendo un área de 820 metros de largo y 147 metros de altitud, nadie lo podía creer, casas, carreteras, puentes, poblados, valles y cultivos eran devorados por el agua. La magnitud del desastre fue tan grande que obligó la presencia de técnicos, expertos y funcionarios de alto nivel, inclusive el Presidente de esa época el Arq. Sixto Durán Ballén, estuvo presente para constatar la gravedad del fenómeno, un hecho que conmocionó al país y al continente.

[1]Tomado del artículo publicado por el Diario el Hoy [9]

Transcurrida la pesadilla, aquellas personas que lograron sobrevivir a esta desgracia, formaron pequeños campamentos para refugiarse y poder apoyarse mutuamente, mientras atravesaban por momentos muy difíciles.

Las aguas se calmaron y fue el momento de recobrar lo perdido, transformar la vida y activar la economía. Nació entonces la idea de formar una cooperativa, la falta de experiencia provocó que se busque ayude en un banco establecido en Babahoyo, el mismo que fue el primero en cerrar tras la crisis bancaria de 1999. Luego un equipo de expertos en cooperativismo trato de apoyarlos, pero la naturaleza nuevamente hizo de las suyas y el equipo quedó atrapado en el río Paute, lo que les desmotivó totalmente y no regresaron nunca más.

Realmente no era una de las mejores épocas para crear una cooperativa, poco antes del desastre de la Josefina, "Mi Cooperativa" había pasado por Paute llevándose todos los ahorros depositados en ella, pero esto no detuvo la esperanza de ese pueblo que buscaba reconstruir su cantón y reactivarlo económicamente. Así en febrero de 1996 ciento veinte pauteños constituyeron la cooperativa “Jardín Azuayo” con un aporte inicial de ciento setenta millones de sucres que fueron entregados por el Centro de Capacitación Campesina del Azuay - CECCA, Programa para el Mundo y Bilance; y la población pauteña.

La Cooperativa de Ahorro y Crédito "Jardín Azuayo" se formó con una visión social y el plan de recuperación post-desastre evolucionó a una visión de desarrollo estratégico de la región austral.

1.1.2 Situación actual de la COAC “Jardín Azuayo”[13]

Desde su conformación la Cooperativa de Ahorro y Crédito “Jardín Azuayo” ha tenido que enfrentar las profundas crisis del sistema financiero ecuatoriano, tal como fue el famoso “Feriado Bancario” de 1999 que provocó desconfianza en la banca y que las Cooperativas de Ahorro y Crédito experimentarán un crecimiento sistemático en casi todos sus indicadores financieros.

Actualmente la Cooperativa “Jardín Azuayo” es una de las cooperativas de ahorro y crédito más importantes de la Región del Austro y del país, opera en las provincias de Azuay, Cañar, El Oro, Loja y Morona Santiago con 31 sucursales para atención a sus socios que hasta abril del 2011 llegan a 195.082, según lo indica el resumen estadístico generado por la Unidad de Riesgos de la Cooperativa.

Figura 1. Ubicación de Agencias de la COAC “Jardín Azuayo” en el Ecuador

Fuente: http://www.jardinazuayo.fin.ec/index.php?option=com_content&view=article&id=53&Itemid=58

A continuación la figura No. 2 presenta la tendencia de crecimiento anual de los socios en la cooperativa.

Figura 2. Crecimiento Anual de Número de Socios desde 1999 hasta abril 2011

Fuente: Base de Datos de Socios de la COAC "Jardín Azuayo".

Este crecimiento impresionante provocó que la Cooperativa se diera cuenta que su estructura no era capaz de soportarlo y sienta la imperiosa necesidad de iniciar un proceso de reestructuración, el cual viene dándose desde hace aproximadamente dos años, estableciéndose una nueva estructura social, la misma que se encuentra formada por los ámbitos de Participación, Directivo y Administrativo, tal como se puede observar en la figura No. 3.

Figura 3. Estructura Social de la COAC "Jardín Azuayo"

Fuente: http://www.jardinazuayo.fin.ec/index.php?option=com_content&view=article&id=85&Itemid=29

1.1.3 Proyección Futura de la COAC "Jardín Azuayo"[10][9]

Dentro de este marco de reestructuración la Cooperativa ha establecido un Plan Estratégico hasta el año 2013 en el cual se propone ser *"Una sociedad de personas con cultura cooperativa que busca el buen vivir de las comunidades y la sociedad en general, privilegiando a los sectores populares, contando con una organización solidaria, confiable, solvente, referente del cooperativismo nacional e internacional para lo cual deberían desarrollarse actividades sociales y financieras eficientes, competitivas y de calidad, integrando pueblos y culturas".*

Para lograr esto, se han plantado los siguientes objetivos:

- Consolidar el plan de formación cooperativa para socios, directivos y empleados.

- Desarrollar e institucionalizar sistemas de información y comunicación social y financiera, tanto personalizados como masivos y sistemas de investigación y planificación institucional.
- Estructurar la Cooperativa para responder a los requerimientos de su misión y crecimiento.
- Fortalecer la capacidad de gestión de riesgos.
- Construir un sistema de gestión de la calidad.
- Diseñar estrategias de posicionamiento de productos y servicios que permitan la recirculación de los recursos financieros.

El gerente de la cooperativa señaló el rumbo, según su criterio "Es importante señalar que en este momento en la cooperativa Jardín Azuayo nos encontramos en un proceso particularmente importante, en razón de que hemos asistido a un crecimiento impresionante de la institución: tenemos casi 180 000 socios, algunos millones de dólares – propiedad de los socios - , que estamos administrando, unos 44 000 socios en este momento tienen créditos, estamos actuando en 5 provincias del sur del Ecuador y todo esto nos da una idea del tamaño aproximado de nuestra cooperativa, pero al mismo tiempo estas dimensiones reclaman nuevos desafíos para nosotros, un desarrollo mucho más grande de nuestras capacidades para hacer frente a la perspectiva de crecimiento aún mayor que se avecina en el futuro".[2]

1.2 ANTECEDENTES

El desarrollo de software de manera general se ha enmarcado únicamente en plasmar las necesidades que tiene el usuario final en un determinado lenguaje de programación, sin importar la calidad que puedan tener estos programas, ni los problemas posteriores que se presentan cuando se pasan por alto ciertos detalles no definidos en su momento.

[2] Tomado del Plan Estratégico 2009 - 2013 de la Cooperativa de Ahorro y Crédito Jardín Azuayo [10]

Proyectos fuera de tiempo, inconclusos y con pérdidas económicas, tanto para quien contrata el servicio de desarrollo de software como para quien lo ejecuta, es el resultado de realizar este proceso de una forma absolutamente empírica. De hecho, una vez implementada la *"solución"* inician los *"problemas"* de mantenimiento y soporte de un producto que se transforma en una carga para el desarrollador y un verdadero dolor de cabeza para el usuario, puesto que aquello que debía ser una herramienta de trabajo y apoyo se convierte en el principal obstáculo para el cumplimiento de sus tareas, objetivos y metas.

Todos estos inconvenientes producidos por la falta de organización, disciplina, normas y buenas prácticas que guíen el proceso de desarrollo de aplicaciones informáticas, se han convertido en situaciones muy comunes y parte de nuestra ***"Cultura de desarrollo"***. Por lo tanto, es de vital importancia empezar a cambiar dicha cultura, definiendo como meta principal el obtener ***"Software de Calidad"*** generados a través de un ***MODELO*** basado en una serie de normas, estándares y buenas prácticas de aseguramiento de la calidad, existentes en la actualidad y que estén acordes a las necesidades de la organización.

Realmente el camino por recorrer no es sencillo y mucho menos se pueden esperar resultados inmediatos, ya que esto no sólo implica tener que vender la idea a los líderes del negocio, sino principalmente convencerse uno mismo que la única manera de mejorar y empezar a cambiar ésta metodología de desarrollo caduco e ineficiente es proponiéndose firmemente cumplir al menos con una pequeña parte de las buenas prácticas de desarrollo, las mismas que irán evolucionando a medida que se vayan incorporando y utilizando en el proceso.

1.3 PLANTEAMIENTO DEL PROBLEMA

La institución para cumplir con el propósito de brindar un servicio de calidad a sus socios se soporta en la nueva estructura organizacional en la que se establecen algunas áreas, como el "***Área de Servicios Informáticos y Redes de***

Comunicación", la misma que se encuentra conformada por cinco unidades que son:

- Ingeniería de Software
- Desarrollo de Software
- Administración de Aplicativos y Base de Datos
- Infraestructura, Redes, Telecomunicaciones y Seguridad y
- Soporte a Usuarios.

La Unidad de Desarrollo de la Cooperativa actualmente genera productos sin considerar ningún tipo de estándar y documentación que soporte el trabajo realizado, el proceso que se debe seguir no está definido formalmente y carece de indicadores que le permitan evaluar el rendimiento de la misma. Estas deficiencias han generado dificultades en los tiempos de entrega sobre todo cuando el recurso humano es nuevo, así como también la dependencia de un miembro del equipo puesto que nadie más que éste tiene el conocimiento para solucionar cualquier inconveniente presentado, lo que no permite a la Unidad aportar efectivamente en el logro de los objetivos de la Institución.

1.4 JUSTIFICACIÓN

Cada unidad debe establecer los procedimientos bajo los cuales desempeñarán sus funciones para cumplir con los objetivos y metas propuestas, tanto a nivel organizacional como a nivel de unidad. Por tal razón y considerando la importancia que tiene el entregar un software que sirva de apoyo en cada una de las Áreas de la Institución, es necesario Diseñar una propuesta del Modelo de Procesos para la Unidad de Desarrollo, basándose en estándares y/o modelos de calidad que estén de acuerdo a las necesidades de la Institución y que le permitan a la Unidad de Desarrollo gestionar la calidad del producto entregado.

1.5 OBJETIVO GENERAL Y OBJETIVOS ESPECÍFICOS

1.5.1 Objetivo General.

Diseñar el Modelo de Procesos para la Unidad de Desarrollo de Software de la Cooperativa de Ahorro y Crédito “Jardín Azuayo”, basándose en normas, estándares y buenas prácticas que estén de acuerdo a las necesidades de la unidad y la organización permitiéndoles gestionar el aseguramiento de la calidad del software producido.

1.5.2 Objetivos Específicos

1. Profundizar sobre los temas relacionados con modelos de procesos de desarrollo, métricas y aseguramiento de calidad de software, de tal manera que los conceptos teóricos estén claramente entendidos.
2. Definir la situación actual del proceso de desarrollo en la Unidad de Desarrollo de Software de la Cooperativa de Ahorro y Crédito “Jardín Azuayo”.
3. Mejorar el proceso existente o diseñar un nuevo modelo de proceso, tomando como guía las normas y estándares investigados y que sean aplicables a la Cooperativa.
4. Definir una serie de métricas que permitan evaluar los resultados del Modelo de Proceso propuesto.
5. Proponer las herramientas necesarias para la aplicación del Modelo.

1.6 PRODUCTOS ESPERADOS

Los productos que se esperan entregar para la consecución de los objetivos de la unidad son:

1. Diseño del Modelo de Procesos de la Unidad
2. Métricas a Utilizar.
3. Propuesta de herramientas que se pueden utilizar para la implementación y gestión del modelo.

1.7 ALCANCE Y DELIMITACIÓN

El Diseño de la Propuesta establecerá los procedimientos a seguir desde el ingreso del requerimiento, pruebas y entrega del producto a la Unidad de Administración de Aplicativos y Base de Datos.

CAPITULO II

MODELOS DE PROCESOS DE DESARROLLO DE SOFTWARE

El presente capítulo tiene como objetivo dar a conocer de forma general sobre los distintos modelos que existen y que se puede aplicar dentro del proceso de desarrollo de software, dado que para algunos ingenieros de sistemas de nuestro medio, el tema puede resultar completamente desconocido. Con esta pequeña recopilación el lector podrá comprender mejor los temas tratados en los capítulos posteriores y sobre todo el tema principal al que hace referencia éste trabajo.

2 MODELOS DE PROCESOS DE DESARROLLO DE SOFTWARE

2.1 INTRODUCCIÓN.

En la década de los sesenta existía poco conocimiento sobre cómo desarrollar software, el modelo predominante era el de codificar y probar, los requerimientos eran ambiguos y no existían especificaciones puntuales. Los esfuerzos se centraban en la implementación de los programas y poca atención se presentaba en el diseño del software. Al final de la década, la tecnología soportaba desarrollos grandes, pero la dificultad radicaba en organizar y coordinar las actividades de programación de los proyectos, lo que produjo la denominada "crisis del software" en donde el ingeniero de software se sentía incapaz de desarrollar proyectos de una dimensión acorde con los avances tecnológicos de la época.

Es así que en 1968 el comité científico de la OTAN patrocinó una conferencia en Alemania, cuyo objetivo era el de identificar, clasificar y discutir los problemas que se producían en el desarrollo de grandes proyectos. Se formó un grupo de trabajo, presidido por F. L. Bauer, cuya misión era la evaluar técnicas para poder desarrollar un software flexible y de calidad y se constató que el desarrollo de software era una tarea de ingeniería tomando desde entonces el nombre de *"Ingeniería de Software" y* debería ser modelado de acuerdo con métodos establecidos de tal manera que se produzcan programas eficientes, fiables y robustos con el mínimo coste y tiempos posibles.

Desde entonces los modelos de procesos de desarrollo de software han ido evolucionando dependiendo del conjunto de características de un proyecto dado.

2.2 MODELOS DE PROCESOS DE DESARROLLO DE SOFTWARE

Los modelos de procesos de software representan una secuencia de actividades en red, objetos, transformaciones y eventos que incorporan estrategias para el logro de la evolución del software. Dichos modelos pueden ser utilizados para desarrollar una descripción más precisa y formal de las actividades del ciclo de vida del software. Su poder surge en la utilización de una notación lo suficientemente rica en sintaxis o semántica adecuada para el procesamiento computacional. A continuación de manera general podemos mencionar los siguientes:

- Modelo en Cascada
- Modelo por Prototipos
- Modelo iterativo e incremental
- RUP
- Modelo Espiral
- Métodos Ágiles

2.2.1 Modelo en Cascada. [2][3][4][5][6][7][11]

El modelo en Cascada introdujo una disciplina en la Ingeniería del Software y corrigió deficiencias tales como la codificación sin un previo y correcto análisis y diseño. Este modelo se ajusta bien cuando las especificaciones están bien definidas y se trata de implementar un software que satisfaga esas especificaciones. En la realidad no siempre es así puesto que hay que reconocer que muchos de los requerimientos no son comprendidos inicialmente y que las necesidades varían dinámicamente.

Este modelo fluye en una sola dirección es decir que una fase no puede iniciar sin haber terminado completamente la anterior como se muestra en la figura No. 4.

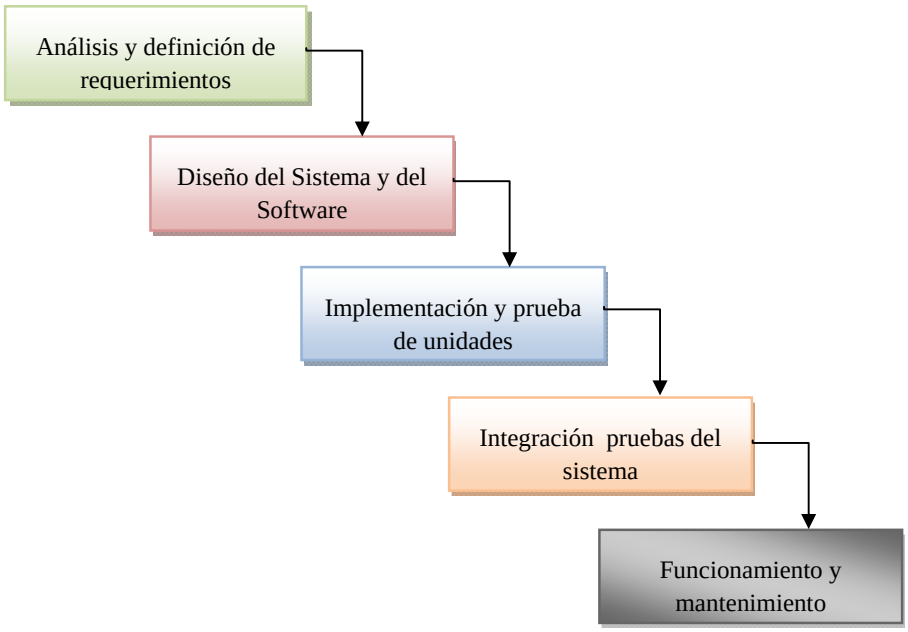


Figura 4. Modelo en Cascada

Fuente: Elaboración Propia

Las principales etapas de este modelo son:

1. **Análisis y definición de requerimientos.** A partir de consultas a los usuarios se establecen los servicios, restricciones y metas del sistema y se definen en detalle lo que se conoce como especificaciones del sistema.
2. **Diseño del sistema y del Software**. En esta fase se divide los requerimientos en hardware y software. El diseño identifica las abstracciones fundamentales del sistema software y sus relaciones.

3. **Implementación y prueba de unidades.** El diseño del software se lleva a cabo como un conjunto o unidades de programas. La prueba de unidades implica verificar que cada una cumpla su especificación.
4. **Integración y pruebas del sistema.** Los programas se integran y se prueban como un sistema completo, para asegurar que se cumplan con los requerimientos del sistema. Después de las pruebas se entrega el producto al cliente.
5. **Funcionamiento y mantenimiento.** Es la fase más larga del ciclo de vida del software. El sistema se instala y se pone en producción. El mantenimiento es una tarea que implica corregir errores que no se identificaron en las etapas anteriores y mejorar su funcionamiento una vez que se descubren nuevos requerimientos.

Ventajas:

- Cada fase se completa antes de que inicie la siguiente.
- Modelo y planificación sencillas
- Los desarrolladores conocen exactamente cada fase. lo que deben hacer y hasta dónde.
- Es el modelo más conocido y por lo tanto el más utilizado por los profesionales.
- Es aplicable a la mayoría de proyectos de gestión.

Desventajas:

- Riesgo alto en sistemas nuevos debido a problemas en el diseño y especificaciones.
- Dificultad para localizar un problema puesto que todo el diseño debe investigarse.
- No se pueden asumir con facilidad cambios en el desarrollo.
- Los resultados no se ven hasta muy avanzado el desarrollo.

2.2.2 Modelo por prototipos.[1][2][3][4][5][6][7] [11]

Los prototipos se crearon para acelerar la definición de los requisitos de software por desarrollar y se puede aplicar de las siguientes maneras:

- Se realiza un modelo en papel que indica la iteración entre el usuario y el sistema, de tal manera que sea fácil de entender para el usuario, quien puede tener una idea de cómo funcionará el sistema sin tener que esperar hasta su entrega.
- Puede simular los pasos a seguir internamente.
- Adecuar un programa real en parte.

Este modelo está formado por los siguientes pasos:

1. Recolección de los Requisitos. Se definen los objetivos globales del software y los más específicos que se desean plasmar en el prototipo.
2. Diseño Rápido. Se centra en los aspectos visibles al usuario.
3. Construcción del Prototipo.
4. Evaluación del Prototipo. Los usuarios involucrados concretan los requisitos del software a desarrollar.
5. Refinamiento del prototipo. El producto es refinado hasta que satisfaga las necesidades del usuario, al mismo tiempo que facilita un mejor conocimiento del sistema al ingeniero de software.
6. Producto. En la mayoría de los casos se desecha el programa piloto y se crea uno nuevo, por eso es importante el acuerdo con el usuario.

J.L. Connell realiza las siguientes consideraciones al modelo de prototipos:

- Es básicamente una técnica de análisis que permite completar el conjunto de requisitos funcionales de un sistema.
- Lo importante es evolucionar hasta obtener el producto final, en lugar de deshacerlo y construir un nuevo, lo que no suele ser fácil.

- Si una aplicación puede presentar el riesgo de no ser aceptada por el usuario, es una aplicación candidata para que se desarrolle un prototipo rápido.
- El prototipado rápido es una implementación incremental “top-down” en donde se desarrollan primero los niveles más externos y los detalles se dejan para más adelante, utilizando rutinas ficticias o “stubs”.
- Planificando correctamente el prototipado el 50% del esfuerzo de desarrollo, es contribución del usuario. Los equipos están formados por la mitad de usuarios y la otra mitad por desarrolladores de software.
- Es aconsejable que la primera demostración sea imperfecta, para que se pueda garantizar que el sistema final se ajusta mejor a los requisitos del usuario.

Ventajas

- Se puede aplicar en aplicaciones no muy complejas.
- Es útil cuando el usuario conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida.
- Cuando el responsable del desarrollo no está seguro de la eficacia de un algoritmo, o de la forma en la que debería realizar la interacción hombre-máquina.

Desventajas

- El prototipo es un borrador no un producto de ingeniería y puede causar falsas expectativas al usuario en el sentido de que es un producto integrado. Si los ingenieros tardan en entregar el producto, puede darse un efecto de ansiedad en los usuarios.

- Si no se tiene definido claramente el alcance del producto, los requisitos pueden ser infinitos, convirtiéndose en un proyecto sin fin.
- No es aconsejable utilizarlo para casos experimentales
- Genera una lenta gestión de desarrollo porque tiene que esperar hasta que el usuario este de acuerdo y establezca los límites.
- No se puede definir con exactitud el tiempo de desarrollo.

2.2.3 Modelo iterativo e incremental [1][3][4][5][6][7][11]

Se denomina iterativo porque se repite dentro del mismo proyecto e incremental porque se ejecuta por partes. El modelo en cascada evalúa el proyecto sobre bases no muy sólidas y con un gran margen de error, puesto que el análisis de los requisitos casi siempre son incompletos o a menudo cambian antes de haber terminado el desarrollo del software.

Si consideramos que la especificación de los requisitos es insegura e incompleta, naturalmente el costo del proyecto aumentará por retrasos del desarrollo que no estaban previstos realizar o rehacer. Es complicado que los usuarios reflexionen sobre lo que realmente quieren conseguir y sus peticiones nunca serán concretas, el modelo iterativo e incremental modifica el modelo en cascada proponiendo analizar una pequeña parte de los requisitos para realizar las etapas de diseño, programación y pruebas. Una vez que el usuario haya verificado su correcto funcionamiento se continuará con otra parte y si se parte de un software ya desarrollado los requisitos serán muchos más precisos y se podrán estimar con más fiabilidad tanto tiempo, costes y recursos.

En este modelo también se debe considerar la posibilidad de que al construir una parte, sea necesario modificar otra desarrollada previamente.

Ventajas:

- Resolución de problemas de alto riesgo en tiempos tempranos del proyecto

- Visión de avance en el desarrollo desde las etapas iniciales del desarrollo.
- Se obtiene el *feedback* del usuario lo antes posible, para orientar el desarrollo al cumplimiento de las necesidades y realizar los ajustes identificados para cumplir con los objetivos planteados.
- Proyectos con menor tasa de errores.
- Mayor productividad del equipo
- Menor cantidad de defectos.
- Mejora el aprendizaje y la experiencia del equipo.
- Permite optimizar el proceso en corto plazo.

Desventajas

- Este modelo exige un cliente involucrado en todo el curso del proyecto y en ciertas ocasiones no están dispuestos a invertir el tiempo necesario.
- Las metas deben estar claras para conocer el estado del proyecto.
- Requiere una muy buena planificación tanto administrativa como técnica.

2.2.4 Rational Unified Process (Proceso Unificado de Rational)[11]

Es un proceso que proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en un área u organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles. RUP intenta integrar todos los aspectos a tener en cuenta durante todo el ciclo de vida del software, con el objetivo de hacer abarcables tanto pequeños como grandes proyectos de software.

Las características principales de RUP son:

- Guiado por casos de uso: Un caso de uso es una facilidad que el software debe proveer a sus usuarios, estos reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- Centrado en arquitectura: Involucra los elementos más significativos del sistema y está influenciada por plataformas de software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo, pero lo suficientemente simple como para que si quitamos algo una parte importante del sistema quede sin especificar. Se representa mediante varias vistas que se centran en aspectos concretos del sistema
- Iterativo e Incremental: RUP recomienda dividir un proyecto en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. En concreto RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en los distintas actividades.
- Utilización de un único lenguaje de modelado: UML es adoptado como único lenguaje de modelado para el desarrollo de todos los modelos.
- Proceso Integrado: Se establece una estructura que abarque los ciclos, fases, flujos de trabajo, mitigación de riesgos, control de calidad, gestión del proyecto y control de configuración; el proceso unificado establece una estructura que integra todas estas facetas. Además esta estructura cubre a los vendedores y desarrolladores de herramientas para soportar la

automatización del proceso, soportar flujos individuales de trabajo, para construir los diferentes modelos e integrar el trabajo a través del ciclo de vida y a través de todos los modelos.

A continuación la figura No. 5 muestra el ciclo de vida RUP .

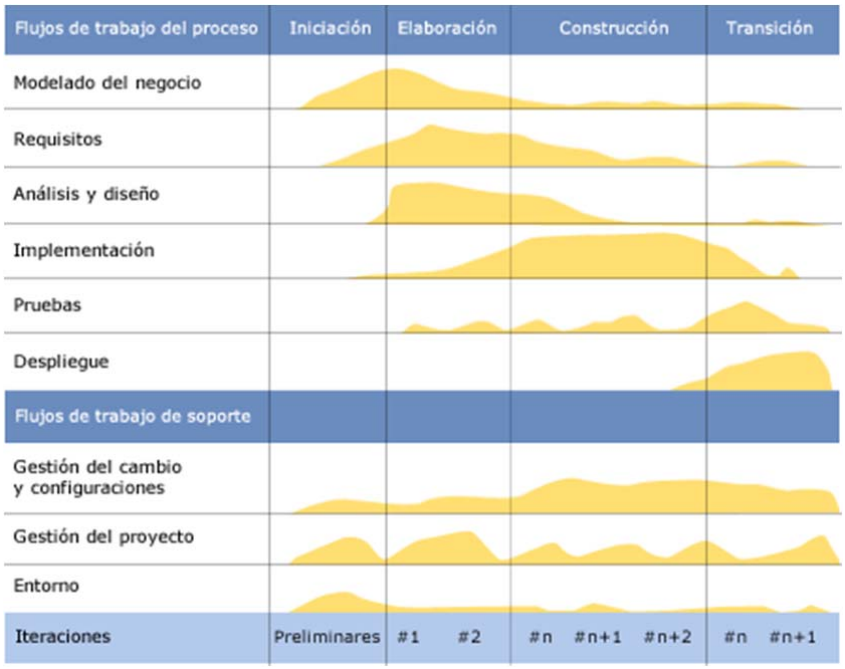


Figura 5. Modelo Rational Unified Process (RUP)

Fuente: http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational

Ventajas

- Mitigación temprana de posibles riesgos altos.
- Progreso visible en las etapas tempranas.

- El conocimiento adquirido en una iteración puede aplicarse de iteración a iteración.
- Los usuarios están involucrados continuamente.

Desventajas

- Por el grado de complejidad puede no resultar muy adecuado.
- El RUP es generalmente mal aplicado en el estilo cascada.
- Requiere conocimientos del proceso y de UML.

2.2.5 Modelo en Espiral [11][24]

Este modelo se basa en recorrer de manera controlada y cíclica por cuatro actividades que son:

1. Planificación: Se determinan los objetivos, alternativas de solución y restricciones que se dan en el proyecto.
2. Análisis de Riesgo: Se identifican los riesgos y se aplican soluciones a los mismos. Los riesgos pueden ser técnicos, administrativos, humanos, de seguridad, etc.
3. Ingeniería: Se construye el software (diseño, programación, pruebas, puesta en producción)
4. Evaluación del cliente: Esta actividad es importante debido al control de calidad y la participación de los usuarios en el desarrollo del producto.

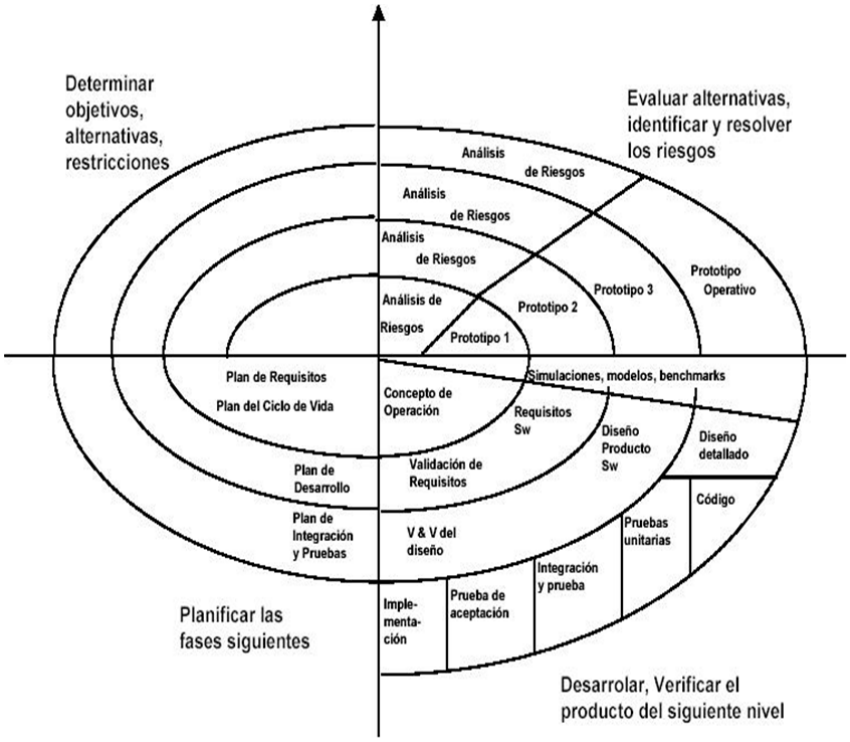


Figura 6. Modelo Espiral

Fuente: http://ingeniumetsomnia.blogspot.com/2011/04/ciclo-de-vida-de-desarrollo-de-software.html

El desarrollo del producto es evolutivo e inicia con la planificación, luego se evalúan los riesgos y si éste es muy grande, el proyecto se detiene. Posteriormente se continua con la ingeniería y la evaluación del producto. De las observaciones que resulten se iniciará un nuevo ciclo desde la planificación, una vez que el usuario dé por aprobado el producto se culmina con la puesta en producción. Ha sido muy utilizado en proyectos en los cuales la experiencia del equipo de desarrollo es nueva.

Pressman, indica que este es el modelo más “realista” para el desarrollo de software, sobre todo en proyectos de mucha complejidad.

Ventajas:

- Se centra desde el principio en lo que se puede reutilizar del software existente.
- Se adapta a la preparación para la evolución del ciclo de vida, crecimiento, y cambios del producto.
- Proporciona un mecanismo para incorporar los objetivos de calidad en el desarrollo de software.
- Provisiona un ambiente de trabajo viable para integrar tanto hardware y software.

Desventajas:

- Requiere gran habilidad para identificar riesgos. Si un riesgo no es detectado, el proyecto puede verse afectado.
- Es difícil convencer a los usuarios sobre el enfoque evolutivo, ya que no se visualiza el final del proyecto.

2.2.6 Métodos Ágiles[33]

En febrero de 2001, nace el término "ágil" aplicado al desarrollo de software, en una reunión en la que participaban 17 expertos de la industria del software, junto con creadores e impulsadores de metodologías, cuyo objetivo fue establecer principios que permitan a los desarrolladores responder rápidamente a los cambios que puedan surgir a lo largo de un proyecto, así como también ofrecer una mejora a los modelos de desarrollo tradicionales que se caracterizan por ser rígidos y guiados por la documentación generada en cada una de las etapas. Tras esta reunión se creó la organización "*The Agile Alliance*", sin ánimo de lucro, dedicada a promover el desarrollo ágil de software y ayudar a las organizaciones

para que implementen este concepto, partiendo del "Manifiesto Ágil", documento en el que se resume esta filosofía.

Lo más importante es conocer que no existen métodos o proceso ágiles, sino equipos ágiles. Un equipo que trabaja en conjunto es mucho más importante que un proceso, de hecho un nuevo proceso puede mejorar la productividad del equipo en una fracción, pero el trabajo efectivo como una unidad compacta, puede mejorar la productividad en varias ocasiones. En segundo lugar se debe mejorar la relación con el usuario, haciéndole conocer que es un miembro valioso y esencial del equipo y que puede tomar decisiones a lo largo de todo el proyecto, en lugar de que el desarrollador decida al final lo que funciona o no correctamente. Esta interrelación permite al equipo de desarrollo adquirir experiencia para encontrar una solución sencilla y correcta. El mayor problema con el desarrollo de software es que las necesidades cambian, los procesos ágiles aceptan la realidad del cambio versus la búsqueda de especificaciones completas y rígidas, existirán ámbitos en los que los requisitos no pueden cambiar, pero son casos relativamente pequeños.

Podemos aprender mucho más sobre los requisitos del proyecto si se muestra el producto en desarrollo al usuario, de tal manera que los cambios se vayan dando durante el proceso y no esperar al final del mismo. La aplicación de una metodología ágil es un cambio difícil, pero abre nuevos caminos para el desarrollo de software.

Como ejemplos de éste modelo tenemos los siguientes:

- **Manifiesto Ágil**

  Según los autores a través de este método se aprende a valorar:

- **A los individuos e interacciones sobre procesos y herramientas.** El principal factor de éxito es el ser humano y se considera mucho más

importante la construcción de un buen equipo de trabajo antes que el entorno.

- **Un software funcionando sobre documentación extensiva.** Se debe seguir la siguiente regla: “No producir documentos a menos que sean estrictamente necesarios de forma inmediata y que ayuden a tomar decisiones”.
- **La colaboración con el cliente sobre negociación contractual.** El éxito del proyecto estará marcada por la interacción entre el desarrollador y el usuario.
- **Respuesta ante el cambio sobre seguir un plan.** Es importante adquirir la habilidad de responder ante los cambios que pueden ir surgiendo durante el proyecto.

Doce principios del Manifiesto Ágil

Los principios que hacen que este proceso sea diferente de los tradicionales, se describen a continuación:

1. La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptar que los requisitos cambian, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Se entrega software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
12. En intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento.

- **Metodologías de Desarrollo Ágil**

Existen varias metodologías sin embargo se dará énfasis a la más conocida como es la Programación Extrema o Extreme Programming.

Extreme Programming (XP)[26]

Extreme Programming es una metodología ágil que hace hincapié en la satisfacción del usuario. Los administradores, clientes y desarrolladores son parte del equipo de colaboración, implementa un entorno simple, pero eficaz permitiendo a los equipos ser altamente productivos. .El equipo se auto-organiza en torno al problema para resolverlo lo más eficientemente posible.

Un proyecto de software mejora en cinco aspectos esenciales: la comunicación, la sencillez, la retroalimentación, el respeto y el coraje.

- **Gestión de las Metas**

Un proceso tradicional agenda todas las actividades una tras otra, hasta una fecha de finalización del proyecto y únicamente en ese momento se puede comprobar si el producto entregado es de agrado del usuario final.

Desafortunadamente los requisitos en la mayoría de los casos cambian antes que el proyecto finalice y es éste un factor importante que influye en los costos del proyecto.

Las metodologías ágiles aceptan dichos cambios y crean oportunidades de mejora y ventaja competitiva, tomando a cada requisito como una historia del usuario. Las tarjetas de historia son un mecanismo de bajo costo para determinar el alcance del proyecto.

El proceso ágil toma a los procesos tradicionales y les da un giro de 90˚ estimando costos por función en lugar de actividad y uno de los costos de la gestión es estar siempre preparados para nuevos requerimientos, manteniendo la calidad del código y el diseño a lo largo del proyecto.

- **Historias de usuario**

Las historias de usuario sirven como los casos de uso pero no son iguales. Se utilizan para describir brevemente las características que el sistema debe poseer, ya sean requisitos funcionales o no funcionales. Cada historia de usuario está escrita en un formato lo suficientemente comprensible sin terminología técnica y delimitada de tal manera que los programadores puedan implementarla en unas semanas.

Dichas historias también se pueden manejar en las pruebas de aceptación, para poder determinar si están correctamente implementadas.

La mayor diferencia entre los requerimientos de los modelos tradicionales y las historias de usuario es el nivel de detalle. Las historias sólo deben proporcionar los detalles suficientes para hacer una estimación de riesgo razonable. El tiempo de desarrollo ideal de una historia está entre dos y tres semanas siempre y cuando no existan otras asignaciones, ni distracciones y se conoce exactamente lo que se tiene que hacer.

Otra diferencia entre las historias y los documentos de requerimientos es el enfoque sobre las necesidades de los usuarios, se debe evitar los detalles de tecnología específica, diseños de base de datos y algoritmos.

- **Roles**
  - **Programador.** Quien produce el código del sistema.
  - **Cliente.** Escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio.
  - **Encargado de pruebas (Tester).** Ayuda al cliente a escribir las pruebas funcionales.
  - **Encargado de seguimiento (Tracker).** Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

- **Entrenador (Coach).** Es responsable del proceso global. Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- **Consultor.** Miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas.
- **Gestor (*Big boss*).** Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

- **Otras Metodologías Ágiles**

Existen otras metodologías ágiles y coinciden con los principios enunciados anteriormente, la mayoría están siendo utilizadas con éxito pero sin mayor difusión y reconocimiento. Entre esas tenemos:

SCRUM. Utilizada para proyectos con un rápido cambio de requisitos. Sus principales características son el desarrollo de software mediante iteraciones y las reuniones a lo largo proyecto, entre ellas destaca la reunión diaria de 15 minutos conocida como "*Standup Meeting*", cuya finalidad es brindar una forma sencilla para que los miembros del equipo estén enterados de lo que pasa y puedan obtener actualizaciones oportunas sobre el avance de otros, sin tener que pasar mucho tiempo en largas reuniones.

Existen dos aspectos claves de esta técnica:

1. Fomentar la discusión del progreso, planes y problemas estando de pie ya que al sentarse la gente se siente más cómoda y no existe problema en expandir algunos puntos de sus actualizaciones.

Cuando están de pie, por el contrario, existe una sensación de urgencia de realizar una conversación rápida y los miembros del equipo tienden a concentrarse en información realmente crítica. Si estas reuniones diarias son cortas, concisas y llenas de información, los miembros del equipo las apreciarán más, y no tratarán de evitarlas como lo hacen cuando se trata de reuniones tradicionales de una vez a la semana y que además duran más de tres horas.

2. Guiar al equipo hacia una mejor colaboración, fomentando que provean actualizaciones entre ellos, y no por parte del administrador del proyecto. Con esto, el equipo comienza a ver a la "*Standup Meeting*" como una herramienta para coordinar su trabajo y mantenerse al tanto de lo que hacen los otros, en lugar de verla como un mal necesario que sirve únicamente para contestar a la pregunta ¿ya terminaste?.

En los enfoques ágiles, la "*Standup meeting*" es recomendada para equipos que se encuentran físicamente en el mismo lugar, pero cuando estos equipos se vuelven altamente efectivos, la necesidad de una "*Standup meeting*" puede desaparecer (porque el equipo está en un estado de continua colaboración y a la vez actualizándose entre ellos). Los equipos distribuidos pueden usar modificaciones, como la teleconferencia.

A continuación se detallan una serie de pasos que se deben tomar en cuenta en ésta técnica:

1. **Decidir en qué proyecto es apropiado usar una "*Standup meetings*".** Sabiendo que son útiles en su mayoría, para equipos pequeños que trabajan en el mismo proyecto o parte de él, puesto que tienden a sufrir cambios y necesitan de una frecuente coordinación.

2. **Introducir el concepto de "*Standup meetings*" al equipo**, incluyendo las reglas y estructura, y por qué es importante seguir esas reglas.
3. **Realizar la primera "*Standup meeting*".** Animando a los miembros del equipo a generar "*feedback*".
4. **Recolectar información después de la reunión para utilizarla, recolectar información y usarla**. Esto mostrará el equipo que realmente algo positivo saldrá de la reunión.
5. **Llevar a cabo una reunión cada día por el resto de la semana.** Luego discutir con el equipo si realizar la reunión a diario es lo mejor, y tener una mini-retrospectiva sobre la técnica para identificar algunos ajustes necesarios para las reuniones futuras.

- *Crystal Methodologies* Se caracteriza por estar centradas en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos. El equipo de desarrollo es un factor clave, se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo en equipo definidas. Estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores, por ejemplo *Crystal Clear* (3 a 8 miembros) y *Crystal Orange* (25 a 50 miembros).

- *Dynamic Systems Development Method (DSDM).* Define el marco para desarrollar un proceso de producción de software. Sus principales características son: es un proceso iterativo e incremental y el equipo de desarrollo y el usuario trabajan juntos. Propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación.

- *Adaptive Software Development (ASD).* Iterativo, orientado a los componentes software más que a las tareas y tolerante a los cambios. El

ciclo de vida tiene tres fases esenciales: especulación, colaboración y aprendizaje.

- *Feature Driven Development (FDD).* Las iteraciones son cortas (hasta 2 semanas). Se centra en las fases de diseño e implementación del sistema partiendo de una lista de características que debe reunir el software. Los cambios se consideran riesgos, pero si se manejan adecuadamente se pueden convertir en oportunidades de mejora.

CAPITULO III

ASEGURAMIENTO DE LA CALIDAD EN DESARROLLO DE SOFTWARE

Este capítulo, como información general, menciona a los gurús de la calidad en desarrollo de software, esto como una introducción del proceso que la calidad en software ha tenido en distintas épocas y sobre la importancia que actualmente tiene el asegurar la calidad de un producto a través de la aplicación de una serie de controles, normas y estándares durante la construcción del mismo.

Las fuentes bibliográficas utilizadas en este capítulo son:

[8][14][15][16][17][18][19][20][21][22]

3 ASEGURAMIENTO DE LA CALIDAD EN DESARROLLO DE SOFTWARE

3.1 INTRODUCCIÓN A LA CALIDAD.

La “Calidad” tiene diferentes implicaciones, dependiendo del punto de vista desde donde es enfocado. Así, para un cliente consumidor de un producto, la calidad, implica que éste es libre de defectos, tiene un correcto funcionamiento, es fiable y fácil de usar, en cambio, para una persona que recibe un servicio, la calidad, implica la facilidad de obtener el servicio, la entrega de un servicio postventa o contar con la protección necesaria en caso de existir daños consecuentes. Para un proveedor de servicios, la calidad significa entre otros aspectos, el cumplimiento de plazos, la entrega del servicio ajustado a las especificaciones del cliente y el apoyo para que disfrute o utilice el producto o servicio a lo largo de su vida.

Cualquier producto o servicio que cumpla con los requisitos del cliente es definido como un “producto o servicio de calidad” y cualquier producto o servicio que no los cumple es calificado como de "mala calidad". Esta definición es totalmente aplicable al Desarrollo de Software.

3.2 EVOLUCIÓN DE LOS CONCEPTOS DE CALIDAD

Pese a que la palabra Calidad es muy antigua, su nivel de comprensión en las organizaciones se ha desarrollado en los últimos tiempos, especialmente desde la Segunda Guerra Mundial. Inicialmente, se creía que sólo el artesano podía lograr "calidad" en un producto. Sin embargo, con la Revolución Industrial se reemplazó la manufactura artesanal por la implementación de fábricas con artesanos que trabajaban en un sólo producto y el supervisor se convirtió en un factor importante para el logro de un producto de calidad. Sin embargo, la presión que existía sobre este, por fallas encontradas en el producto o pérdida de piezas para la

fabricación del mismo, hicieron que el aseguramiento de la calidad pasara a un segundo plano. Posteriormente se nombra a un inspector independiente que asegure que cada parte se encuentre en buenas condiciones, creando de esta manera una nueva profesión, junto con una serie de herramientas de inspección, técnicas y métodos.

La inspección se convirtió en un eslabón muy importante en la cadena de fabricación y sirvió muy bien por algún tiempo, pero se volvió inadecuada cuando la funcionalidad del producto llegó a ser más variada. Asegurar que cada componente estuviera bien montado no garantizaba el correcto funcionamiento de los productos. Esto sucedió especialmente en productos eléctricos, como motores y máquinas, que requerían de pruebas de funcionalidad.

Casi al mismo tiempo, la subcontratación de la fabricación de piezas comenzaron a llevarse a cabo sobre todo para la industria automotriz, situación que conllevó un nuevo problema, garantizar la calidad de los insumos. La fabricación por lotes comenzó a surgir en la misma época dando como resultado un nuevo concepto: ***"Control de Calidad".*** Una serie de nuevos estudios sobre los métodos de control de calidad entraron en vigor, incluyendo la inspección por muestreo, el control estadístico de calidad, gráficos de control, y así sucesivamente. Hasta ese momento, el énfasis en términos de calidad era el asegurar la calidad de la fabricación, pero se descubrió que los productos pueden fallar debido a defectos en su diseño, aunque la calidad en la fabricación esté excesivamente controlada.

Con el fin de lograr un mejor diseño, se hizo necesario el establecimiento de pautas por parte de los fabricantes, aprovechando la experiencia y el conocimiento de las organizaciones, lo que dio lugar a la elaboración de normas y directrices para garantizar la calidad del diseño y sus especificaciones. Estos desarrollos dieron lugar al concepto de "***Aseguramiento de la Calidad"***, una

parte integral de fabricación que incluye la inspección, pruebas y normas para el diseño.

Un impacto en la evolución de los conceptos de calidad fue la transformación que sufrieron las compañías japonesas, que pasaron de ser proveedores baratos con productos de mala calidad a fabricantes con productos de alta calidad. Esta fue una transformación revolucionaria y condujo a estudios de métodos de manufactura japonesa tales como "Control de círculos de calidad", "Zero Defectos" y "*Right First Time*".

Si bien el concepto de círculos de calidad fue recibido por otras economías, su aplicación no produjo resultados tan espectaculares. En la India, particularmente, adoptaron este concepto e invirtieron una considerable cantidad de recursos para ponerlo en práctica, pero no alcanzaron ningún resultado positivo. Sin embargo, la transformación de la calidad de los productos japoneses dio como resultado la conciencia que la calidad no es sólo responsabilidad del departamento involucrado sino de toda la organización. Si la calidad se descuida, la supervivencia de las organizaciones puede estar en juego.

Esta realidad hizo que se desarrollara el concepto de "***Gestión Total de la Calidad***", donde toda la organización debe participar en el logro de la misma, considerándola como un ingrediente fundamental en todas sus actividades. El desarrollo de la tecnología creó una nueva dimensión para ayudar a lograr la calidad: Los robots. Japón tomó nuevamente la iniciativa y los robots se desplegaron ampliamente en sus fábricas y las probabilidades de fallas humanas fueron descartadas en un gran número de operaciones con lo que la inspección se convirtió en algo innecesario.

Al no realizar inspecciones se generaba otro inconveniente, si un producto no funcionaba correctamente, había que desmontarlo, todo para determinar en donde se encontraba el problema, lo que dio lugar al concepto de ***"Calidad de Procesos".***

La figura No. 7 muestra como la Calidad ha evolucionando y tomando en cada cambio nombres diferentes.

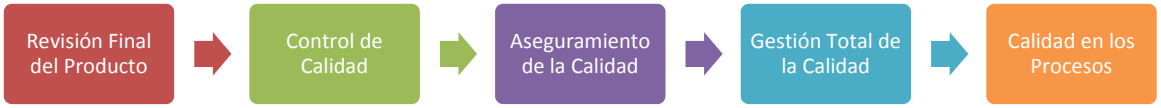


Figura 7. Evolución de la Calidad

Fuente: Elaboración propia

3.3 GURUS DE LA CALIDAD

3.3.1 William Edwards Deming

El Dr. William Edwards Deming es considerado como el padre de la filosofía moderna en la calidad. Deming fue asesor de Japón a principios de los años 50, colaborando en alcanzar el éxito en todo el mundo. En la década de 1970, la filosofía de Deming fue resumida por sus discípulos japoneses de la siguiente manera:

- Cuando las organizaciones se concentran en la calidad, ésta tiende a aumentar y los costos tienden a bajar.
- Si las organizaciones se centran en los costos, estos tienden a aumentar y la calidad tiende a disminuir.

En resumen, la calidad mejora la productividad. Esta filosofía fue probada por las compañías japoneses y está entre las mejores del mundo hoy en día. En 1981, después que la Compañía Ford Motor perdiera cerca de $ 3 millones de dólares, contrataron a Deming como consultor, quien la convirtió en la compañía de fabricantes de automóviles más rentable de América y lo logró proponiendo 14 principios fundamentales para el éxito del negocio, publicados en su libro "Cómo Salir de la Crisis", en el año 1986. Estos 14 principios se describen a continuación:

1. **Constancia en el propósito.** Crear constancia en el propósito de mejorar los productos y servicios. El objetivo es importante, y tiene que ser constante a lo largo de un período de tiempo.
2. **Adoptar la nueva filosofía.** Las condiciones cambian, y la filosofía debe estar alineada con las condiciones actuales.
3. **Inferencia estadística.** Deming promovió el uso de técnicas estadísticas para el control de la calidad en lugar de la inspección del 100% de los componentes.
4. **Precio. Deming sugiere** acabar con la práctica de la adjudicación de los contratos basándose en los precios más bajos. Este razonamiento dio origen al sistema de selección de proveedores.
5. **Mejora continua**. Encontrar los problemas y resolverlos.
6. **Capacitación en el trabajo.** Deming avocó a que las organizaciones ofrezcan oportunidades de aprendizaje en el trabajo.
7. **Instituir el liderazgo.** Deming defendió el Liderazgo en lugar de la supervisión
   en las organizaciones.
8. **Desterrar el temor.** Para Deming el miedo no debe utilizarse como un motivador
   en las organizaciones. Sugirió desterrarlo para que todos puedan trabajar eficazmente.

9. **Derribar las barreras entre los departamentos.** Deming recomienda que las personas trabajen juntas para que puedan aprender unos de otros.
10. **Métodos.** Deming recomendó el desarrollo y suministro de métodos adecuados en el trabajo para obtener los resultados esperados. Estaba en contra de las exhortaciones y las consignas y sostenía que los objetivos sin métodos para alcanzarlos, no tienen sentido.
11. **Eliminación de las cuotas.** Deming reconoció el potencial ilimitado de los seres humanos para mejorar la productividad. Por lo tanto, argumentó

    que las cuotas numéricas deben ser eliminados y sugirió la gestión por objetivos y el liderazgo con el fin de mejorar la producción.
12. **Orgullo.** Deming sostuvo que los trabajadores deben sentirse orgullosos de su trabajo cuando obtienen logros, y no se puede privar de este orgullo por evaluaciones anuales de desempeño. Sugirió la eliminación de cualquier barrera que se interponga entre los trabajadores y el orgullo de su mano de obra.
13. **Educación y reentrenamiento.** Es importante incrementar los conocimientos y mejorar el sentido de responsabilidad y pertenencia de los empleados hacia la organización.
14. **Gestión.** Sugiere una estructura de gestión para conducir los 13 puntos descritos anteriormente en la organización para lograr su transformación.

Deming considera además 4 aspectos que tienen relación con la administración y que pueden ser un tropiezo en la transformación de la organización, estos aspectos son:

1. Descuidar la planificación a largo plazo.
2. Resolver problemas basándose en la tecnología.
3. La búsqueda de métodos de probada eficacia en lugar de desarrollar nuevas soluciones
4. Excusarse diciendo siempre "nuestros problemas son diferentes".

Y por el contrario, abogó por un ciclo de cuatro pasos para la transformación de un negocio exitoso:

1. Planificar. Establecer un plan de acción.
2. Hacer. Llevar a cabo y ejecutar el plan.
3. Verificar. Verificar los resultados de la acción y hacer inferencias.
4. Actuar. Modificar plan según sea necesario.

Este ciclo es utilizado con mucho éxito en diferentes áreas de fabricación.

3.3.2 Dr. Joseph Juran Moisés

Joseph Juran inició su carrera en la fábrica *Hawthorne* de la *Western Electric* como ingeniero industrial, más tarde se convirtió en Presidente del Departamento de Ingeniería Administrativa en la Universidad de Nueva York. Fue consultor y autor de varios libros entre los que podemos destacar "*Quality Control Handbook*" que fue publicado en 1951. Desarrolló una trilogía de calidad, que puede resumirse de la siguiente manera:

1. **Planificación de la calidad.** Se inicia identificando a los clientes, sus necesidades, y las de la organización, para luego desarrollar un producto que las satisfaga. Es decir, la calidad empieza con las especificaciones y el diseño.
2. **Mejora de la calidad.** Definir un proceso de producción y optimizarlo. Es decir, la calidad depende del proceso.
3. **Control de calidad.** Probar que los procesos generen un producto satisfactorio y luego implementar el proceso probado en las operaciones.

Juran fue el primero en incorporar los aspectos humanos en la Gestión de Calidad,
lo que ayudó a dar forma al concepto de Gestión de Calidad Total y se le atribuye la definición popular de la calidad: *"La aptitud para su uso".*

3.3.3 Philip Bayard Crosby

Hombre de negocios y autor de libros como "*Quality is Free*" (1979), inició su carrera en ITT Corporation, promovió la frase popular "Hacerlo bien desde la primera vez", o DIRFT (*Do it right the first time*) y el concepto "Cero defectos". Para Crosby la administración tiene la responsabilidad primordial de garantizar la calidad en la organización.

Los cuatro principios de la calidad de Crosby son:

1. La definición de calidad está en conformidad con los requisitos, no con los conceptos de bondad o elegancia.
2. El sistema de calidad es la prevención, lo cual es preferible a las inspecciones de calidad.
3. La norma de desempeño de calidad es cero defectos.
4. La medición de la calidad es el precio de las no conformidades (calidad pobre). Es el precursor del concepto del costo de mala calidad.

Además define 14 pasos para lograr y mejorar la calidad:

1. Estar comprometidos con la calidad y asegurar que este compromiso sea claro para todos los miembros de la organización.

2. Crear equipos de mejora de la calidad, con representantes de todos los departamentos.
2. Medir el proceso para determinar los actuales y potenciales problemas de calidad.
3. Calcular el costo de la calidad o de la mala calidad.
4. Incrementar la conciencia de calidad en todos los empleados.
5. Tomar medidas visibles para corregir problemas de calidad.
6. Monitorear el progreso de la mejora de la calidad, y establecer mecanismos para monitorear el concepto de cero defectos.
7. Capacitar a los supervisores en la mejora de la calidad.
8. Días "cero defectos".
9. Alentar a los empleados para crear sus propias metas para mejorar la calidad.
10. Fomentar la comunicación entre los empleados y la administración sobre los obstáculos con la calidad.
11. Reconocer el esfuerzo de los participantes en el logro de mejora de la calidad.
12. Crear consejos de calidad.
13. Repetir el proceso de calidad.
14. La mejora continua de la calidad es interminable.

Crosby hace referencia a 5 características claves para el éxito de una organización:

1. Que la gente haga las cosas "bien desde el primer momento."
2. Aprovechar el cambio de la organización
3. Crecimiento consistente y rentable.
4. Desarrollar nuevos productos y servicios cuando sea necesario.
5. Todos son felices de trabajar en la organización.

A continuación se describen algunas de las citas más populares de Crosby:

1. "La calidad tiene que ser provocada, no controlada".
2. "La calidad es el resultado de un ambiente cuidadosamente construido culturalmente. Tiene que ser la estructura de la organización, no parte de la tela. "
3. "Hay que liderar a la gente con suavidad hacia lo que ya sabemos que es correcto".
4. "El cambio debe ser un amigo. Debería ser planificado, no por accidente".
5. "En un verdadero enfoque de cero defectos, no hay elementos importantes".

3.4 GESTIÓN TOTAL DE LA CALIDAD (*Total Quality Managment*)

La Calidad Total busca la satisfacción del cliente y obtener beneficios para todos los miembros de la organización. Por lo tanto, no pretende únicamente fabricar un producto con el objetivo de comercializarlo, sino que abarca otros aspectos tales como mejoras en las condiciones de trabajo y la formación del personal. Es una alusión a la mejora continua cuyo objetivo es lograr la calidad óptima en todas las áreas.

Kaoru Ishikawa define la Calidad Total como "Filosofía, cultura, estrategia o estilo de gerencia de una empresa según la cual todas las personas dentro de ella, estudian, practican, participan y fomentan la mejora continua de la calidad".

Al mejorar la calidad de un producto, disminuye el costo de la garantía al cliente, al igual que los gastos de revisión y mantenimiento. Si se empieza por hacer bien las cosas, los costes de los estudios tecnológicos y de la disposición de máquinas y herramientas también disminuyen, a la vez que la empresa acrecienta la confianza y la lealtad de los clientes.

En Japón la Gestión Total de Calidad incluye 4 pasos:

1. Kaizen: Enfocarse en la mejora continua de procesos para hacer de cualquier proceso de la organización visible, repetible y medible.
2. Atarimae Hinshitsu. Crear productos que funcionen tal como los clientes diseñaron la funcionalidad.
3. Kansei: Estudiar la forma como el usuario utiliza el producto, para facilitar el mejoramiento del producto.
4. Miryoketuki Hinshitsu. Crear productos con características estéticas y fáciles de utilizar.

3.4.1 Objetivos de la calidad en la organización.

Los objetivos de calidad dentro de la organización son:

1. Alcanzar y superar los puntos de referencia de la industria para la calidad del producto.
2. Alcanzar y superar los puntos de referencia de la industria para la fiabilidad del producto.
3. Reducir tiempo en inspección y pruebas y dedicarlo a actividades relacionadas con la mejora de procesos y calidad.
4. Establecer objetivos de mejora de la calidad específica de una organización.

Como se puede observar, los dos primeros objetivos se enfocan al producto y los otros dos a la organización.

El CEO es responsable de alcanzar los objetivos de la organización con respecto a todas las funciones, con las responsabilidades delegadas respectivamente a

cada uno de los administradores de menor nivel. De acuerdo a esto el departamento de calidad es el responsable de alcanzar los objetivos de calidad en la organización.

3.4.2 Escenario de la calidad en las organizaciones de Desarrollo de Software

En la industria de desarrollo de software la calidad no ha tenido la importancia debida, algunas organizaciones disponen de un departamento de pruebas independiente para llevar a cabo las pruebas del sistema y coordinar las pruebas de recepción y otras se enfocan en únicamente a conseguir certificaciones de calidad en sus procesos. Esta serie de circunstancias hace que el escenario de muchas de las organizaciones o departamentos de Desarrollo de Software presenten el siguiente escenario:

1. La mayoría de las empresas tienen un departamento de calidad únicamente de nombre y otras empresas ni siquiera disponen de uno.
2. Muy pocas organizaciones tienen un departamento de calidad con personal competente, dirigidos por un profesional experto en materia de la calidad.
3. El aseguramiento de la calidad se reduce a la ejecución de pruebas únicamente.
4. No existe auditoría de datos, métricas o estándares que garanticen la calidad del producto que ofrecen.
5. No existen metas trazadas sobre calidad.
6. No existen equipos independientes para inspección y pruebas, el mismo equipo de desarrollo realizan estas actividades.

3.5 DIMENSIONES DE LA CALIDAD

La calidad tiene cuatro dimensiones:

1. Especificaciones
2. Diseño
3. Desarrollo (Construcción de software)
4. Conformidad

3.5.1 Especificaciones

Esta es la primera actividad en la construcción de un producto o un servicio. Es una actividad netamente creativa y en la industria del software se conocen como necesidades de los usuarios, las mismas que pueden obtenerse realizando las siguientes actividades:

1. El analista de negocio redacta un informe basado en un estudio de viabilidad, en el cual se establecen los requisitos del usuario, para lo cual:
   a. Se reúne con todos los usuarios finales, jefes y administradores para anotar sus necesidades y preocupaciones.
   b. Consolida los requisitos y los presenta a usuarios finales, jefes y administradores para *feedback*.
   c. Implementa el *feedback* y finaliza las especificaciones

2. Se presenta una lista de requerimientos de usuarios como parte de la solicitud de la propuesta.

3. Se genera una propuesta del producto con las solicitudes y la personalización de cada una.

El éxito de las tareas posteriores dependen de las especificaciones, por lo tanto si éstas no tienen un buen diseño o están incompletas el resultado será un producto de baja calidad o incorrecto, y el esfuerzo en asegurar la calidad sería en vano.

Es muy importante que las especificaciones estén completas y bien definidas y que tengan en cuenta todos los aspectos posibles que inciden en la calidad del producto.

Las especificaciones deben incluir las siguientes características:

1. Funcionalidad. Especificar qué funciones se van alcanzar con el producto o servicio.
2. Capacidad. Especificar la carga que el producto puede soportar tanto en información como el número de personas que utilizará el mismo.
3. Uso adecuado. Especificar la necesidad o las necesidades que el producto o servicio satisface.
4. Fiabilidad. Especificar el tiempo de vida del producto para garantizar la prestación del servicio.
5. Seguridad. Especificar las amenazas para las cuales el producto o servicio tiene que estar preparado.

Para asegurar que las especificaciones están correctamente elaboradas deben participar personas calificadas, tales como analistas de negocios o analistas de sistemas, capacitados para llevar a cabo la ingeniería de requisitos.

Una vez listas las especificaciones con las características descritas anteriormente, el siguiente paso es asegurar la calidad de las mismas, garantizando que sean exhaustivas y que cubren todas las áreas. Para lo cual se utilizan las siguientes herramientas:

1. Documentación de Procesos para detallar las metodologías de recolección, desarrollo, análisis y finalización de las especificaciones.
2. Guías, normas, formatos y plantillas para definir el mínimo conjunto de especificaciones necesarias para el desarrollo.
3. Listas de control para asegurar la compresión de las especificaciones.

3.5.2 Diseño.

Es la transformación de las especificaciones recopiladas en documentos de diseño, los mismos que serán utilizados por los programadores para desarrollar el código fuente necesario para el producto que se está construyendo.

El diseño determina la forma y los puntos fuertes del producto. Por lo tanto, si el diseño es débil, el producto fracasará, aunque las especificaciones estén muy bien definidas.

El diseño se divide en dos fases:

- Diseño Conceptual: Es la parte creativa del proceso, por ejemplo si vamos a construir un puente, el diseño conceptual indica si es un puente colgante o un puente estático, cuántos pilares deben sostenerlo y el tráfico que fluirá sobre él. En términos de software, el diseño conceptual se refiere a la documentación de la arquitectura de software, navegación, número de niveles y las características de flexibilidad, portabilidad, facilidad de mantenimiento, etc, la misma que posteriormente será utilizada para el diseño de ingeniería.

- Diseño de Ingeniería: En esta fase se detallan las dimensiones de cada componente, en el caso del puente sería, la selección de materiales, métodos de unión, etc. En desarrollo de software se refiere al diseño de bases de datos, las especificaciones del programa, diseño de pantallas, informes, etc. Esta documentación será utilizada por los desarrolladores y contiene los siguientes elementos:

  1. Funcionalidad.
  2. Arquitectura de software

3. Navegación
4. Base de datos
5. Desarrollo
6. Despliegue de la plataforma
7. Diseño de la interfaz de usuario
8. Informe sobre el diseño
9. Seguridad
10. Tolerancia a fallos
11. Capacidad
12. Confiabilidad
13. Capacidad de mantenimiento
14. Eficiencia y concurrencia
15. Acoplamiento y cohesión
16. Las especificaciones del programa
17. Diseño de pruebas

Para asegurar que el diseño esté correcto antes de iniciar el desarrollo es importante que el personal calificado verifique que cumple con las nomas establecidas, contar con el aporte de los involucrados en el proyecto y crear un prototipo para evaluarlo son prácticas que permitirán corregirlo y mejorarlo.

Como herramientas para construir un diseño de calidad se dispone de:

1. Documentación de procesos en donde se detalle las alternativas de metodología de diseño a considerar, los criterios para la selección de la alternativa para el proyecto y la finalización del diseño conceptual.
2. Guías, normas, formatos y plantillas para especificar la posible arquitectura del software, ventajas y desventajas, la metodología para la preselección de las alternativas de diseño, y así sucesivamente.

3. Listas de control para que los diseñadores puedan asegurar que el diseño es integral y apropiado.

De esta manera se pueden desarrollar diseños claros, completos y que estén disponibles para cualquier revisión.

3.5.3 Desarrollo (Construcción de software).

El desarrollo es el acto de construir el producto de software de conformidad con el diseño. En esta etapa, el código fuente es desarrollado y se convierte en código ejecutable. El desarrollo debe cumplir con el diseño de software establecido previamente.

En esta etapa se realizan las siguientes actividades:

1. Crear las estructuras de base de datos.
2. Desarrollo de bibliotecas o rutinas comunes
3. Desarrollo de pantallas
4. Elaboración de informes
5. Implementación de los planes de pruebas (El desarrollo o creación de planes de pruebas se debe hacer en una etapa anterior, por ejemplo al diseñar)
6. Desarrollar las rutinas asociadas para asegurar todas las características de seguridad, eficiencia, tolerancia a fallos, etc.

Para lograr un desarrollo de calidad se deben predefinir las directrices de codificación, convenios de denominación, formato de código, normas de optimización, y las directrices de prevención de defectos que ayudarán al desarrollador a escribir un código confiable y libre de defectos, gestión de

cambios, gestión de configuración y por supuesto, seleccionar personal capacitado para esta etapa.

3.5.4 Conformidad.

En esencia, la conformidad de la calidad examina en qué medida el control de calidad es aplicado en la organización. Una forma de determinar la eficacia de las actividades de aseguramiento de la calidad consiste en utilizar un conjunto de métricas de calidad. Estas métricas incluyen la eficiencia de eliminación de defectos de cada uno de los controles de calidad en las actividades, la calidad del producto, y la densidad de defectos. Otra manera de determinar la eficacia de las actividades de aseguramiento de la calidad consiste en comparar los datos de referencia de las métricas de calidad de la organización con las métricas de la organización.

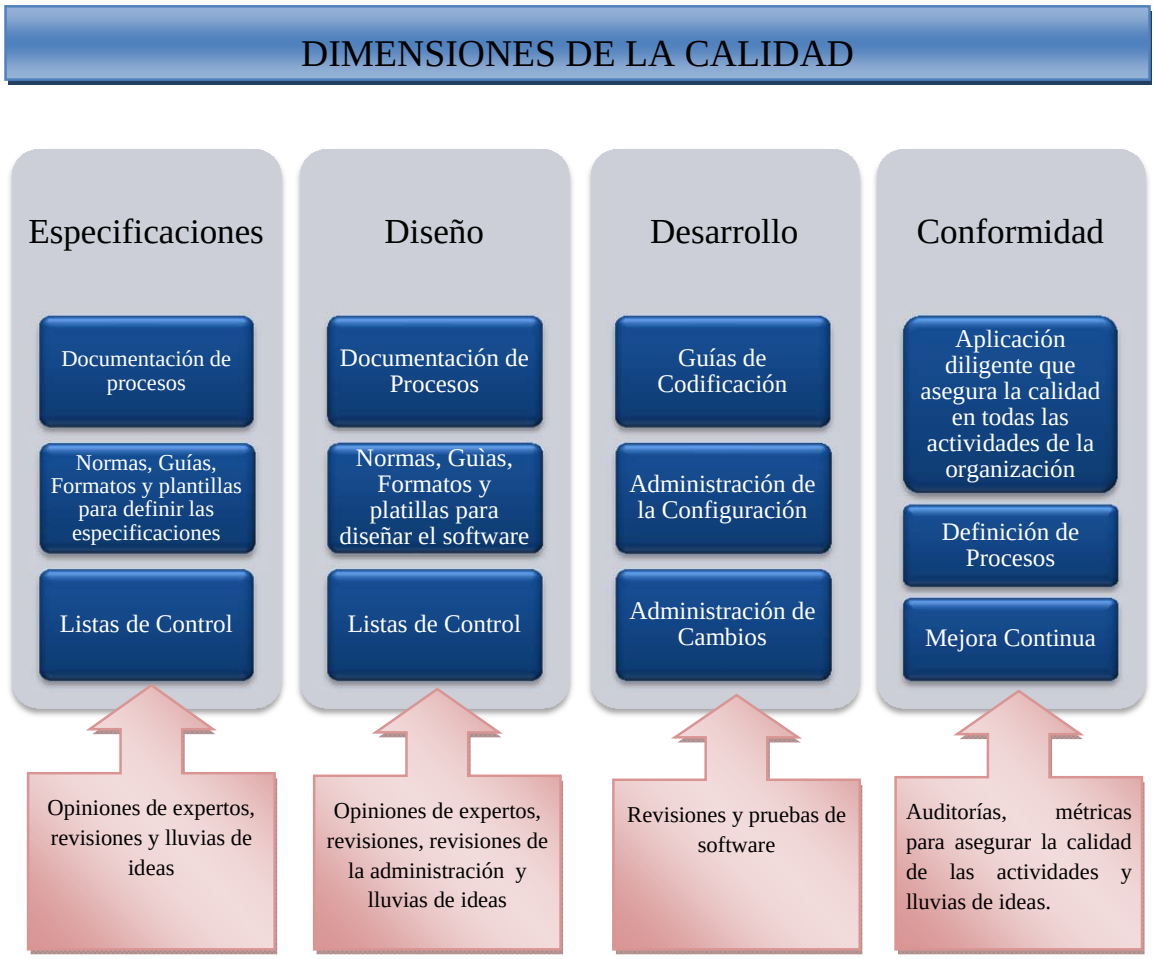


Figura 8. Dimensión de la Calidad

Fuente: Elaboración Propia

La figura No. 8 muestra cada una de las dimensiones que la Calidad debe abarcar en el desarrollo de un producto

3.6 CALIDAD EN EL PRODUCTO DE SOFTWARE

La calidad en un producto de software puede ser visto desde los siguientes puntos de vista:

3.6.1 Desde el Punto de Vista de Funcionalidad.

Es decir el producto de cumplir con las principales funcionalidades para las cuales está diseñado y sin las cuales el producto no serviría para nada Un producto tiene dos tipos de funcionalidades:

- Básicas y
- Auxiliares.

Funcionalidades Básicas. Se refieren a las principales funciones que el producto debe cumplir y sin las cuales éste, sería inútil.

Funcionalidades Auxiliares. Son las funciones que complementan a las funcionalidades básicas. Si una funcionalidad auxiliar no está presente, el producto sigue siendo útil.

Las funcionalidades auxiliares se clasifican en las siguientes subcategorías:

1. **Seguridad y Protección**. Esta funcionalidad se logra a través de las especificaciones, el diseño y las pruebas.

2. **Usabilidad.** Permite que el producto se pueda utilizar de forma conveniente. Esta funcionalidad se logra a través de directrices de usabilidad, pautas de diseño, y las directrices de arquitectura de software. Su calidad está garantizada a través de revisiones y pruebas de usabilidad.
3. **Tolerancia a fallos.** El mal uso o uso no adecuado no deben dañar el producto. El software debe tolerar a un nivel razonable. La tolerancia a fallos se logra a través de los estándares de diseño, interfaz de usuario, directrices de diseño y validación de los datos. La calidad de estas funciones está garantizada a través de pruebas negativas.
4. ***Feel-Good*.** Esta funcionalidad se logra a través de la apariencia del producto y se consigue basándose en las directrices estéticas. Se garantiza con la revisión de expertos y de la administración.
5. **Estimación.** Se logra mejorando la apariencia del producto a través de pautas de diseño de interfaz de usuario y lluvia de ideas. Su calidad se garantiza a través de las pruebas positivas.
6. **Rivalidad (Ventaja competitiva).** Superioridad ante otros productos, se logra a través de una guía de gestión. La calidad de estas funciones está garantizada a través de pruebas normales.

3.6.2 Desde el punto de vista de la Caja Blanca

Es decir desde la forma en la que fue construido el producto, por lo debería tomarse en cuenta lo siguiente

1. **Capacidad de mantenimiento.** Lo que significa que es fácil de añadir, eliminar o modificar. Para lograr esto, el código debe ser legible y comprensible por los programadores que no desarrollaron el producto.

   Es importante entonces al momento de generar el código tener presente las siguientes consideraciones:

- Utilizar estándares en las convenciones de nomenclatura para poder diferenciar entre las variables del programa, constantes, campos, etc.
- Formato del código estandarizado de tal manera que sea fácil de encontrar el principio y el fin.
- Utilizar documentación para entender la lógica aplicada.
- Utilizar construcciones simples.

2. **Portabilidad.** Significa que un producto de software puede ser pasado de una plataforma a otra. Esto se logra utilizando estándares de lenguaje de programación durante la construcción.
3. **Flexibilidad.** La flexibilidad hace que sea factible de utilizar, sin importar si algunos de los valores o parámetros cambian. Por ejemplo, si cambia el porcentaje de la tasa de interés financiera no debe cambiarse en el código.
4. **Eficiencia.** Reducir al mínimo el consumo de los recursos del sistema y tiempos de ejecución. La minimización se logra utilizando normas y guías de optimización de código como cierre de cursores, tablas, etc.
5. **Modularidad.** Se refiere a la construcción del producto de software con módulos independientes de tal manera que si un módulo cambia, éste no tiene impacto sobre otros módulos. La modularidad se logra teniendo en cuenta las guías de arquitectura de software y diseño de software.
6. **Reusabilidad.** Es poder utilizar varias veces un código fuente en otros desarrollos. Se logra aplicando guías de codificación.
7. **Legibilidad.** Facilitar el reconocimiento de la estructura codificada.
8. **Capacidad de pruebas.** Cada unidad del software deben probarse por separado, para no esperar a realizar pruebas al final cuando este el producto completamente desarrollado. La capacidad de prueba hace que sea más fácil corregir los defectos durante el desarrollo.

3.7 DEFECTOS EN EL PRODUCTO

Los defectos en algunos casos son aceptados en un producto de software, siempre y cuando no afecten de manera sustancial al mismo. Las actividades de

aseguramiento de la calidad están diseñadas para llevar a cabo una disminución gradual de éstos ya sea por especificaciones, diseño, programación o pruebas ineficientes o inadecuadas.

Dentro de este contexto es importante aclarar los siguientes conceptos:

- **Error.** Un error es un paso incorrecto en un programa, una definición incorrecta de datos ya sea en su tipo o tamaño. Un error se genera durante la etapa de desarrollo del software.
- **Defecto.** Consecuencia de un error y puede originarse en cualquiera de las etapas de especificación, diseño o desarrollo.

  Los defectos se pueden dividir en tres clases:

  - **Defectos Críticos:** Los defectos críticos causan el fracaso. Permanecen en el producto hasta que se tomen medidas correctivas para eliminarlas, estos defectos deben ser corregidos inmediatamente.

    Un "producto de calidad" no puede tener defectos críticos, puesto que los usuarios finales pueden utilizar la funcionalidad hasta que el defecto no haya sido eliminado. Los defectos críticos pueden encontrarse a través de revisiones de software, con resultados negativos, y pruebas de estrés, además de las pruebas habituales a las que se somete el producto.
  - **Defectos Mayores.** Son las condiciones de falla, mientras que no interfieran con el uso de otras funcionalidades, puede permanecer hasta que se resuelva con la mayor brevedad posible.

    Los usuarios comprenden que existieron un conjunto de especificaciones que produjeron la falla y puede continuar utilizándolo mientras se corrige, este no provoca una evaluación negativa por parte del usuario hacia el software.

- ❖ **Defectos Menores.** Son lo que no causan fracasos y no es necesario establecer medidas alternativas. El producto puede ser utilizado sin interrupciones, pero igual son defectos.

  Estos defectos aunque menores, generan también críticas sobre la calidad de un producto, puesto que hace evidente la carencia de actividades de aseguramiento de calidad. Por lo que el desarrollador debe tomarse el tiempo necesario para eliminarlos.

- **Falla.** Es la manifestación del defecto. Es la diferencia entre el resultado esperado y el real.
- **Tolerancia a fallos.** Es la propiedad que permite a un sistema continuar operando adecuadamente en caso de falla en alguno de sus componentes. La tolerancia de fallas es muy importante en aquellos sistemas que deben funcionar todo el tiempo.

  Ante una falla, otro componente o un procedimiento especial de respaldo pueden tomar el control para subsanar o amortiguar los efectos del fallo. Una forma de lograr tolerancia de fallas, es duplicar cada componente del sistema.
- **Fracaso.** Es el resultado de encontrar fallas en operación puesto que la tolerancia a fallas no está presente. El fracaso impide el paso sin problemas a otra funcionalidad y se repite hasta que las fallas no sean eliminadas del producto.

3.8 VERIFICACION DEL SOFTWARE

La verificación es una actividad que se lleva a cabo para confirmar que algo se ajusta a las especificaciones documentadas, normas, reglamentos, etc, es decir que hace exactamente lo que dice hacer y que los componentes necesarios para

su correcto funcionamiento estén presentes, lo que no implica necesariamente que este proceso confirme la funcionalidad de un objeto.

Para CMMI (*Capability Maturity Model Integration*), la verificación es *"La confirmación que los productos funcionen correctamente y reflejen los requisitos especificados para ellos. En otras palabras, la verificación asegura que lo construido este bien".*[3]

Este proceso dentro del desarrollo de software juega un papel muy importante y dejarlo de lado perjudica notablemente el resultado esperado del producto en construcción, ya que sería muy difícil detectar los siguientes inconvenientes:

- La declaración innecesaria de variables y constantes no utilizadas en el código, aparentemente esto puede ser inofensivo, pero afecta el rendimiento de la memoria RAM provocando lentitud en el sistema, especialmente en aplicaciones WEB.
- La inclusión de bibliotecas que no se utilizan, lo que aumenta el tamaño del archivo y por lo tanto ocupa espacio físico innecesario en el disco.
- Detectar algoritmos cuyo rendimiento no sea adecuado.
- Detectar estructuras de control ineficientes, que sería difícil encontrar únicamente con pruebas normales.
- Detectar código malicioso.
- Detectar código basura como mensajes temporales que el desarrollador utiliza y que le sirven únicamente a él como referencias mientras realiza su trabajo.

3.8.1 Objetivos de la Verificación.

[3] CMMI for Development, versión 1.3. Pág. 466[12]

A parte del objetivo fundamental de la verificación que es confirmar la correcta construcción de un artefacto, a continuación se describen otros objetivos que también se buscan a través de este proceso:

1. Asegurar que los artefactos creados se integren correctamente al *core* del negocio.
2. Asegurar la exhaustividad e integridad de los artefactos, es decir que cualquier aspecto que se requiere en el artefacto no se quede fuera.
3. Asegurar la Aplicación de los estándares y normas definidas en los objetos creados dentro del proyecto, para exista uniformidad en la interpretación y facilitar su posterior mantenimiento.
4. Obtener eficiencia y eficacia en los artefactos.
5. Construir artefactos con claridad y la exactitud de tal manera que sean fácilmente comprensible para otros.
6. Controlar que los artefactos tengan las seguridades necesarias.
7. Desechar funcionalidades que ya no se utilizan dentro del artefacto.

3.8.2 Técnicas de Verificación en Desarrollo de Software.

Para realizar la verificación de un producto se pueden tener algunas técnicas tales como:

1. Recorridos o Revisión por Pares
2. Inspecciones
3. Auditorías

Las mismas que se detallan a continuación:

1. **Recorridos o Revisión por pares.**

Casi nunca se puede estar seguro que los requisitos han sido plasmados exactamente en el desarrollo, de hecho en el desarrollo de software, por su naturaleza subjetiva, el proceso de calidad debe estar inherente. De ahí que se sugiere realizar la comprobación llamada “Revisión por pares”, que consiste en permitir que otros profesionales distintos a los que han creado el producto juzguen y opinen sobre el mismo. De esta manera se puede alcanzar un nivel de verificación al menos independiente del autor original.

Esto no implica que sea un consultor externo quien realice dicha verificación, se puede confiar en la profesionalidad de otros recursos ya que la finalidad es encontrar defectos, no para señalar errores de los autores, sino para identificar oportunidades de mejora. Algunos egos se pueden ver afectados durante una revisión rigurosa del diseño; y este no es el efecto que se busca. La idea es incorporar beneficios de crítica constructiva y no la de señalar errores personales.

Como todo proceso, la revisión consume tiempo ya que los diseños se envían a evaluación para que sean analizados correctamente. Posteriormente en reunión con el equipo de diseño, los auditores pueden emitir sus observaciones, abriendo un espacio de dialogo en el cual se exponen las ideas entre los involucrados. Cabe recalcar que la revisión por pares solo es posible aplicar en aquellos proyectos que estén correcta y suficientemente documentados.

Una vez realizada la revisión por pares se puede obtener un reporte que contenga la siguiente información:

1. Nombre del artefacto revisado, fecha de la revisión, los nombres del autor y de quien revisa, información relativa para el cierre de defectos, y un lugar para dar respuesta a las oportunidades o sugerencias de mejora.

2. Los defectos encontrados durante el proceso, que incluyen errores en lógica, código malicioso o código basura, variables inutilizadas, código quemado, incumplimiento de normas y estándares, etc.

3. Sugerencias y oportunidades de mejora.

Recorridos Independientes.

No es necesario que el autor del artefacto esté presente mientras se efectúa la revisión y quien realiza la misma se contactará con el autor en el caso de necesitar alguna aclaración. Sin embargo pueden existir problemas de comprensión para el revisor si existen algoritmos complejos. El autor es libre de dedicar tiempo a otra actividad, mientras se realiza este proceso.

En este tipo de revisión se consideran los siguientes pasos:

1. El autor del artefacto al terminar el desarrollo informa al líder del proyecto
   (PL-*Project Leader*) o al director del proyecto de software (SPM-*Software Project Manager*), quien planifica la revisión.
2. El PL o SPM entrega el artefacto a control de configuración.
3. El PL o SPM asigna la revisión del artefacto a un compañero
4. El artefacto es puesto a disposición del revisor.
5. El revisor chequea el artefacto, prepara el informe y se lo entrega al PL SPM.
6. El PL o SPM examina el informe y se concentra en la corrección de los defectos señalados.
7. El PL o SPM solicita al revisor que verifique si los problemas detectados fueron corregidos.
8. El revisor verifica las rectificaciones.

9. Si todos los defectos se corrigieron, el revisor registra los detalles en el informe y lo da por cerrado.
10. Si los defectos permanecen en el artefacto, los pasos 6, 7 y 8 se repite hasta que todos los defectos sean corregidos.
11. Cuando todos los defectos están cerrados, la actividad de revisión se da por completada.

Recorridos con Guía

En este tipo de revisión el autor del artefacto está presente y sirve de guía para el revisor, disminuyendo el tiempo de duración de este proceso. Sin embargo existe la posibilidad de que el autor y el revisor no lleguen a un acuerdo en las oportunidades de mejora o que el autor convenza al revisor que el defecto realmente no es un defecto, lo que desembocaría en una discusión.

En esta revisión se siguen los siguientes pasos:

1. El autor del artefacto al terminar el desarrollo informa al líder del proyecto
   líder (*PL-Project Leader*) o al director del proyecto de software (*SPM-Software Project Manager*), quien planifica la revisión.
2. El PL o SPM asigna la revisión del artefacto a un compañero
3. El autor interactúa con el revisor y establecen un tiempo para la revisión.
4. El autor explica al revisor el contenido del artefacto y su funcionamiento.
5. El revisor solicita una explicación adicional si es necesaria.
6. Cuando existe una oportunidad de mejora el revisor lo socializa con el autor para llegar a un consenso y el autor anota la corrección que debe realizarse.

7. Al final de la revisión se han observado todas las oportunidades de mejora y aceptado la manera en las que se llevarán a cabo.
8. Opcionalmente, el autor puede corregir en el momento de la revisión y dar por cerrado cada defecto descubierto.
9. El autor aplica las correcciones y cierra los defectos en concurrencia con la aceptación del revisor.
10. El revisor informa al PL o SPM que el artefacto ha pasado la revisión.
11. El PL o SPM entrega el artefacto a control de configuración.
12. Opcionalmente, el revisor puede preparar un informe de la revisión formal.

Recorridos en Grupo

Se utiliza cuando se determina que el conocimiento lo tiene más de una persona y se pueden realizar 3 tipos de revisiones grupales.

1. **Revisión Postal.** El PL o SPM asume la responsabilidad para coordinar la revisión y tiene como ventaja que los revisores pueden realizar en el lugar y el momento que deseen. Cada miembro del grupo de revisión se centra en el artefacto y pueden emitir una cantidad superior de mejoras que un solo revisor.

   La desventaja es que la revisión puede tomar más tiempo, porque el proceso de
   no se puede completar hasta que finalice el último miembro.

   Los pasos se llevan a cabo en este tipo de revisión son:

Cuando un artefacto está construido y listo para su revisión el PL SPM selecciona un equipo de revisión compuesto por miembros con un poco más de experiencia que el autor.

1. Se nombra un coordinador de revisión ya sea por el equipo de revisión
   o por el PL o SPM. A veces, el PL o SPM actúan como coordinadores de revisión.
2. El artefacto es entregado a cada miembro del equipo de revisión.
3. Cada miembro elabora su informe de revisión individual y se la entrega al coordinador de revisión.
4. El coordinador de revisión consolida los informes eliminando información duplicada y prepara un informe final de evaluación.
5. El coordinador de revisión organiza una reunión para cotejar los resultados de la revisión y preparar el informe final de evaluación.
6. El informe final se entregará al autor del artefacto.
7. El autor y coordinador de revisión se mantienen en contacto en caso de necesitar aclaraciones, se corrige y se implementa en base al informe de revisión.
8. Se dan por cerradas las correcciones
9. La revisión se completa si el artefacto se promueve a la siguiente etapa en la gestión de la configuración.

2. **Reuniones de Revisión.** Se utiliza para reducir el tiempo de revisión sin embargo tiene una desventaja y es que algunos miembros del grupo no centran su atención en el artefacto, dejando de contribuir plenamente a su
   mejora. En este tipo de revisión se realizan los siguientes pasos:

1. El PL o SPM selecciona un equipo de revisión compuesto por miembros con un poco más de experiencia que el autor (es) una vez que el artefacto esté listo.
2. El PL o SPM nombra un coordinador de revisión
3. El artefacto es entregado a cada miembro del equipo de revisión antes de la reunión.
4. El coordinador de revisión convoca a una reunión a todos los miembros del grupo.
5. Los miembros del equipo asisten a la reunión para dar sus comentarios acerca del artefacto y los cambios que deben realizarse.
6. El coordinador de la revisión recoge dichos comentarios y se discuten en la reunión.
7. El informe de revisión se concluye en la reunión.
8. Los miembros del equipo de evaluación son liberados y el coordinador de revisión sigue los siguientes pasos en el proceso.
9. El coordinador de la revisión envía el informe a la PL, o SPM, indicando los defectos que tiene que corregir el autor.
10. El autor y el coordinador de revisión deben interactuar hasta que todos los defectos sean corregidos y cerrados por el coordinador de revisión.
11. El artefacto se mueve a control de configuración.

3. **Reuniones de revisión guiadas.** Este tipo de revisión se lleva a cabo que el tipo anterior con la diferencia que los miembros del equipo no tiene que preparar previamente sus comentarios. El artefacto es presentado por su autor, y los miembros del equipo dan a conocer sus puntos de vista. El Coordinador de revisión consolida las observaciones y prepara el informe de revisión. La ventaja de este método es que el tiempo de respuesta se reduce notablemente ya que los miembros del equipo de

revisión no tienen que leer el artefacto y elaborar sus comentarios antes de la reunión. Una desventaja, es que los miembros del equipo de revisión no se pueden centrar adecuadamente para agregar valor al artefacto.

2. Inspecciones

Las inspecciones juegan un papel importante en el control de calidad del software, puesto que aseguran que estén listos todos los componentes necesarios para la siguiente etapa. La salida de una actividad de inspección se plasma en un informe de inspección que especifica si el sistema ha pasado o no las revisiones.

Las inspecciones básicas que deberían realizarse son:

1. **Inspección de la preparación del sistema de pruebas**. Esta inspección debe ser realizada por personal del departamento de control de calidad para asegurar que todos los componentes necesarios estén listos y evitar que el proceso de pruebas se detenga, provocando retrasos en el proyecto.

   Dentro de los componentes a inspeccionar están los equipos a utilizar para pruebas, servidores, software de seguridad, datos y el producto desarrollado. Una vez realizada la inspección se emite un informe en el que se define los defectos encontrados para que sean resueltos por el PL o el SPM y continuar con las pruebas.

Con esta inspección se cumple con las siguientes condiciones:

1. Planificación de pruebas y casos de prueba sometidos a control de calidad.
2. Defectos corregidos
3. Todos los involucrados en las pruebas debidamente informados.
4. Criterios para el cierre de la actividad de las pruebas aprobadas y conocidas
   por las personas afectadas.

2. **Aceptación de la inspección de la preparación del sistema de pruebas.** La aceptación de las pruebas se lleva a cabo como requisito previo para la obtención del visto bueno del usuario y la entrega del producto.. Esta etapa es crucial ya que cualquier defecto descubierto aquí, refleja la mala la calidad del producto. Por lo tanto, la inspección llevada a cabo en esta etapa asegura que todo está bien y listo para la aceptación de las pruebas.

Esta inspección contempla el aseguramiento de lo siguiente:

1. El software se ha completado en todos sus aspectos.
2. Todas las actividades planificadas de control de calidad, incluyendo otras inspecciones, revisiones, se han llevado a cabo en forma exhaustiva y todas las no conformidades se cerraron satisfactoriamente. Los datos de prueba se ha creado e inspeccionado por su idoneidad y se encuentra para ser exactos.
3. El plan de aceptación aprobado.
4. Casos de pruebas aprobados
5. Todo el hardware requerido, con la configuración adecuada según lo especificado.
6. Todo el software del sistema y middleware, así como la base de datos se han cargado correctamente en el hardware y que están funcionando como deberían.

7. Los datos de prueba están cargados y listos.
8. El usuario da la confirmación para llevar a cabo la aceptación de las pruebas.
9. Todas las entidades de la organización que están involucradas interesados están informados del calendario de aceptación de las pruebas.

La inspección puede ser realizada por el departamento de control de calidad o por el PL o el SPM quienes se encargan de entregar el informe con los defectos encontrados y de cerrarlos con sus respectivas correcciones.

3. **Inspección de Entrega.** Tiene como propósito garantizar que la entrega tenga todos los paquetes y las versiones correctas del desarrollo. Se cubren los siguientes aspectos:

1. El registro de configuración del proyecto se utiliza como referencia.
2. Se inspecciona la fecha y la hora de desarrollo para asegurarse de que está de acuerdo con el registro de configuración.
3. Se inspecciona de que todos los componentes mencionados en la nota de entrega de software, se encuentren en la entrega.
4. Todas las actividades de control de calidad previstas se llevan a cabo en todos los componentes de la entrega establecida, y todos los defectos descubiertos están cerrados
5. Los números de versión de cada uno de los componentes del sistema de la entrega se comparan con la nota de entrega de software y el registro de configuración.

Las inspecciones se pueden llevar a cabo en cualquier fase de desarrollo de software si fuese precios, en función de la naturaleza del proyecto y la necesidad.

3. Auditorías.

Las auditorías se utilizan principalmente en organizaciones que tienen un proceso de desarrollo de software definido e implementado en sus proyectos.

Las auditorías se basan en documentos de verificación en donde se compara el proyecto contra los registros de estándares o procesos definidos. Por lo general son de corta duración, con una o dos horas dedicadas a la auditoría de un proyecto o una función.

Las auditorías se utilizan como una herramienta de control de calidad asegurando que un proyecto esté siendo ejecutado de conformidad con los procesos definidos, de encontrarse desviaciones estas toman el nombre de ***No Conformidades.***

Dentro de un proyecto se pueden realizar auditorías en las siguientes fases:

- Al inicio del proyecto, para asegurar que inicia cumpliendo con los procesos establecidos y debe realizarse en todos los proyectos.
- Análisis de los requisitos de software, pero se evitará para proyectos pequeños o de corta duración. Se asegura que las actividades de control de calidad se llevaron a cabo y que las próximas fases de desarrollo de software estarán libres de problemas.
- Diseño de software, sobre todo en proyectos donde el diseño sea amplio.
- Construcción del software, para asegurar que el producto fue desarrollado con pruebas de integración revisión y se aplicaron todos los procesos establecidos.
- Pruebas, puede ser omitido proyectos pequeños, la auditoría asegura de que las pruebas del sistema se completaron con éxito y

que todos los defectos descubiertos se resolvieron satisfactoriamente en cumplimiento con el proceso de pruebas.

- Cierre del proyecto, esta auditoría se realiza para todos los proyectos, justo antes de que un proyecto sea formalmente cerrado. Se asegura de que todas las actividades de cierre del proyecto, como la documentación de las mejores y peores prácticas del proyecto, archivo de artefactos, identificación de los componentes reutilizables, etc. sean entregados satisfactoriamente. Una auditoría de cierre del proyecto asegura que la experiencia del proyecto sea compartido con el SPM.

3.9 VALIDACION

En el contexto de desarrollo de software, la validación hace referencia a las actividades realizadas en un producto de software para confirmar que todos los diseños se han construido correctamente y están trabajando de acuerdo con las especificaciones originales dando seguridad y facilidad de uso.

Para CMMI la validación es *"la confirmación que el producto o servicio cumplirá con su uso previsto. En otras palabras, la validación asegura que lo que se construyó hace las cosas bien".*[4]

Los factores que se deben aplicar para este proceso son:

a. La validación se realiza por personas independientes.

[4] CMMI for Development, versión 1.3. Pág. 466[12]

b. La validación no se realiza sólo con las especificaciones de la demandante,
   sino también con las especificaciones externas.
c. La validación es un esfuerzo planificado y coordinado realizado con el propósito de fundamentar un reclamo e infundir confianza en los actores.

3.9.1 Validación de diseños de software.

Se validan a través de revisiones por pares o grupo de expertos. Sólo si los miembros del equipo de revisión han sido cuidadosamente seleccionados hace que este método sea eficaz para la validación de los mismos. También es importante que los usuarios sean tomados en cuenta como miembros del equipo de validación.

3.9.2 Validación de las especificaciones de productos

Cuando las especificaciones no son correctas, obviamente, el producto no puede ser construido satisfactoriamente. La definición de las especificaciones del producto es el primer paso en el desarrollo de software y deben ser validados. Al igual que con el diseño, la validación de las especificaciones del producto a menudo se logran con grupos de revisión que incluyen expertos en la materia.

La ventaja de utilizar una revisión postal es que los expertos geográficamente dispersos pueden ser incluidos, aunque puede ser que tome más tiempo para obtener retroalimentación de ellos y para finalizar las especificaciones del producto.

Las reuniones de revisión para la validación de las especificaciones pueden acelerar el tiempo de respuesta de este proceso, sin embargo como se explicó anteriormente, en este tipo de reuniones el inconveniente es que los expertos están geográficamente dispersos y tienen que ser transportados a la

organización, incurriendo en un gasto para la misma. La alternativa es utilizar sólo expertos locales.

La lluvia de ideas es otra técnica que se utiliza para validar las especificaciones del producto, los expertos interesados se reúnen formalmente y validan el artefacto.

3.9.3 Validación del Producto de Software.

Las pruebas de software son la herramienta principal para la validación del producto de software final.

La Norma de *British Standards Institution* BS7925-1, lo define como *"El proceso de realización de software para comprobar si se cumplen los requisitos especificados*
y para detectar fallas".

Las pruebas son capaces de detectar errores, pero no puede confirmar que no hay otros defectos que acechan el producto de software.

Las pruebas exhaustivas implican probar todas las combinaciones posibles de entradas y salidas, y asegurar que los resultados son los correctos.

El Desarrollo de Software es un dominio donde la flexibilidad para las pruebas no tiene precedentes. El número de funciones que deben ser evaluados es tal vez el más alto en comparación con otros ámbitos, como la fabricación, incluso el costo de las pruebas iguala o supera el costo de desarrollo mismo.

La creciente complejidad y el tamaño del software han dado lugar a pruebas complejas, así como varios tipos de pruebas.

3.9.4 Fundamentos de las Pruebas

Las pruebas se fundamentan bajo seis principios:

1. Los Requerimientos de los Usuarios deben ser la base de todas las pruebas.
2. Las Pruebas de software deben ser planificadas con anterioridad antes de ser iniciadas.
3. Las Pruebas de software está sujeta al Principio de Pareto*: "el 80% de los fallos de un software es generado por un 20% del código de dicho software, mientras que el otro 80% genera tan solo un 20% de los fallos".*
4. Las Pruebas de software deberían comenzar con la unidad más pequeña de software y avanzar gradualmente hacia todo el sistema.
5. Pruebas exhaustivas del 100%, es decir, todos los casos posibles no es práctico.
6. Para que las pruebas sean más efectivas, las pruebas del software deben ser realizadas por personas independientes que no estuvieron involucrados en el desarrollo.

Existen dos tipos de técnicas para las pruebas:

1. **Pruebas de caja negra**

En este tipo de pruebas el software es considerado como una “Caja Negra” donde la lógica interna y el procesamiento de los datos no es considerado. Los datos se ingresan y las salidas entregadas por el software son comparadas con las salidas esperadas, esto determina si el mismo funciona correctamente.

Figura 9. Pruebas de Caja Negra

Fuente: “Mastering Software Quality Assurance. Best Practices,Tools and Tecniques for Software Developers”. Pág. 139[8]

La figura No. 9 muestra que la información de datos de prueba pasa por la llamada caja negra, dentro de la cual se analizarán los inconvenientes, los mismos que generan el resultado de dicha prueba.

La eficacia de las pruebas de caja negra dependen de como los casos de prueba y datos de prueba estén diseñados. Si los casos de prueba son completos y las pruebas son exhaustivas, se tiene una mejor oportunidad de detectar anomalías en el software.

A continuación se detallan los pasos a seguir para la realización de las pruebas de caja negra:

1. Preparar la unidad de pruebas de software instalando los productos entregados por el equipo de desarrollo.
2. Preparar los datos maestros que se requieren para ejecutar las pruebas.
3. Estudiar el plan de prueba y anotar los objetivos de prueba.
4. Estudiar los casos de prueba diseñados.
5. Ejecutar el programa desde la línea de comandos o la interfaz de usuario.
6. Ejecutar los casos de prueba en la secuencia especificada. Al final de cada caso de prueba registrar los resultados reales y determinar si la ejecución de pruebas fue satisfactoria o no. Anotar los resultados.
7. En caso de duda sobre los resultados se puede restablecer los datos de prueba preliminares y volver a ejecutar la prueba.
8. Después de ejecutar todos los casos de prueba, registrar los resultados reales junto conla decisión de aprobarlos o no, se debe presentar un informe al solicitante de las pruebas.

9. Se debe clarificar tanto al autor la solicitud de la prueba y a las personas involucradas sobre los defectos descubiertos durante las pruebas.
10. Cuando sea necesario, llevar a cabo las pruebas de regresión para asegurar que los defectos se resuelven de manera satisfactoria. Cuando se han resuelto, desactivar el software de la unidad de pruebas para la siguiente etapa.
11. Si las pruebas de regresión revela defectos nuevos o antiguos que no se han resuelto de manera satisfactoria, se debe repetir los pasos del 7 al 9 hasta que todos los defectos se resuelvan.

Una precaución que debe tomarse en las pruebas de recuadro negro es conservar los datos y el software inicial de las pruebas, ya que durante el transcurso de las pruebas estos pueden sufrir alteraciones.

2. Pruebas de Caja Blanca.

Las pruebas de caja blanca consideran la lógica interna del programa de software. Se trata de pasar a través por cada línea de código.

Para utilizar esta técnica, el auditor debe tener conocimiento del lenguaje de programación utilizado y debe comprender la estructura del mismo con el objetivo de asegurar la eficiencia de las estructuras de control.

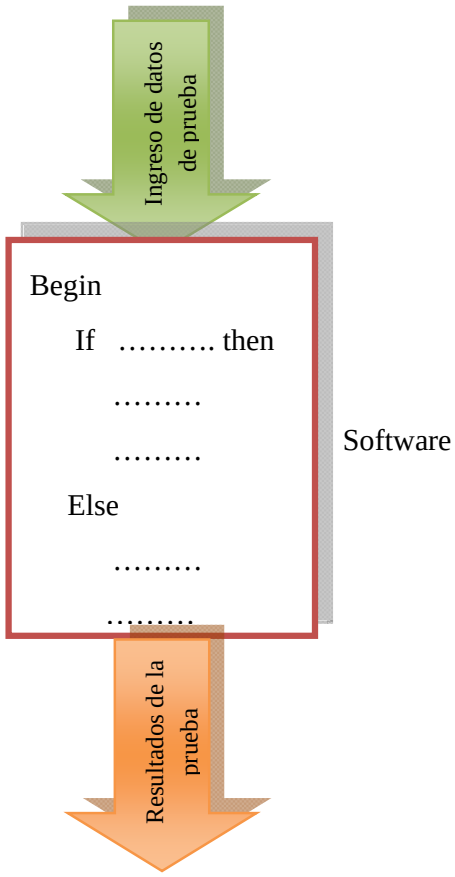


Figura 10. Pruebas de Caja Blanca

Fuente: "Mastering Software Quality Assurance. Best Practices,Tools and Tecniques for Software Developers". Pág. 141[8]

Las pruebas de caja blanca, tal como lo ilustra la figura No. 10, pueden llevarse a cabo desde los siguientes frentes:

- Línea de comandos
- Interfaz de usuario
- Programa

Si la prueba se lleva a cabo desde la línea de comandos o desde la interfaz de usuario la exhaustividad de la prueba depende de los casos de prueba para recorrer cada ruta en el software.

Si se realiza desde el programa, las herramientas con las que se desarrolla el mismo permiten llevar a cabo lo siguiente:

1. Paso a través de cada línea de código.
2. Establecer puntos de interrupción en el código durante la ejecución.
3. Establecer el valor inicial o cambiar el valor de las variables o constantes, sin la necesidad de cambiar el programa.
4. Recorrer a través de todas las estructuras de control de forma dinámica establecimiento de variables de control.
5. Detener la ejecución en cualquier punto del programa y continuar con las pruebas desde el principio o desde cualquier parte del programa.
6. Mover la ejecución de un punto a cualquier otro.

Esto permite probar a fondo el software individualmente y para poder llevarlo a cabo es necesario tener en cuenta lo que se describe a continuación:

1. Preparar los datos maestros.
2. Estudiar el plan de pruebas y los objetivos.
3. Estudiar las pautas aplicables a las pruebas.
4. Obtener listas de control aplicables y tenerlos a mano.
5. Estudiar los casos de prueba y estar listo para su ejecución.
6. Cargar la unidad de software en el sistema de pruebas.
7. Cargar la unidad de software y abrirlo con el depurador
8. Establecer los puntos de ruptura que se desee durante la ejecución del programa.
9. Ejecutar los casos de prueba y registrar los resultados actuales.
10. Determinar la aprobación o no de los resultados para cada caso de prueba mediante la comparación de los resultados reales con los resultados deseados
11. Registrar la aprobación o no de resultado.
12. Ejecutar todas las pruebas como se sugiere en las directrices de pruebas aplicables y listas de verificación y registrar los resultados.

13. Realizar el informe de pruebas y presentarlo al autor de la solicitud de pruebas.
14. Asistir al equipo en la comprensión y resolución de los defectos.
15. Realizar pruebas de regresión para asegurar la resolución satisfactoria de todos los defectos.
16. Si las pruebas de regresión revelan nuevos defectos o defectos originales que no han sido resueltos manera satisfactoria, repetir los pasos del 12 al 15.

3.10 PROCESO DE CALIDAD

Todas las actividades de la vida siguen un proceso, incluso si este proceso no es explícitamente definido o documentado. Para algunas actividades, el proceso debe cumplirse estrictamente o para otras actividades, si se omiten algunos pasos el resultado puede ser aceptable. Sin embargo el Desarrollo de Software era considerado como un trabajo creativo que no necesita manejarse a través de normas o estándares hasta que en el año de 1980 el Instituto de Eléctricos y Electrónicos lanza la serie ISO 9000 de estándares de procesos que podían ser aplicados al desarrollo de software considerándolo como una industria manufacturera.

Hoy en día, en lugar de permitir que los defectos se produzcan y utilizar inspecciones y pruebas para descubrirlos después de realizado el producto, la aplicación de procesos permite desarrollar productos libres de defectos.

Existen tres pasos para lograr un proceso de calidad y estos se detallan a continuación:

1. Definición del Proceso
2. Mejora del Proceso

3. Estabilización del Proceso.

3.10.1 Definición del proceso

El primer paso en la definición del proceso es asignar a una entidad de la organización la responsabilidad de la definición y mejora de los procesos, en algunos casos se asigna al departamento de aseguramiento de calidad quien se encarga de levantar los procesos bajo los cuales la organización trabaja.

Posteriormente se realizan reuniones en las que el departamento responsable solicita sugerencias o mejoras y evalúa el costo-beneficio de cada uno. Finalmente se implementa la real definición del proceso.

Hay dos enfoques para la definición de un proceso:

1. ***Top-Down***. Este se recomienda cuando la organización es nueva y los procesos se están creando y se compone de los siguientes pasos:

    1. Dividir las operaciones de la organización por funciones. En el primer nivel están las actividades principales de las operaciones de la organización. En el desarrollo de software tenemos en: Análisis de los requisitos, diseño de software, desarrollo, pruebas, etc. Luego cada una de las actividades de primer nivel se subdividen por ejemplo el diseño de software se divide en diseño de arquitectura, modelado de datos, diseño de base de datos, diseño de interfaz de usuario, informe del diseño, etc. y continuar dividiendo hasta que la división no entregue un valor agregado.
    2. Definir un proceso para cada actividad en el primer nivel de desglose.
    3. Definir un subproceso para el siguiente nivel si se trata de subniveles.

4. Definir un procedimiento para una actividad que no se divide en más niveles.
5. Siempre que sea posible, definir las normas y directrices para ayudar a los profesionales en la adhesión a los procedimientos.
6. Definir los modelos y plantillas para registrar y presentar la información.
7. Definir listas de comprobación para ayudar a los profesionales en la adhesión a los procedimientos y la realización de las actividades de manera integral y exhaustiva.
8. Definir medidas y analizar los resultados del proceso definido con el fin de evaluar la eficacia del mismo
9. Arreglos para la revisión del proceso definido por los profesionales en la organización para asegurarse que refleje la realidad y se defina con precisión.

.

2. ***Bottom-Up***. Este se recomienda cuando la organización ya existe y realiza operaciones desde hace algún tiempo, se compone de los siguientes pasos:

1. Estudiar cómo los profesionales están desempeñando sus funciones y documentar esta información incluyendo las prácticas más comunes y las prácticas de las personas que se conoce que generan resultados de mejor calidad.
2. Elaborar formatos para utilizar en la organización.
3. Capturar y documentar los procesos de los directores de proyectos y altos directivos. Seleccionar las mejores prácticas.
4. Organizar el material en los procesos, procedimientos, formatos, listas de control, normas y directrices.
5. Generar una versión preliminar del proceso de la organización y socializar con todos los interesados para recibir comentarios y sugerencias.

6. Después de implementar y aplicar los procesos establecidos en la versión, analizar los resultados obtenidos para mejorarlos.

3.10.2 Mejora del Proceso.

Una vez que la organización ha definido un conjunto de procesos y ha logrado alinearse con un modelo, su rendimiento debe ser monitoreado continuamente comparando el rendimiento real con el rendimiento deseado.

Es muy importante dicho monitoreo ya que la organización cambia frecuentemente por cambios en la tecnología, uso de mejores herramientas y técnicas, disponibilidad de nuevas herramientas de desarrollo, disponibilidad de nuevos productos para los niveles medios, etc. lo que hace que la actualización de los procesos deben realizarse según los cambios del entorno para poder ser más competitivos.

Un proceso para mejorar debe cubrir las siguientes áreas:

1. Eventos para la mejora de procesos. Un evento puede ser una auditoría tanto interna como externa en donde el evaluador emite un informe de la organización y sobre el cual se pueden establecer las mejoras en los procesos de la organización.
2. Fuentes de información para la mejora del proceso. Pueden ser fuentes internas, como las sugerencias de los miembros del equipo de desarrollo, gerentes y altos directivos o fuentes externas, como las sugerencias de un auditor externo o un organismo de normalización o el propietario del modelo de proceso.
3. Procedimientos para solicitud de mejora en los procesos

4. Las personas autorizadas que pueden solicitar la mejora de los procesos
5. Procedimiento para el análisis y la aceptación de las solicitudes de mejora.
6. Procedimientos para la implementación de la mejora de los procesos.
7. Implementación de los procedimientos pilotos para obtener *feedback.*
8. Implementación del *feedback* en las versiones de los procesos.

3.10.3 Estabilización del Proceso

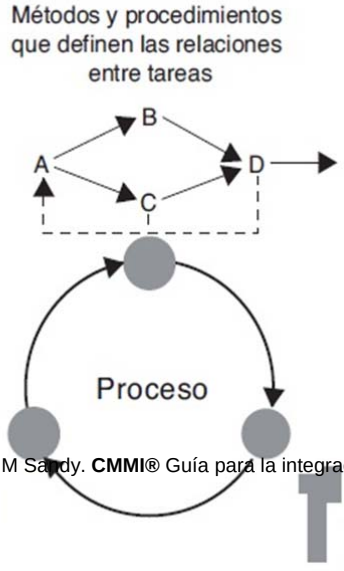
La estabilización del proceso se hace posible solo después de definirlo. Una organización por lo general pasa a través de las siguientes etapas antes de alcanzar la estabilización de los procesos:

1. **Etapa inicial.** Cuando la organización es nueva y está tratando de establecerse comercialmente. Las operaciones se realizan sobre la base de la dirección personal de los propietarios, director ejecutivo y la alta dirección.
2. **Definición del proceso**. Una vez que la organización ha logrado establecerse y define sus procesos para la realización de operaciones.
3. **Implementación del proceso.** El proceso definido se lleva a cabo en la organización. Todas las operaciones se ejecutan sobre la base de los procesos definidos y el proceso se ha institucionalizado en la organización.
4. **Mantener el proceso.** Una vez que el proceso se lleva a cabo, se monitorea y se mejora según se requiera basándose en los siguientes pasos:
   a. Analizar los resultados de las operaciones como aumento o disminución de la productividad, etc.
   b. Realizar análisis de la causa raíz de las variaciones sobre todo cuando los resultados no son los deseados
   c. Mejorar los procesos pilotos para observar la eficacia de las mejoras. Si los resultados no producen las mejoras deseadas, se repiten los pasos b y ce hasta lograr las mejoras deseadas.
   d. Implementar la mejora de procesos. Una vez que el proceso piloto haya mejorado su nivel, se implemente en el proceso real.

5. **Proceso estable.** Cuando la mayoría de las variaciones se deben a causas aleatorias (errores de azar), entonces el proceso se considera estable.

3.11 PROCESOS DE DESARROLLO DE SOFTWARE

"Un enfoque centrado en el proceso proporciona la infraestructura necesaria para hacer frente a este mundo en constante evolución, maximizar la productividad de las personas y utilizar la tecnología con el fin de ser más competitivo."[5]



[5] CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da Ed. 2009. 4p.[12]

Figura 11. Las Tres Dimensiones Crítica
Fuente: CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 4p.[12]

Un proceso bien definido e institucionalizado es un requisito previo para fomentar la cultura de calidad en una organización. Para el Desarrollo de Software tenemos cuatro procesos básicos que son:

1. **Ingeniería de software.** En dónde se define cómo se entregará el desarrollo, por lo general se componen de la gestión de los requisitos, diseño del software, desarrollo e implementación.
2. **Aseguramiento de calidad en los procesos.** Define cómo la calidad se integra en el desarrollo y se compone generalmente de verificación, validación, inspección, medición, análisis y auditorías.
3. **Administración de los Procesos.** Se define como los demás procesos son administrados, tales como el proceso de dirección del proyecto, el proceso de inicio del proyecto, estimación del software, proceso de planificación, gestión de configuración, gestión de calidad, gestión del trabajo, gestión de recursos, gestión de expectativas de los interesados y el proceso de cierre del proyecto.
4. **Procesos de Soporte.** Apoyan la ejecución del proyecto y se compone de la red, sistemas de administración de procesos, proceso de recursos humanos, proceso de gestión de instalaciones, etc.

3.11.1 Componentes de un proceso de desarrollo de Software

1. **Procedimientos**. Son las instrucciones para la realización de una sub actividad de un proceso, como por ejemplo procedimiento de planificación, procedimiento de estimación de software, procedimiento de auditoría, procedimiento de entrega de informes, etc.
2. **Normas y directrices.** Es definir una manera común de construcción de los artefactos en la organización para que la salida sea uniforme por ejemplo normas y directrices de pantalla, directrices de diseño, directrices de diseño base de datos, estándares de nomenclatura, directrices de codificación, normas de análisis de defectos, etc.
3. **Formatos y plantillas.** Utilizar formatos y plantillas para presentar la información de manera uniforme, por ejemplo plantillas de presentación, formularios de no conformidades, formularios de revisión de defectos, etc.
4. **Listas de verificación.** Esto ayuda a que la persona realice una actividad completa y ayuda a los revisores a asegurar la integridad de la revisión. Una lista de control contiene una serie de ítems y junto a cada ítem un "sí", "no" o "no aplica". Cuando una actividad se ha completado, esta lista deben tener una referencia de orden para asegurar que todos los puntos han sido revisados.

3.12 CMMI PARA DESARROLLO[12]

"La calidad de un sistema o de un producto está muy influenciada por la calidad del proceso empleado para desarrollarlo y para mantenerlo"[6].

"El CMMI se concentra en la mejora de los procesos de una organización. Contienen los elementos esenciales de eficacia de los procesos en una o más disciplinas y describen un camino de mejora evolutivo que permite pasar desde

[6] CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 5p.[12]

procesos inmaduros ad hoc a procesos disciplinados y maduros de mejor calidad y más eficaces."[7]

La SEI (*Software Engineering Institute*) ha definido al CMMI *(Capability Maturity Model Integration*) para Desarrollo como un modelo de referencia que se basa en buenas prácticas relacionadas con actividades de desarrollo de productos y servicios, cubriendo el ciclo de vida del producto desde su concepción hasta la entrega del mismo.

3.12.1 Componentes de CMMI

- **Área de Proceso:** Son un grupo de prácticas relacionadas en un área que al ser implementadas satisfacen sus objetivos. Hay 22 áreas de proceso:

1. Análisis causal y resolución (CAR).
2. Gestión de configuración (CM).
3. Análisis de decisiones y resolución (DAR).
4. Gestión integrada del proyecto + IPPD (IPM + IPPD)$_1$.
5. Medición y análisis (MA).
6. Innovación y despliegue en la organización (OID).
7. Definición de procesos de la organización + IPPD (OPD + IPPD)$_1$.
8. Enfoque en procesos de la organización (OPF).
9. Rendimiento del proceso de la organización (OPP).
10. Formación organizativa (OT).
11. Integración de producto (PI).
12. Monitorización y control del proyecto (PMC).
13. Planificación de proyecto (PP).
14. Aseguramiento de la calidad de proceso y de producto (PPQA).
15. Gestión cuantitativa de proyecto (QPM).
16. Desarrollo de requerimientos (RD).

[7] CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 8p.[12]

17. Gestión de requerimientos (REQM).
18. Gestión de riesgos (RSKM).
19. Gestión de acuerdos con proveedores (SAM).
20. Solución técnica (TS).
21. Validación (VAL).
22. Verificación (VER).

- **Declaración de Propósitos:** Describe la finalidad del área de proceso y es un componente informativo.
- **Notas Introductorias:** Describe los conceptos principales cubiertos por el área de proceso. Es un componente informativo.
- **Áreas de Proceso relacionadas:** Lista las referencias a áreas de proceso que están en relación y refleja las relaciones de alto nivel entre las áreas de proceso. Es un componente informativo.
- **Metas Específicas:** Describe las características únicas que deben estar presentes para satisfacer el área de proceso. Es un componente requerido utilizado en las evaluaciones para determinar si se satisface un área de proceso.
- **Metas Genéricas:** Describe las características que deben estar presentes para institucionalizar los procesos que implementan un área de proceso. Es un componente requerido y se utiliza en evaluaciones para determinar si se satisface un área de proceso.
- **Resúmenes de Metas específicas y prácticas específicas:** Resumen de alto nivel de las metas específicas, que son componentes requeridos, y de las prácticas específicas, que son componentes esperados. Es un componente informativo.
- **Prácticas específicas:** Describe una actividad que se considera importante para alcanzar la meta específica asociada. Las prácticas específicas describen las actividades que se espera que produzcan la consecución de las metas específicas de un área de proceso. Es un componente esperado del modelo.

- **Productos de Trabajo Típicos:** Lista muestras de resultados de una práctica específica. Es un componente informativo del modelo.
- **Subprácticas:** Es una descripción detallada que proporciona una guía para interpretar e implantar una práctica específica o genérica. Es un componente informativo indicado sólo para proporcionar ideas que puedan ser útiles para la mejora de proceso.
- **Prácticas Genéricas:** Es la descripción de una actividad que se considera importante para el logro de la meta genérica asociada. Es un componente esperado. **Elaboraciones de las Prácticas Genéricas:** Aparece después de una práctica genérica en un área de proceso, y proporciona una guía sobre cómo la práctica genérica debería aplicarse de forma exclusiva al área de proceso. Es un componente informativo**.**

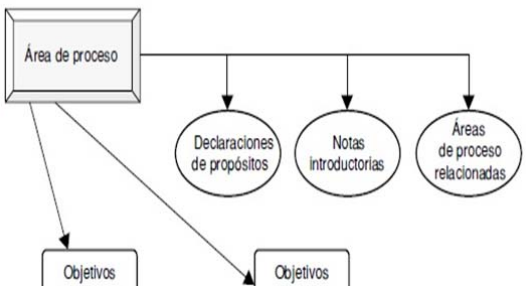


Figura 12. Componentes del Modelo CMMI
Fuente: CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 31-36p.[12]

3.12.2 Representaciones del Modelo CMMI

1. **Representación Continua:** Selecciona una área de proceso (o un grupo de áreas de proceso) y mejora los procesos relacionados con ésta. Esta representación utiliza niveles de capacidad para caracterizar la mejora concerniente a un área de proceso individual. Si se conoce previamente los procesos que se necesitan ser mejorados y las dependencias existentes entre las áreas de proceso descritas en el CMMI, la representación continua es correcta.
2. **Representación por Etapas:** Esta utiliza conjuntos predefinidos de áreas de proceso para definir un camino de mejora para una organización. Este camino de mejora se caracteriza por diversos niveles de madurez con un conjunto de áreas de proceso que caracterizan diferentes comportamientos organizativos. Si no se conoce por dónde comenzar ni qué procesos elegir para mejorar, la representación por etapas es la apropiada.

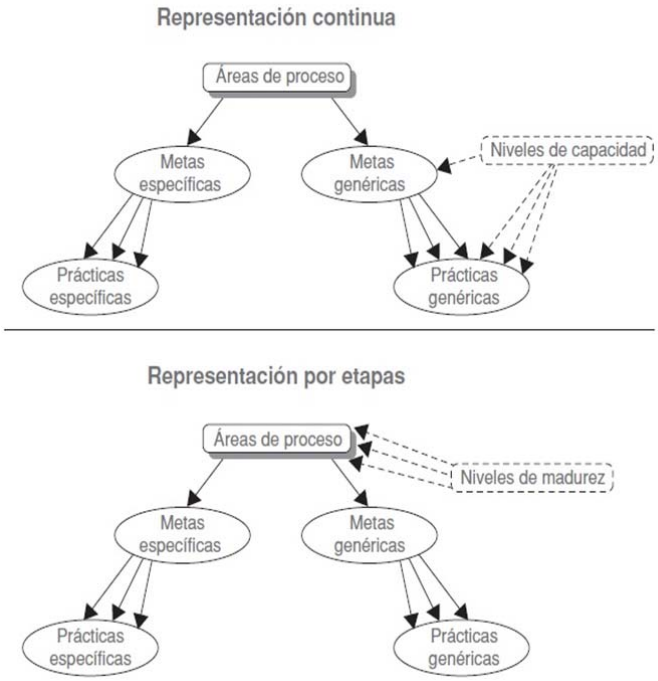


Figura 13. Estructuras de las representaciones continua y por etapas
Fuente: CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 45p.[12]

A continuación se presentan los niveles de cada representación:

Nivel	*Representación continua* *Niveles de capacidad*	*Representación por etapas* *Niveles de madurez*
Nivel 0	Incompleto	N/A
Nivel 1	Realizado	Inicial
Nivel 2	Gestionado	Gestionado
Nivel 3	Definido	Definido
Nivel 4	Gestionado cuantitativamente	Gestionado cuantitativamente
Nivel 5	En optimización	En optimización

Figura 14. Comparación de los niveles de capacidad y madurez
Fuente: CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 45p.[12]

3.12.3 Categorías

Las áreas de proceso se puede agrupar e 4 categorías:

- Gestión de procesos.
- Gestión de proyectos.
- Ingeniería.
- Soporte.

Área de proceso	*Categoría*	*Nivel de madurez*
Análisis causal y resolución – CAR	Soporte	5
Gestión de configuración – CM	Soporte	2
Análisis de decisiones y resolución – DAR	Soporte	3
Gestión integrada de proyecto + IPPD – IPM + IPPD	Gestión de proyectos	3
Medición y análisis – MA	Soporte	2
Innovación y despliegue en la organización — OID	Gestión de proyectos	5
Definición de procesos de la organización + IPPD —OPD + IPPD	Gestión de proyectos	3
Enfoque en procesos de la organización — OPF	Gestión de proyectos	3
Rendimiento de procesos de la organización — OPP	Gestión de proyectos	4
Formación organizativa — OT	Gestión de proyectos	3
Integración de producto — PI	Ingeniería	3
Monitorización y control de proyecto — PMC	Gestión de proyectos	2
Planificación de proyecto — PP	Gestión de proyectos	2
Aseguramiento de la calidad de proceso y de producto — PPQA	Soporte	2
Gestión cuantitativa de proyecto — QPM	Gestión de proyectos	4
Desarrollo de requerimientos — RD	Ingeniería	3
Gestión de requerimientos — RD	Ingeniería	2
Gestión de riesgos — RSKM	Gestión de proyectos	3
Gestión de acuerdos con proveedores — SAM	Gestión de proyectos	2
Solución técnica — TS	Ingeniería	3
Validación — VAL	Ingeniería	3
Verificación — VER	Ingeniería	3

Figura 15. Areas de Proceso, sus categorías y niveles de madurez
Fuente: CHRISSIS, Mary B., KONRAD Mike, SHRUM Sandy. **CMMI®** Guía para la integración de procesos y la mejora de productos 2da.Ed. 2009. 63p.[12]

3.13 NORMA DE CALIDAD SQUARE (ISO 25000) (PRODUCTO).[23][34]

Tanto la calidad del producto como la calidad del proceso en el desarrollo de software son aspectos muy importantes. La Norma ISO 25000 proporciona una guía para el uso de las nuevas series de estándares internacionales llamados

Requisitos y Evaluación de Calidad de Productos de Software *(SQuaRE*) cuyo objetivo principal es guiar el desarrollo de los productos de software estableciendo especificaciones, métricas y evaluación. Está formada por las divisiones :

Figura 16. Divisiones de la Norma de Calidad SQUARE

Fuente: "Normas Square Software Product Quality Requirements and Evaluation".[34]

Como se puede observar en la figura 16 la Norma

- ISO/IEC 2500n División de Gestión de Calidad. Los Estándares que forman ésta división definen todos los modelos comunes, términos y referencias a los que hacen referencia las demás divisiones de *SQuaRE*.
- ISO/IEC 2501n. División del Modelo de Calidad. Presenta un modelo de calidad a detalle donde se incluyen características para calidad interna, externa y en uso.
- ISO/IEC 2502n. División de Mediciones de Calidad. Incluye un modelo de referencia de calidad del producto de software, definiciones de métricas de calidad y una guía para su aplicación. Presenta métricas para calidad de software interna, externa y en uso.

- ISO/IEC 2503n. División de Requisitos de Calidad. Los estándares que forman ésta división ayudan a especificar los requisitos de calidad, los mismos que se utilizan en el proceso de especificación de requisitos de calidad para un producto que va a desarrollarse o cómo entrada para un proceso de evaluación. El proceso de definición de requisitos se guía por el establecido en la norma ISO/IEC 15288.
- ISO/IEC 2504n División de evaluación de la calidad. Proporcionan requisitos, recomendaciones y guías para la evaluación de un producto.
- ISO/IEC 25050-25099. Estándares de extesión *SQuaRE*. Proporciona requisitos para la calidad de productos de software *"Off-The_Self".*

Los beneficios que proporciona utilizar *SQuaRE* son:

1. El modelo representa la calidad esperada del producto de software.
2. Planteo del desdoblamiento de las necesidades o expectativas en calidad en uso, calidad externa y calidad interna.
3. Permite una mayor eficacia en la definición del software.
4. Plantea la evaluación de productos intermedios.
5. Propone una calidad final a través de las evaluaciones intermedias.
6. Permite efectuar un rastreo entre las expectativas, requisitos y medidas de evaluación y
7. Mejora la calidad del producto.

El modelo de calidad se divide en dos partes:

- Calidad interna y Calidad externa: Especifica 8 características que se subdividen en subcaracterísticas. Estas se manifiestan externamente cuando el software es utilizado y son el resultado de las cualidades internas del software.

  Las características son:

1. Funcionalidad
2. Seguridad
3. Interoperabilidad
4. Fiabilidad
5. Usabilidad
6. Eficiencia
7. Mantenibilidad
8. Portabilidad

La calidad interna del software se relaciona con las propiedades estáticas del software y la calidad externa proporciona una “caja negra” tratando las características del producto que están disponibles en el desarrollo.

- Calidad en uso: Se especifica en 6 características de calidad en uso que son:
    1. Usabilidad
    2. Contexto
    3. Protección
    4. Seguridad
    5. Soporte
    6. Adaptabilidad

La calidad interna tiene impacto en la calidad externa y ésta a su vez en la calidad en uso.

Figura 17. Modelo de la Calidad

3.14 MÉTRICAS

Las métricas sirven para cuantificar algo. La cuantificación facilita la comparación de algo contra cosas de similares características y especificar si es igual, menor o mayor entre los objetos comparados. Ahora bien cómo se mide la calidad de un software?.

El Software es un tipo de producto especial, ya que no se puede dividir en partes , ni sufre deterioro a través del tiempo, sin embargo debe ser medido. Su calidad es tan buena como los requisitos que detallan el problema, el diseño de la solución, el código y las pruebas que se ejecutan para detectar errores. De ahí la importancia de medir objetivamente la calidad y los modelos de diseño, así como el código fuente y los casos de pruebas que se establecen.

La aplicación continua de mediciones en el proceso de desarrollo de software y sus productos permite suministrar información relevante a tiempo, y por lo tanto una gestión eficaz del mismo.

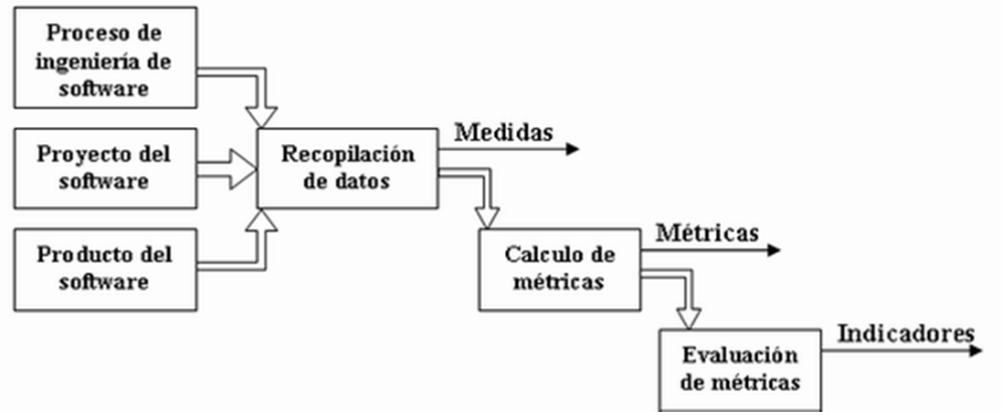


Figura 18. Proceso de Recopilación de Métricas

Fuente: http://www.monografias.com/trabajos55/proceso-de-desarrollo-software/proceso-de-desarrollo-software2.shtml

Las métricas del Software comprenden un amplio rango de actividades y se han definido dos aspectos importantes bajo los cuales se pueden enfocar las mismas:

- **Métricas del Proceso:** Estas permiten obtener un conjunto de indicadores del proceso de software, que conduzcan a la mejora del mismo a largo plazo. Estas métricas se utilizan con fines estratégicos.

- **Métricas del Producto:** Permiten valorar el estado de un proyecto en curso, así como también rastrear los riesgos potenciales y descubrir las aéreas problema antes que se vuelvan “criticas”, también permiten ajustar el flujo de trabajo o las tareas y evaluar la habilidad del equipo del proyecto. Las métricas del proyecto se usan con fines tácticos.

CAPITULO IV

SITUACIÓN ACTUAL (*AS-IS*) DE LOS PROCESOS DE LA UNIDAD DE DESARROLLO DE SOFTWARE DE LA COOPERATIVA DE AHORRO Y CRÉDITO "JARDÍN AZUAYO".

El presente capítulo contiene los procesos y procedimientos que actualmente se utilizan en la Cooperativa "Jardín Azuayo" para desarrollar el software requerido, se realiza un breve análisis del cumplimiento de las órdenes ingresadas a la Unidad y se define el FODA del proceso como tal, con el objetivo de establecer claramente la situación (AS-IS) del mismo.

Con esta información y la recopilación de los conceptos teóricos de los capítulos anteriores, se define la propuesta de un nuevo modelo en el proceso de desarrollo de software de la cooperativa, el cual espera mejorar tanto la manera de construcción del software, como del producto final.

4 SITUACIÓN ACTUAL (*AS-IS*) DE LOS PROCESOS DE LA UNIDAD DE DESARROLLO DE SOFTWARE DE LA COOPERATIVA DE AHORRO Y CRÉDITO "JARDÍN AZUAYO".

4.1 ANTECEDENTES

Como se mencionó en el capítulo No. 1, la Cooperativa de Ahorro y Crédito Jardín Azuayo, en su compromiso de brindar un buen servicio a sus socios y a la comunidad, ha buscado siempre el apoyo de herramientas tecnológicas que le permitan cumplir con este objetivo. Es así como inició el proceso de selección para la compra de un software que en permitiera llevar el control de las transacciones realizadas de cada uno de sus socios. CR Soluciones, hoy en día CompuFácil, fue la empresa escogida para el desarrollo e implementación de dicho software.

La plataforma escogida fue *Fox Pro* Lan para DOS y el desarrollo estuvo a cargo de una sola persona, el Ing. David Ávila. Para esto la Cooperativa contaba únicamente con dos equipos, uno para desarrollo y otro para atención al socio, sin embargo el primer producto entregado al cabo de dos meses desde la contratación fue el Módulo de Libretas de Ahorro, el mismo que se mantuvo durante 4 meses en producción sin ningún cambio.

El incremento de socios a la cooperativa, hizo que ésta se viera en la necesidad de adquirir dos nuevos equipos con los cuales se formó una pequeña red, contratando al Ing. Ávila como parte de la institución, para que se encargue del soporte tanto de software como de hardware de la Cooperativa.

Al cumplir un año de fundación, nace la primera oficina y las necesidades operativas cambian, ya que se debía consolidar la información de la oficina con la información de la matriz, utilizando en varias ocasiones conexión *Dial UP* y *módems*, sin tener resultados favorables debido a la pésima calidad de la comunicación, por esto, se decide adquirir e instalar en cada equipo, unidades de CD *Writer* con el fin de respaldar la información para que sea enviada cada 3 meses a la matriz y realizar la respectiva consolidación, esto implicaba un retraso en la Contabilidad de mínimo uno o dos meses.

La Cooperativa seguía creciendo y con 5 oficinas ya establecidas se hizo necesario mejorar el software a través de la integración de la información, optando por *Visual Basic* para desarrollar un aplicativo que brindara estas facilidades. Se contrata entonces una empresa dedicada al desarrollo de software en la ciudad de Cuenca, quien establece un año como tiempo de entrega del producto, situación que en la realidad no se dio debido a la contratación de los desarrolladores de dicha empresa para agilizar el proceso, formándose de esta manera el primer equipo de desarrollo como tal, dentro de la Institución y el Ing. Ávila pasa a cubrir el área de redes y comunicaciones, la misma que se encontraba sin la atención requerida.

Alrededor del año 2004 ya se contaba con una red integrada de datos y telefonía, sin embargo el alto consumo de ancho de banda por la base de datos utilizada, complicaba el normal desenvolvimiento de las tareas, y se cambia la plataforma a una base de datos más robusta como Oracle incluyendo sus herramientas de desarrollo como *Forms y Report*.

En el año 2010 la Cooperativa inicia un cambio interno enfocándose en definir y mejorar sus procesos, motivo por el cual se replantea su estructura organizacional. Inmerso en este proceso se genera la división del área tecnológica en 5 unidades y se establece mejorar el software utilizado, para lo cual se contrata una consultoría la misma que sugiere la contratación de un core desarrollado por una empresa de la ciudad de Quito cuya propuesta fue desarrollar el software en Punto Net, utilizando Oracle como Base de Datos. El acuerdo y la contratación se realizan en el año 2011.

Con estos antecedentes y considerando que las unidades del Área Tecnológica son relativamente nuevas, es importante definir los lineamientos que regirán cada una, de aquí que el presente capítulo se enfoca en la Unidad de Desarrollo de Software con el propósito de analizar y proponer un modelo que se ajuste a las necesidades propias como institucionales y que le permita ser mucho más efectiva, eficiente y eficaz para enfrentar los retos propuestos.

4.2 PROCESOS ACTUALES (AS-IS) DE LA UNIDAD DE DESARROLLO DE LA COOPERATIVA JARDINA AZUAYO.

"No podemos cambiar aquello que no conocemos"

Según la Norma ISO 9001: 2008 uno de los requisitos generales del Sistema de Gestión de Calidad es:

"Determinar los procesos necesarios para el sistema de gestión de la calidad y su aplicación a través de la organización"

Es importante entonces establecer "qué" actividades y "cómo" se realizan las tareas dentro de cada una de ellas para visualizar de mejor manera la realidad

actual del entorno sobre el cual se está trabajando. Únicamente así se pueden aplicar los correctivos necesarios que permitan una mejora continua para la entrega de un mejor producto o servicio.

Actualmente la Cooperativa tiene disponible en la Intranet un Manual de Procesos que define las actividades de cada Unidad del Área de Sistemas, como se puede observar en el Anexo I, sin embargo no son los procesos y procedimientos que rigen actualmente el desarrollo de aplicativos.

Por tal motivo se ha considerado conveniente en primera instancia actualizar el esquema, para tener un punto de partida mucho más claro y posteriormente aplicar el análisis FODA a la Unidad de Desarrollo con el propósito de determinar los problemas fundamentales que tiene el mismo y a los que se deben enfocar las mejoras.

La documentación generada se presenta en el Anexo II.

4.2.1 Proceso de Automatización de Requerimientos

Como se puede identificar el proceso compone de 5 subprocesos que son:

1. Análisis y diseño de la Orden de Pedido
2. Desarrollo de la Orden de Pedido
3. Implementación en ambiente de pruebas
4. Pruebas
5. Implementación en ambiente de producción.

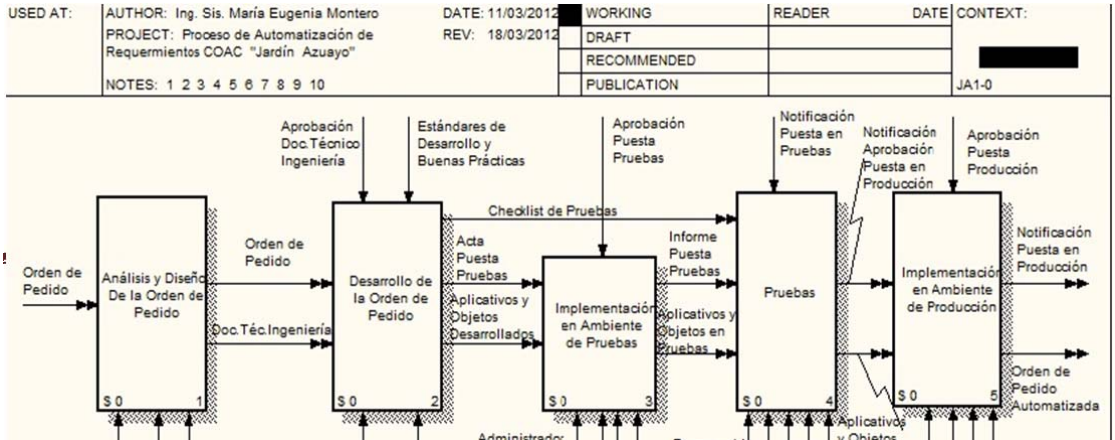


Figura 19. Proceso de Automatización de Requerimientos

Fuente: Elaboración Propia

Análisis y Diseño de la Orden de Pedido

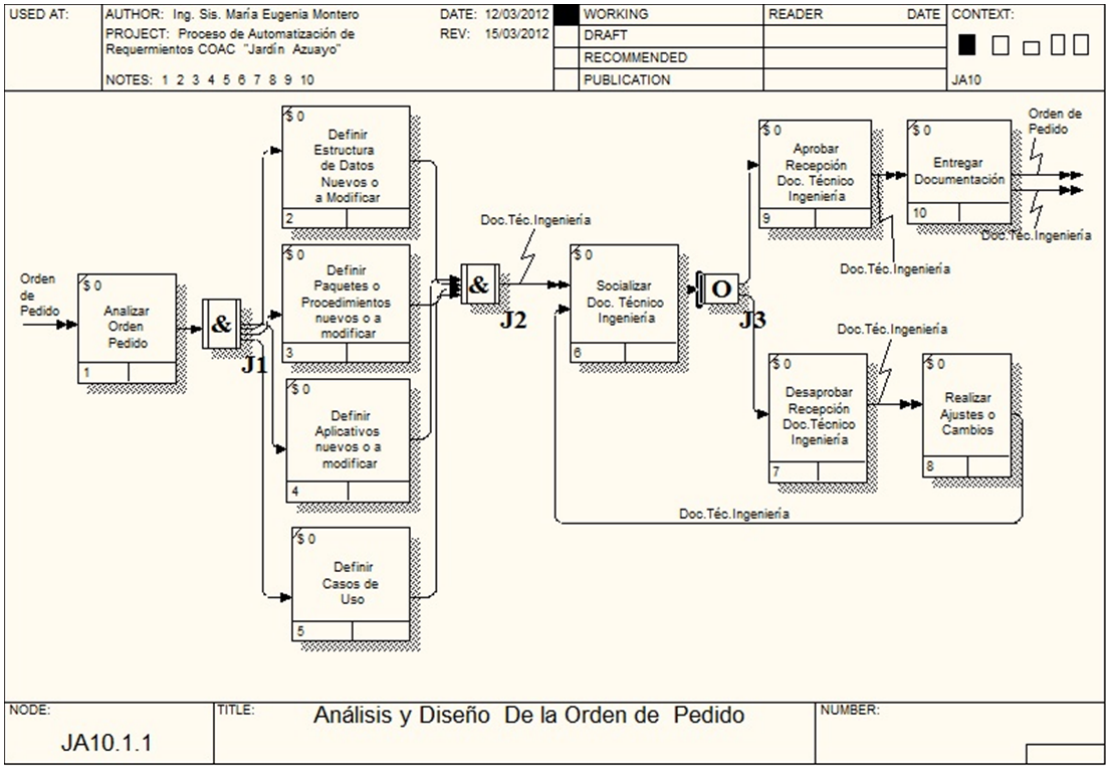


Figura 20. Subproceso de Análisis y Diseño de la Orden de Pedido

Fuente: Elaboración Propia

Nombre	**JA10.1.1 Análisis y Diseño de la Orden de Pedido**
Procedimiento	**1.** Ingresa la orden de pedido y es recibida por el Ingeniero de Software asignado como recurso en la Red de Proyectos. **2.** El Ingeniero en Software analiza la orden de pedido **3.** El Ingeniero de Software realiza el diseño, generando el Documento Técnico de Ingeniería que contiene la siguiente información: • Definición de estructuras de datos nuevos o a modificar • Definición de Paquetes y Procedimientos nuevos o a modificar

	• Definición de Aplicativos que se deben crear o modificar • Definición de Casos de Uso • Definición de Permisos. **4.** El Ingeniero de Software realiza una reunión con el Desarrollador asignado en la Red del Proyecto para socializar el Documento Técnico de Ingeniería. **5.** Si el Documento Técnico está de conformidad para el Desarrollador y el Ingeniero de Software, se da por recibido el documento. **6.** El Ingeniero de Software le entrega al Desarrollador la Orden de Pedido y el Documento Técnico. **7.** Si el Documento Técnico no está de conformidad con el Desarrollador y el Ingeniero de Software, el Desarrollador solicita los ajustes y cambios necesarios al Ingeniero de Software. **8.** El Ingeniero de Software realiza los ajustes y cambios solicitados. **9.** Se repite desde el punto 4 al 7.

Tabla No. 1. Procedimiento de Análisis y Diseño de la Orden de Pedido

Desarrollo de la Orden de Pedido

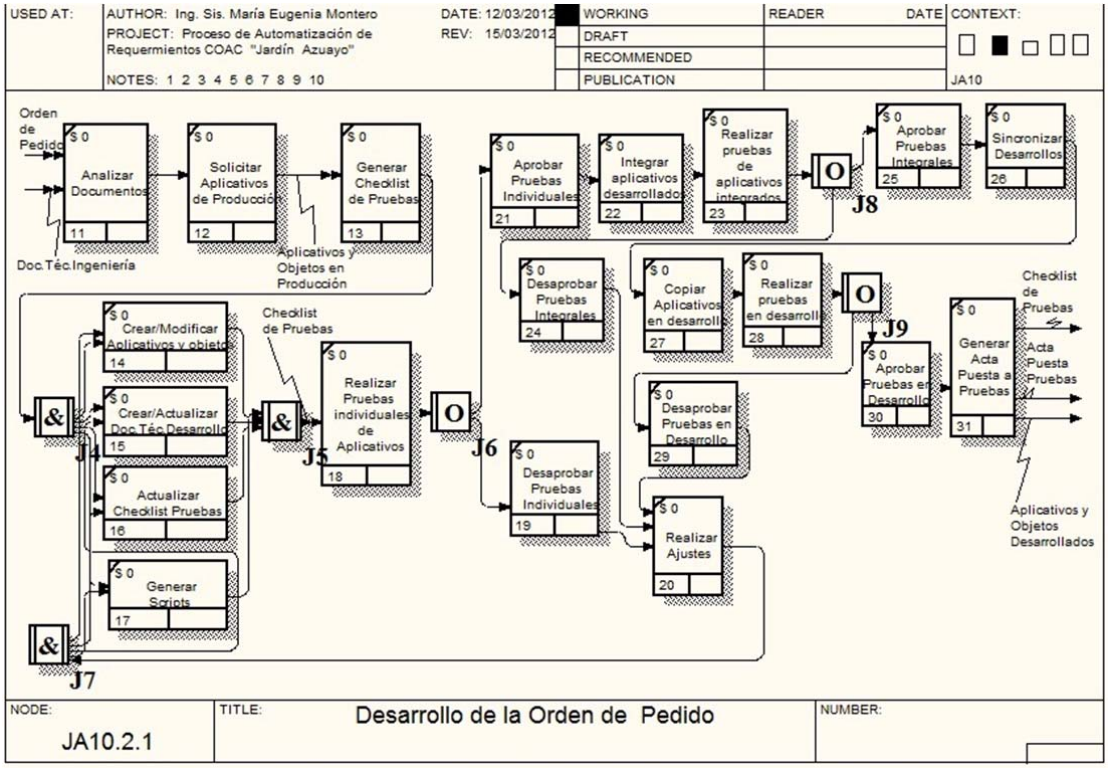


Figura 21. Subproceso de Desarrollo de la Orden de Pedido

Fuente: Elaboración Propia

Nombre	**JA10.2.1 Desarrollo de la Orden de Pedido**
Procedimiento	**1.** El Desarrollador recibe el Documento Técnico de Ingeniería y la Orden de Pedido. **2.** El Desarrollador analiza los documentos recibidos. **3.** El Desarrollador solicita al Administrador de Aplicaciones y Base de datos los aplicativos que se deben modificar y que se encuentran en el ambiente de producción. **4.** El Administrador de Aplicativos y Base de Datos entrega los aplicativos solicitados al Desarrollador.

	5. El Desarrollador genera el *Checklist* de pruebas. **6.** El Desarrollador copia los aplicativos entregados por el Administrador de Aplicativos en su equipo. **7.** El Desarrollador crea o modifica tablas, paquetes, procedimientos y funciones aplicando los Estándares de Desarrollo y Buenas Prácticas **8.** El Desarrollador almacena los scripts correspondientes. **9.** El Desarrollador crea o actualiza el Documento Técnico de Desarrollo con la siguiente información : • Tipo de Objeto • Nombre del Objeto y Parámetros • Estructuras de datos • Funcionalidades • Permisos **10.** El Desarrollador crea o modifica aplicativos **11.** El Desarrollador actualiza el Documento Técnico de Desarrollo. **12.** El Desarrollador actualiza el *Checklist* de Pruebas **13.** El Desarrollador realiza pruebas individuales de los aplicativos **14.** Si el aplica no cumple con la funcionalidad especificada se realizan ajustes. **15.** Si el aplicativo cumple con las funcionalidades especificadas se integran los aplicativos desarrollados **16.** El Desarrollador realiza pruebas de integración de los aplicativos. **17.** El Desarrollador consulta si otro compañero está trabajando con el mismo aplicativo en el ambiente de desarrollo. **18.** Si ningún compañero se encuentra trabajando con los mismos aplicativos en el ambiente de desarrollo, el desarrollador copia los aplicativos de su equipo al ambiente de aplicativos de desarrollo. **19.** Si otro compañero se encuentra trabajando sobre los mismos aplicativos, se sincronizan los desarrollos, es decir quien

	tenga menor número de cambios, actualiza sus desarrollos en los aplicativos que se encuentran en el ambiente de desarrollo. **20.** El Desarrollador realiza pruebas en el ambiente de desarrollo validando y verificando las funcionalidades con el *Checklist*. **21.** Si las pruebas no son satisfactorias el Desarrollador copia los aplicativos de desarrollo a su equipo. **22.** El Desarrollador realiza los ajustes necesarios. **23.** Se repite desde el punto 7 al 22 **24.** Si las pruebas realizadas son satisfactorias, el desarrollador notifica al Coordinador de Desarrollo. **25.** El Coordinador de Desarrollo realiza pruebas en el ambiente de Desarrollo validando y verificando el *Checklist* de Pruebas **26.** Si existen inconvenientes en las pruebas el Coordinador de Desarrollo solicita realizar ajustes y cambios al Desarrollador. **27.** Se repite desde el punto 7 al 26. **28.** Si las pruebas son satisfactorias el Coordinador de Desarrollo aprueba pasar el desarrollo al ambiente de pruebas **29.** El Desarrollador genera el Acta de pruebas. **30.** El Desarrollador envía el Acta de Puesta a Pruebas y los aplicativos y objetos desarrollados al Administrador de Aplicaciones y Base de Datos **31.** El Desarrollador envía el *Checklist* de Pruebas al Ingeniero de Software **32.** El Ingeniero de Software envía el *Checklist* de Pruebas a los Usuarios y al Responsable de la orden para las pruebas.

Tabla No. 2. Procedimiento de Desarrollo de la Orden de Pedido

Implementación en Ambiente de Pruebas

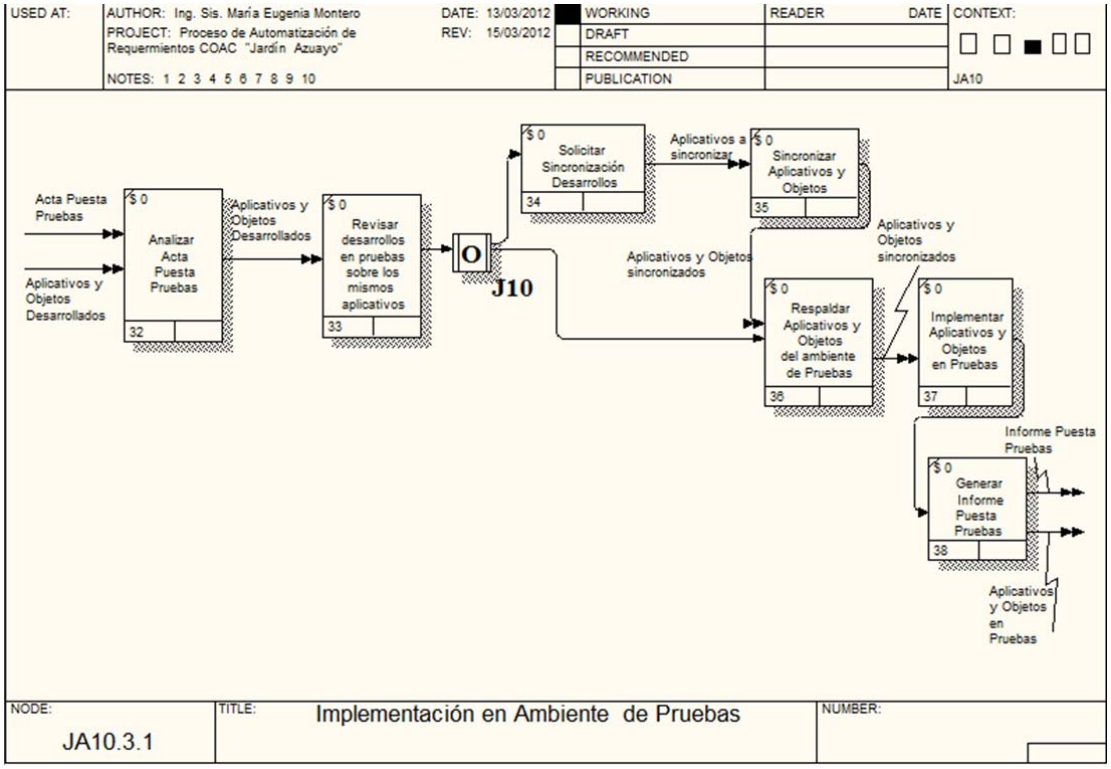


Figura 22. Subproceso de Implementación en Ambiente de Pruebas

Fuente: Elaboración Propia

Nombre	**JA10.3.1 Implementación en Ambiente de Pruebas**
Procedimiento	**1.** El Administrador de Aplicaciones y Base de Datos recibe el Acta de Puesta a Pruebas y los aplicativos y objetos desarrollados. **2.** El Administrador de Aplicaciones y Base de Datos analiza el Acta de Puesta a Pruebas. **3.** Si los mismos aplicativos se encuentran en pruebas el Administrador de Aplicaciones y Base de Datos solicita al

	Desarrollador sincronizar los aplicativos. **4.** El Administrador le envía al Desarrollador los aplicativos que están en pruebas para que sean sincronizados. **5.** El Desarrollador implementa los cambios realizados en los aplicativos que están en pruebas. **6.** El Desarrollador envía al Administrador de Aplicaciones y Base de Datos los aplicativos sincronizados. **7.** El Administrador de Aplicaciones y Base de Datos respalda aplicativos y objetos que se encuentran en pruebas. **8.** El Administrador de Aplicativos y Base de Datos actualiza paquetes, procedimientos y aplicativos en el ambiente de pruebas. **9.** El Administrador de Aplicativos y Base de Datos genera el documento de informe de puesta a pruebas. **10.** El Administrador de Aplicativos y Base de Datos envía un correo notificando al Ingeniero de Software y al Desarrollador, que el desarrollo está en el ambiente de pruebas junto con el informe de puesta de pruebas. **11.** El Ingeniero de Software envía un correo a los usuarios asignados para las pruebas en la red de proyectos y al responsable de la orden, para que inicien las pruebas del producto. **12.** El Ingeniero de software envía el *Checklist* de Pruebas a los usuarios y al responsable de la orden

Tabla No. 3. Procedimiento de Implementación en Ambiente de Pruebas

Pruebas

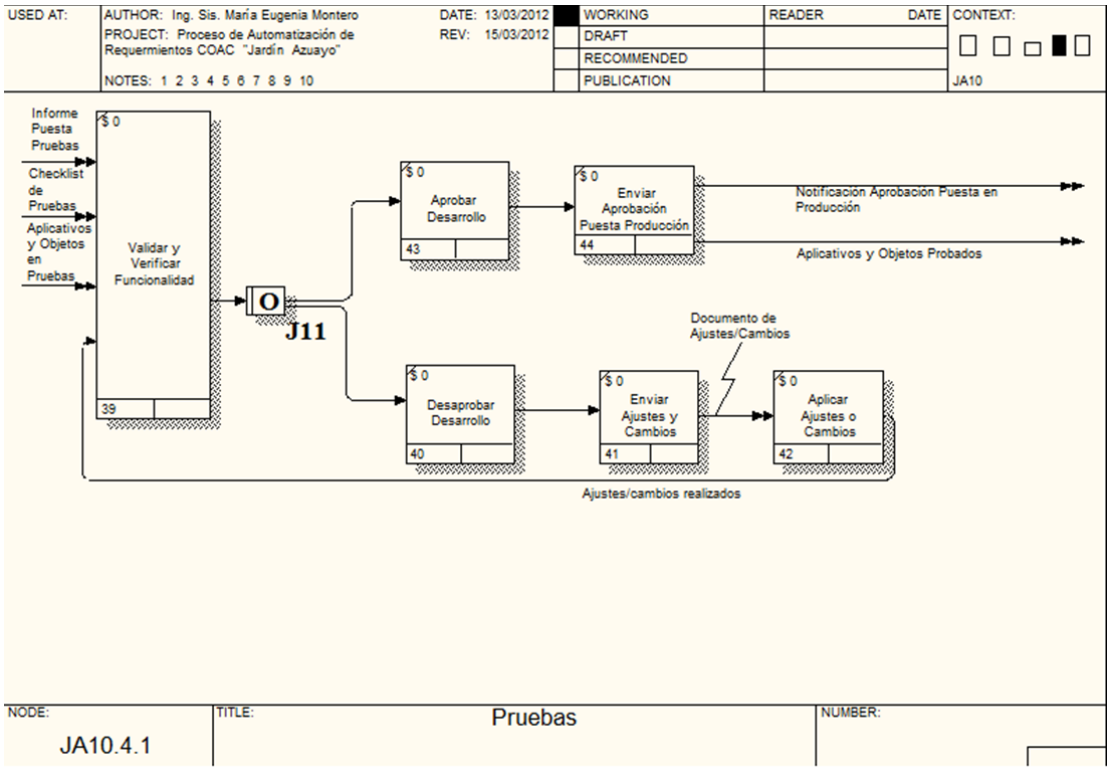


Figura 23. Subproceso de Pruebas

Fuente: Elaboración Propia

Nombre	**JA10.4.1 Pruebas**
Procedimiento	**1.** Los usuarios y el responsable de la orden reciben el *Checklist* de Pruebas y el Informe de Puesta a Pruebas y los aplicativos puestos en pruebas. **2.** Los usuarios y el responsable de la orden validan y verifican la funcionalidad del desarrollo. **3.** Si existen inconvenientes tanto el propietario de la orden y los usuarios envían el documento de ajustes o cambios al Ingeniero de Software **4.** El Ingeniero de Software envía un documento con los ajustes

	o cambios que se deben realizar al Desarrollador. **5.** El Desarrollador solicita al Administrador de Aplicaciones y Base de Datos los aplicativos correspondientes. **6.** El Administrador de Aplicaciones envía al Desarrollador los aplicativos solicitados. **7.** El Desarrollador realiza los cambios y ajustes solicitados en su equipo. **8.** El Desarrollador copia los aplicativos en el ambiente de desarrollo para verificar su correcto funcionamiento. **9.** El Desarrollador envía al Administrador de Aplicaciones y Base de Datos los aplicativos con los ajustes realizados. **10.** El Administrador de Aplicaciones y Base de Datos respalda los aplicativos y objetos de la orden. **11.** El Administrador de Aplicaciones y Base de Datos actualiza los aplicativos y objetos con los ajustes o cambios. **12.** El Administrador de Aplicaciones y Base de Datos envía un correo al Desarrollador notificando que se actualizaron los aplicativos y objetos enviados. **13.** El Desarrollador envía un correo al Ingeniero de Software notificando que los ajustes y cambios están realizados. **14.** El Ingeniero de Software envía un correo a los usuarios y al propietario de la orden notificando que se realizaron los ajustes y que pueden continuar con las pruebas. **15.** Se repite desde el punto 2 al 14. **16.** Si las pruebas no presentan inconvenientes, los usuarios y propietario de la orden envían una notificación de aprobación al Ingeniero de Software para la puesta en producción y los aplicativos probados.

Tabla No. 4. Procedimiento de Pruebas

Implementación en Ambiente en Producción

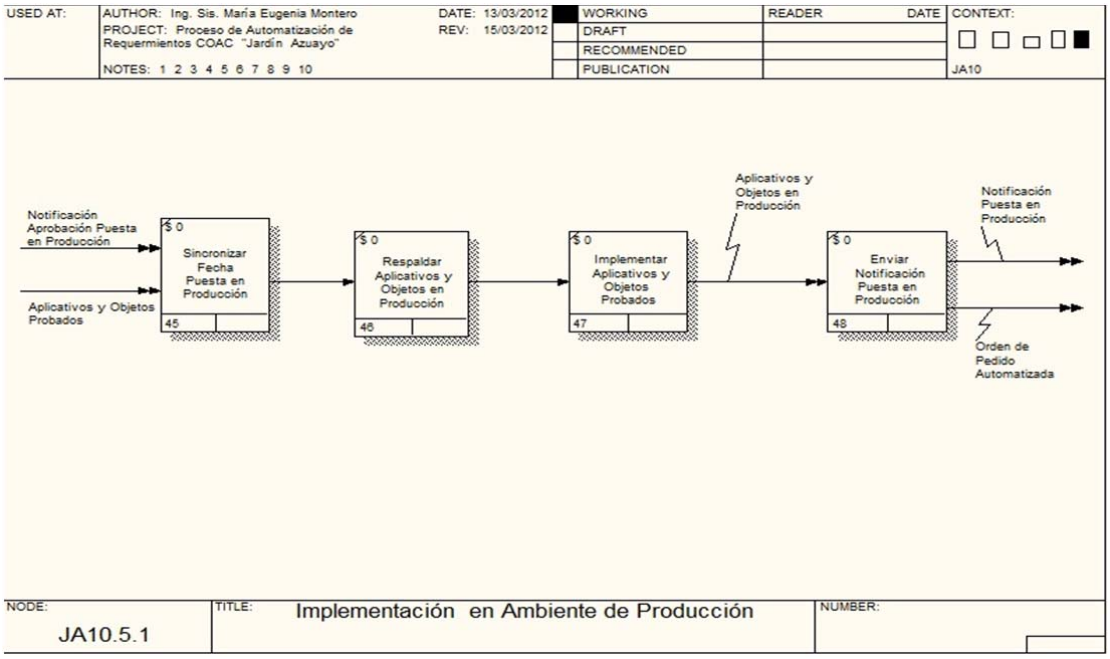


Figura 24. Subproceso Implementación en Ambiente de Producción

Fuente: Elaboración Propia

Nombre	**JA10.5.1 Implementación en Ambiente de Producción**
Procedimiento	**1.** El Ingeniero de Software envía una notificación al Administrador de Aplicaciones y Base de Datos de aprobación para la puesta en producción. **2.** El Administrador de Aplicaciones y Base de Datos recibe el correo de aprobación para la puesta en producción. **3.** El Administrador de Aplicaciones sincroniza la fecha de puesta en producción con el responsable de la Unidad de Operaciones. **4.** El Administrador de Aplicaciones y Base de Datos respalda los aplicativos y objetos que intervienen y que se encuentran en producción. **5.** El Administrador de Aplicaciones y Base de Datos

	implementa el los aplicativos y objetos probados en producción. **6.** El Administrador de Aplicaciones y Base de Datos envía una notificación de que el desarrollo de la orden se encuentra en producción al Ingeniero de Software **7.** El Ingeniero de Software envía una notificación al responsable de proyectos indicando que el desarrollo se puso en producción. **8.** El Responsable de Proyectos cierra la orden.

Tabla No. 5. Procedimiento de Implementación en Ambiente de Producción

4.3 ANALISIS DEL MODELO ACTUAL

Para poder determinar las fases críticas del modelo de proceso de desarrollo actual, se tomará como referencia el año 2011, período en el cual según la bitácora registrada en la Unidad de Proyectos (Figura 25), se han emitido 44 solicitudes de automatización, de las cuales 34 son Órdenes de Pedido y 10 son Proyectos.

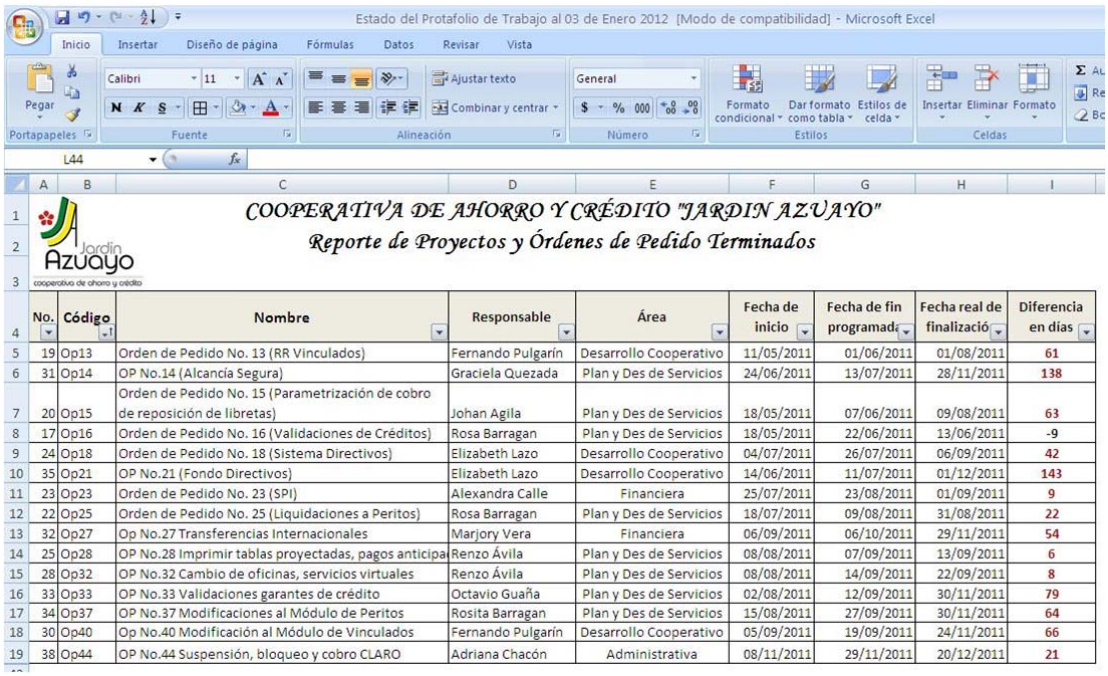
	A	B	C	D	E	F	G
1	OP N	Tipo	Responsabl	Tema	Ingeniería del Sotfware	Desarrollo Informático	Estado
2	1	Proyecto	Graciela Quezada	Seguro de desgravamen	Mauricio Chica	Santiago Araujo	Terminado
3	2	Proyecto	Octavio Guañña	Correción de información	Marco Paredes	Jorge Bonette	Terminado
4	3	Proyecto	Octavio Guañña	Correción automática de información	Marco Paredes	Víctor Astudillo	Terminado
5	4	Orden de Pedido	John Machuca	Consulta de datos			Terminado
6	5	Orden de Pedido	Elizabeth Eras	Cuentas inactivas en la Web	Marco Paredes		Terminado
7	6	Orden de Pedido	Nelly Molina	Remesas			Terminado
8	7	Orden de Pedido	Elizabeth Eras	Cuentas inactivas en cajas	Mauricio Chica	Verónica Vélez	Terminado
9	8	Proyecto	Maira Gonzales	Cambio de firmas autorizadas	Mauricio Chica		Terminado
10	9	Orden de Pedido	Rosita Barragan	Contrato de Crédito		María Eugenia Montero	Terminado
11	10	Proyecto	Octavio Guañña	OP adicional de OP No.2 (Correción de información)	Marco Paredes	Jorge Bonette	Terminado
12	11	Orden de Pedido	Rosita Barragan	Impresión tabla de amortización		María Eugenia Montero	Terminado
13	12	Proyecto	Jorge Luis Alvarado	Débito automático Cidercom	Cristian López	Cristian López	Terminado
14	13	Orden de Pedido	Fernando Pulgarín	Vinculados (RRHH)	Santiago Araujo	Grace Morales	Terminado
15	14	Orden de Pedido	Graciela Quezada	Alcancía Segura (cambio en fórmula de penalización)	Marco Paredes	Grace Morales	Terminado
16	15	Orden de Pedido	Johan Agila	Reposición de libreta	Mauricio Chica	Verónica Vélez	Terminado
17	16	Orden de Pedido	Rosita Barragan	Validación de Crédito	Mauricio Chica	María Eugenia Montero	Terminado
18	17	Proyecto	Johan Agila	Reestructuración de Crédito	Marco Paredes	María Eugenia Montero	Terminado
19	18	Orden de Pedido	Eli Lazo	Sistema de Directivos	Santiago Araujo	Grace Morales	Terminado
20	19	Orden de Pedido	Alexandra Calle	Ingreso de Cheques para Palmas y Mendez	Santiago Araujo	Cristian López	Terminado
21	20	Orden de Pedido	Graciela Quezada	Un cambio en reporte de seguro de desgravamen	Santiago Araujo	Santiago Araujo	Terminado
22	21	Orden de Pedido	Eli Lazo	Fondo de Directivos	Santiago Araujo	Verónica Vélez	Terminado
23	22	Proyecto	Antonia Carchipulla	Proyecto de observaciones SBS-SPLA Cumplimiento	Verónica Vélez	Sandra Abril	Terminado
24	23	Orden de Pedido	Alexandra Calle	SPI	Santiago Araujo	Jorge Bonette	Terminado
25	24	Orden de Pedido	Alexandra Calle	Correciones archivo SCI		Marco Paredes	Terminado
26	25	Orden de Pedido	Rosita Barragan	Pago a Peritos	Santiago Araujo	Grace Morales	Terminado
27	26	Orden de Pedido	Alexandra Calle	Carga de archivo de IESS	Mauricio Chica	Jorge Bonette	Terminado
28	27	Orden de Pedido	Marjory Vera	Transferencias Internacionales	Verónica Vélez	Sandra Abril	Terminado
29	28	Orden de Pedido	Renzo Avila	Imprimir tablas proyectadas pagos anticipados		Paul Zhañay	Terminado
30	29	Orden de Pedido	Octavio Guañña	Reverso pagos de crédito	Romel Cando	Romel Cando	Pendiente
31	30	Orden de Pedido	Renzo Avila	Varios créditos a un certificado de depósito		Sandra Abril	En pruebas
32	31	Orden de Pedido	Octavio Guañña	Notificiones		Paul Zhañay	Pendiente
33	32	Orden de Pedido	Renzo Avila	Cambio de oficina, servicios virtuales		Grace Morales	Terminado
34	33	Orden de Pedido	Octavio Guañña	Validaciones garantes de crédito	Santiago Araujo	María Eugenia Montero	Terminado
35	34	Orden de Pedido	Antonia Carchipulla	Licitud de fondos	Verónica Vélez	Sandra Abril	Terminado
36	35	Orden de Pedido	Marcelo Yunga	Automatización de la estructura R07	Verónica Vélez	Grace Morales	Terminado
37	36	Proyecto	Jorge Luis Alvarado	Fondos Administrados	Marco Paredes	Paúl Zhañay	En pruebas
38	37	Orden de Pedido	Rosita Barragan	Modificaciones del Módulo de Peritos	Santiago Araujo	Grace Morales	Terminado
39	38	Orden de Pedido	Maira Gonzales	Acreditaciones cuentas de socios	Romel Cando	Romel Cando	En pruebas
40	39	Orden de Pedido	Renzo Avila	Modificaciones al Módulo SPI	Grace Morales	Grace Morales	En ejecución
41	40	Orden de Pedido	Fernando Pulgarín	Modificaciones del Módulo de Vinculados	María Eugenia Mont	María Eugenia Montero	Terminado
42	41	Proyecto	Elizabeth Eras	Aplicación a la política de capitalización (Créditos)	Verónica Vélez	María Eugenia Montero	Terminado
43	42	Orden de Pedido	Renzo Avila	Impresión de origen o antecedentes del SPI o SCI	Grace Morales	Grace Morales	Pendiente
44	43	Orden de Pedido	Nelly Molina	Providencias Judiciales	Romel Cando	María Eugenia Montero	Terminado
45	44	Orden de Pedido	Adriana Chacón	Suspensión, bloqueo y cobro CLARO	Marco Paredes	Santiago Araujo	Terminado

Figura 25. Bitácora de Órdenes de Pedido

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

La diferencia entre las órdenes de pedido y los proyectos radican en su extensión, puesto que ambos siguen el mismo esquema de desarrollo, razón por la cual se analizará únicamente el comportamiento de las órdenes de pedido.

En el Estado de Portafolio de Trabajo con corte al 3 de enero del 2012 (Figura 26), se establece que en el año 2011 se han finalizado únicamente 15 órdenes de pedido y se analizarán 13 debido a que la información de la Orden No. 25 y 44 no se encontraron registradas.



No.	Código	Nombre	Responsable	Área	Fecha de inicio	Fecha de fin programada	Fecha real de finalizació	Diferencia en días
19	Op13	Orden de Pedido No. 13 (RR Vinculados)	Fernando Pulgarín	Desarrollo Cooperativo	11/05/2011	01/06/2011	01/08/2011	61
31	Op14	OP No.14 (Alcancía Segura)	Graciela Quezada	Plan y Des de Servicios	24/06/2011	13/07/2011	28/11/2011	138
20	Op15	Orden de Pedido No. 15 (Parametrización de cobro de reposición de libretas)	Johan Agila	Plan y Des de Servicios	18/05/2011	07/06/2011	09/08/2011	63
17	Op16	Orden de Pedido No. 16 (Validaciones de Créditos)	Rosa Barragan	Plan y Des de Servicios	18/05/2011	22/06/2011	13/06/2011	-9
24	Op18	Orden de Pedido No. 18 (Sistema Directivos)	Elizabeth Lazo	Desarrollo Cooperativo	04/07/2011	26/07/2011	06/09/2011	42
35	Op21	OP No.21 (Fondo Directivos)	Elizabeth Lazo	Desarrollo Cooperativo	14/06/2011	11/07/2011	01/12/2011	143
23	Op23	Orden de Pedido No. 23 (SPI)	Alexandra Calle	Financiera	25/07/2011	23/08/2011	01/09/2011	9
22	Op25	Orden de Pedido No. 25 (Liquidaciones a Peritos)	Rosa Barragan	Plan y Des de Servicios	18/07/2011	09/08/2011	31/08/2011	22
32	Op27	Op No.27 Transferencias Internacionales	Marjory Vera	Financiera	06/09/2011	06/10/2011	29/11/2011	54
25	Op28	OP No.28 Imprimir tablas proyectadas, pagos anticipa	Renzo Ávila	Plan y Des de Servicios	08/08/2011	07/09/2011	13/09/2011	6
28	Op32	OP No.32 Cambio de oficinas, servicios virtuales	Renzo Ávila	Plan y Des de Servicios	08/08/2011	14/09/2011	22/09/2011	8
33	Op33	OP No.33 Validaciones garantes de crédito	Octavio Guaña	Plan y Des de Servicios	02/08/2011	12/09/2011	30/11/2011	79
34	Op37	OP No.37 Modificaciones al Módulo de Peritos	Rosita Barragan	Plan y Des de Servicios	15/08/2011	27/09/2011	30/11/2011	64
30	Op40	Op No.40 Modificación al Módulo de Vinculados	Fernando Pulgarín	Desarrollo Cooperativo	05/09/2011	19/09/2011	24/11/2011	66
38	Op44	OP No.44 Suspensión, bloqueo y cobro CLARO	Adriana Chacón	Administrativa	08/11/2011	29/11/2011	20/12/2011	21

Figura 26. Estado del Portafolio de Trabajo con fecha de corte 03/01/2012

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

En base a la información existente en las redes de proyectos (Anexo III), se ha consolidado en las tablas que se presentan a continuación, la siguiente información:

- Número y nombre de la orden.

- Fecha de inicio estimado y real de cada fase de automatización
- Diferencia en días entre fechas estimadas y reales
- Diferencia en días totales de la orden entre el estimado y el real.
- Valoración de la programación de la orden.

No.	Descripción de la Orden	Ingeniería		Desarrollo		Puesta Pruebas		Pruebas		Puesta en Producción		Fecha Finalizacion	
		Proyectado	Real	Proyectado	Real	Proyectado	Real	Proyectado	Real	Proyectado	Real	Proyectado	Real
1	OP13 RR Vinculados	11/05/2011	11/05/2011	12/05/2011	12/05/2011	18/05/2011	20/05/2011	19/05/2011	23/05/2011	24/05/2011	26/07/2011	01/06/2011	29/07/2011
2	OP14 Alcancia Segura	30/05/2011	24/06/2011	30/05/2011	27/06/2011	01/06/2011	05/07/2011	01/06/2011	12/07/2011	06/06/2011	28/11/2011	13/07/2011	28/11/2011
3	OP15 Parametrización de cobro de reposición de libretas	18/05/2011	18/05/2011	18/05/2011	18/05/2011	24/05/2011	24/05/2011	25/05/2011	25/08/2011	30/05/2011	28/07/2011	07/06/2011	09/08/2011
4	OP16 Validaciones de Créditos	18/05/2011	18/05/2011	18/05/2011	19/05/2011	16/05/2011	02/06/2011	17/05/2011	06/06/2011	19/05/2011	10/06/2011	22/06/2011	13/06/2011
5	OP18 Sistema Directivos	06/06/2011	04/07/2011	07/06/2011	08/07/2011	14/06/2011	14/07/2011	15/06/2011	15/07/2011	17/06/2011	25/07/2011	26/07/2011	06/09/2011
6	OP21 Fondo Directivos	14/06/2011	14/06/2011	21/06/2011	21/06/2011	28/06/2011	01/07/2011	29/06/2011	11/07/2011	04/07/2011	28/11/2011	11/07/2011	01/12/2011
7	OP23 SPI	25/07/2011	25/07/2011	29/07/2011	01/08/2011	05/08/2011	09/08/2011	08/08/2011	10/08/2011	10/08/2011	29/08/2011	23/08/2011	01/09/2011
8	OP27 Transferencias Internacionales	06/09/2011	06/09/2011	13/09/2011	14/09/2011	14/09/2011	15/09/2011	16/09/2011	07/10/2011	23/09/2011	28/11/2011	06/10/2011	29/11/2011
9	OP28 Imprimir tablas proyectadas, pagos anticipados	05/08/2011	08/08/2011	10/08/2011	09/08/2011	17/08/2011	17/08/2011	23/08/2011	19/08/2011	23/08/2011	05/09/2011	29/08/2011	13/09/2011
10	OP32 Cambio de oficinas, servicios virtuales	08/08/2011	08/08/2011	11/08/2011	15/08/2011	23/08/2011	25/08/2011	24/08/2011	29/08/2011	29/08/2011	05/09/2011	14/09/2011	22/09/2011
11	OP33 Validaciones garantes de crédito	02/08/2011	02/08/2011	09/08/2011	09/08/2011	19/08/2011	19/08/2011	22/08/2011	23/08/2011	29/08/2011	28/11/2011	12/09/2011	30/11/2011
12	OP37 Modificaciones al Módulo de Peritos	15/08/2011	15/08/2011	22/08/2011	10/10/2011	26/08/2011	13/10/2011	05/09/2011	24/10/2011	08/09/2011	29/11/2011	27/09/2011	30/11/2011
13	OP40 Modificación al Módulo de Vinculados	29/08/2011	05/09/2011	31/08/2011	06/09/2011	28/09/2011	07/09/2011	05/10/2011	08/09/2011	08/11/2011	21/11/2011	08/11/2011	24/11/2011

Figura 27. Consolidado de información de las Órdenes de Pedido

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

No.	Descripción de la Orden	Ingeniería		Desarrollo		Puesta Pruebas		Pruebas		Puesta Producción		Total Días Orden		Diferencia en Días	Valoración Programación
		Proy.	Real	Proy.	Real	Proy.	Real	Proy.	Real	Proy.	Real	Estimado	Real		
1	OP13 RR Vinculados	2	2	5	7	2	2	4	47	7	4	20	62	43	Subestimada
2	OP14 Alcancia Segura	1	2	3	7	1	6	4	100	28	1	37	116	80	Subestimada
3	OP15 Parametrización de cobro de reposición de libretas	1	1	5	5	2	68	4	-21	7	9	19	62	44	Subestimada
4	OP16 Validaciones de Créditos	1	2	-3	11	2	3	3	5	25	2	28	23	-4	Justo a Tiempo
5	OP18 Sistema Directivos	2	5	6	5	2	2	3	7	28	32	41	51	11	Subestimada
6	OP21 Fondo Directivos	6	6	6	9	2	7	4	101	6	4	24	127	104	Subestimada
7	OP23 SPI	5	6	6	7	2	2	3	14	10	4	26	33	8	Subestimada
8	OP27 Transferencias Internacionales	6	7	2	2	3	17	6	37	10	2	27	65	39	Subestimada
9	OP28 Imprimir tablas proyectadas, pagos anticipados	4	2	6	7	5	3	1	12	5	7	21	31	11	Subestimada
10	OP32 Cambio de oficinas, servicios virtuales	4	6	9	9	2	3	4	6	13	14	32	38	7	Subestimada
11	OP33 Validaciones garantes de crédito	6	6	9	9	2	3	6	70	11	3	34	91	58	Subestimada
12	OP37 Modificaciones al Módulo de Peritos	6	41	5	4	7	8	4	27	14	2	36	82	47	Subestimada
13	OP40 Modificación al Módulo de Vinculados	3	2	21	2	6	2	25	53	1	4	56	63	8	Subestimada

Figura 28. Programación proyectada y real de cada fase de automatización de las órdenes

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

La valoración de la planificación que se visualiza en la Figura 28, se ha definido bajo el siguiente criterio:

- Subestimadas: Órdenes con tiempos reales de ejecución mayores al estimado.

- Justo a Tiempo: Órdenes con tiempos reales de ejecución iguales o menores al de estimación

Con estos datos se puede evidenciar que en la mayoría de las órdenes el tiempo real de automatización de las órdenes es superior al tiempo estimado y en relación a porcentajes el 92% pertenecen a órdenes con tiempos subestimados y el 8% son órdenes que se ejecutaron justo a tiempo, como se pueden observar en las siguientes figuras:

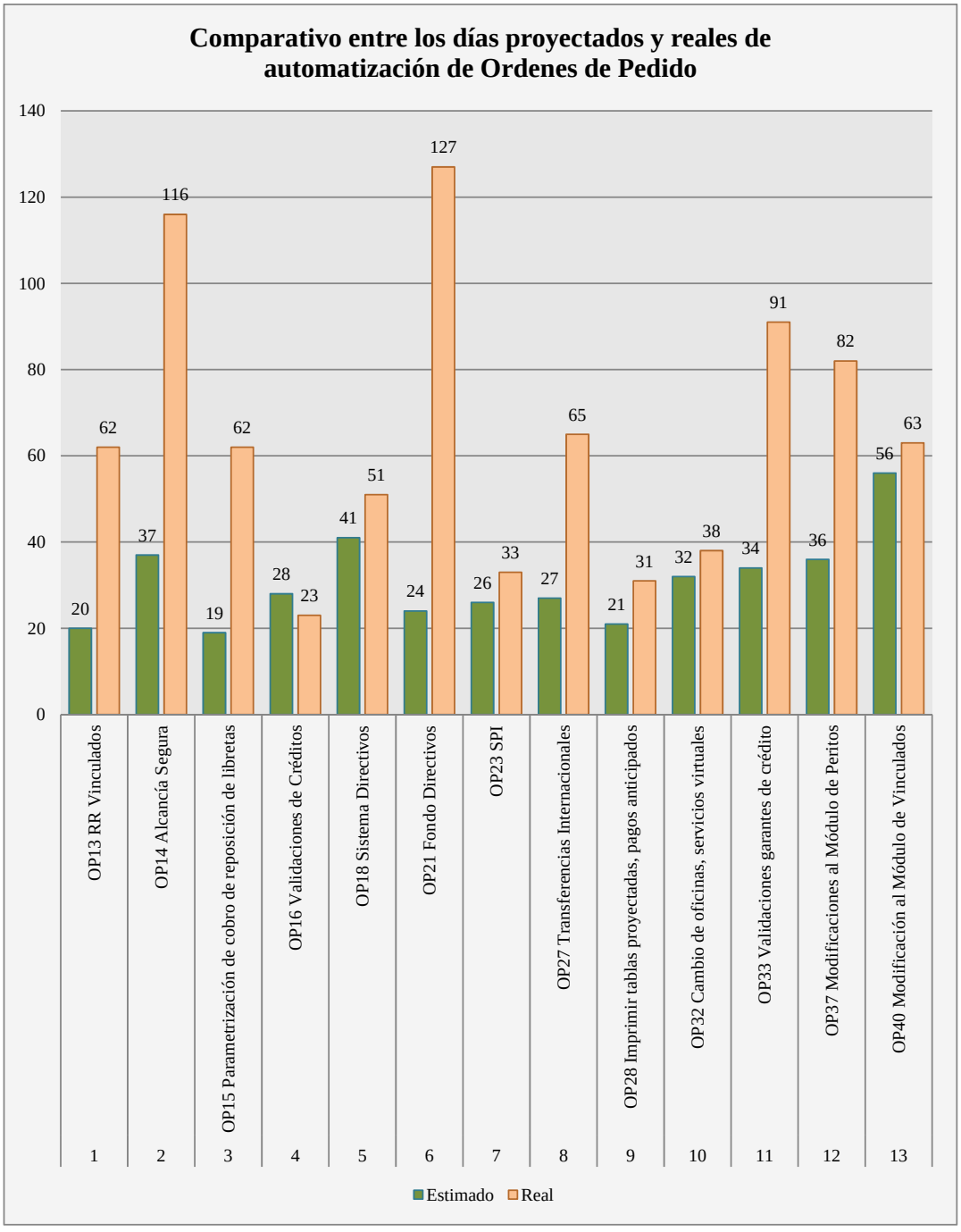


Figura 29. Comparativo días proyectados y reales automatización Órdenes de Pedido

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

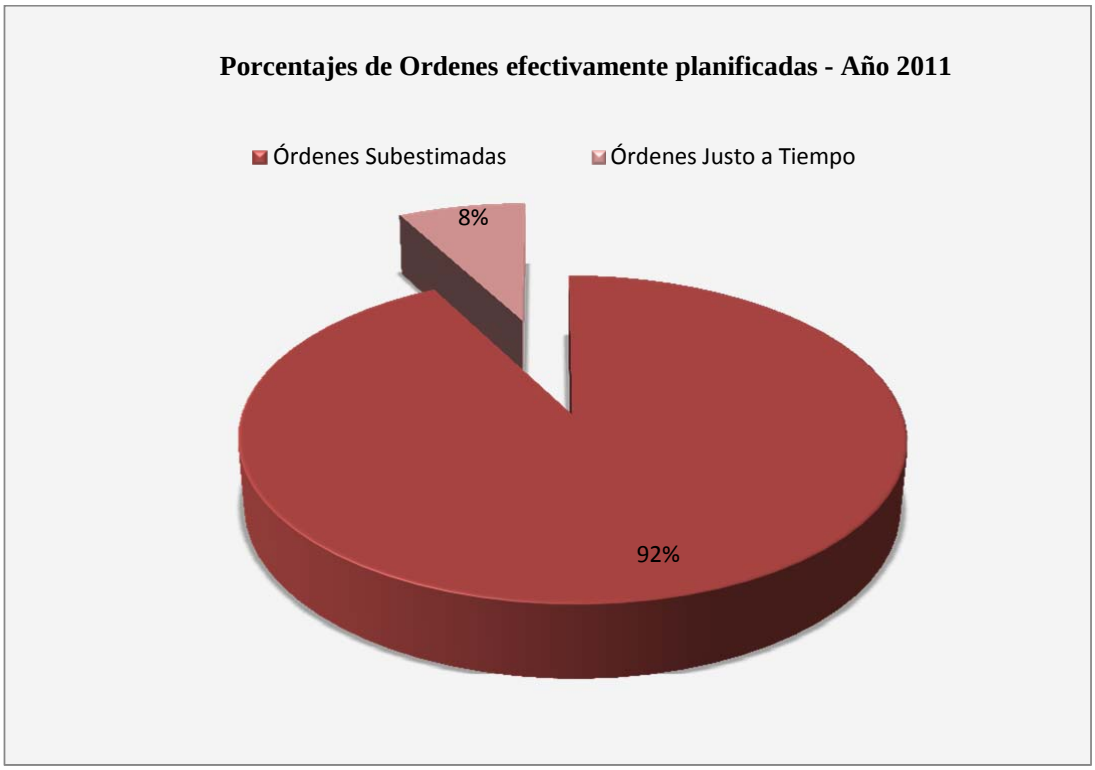


Figura 30. Porcentaje de Órdenes efectivamente planificadas – Año 2011

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

Con los datos obtenidos de los días proyectados y reales en cada fase se pudo establecer dos características importantes:

- El tiempo real de ejecución en cada fase de la orden en la mayoría de los casos es superior al tiempo proyectado para la misma.
- El tiempo proyectado presenta un desfase mayor en la etapa de pruebas.

Los siguientes cuadros comparativos demuestran lo anteriormente especificado.

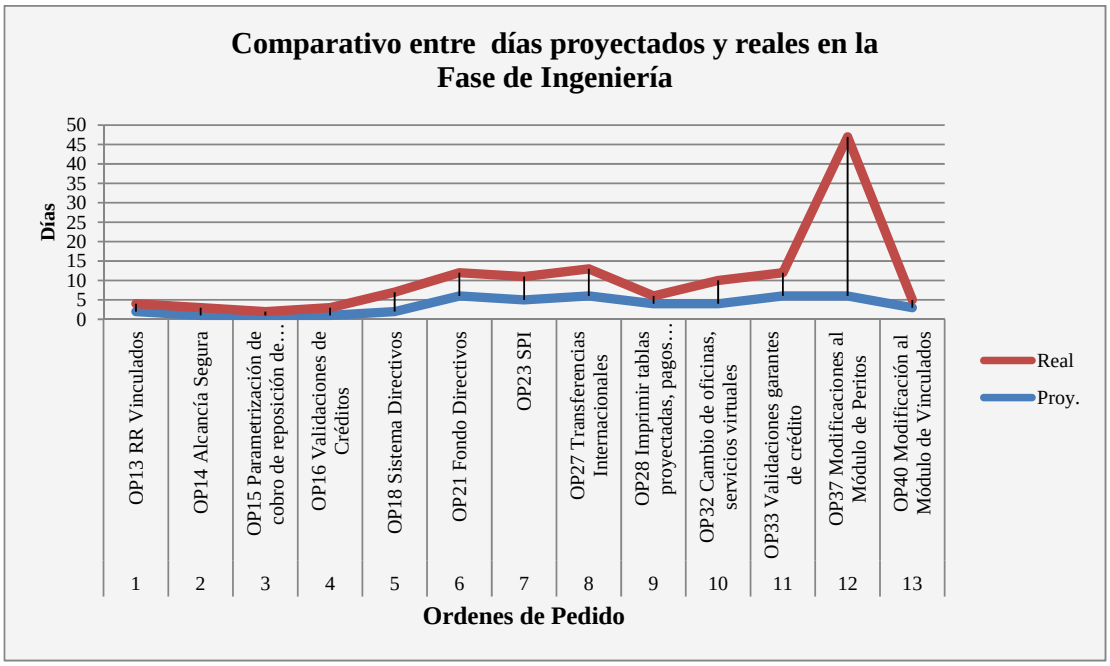


Figura 31. Comparativo días proyectados y reales en la Fase de Ingeniería

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

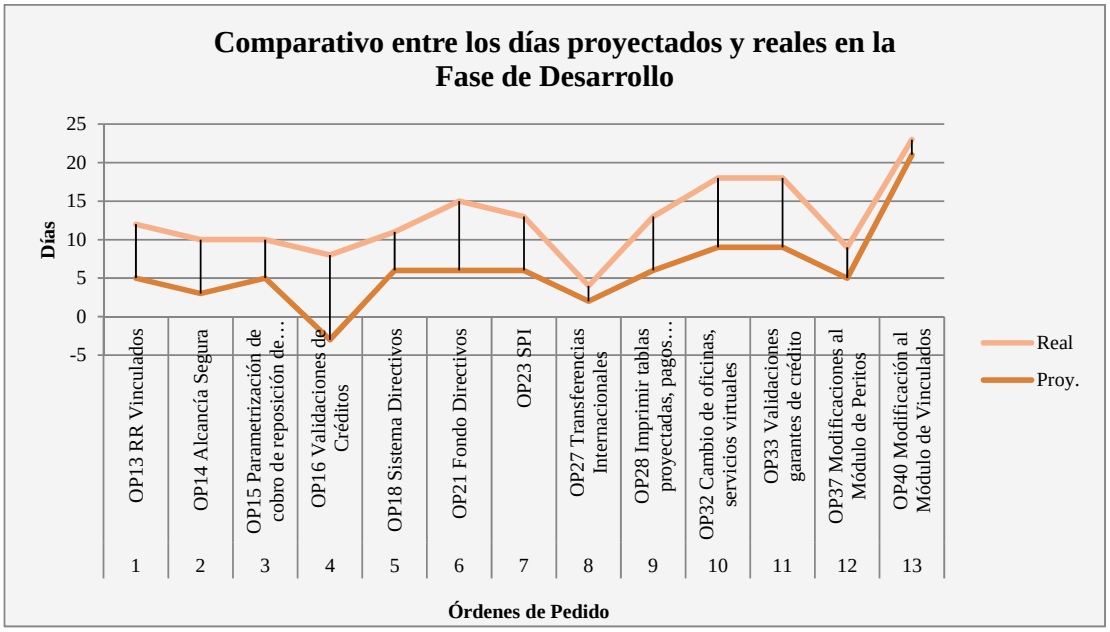


Figura 32. Comparativo días proyectados y reales en la Fase de Desarrollo

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

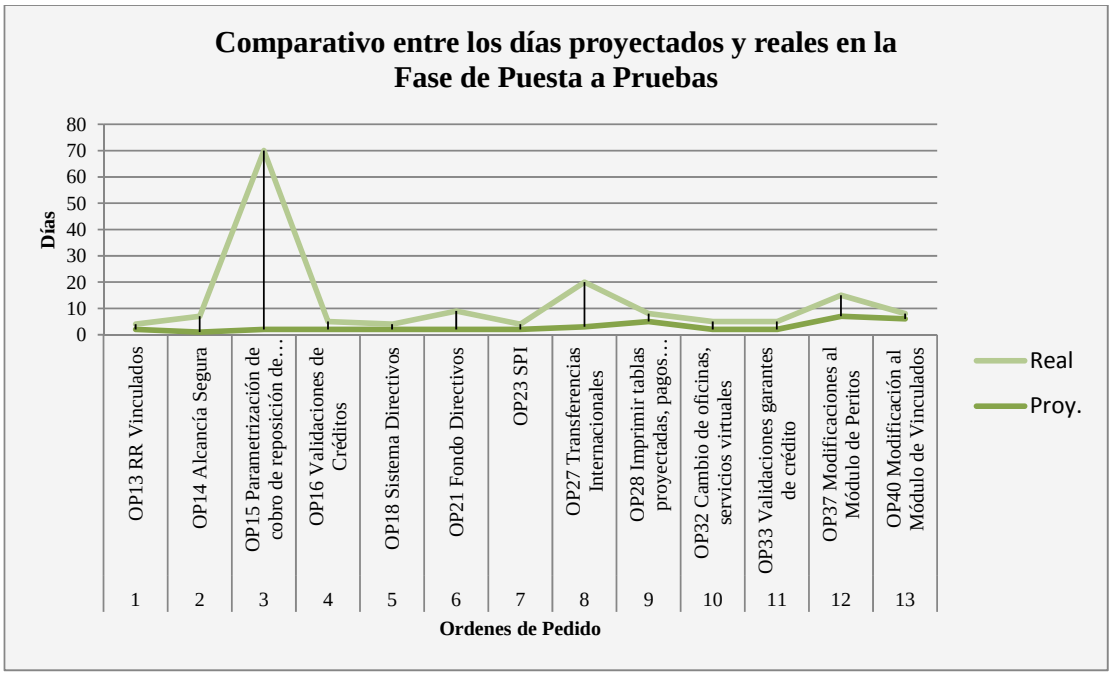


Figura 33. Comparativo días proyectados y reales en la Fase de Puesta a Pruebas

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

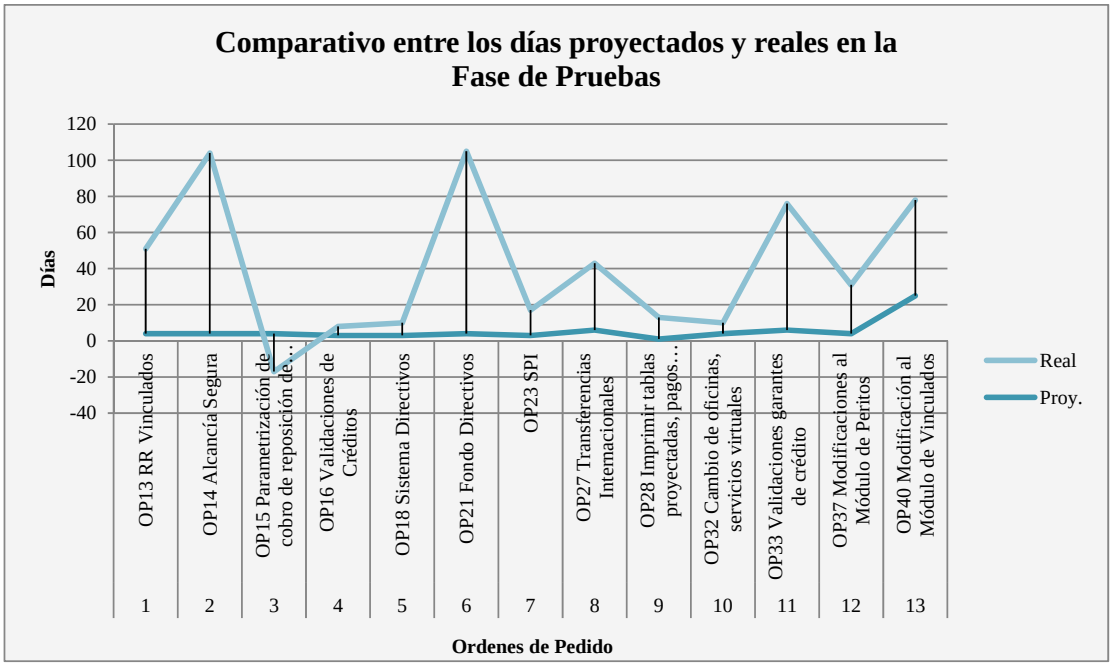


Figura 34. Comparativo días proyectados y reales en la Fase de Pruebas

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

Figura 35. Comparativo días proyectados y reales en la Fase de Puesta a Producción.

Fuente: Unidad de Planificación y Proyectos COAC "Jardín Azuayo"

Estos resultados permiten entonces establecer un Análisis FODA del Modelo de Proceso de Desarrollo utilizado actualmente y que nos dará la oportunidad de definir las estrategias que se pueden aplicar en un nuevo modelo para mejorar la automatización de las órdenes de pedido en la Cooperativa.

4.3.1 Análisis FODA

A continuación se detalla el análisis FODA del Modelo de Proceso de Desarrollo de Software con el que actualmente se realizan las automatizaciones en la Cooperativa de Ahorro y Crédito “Jardín Azuayo”.

FACTORES INTERNOS	**FACTORES EXTERNOS**
Fortalezas	**Oportunidades**
El proceso tiene un responsable definido.	Se cuenta con infraestructura necesaria
Personal con habilidades, formación y motivación	Apoyo de la Alta Gerencia para invertir en el Área de Sistemas tanto en capacitación, herramientas e infraestructura.
Personal con alto sentido de trabajo en equipo	Buen ambiente laboral
Existe documentación de las actividades.	
Están identificadas las partes interesadas dentro del proceso.	
Existe monitorización del proceso	
Se planifican las actividad involucradas en la creación del proyecto	
Debilidades	**Amenazas**
No existe un modelo de desarrollo de software basado en buenas prácticas y acorde a las necesidades de la organización.	Pérdida de Confianza del Usuario final
Entre el subproceso de desarrollo y el subproceso de pruebas no existe un control de calidad del producto por lo que puede producir un cuello de botella ya que siempre estará el producto retornando al subproceso de desarrollo las veces que sean necesarias hasta que las pruebas den un resultado satisfactorio.	Se elevan los costos de los proyectos por el incumplimiento de los tiempos establecidos.
En el Subproceso de Análisis y Diseño de la orden la tarea de "Entregar Documentación" puede provocar retrasos ya que el	Regulaciones de organismos de control

subproceso de desarrollo no iniciará mientras no se ejecute dicha tarea.	
En el Subproceso de Análisis y Diseño de la orden, al existir dos tareas por separado como son la ingeniería y el desarrollo propiamente dicho, "Socializar el Documento Técnico de Ingeniería" produce un cuello de botella ya que no puede continuar con la siguiente tarea mientras no exista la aprobación del mismo.	Usuario sin el conocimiento necesario del proceso de negocio.
En el Subproceso de Desarrollo de la Orden de Pedido la tarea de "Generar *Checklist* de Pruebas" no se debería realizar en ese subproceso.	Ambientes de desarrollo y test no formales ó similares a producción
El Subproceso de Pruebas no dispone de un control para asegurar la calidad de las mismas.	Peticiones de desarrollos en los que intervienen funcionalidades comunes.
En el Subproceso de Pruebas la tarea de "Enviar ajustes y cambios" genera retrasos en el proceso ya que en la mayoría de casos no son ajustes mínimos sino la solicitud de implementar un requerimiento o requisito que no fue contemplado inicialmente.	
Conocimiento disperso del *core* del negocio	
No se tiene definido claramente un responsable del proceso	
En el Subproceso de Análisis y Diseño de la orden y Desarrollo de la Orden se repite la tarea de análisis de la orden de pedido	
Falta de conocimiento de las normativas de organismos de control por parte de los recursos de ingeniería	
Nivel de responsabilidades no definidas claramente dentro de un proyecto	
Personal de base de datos y aplicaciones sin las habilidades necesarias para asesoría en desarrollo e implementación de aplicaciones en producción.	
Bases de datos y aplicaciones que desfasadas a producción	
No existe una herramienta de Control del Proceso de Desarrollo de Software	

Tabla No. 6. Análisis FODA del Modelo de Proceso de Desarrollo (*AS-IS*)

4.3.2 Diagnóstico del Análisis FODA

Como se puede observar en el Análisis FODA es evidente que el número de debilidades del proceso como tal supera las fortalezas, dificultando la elaboración de un producto de software con calidad. El proceso debe estar enfocado hacia el cumplimiento de los objetivos del Área Tecnológica y Organizacionales, sin embargo su débil estructura no lo permite.

"El proceso ayuda a los miembros de una organización a alcanzar los objetivos estratégicos ayudándoles a trabajar más inteligentemente, no más duro, y de un modo más consistente. Los procesos eficaces también proporcionan un medio para introducir y utilizar nuevas tecnologías de forma que permitan responder mejor a los objetivos estratégicos de la organización."[8]

La COAC "Jardín Azuayo" sigue creciendo en su esquema organizativo y junto con ella las necesidades de los usuarios, por lo tanto es imposible continuar desarrollando software como hace 16 años.

El Modelo Propuesto tiene como propósito fundamental dar inicio a una nueva etapa de la Unidad de Desarrollo, mucho más proactiva, apegada a normas y estándares que colaboren en la madurez del mismo.

4.4 MODELO PROPUESTO DEL PROCESO DE DESARROLLO DE SOFTWARE (*TO BE*).

4.4.1 Objetivo

[8] *Tomado de CMMI para el Desarrollo. Pág. 5. Capítulo 1.[12]*

Establecer las políticas, procesos y procedimientos para el desarrollo de software de la Cooperativa de Ahorro y Crédito “Jardín Azuayo”, que aseguren la estandarización del mismo y eleven la calidad de los productos entregados por la Unidad de Desarrollo de Software

4.4.2 Alcance

- Conceptual:
  - Delinear los procesos y procedimientos necesarios para el ciclo de vida de desarrollo de software en proyectos de cualquier tamaño.
- Organizacional:
  - Unidad de Desarrollo de Sistemas

- Geográfico:
  - Cooperativa de Ahorro y Crédito “Jardín Azuayo”.

4.4.3 Políticas para el Proceso de Desarrollo de Software.

1. Del Propietario del Proceso.
   - El proceso debe contar con un único dueño, el mismo que asumirá la responsabilidad de la gestión del mismo.
2. De la Metodología del Proceso.
   - El proceso de desarrollo de Software debe apegarse a una metodología estándar en donde se definan los aspectos más importantes del ciclo de vida del desarrollo de sistemas.
3. De la Gestión del Proceso.
   - El proceso debe ser controlado y evaluado constantemente de tal manera que se puedan aplicar acciones correctivas o preventivas.
4. De la Documentación.
   - Todas las actividades del proceso deben estar debidamente documentadas.

5. De los Tiempos de Entrega.
   - El proceso debe disminuir la cantidad de los proyectos retrasados.
6. Del Producto.
   - El proceso debe elevar la calidad del producto.
   - El proceso debe garantizar el correcto funcionamiento del producto antes de ser liberado a producción.
   - El proceso debe garantizar el control de versiones del producto.
7. De los Recursos.
   - El proceso debe contar con todos los recursos necesarios para el correcto desempeño del mismo.
   - El proceso debe garantizar la correcta utilización de los recursos involucrados.
   - El proceso debe contemplar los roles y responsabilidades de los involucrados en el mismo.

4.4.4 Arquitectura del Modelo Propuesto

Definición de Modelos Estándares a aplicar.

Los procesos y procedimientos para el Desarrollo de Software se basan en el Modelo de Procesos de Desarrollo RUP, aplicando para la calidad del proceso, las mejores prácticas de CMMI para Desarrollo establecidas en el nivel de madurez 2 y la Norma de Calidad *SQUARE* para control de calidad del Producto.

Definición del Dueño del Proceso.

El dueño del proceso debe asumir la responsabilidad de gestionarlo y de su mejora continua, para lo cual debe tener la suficiente autoridad que le permita

implantar los cambios que conduzcan a la mejora de sus resultados. Por lo tanto queda planteada su necesidad en el modelo propuesto, para que la misma sea analizada por la instancia que la Institución considere adecuada.

Definición de Elementos.[32]

Elementos del Modelo Propuesto para el Desarrollo de Software basado en RUP				Fases		
Flujo	**Productos**	**Roles**	**Herramientas**	**Inicio**	**Elaboración y Construcción**	**Trans**
Modelación del Negocio	Plan del Proyecto	Analista de Procesos del Negocio, Jefe del Proyecto, *Stakeholders*		I	R	
	Caso de Negocio			I		
	Lista de Riesgos			I	R	
Requerimientos	Visión	Analista de Procesos del Negocio, Ingeniero/ Analista de Sistemas	*Rational*	I	R	

	Modelo de Casos de Uso	Ingeniero/ Analista de Sistemas	*Requirements Composer, Rational Team Concert, Rational Quality Manager*	I	R	
	Especificación Adicional			I	R	
	Glosario			I	R	
Análisis y Diseño	Modelo de Diseño	Desarrolladores, *Testers,* Administrador de Aplicativos y BD			I – R	
	Documentación de la arquitectura SW				I – R	
Implementación	Modelo de Implementación				I – R	
Pruebas	Plan de Pruebas				I – R	
Despliegue	Plan de Despliegue					I

I=Inicio, R= Refinamiento.

Detalle de los Productos

Producto	Descripción
Plan del Proyecto	Muestra una breve descripción de lo que está asociado en el proyecto y lo que afecta el desarrollo. *(Ver Anexo IV).*
Caso de Negocio	Es un modelo de las funciones de negocio vistas desde la perspectiva de los actores externos (Agentes de registro, solicitantes finales, otros sistemas etc.). permite situar al sistema en el contexto organizacional haciendo énfasis en sus objetivos. *(Ver Anexo IV).*

Lista de Riesgos	Identifica los eventos, en orden decreciente de prioridad, que podrían afectar el éxito del proyecto. *(Ver Anexo IV).*
Visión	Define la visión del producto desde la perspectiva del cliente, especificando las necesidades y características del producto. Constituye una base de acuerdo en cuanto a los requisitos del sistema. *(Ver Anexo IV).*
Modelo de Casos de Uso	Presenta las funciones del sistema y los actores que hacen uso de ellas. Se representa mediante Diagramas de Casos de Uso. *(Ver Anexo IV).*
Especificación Adicional	Este documento capturará todos los requisitos que no han sido incluidos como parte de los casos de uso y se refieren requisitos no-funcionales globales. Dichos requisitos incluyen: requisitos legales o normas, aplicación de estándares, requisitos de calidad del producto, tales como: confiabilidad, desempeño, etc., u otros requisitos de ambiente, tales como: sistema operativo, requisitos de compatibilidad, etc. *(Ver Anexo IV).*
Glosario	Es un documento que define los principales términos usados en el proyecto. *(Ver Anexo IV).*
Modelo de Diseño	Establece la realización de los casos de uso en clases y pasando desde una representación en términos de análisis (sin incluir aspectos de implementación) hacia una de diseño (incluyendo una orientación hacia el entorno de implementación), de acuerdo al avance del proyecto *(Ver Anexo IV).*
Documentación Arquitectura Software	Permite representar la estructura del sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo, ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones. *(Ver Anexo IV).*
Modelo de Implementación	Este modelo es una colección de componentes y los subsistemas que los contienen. Estos componentes incluyen: ficheros ejecutables, ficheros de código fuente, y todo otro tipo de ficheros necesarios para la implantación y despliegue del sistema *(Ver Anexo IV).*
Plan de Pruebas	Establece las condiciones de ejecución de cada prueba, las entradas de la prueba, y los resultados esperados. Estos casos de prueba son aplicados como pruebas de regresión en cada iteración. Cada caso de prueba llevará asociado un procedimiento de prueba con las instrucciones

	para realizar la prueba, y dependiendo del tipo de prueba dicho procedimiento podrá ser automatizable mediante un script de prueba. *(Ver Anexo IV).*
Plan de Despliegue	Muestra el despliegue, la configuración de tipos de nodos del sistema, en los cuales se hará el despliegue de los componentes. *(Ver Anexo IV).*

Detalle de los Roles y Responsabilidades

Roles	Responsabilidades
Analista de Procesos	El analista se encarga de entender las necesidades y oportunidades dentro de un negocio, son traductores de lo técnico a lo "humano", fungiendo como puente entre el equipo de desarrollo y los representantes de la organización.
Jefe del Proyecto	Es el responsable ante la gerencia del desarrollo y mantención del proyecto.
Ingeniero de Sistemas	Es el responsable de definir la arquitectura lógica y tecnológica utilizada para el desarrollo, soporte, mantención y operación del Sistema.
Desarrolladores	Es el responsable de convertir la especificación del sistema en código fuente ejecutable, utilizando uno o más lenguajes de programación, así como herramientas de software de apoyo a la programación
Testers	Es el responsable de las actividades principales del esfuerzo de las pruebas. Actividades que incluyen identificar, definir, implementar y dirigir las pruebas necesarias, como también verificar los resultados de las pruebas y analizar los resultados.
Administrador de Aplicativos y BD	Es el responsable de la gestión de los aplicativos y la base de datos
Cliente	Es el responsable de aceptar el producto.
Stakeholders	Involucra a todas las personas afectados por el Sistema. Puede incluir a miembros del proyecto, proveedores, clientes, usuarios finales y otros.

Detalle de Herramienta IBM Rational. [29][30]

- **IBM Rational Requirements Composer.**

A través del suministro de varias herramientas visuales y de colaboración, *Requirements Composer* permite la captura y la mejor identificación de las necesidades empresariales sobre los requisitos que conducen a una calidad mejorada, reduce la carga de trabajo y del tiempo de salida del producto alineando todo el ciclo de vida. Esta capacidad para visualizar los objetivos empresariales y los requisitos de TI reduce los problemas causados por la falta de información o una mala comunicación entre los diferentes miembros del equipo.

Proporciona también la capacidad de capturar visualmente información de requisitos como diagramas de procesos del negocio, diagramas de casos de uso, guiones gráficos, esbozos de interfaz de usuario y texto enriquecido para comunicar y articular mejor el contexto de los requisitos.

Entre los beneficios que este producto presenta se encuentran:

- Focaliza los equipos en los problemas empresariales que permiten soluciones más eficaces con la ayuda una arquitectura, calidad y procesos de gestión de requisitos mucho mejores.
- Entrega más rápida del producto gracias a proporcionar un contexto y claridad a la validación de los responsables finales, reduciendo la reelaboración de requisitos ya establecidos y controlando la colaboración para la eficacia de las soluciones.
- Maximiza la reutilización de los artefactos de requisitos.
- Proporciona claridad a los responsables finales sobre la solución a desarrollar mediante esbozos de procesos de negocio, esbozos de IU y

guiones gráficos, casos prácticos y glosarios dinámicos que ayudan a eliminar la confusión y a crear consenso alrededor de los requisitos.

- Se integra con *Rational RequisitePro 7.1* para mejorar las capacidades de gestión de requisitos para una mayor facilidad de seguimiento y alineación con los objetivos empresariales de todo el ciclo de vida.

- **IBM Rational Team Concert**

"*IBM Rational Team Concert* ofrece un entorno de colaboración que auxilia a las personas y a los Grupos de trabajo para lograr el máximo rendimiento. Posibilita un control integrado de la versión del software, una gestión del espacio de trabajo y el soporte de desarrollo en paralelo. Además, está diseñado para interconectar equipos de desarrollo dispersos y aumentar de esta manera la productividad individual y del mismo equipo; para compactar los ciclos de desarrollo y para permitir que los equipos puedan suministrar software de alta calidad en forma rápida. Disponible en las ediciones Standard*, Express y Express-C, Rational Team Concert* está diseñado específicamente para equipos de desarrollo pequeños y medianos."

Esta herramienta permite:

- Mejora la Colaboración en Equipo.
- Soporta una gestión transparente del elemento de trabajo
- Brinda valiosas capacidades de gestión de control de fuente
- Provee una guía automatizada del proceso
- Permite construcciones de software más eficientes.
- Mejora la visibilidad mediante una visualización en tiempo real del estado del proyecto.
- Ayuda ampliar sus inversiones empresariales
- Potencializa ecosistema del Asociado de Negocios IBM
- Potencializa la plataforma de tecnología JAZZ.

- **IBM Rational Quality Manager**

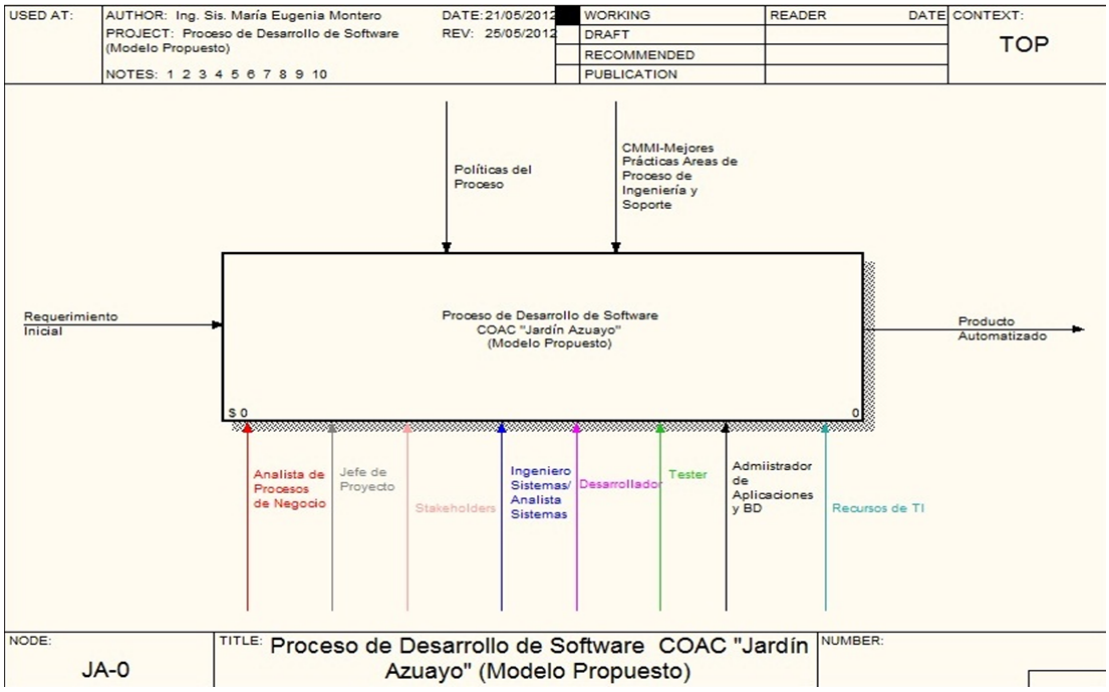
Es una herramienta que ayuda a sincronizar los trabajos en equipo a través del ciclo de vida de desarrollo del software. Automatiza las actividades que requieren un trabajo intensivo, contribuyendo en un mejor control de sus proyectos mediante la provisión de métricas confiables y oportunas. Está construido sobre la plataforma IBM® Jazz™, un entorno colaborativo basado en roles e impulsado por los negocios, que ofrece herramientas para el control de flujos de trabajo, el rastreo y el informe de métricas.

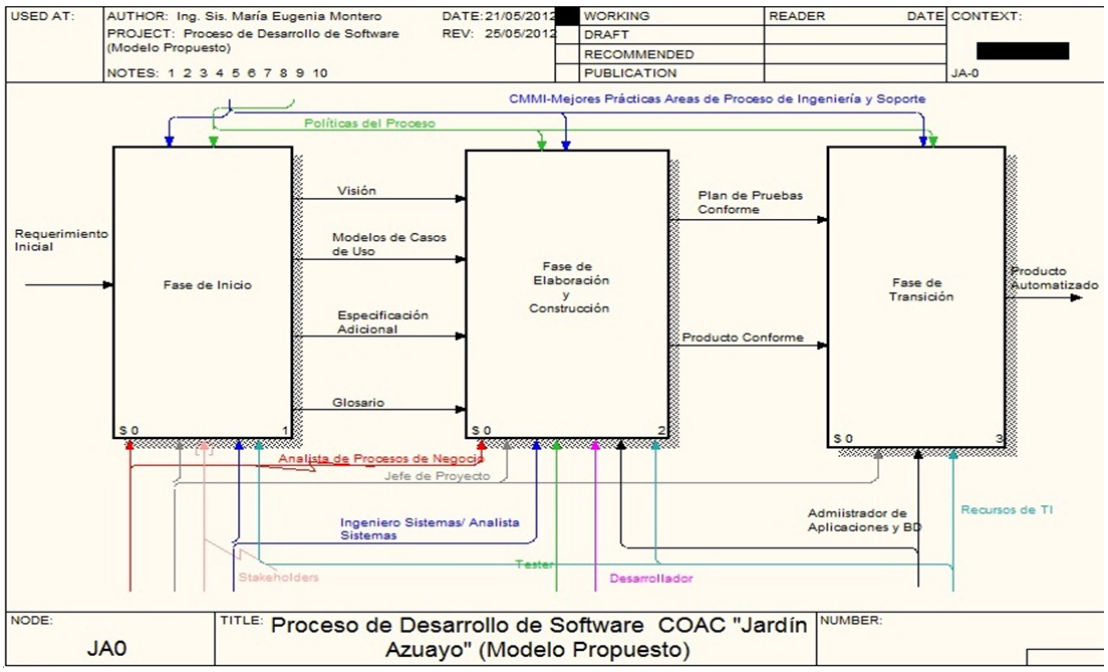
Es un software de gestión de calidad basado en Web utilizado para la planificación de pruebas, testeo manual y la integración con herramientas automatizadas de testeo utilizado por todos los equipos de prueba y soporta una gran cantidad de funciones de usuario – como por ejemplo, gerente de pruebas, arquitecto de pruebas, líder de pruebas y gerente de laboratorio – y de roles fuera de las organizaciones de prueba.

IBM Rational Quality Manager incluye una larga lista de características, sin embargo se resume algunas de las funciones clave que se pueden realizar con ésta herramienta:

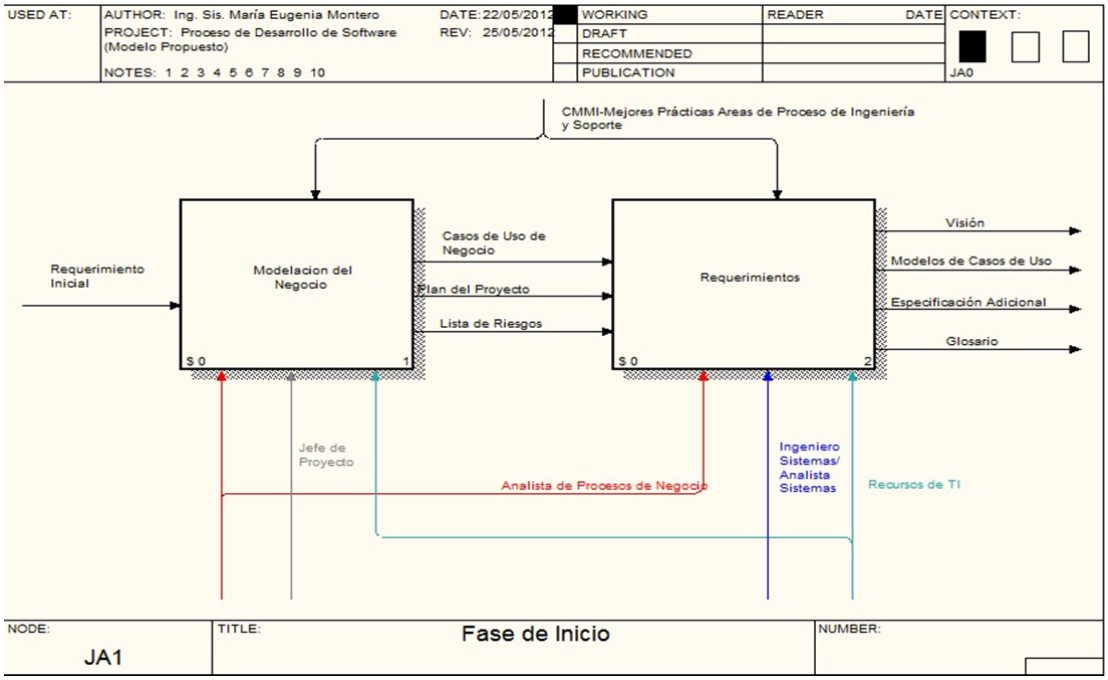
- Se comparte información sin contratiempos
- Gestiona todo el ciclo del vida del proyecto
- Gestiona el testeo manual
- Utiliza la automatización para acelerar los cronogramas del proyecto.
- Informe sobre las métricas del proyecto para la toma de decisiones.

4.4.5 Diseño de los Procesos y Procedimientos del Modelo Propuesto

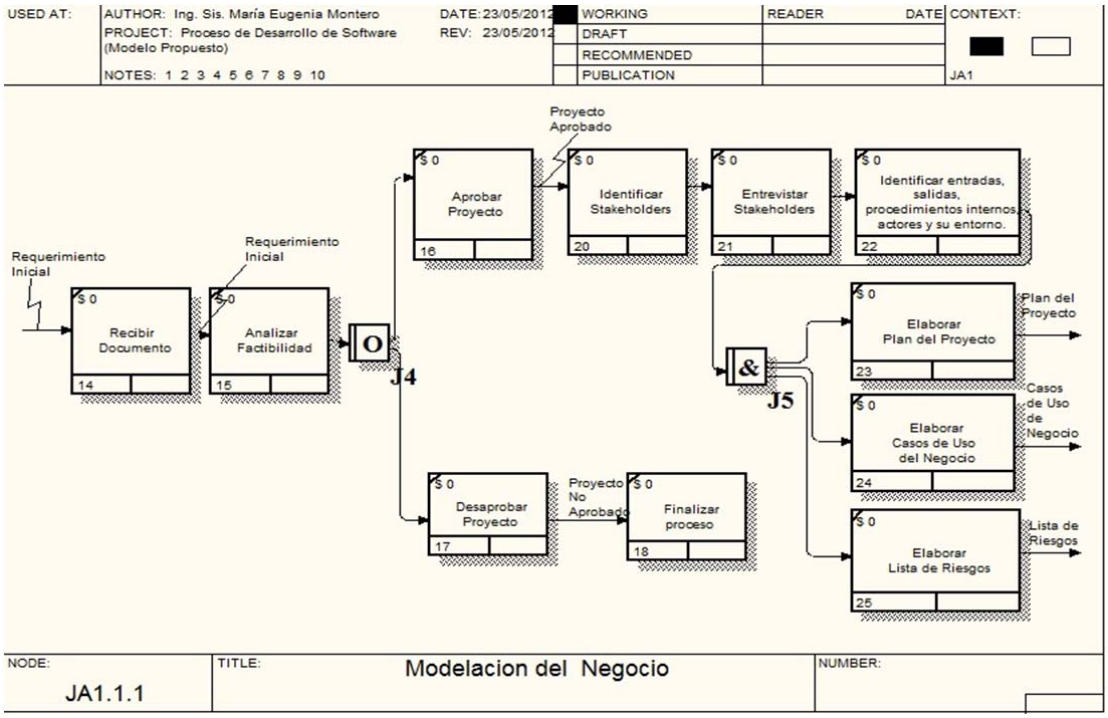




Fase de Inicio



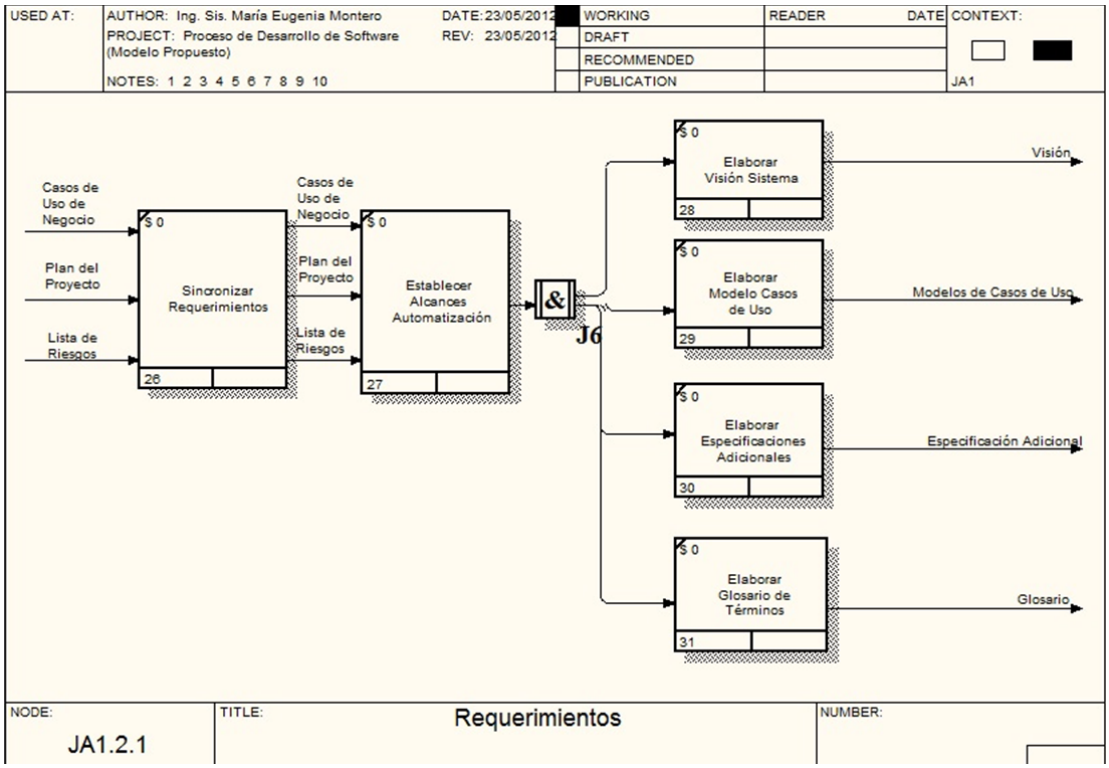
Modelación del Negocio



Nombre:	**Modelación del Negocio**
Entrada:	Requerimiento Inicial
Salida:	Plan del Proyecto, Casos de Uso del Negocio, Lista de Riesgos
Actores:	Analista de Procesos del Negocio Jefe Proyecto *Stakeholders*
Procedimiento:	1. El Analista de Proceso recibe el Documento de Requerimientos Iniciales. 2. El Analista de Procesos y Jefe del Proyecto realizan el estudio de factibilidad identificando las necesidades, los procesos claves del negocio a los cuales se afecta la automatización, recursos y dificultades que representa la consecución del mismo. 3. Si no es factible, el proyecto no se aprueba y se da por finalizado el proceso.

	4. Si es factible, el proyecto se aprueba 5. El Analista de Procesos con el Jefe de Proyecto identifican los principales *stakeholders* del proyecto. 6. El Analista de Procesos realiza entrevistas con los *stakeholders* para determinar su "Visión del Sistema" , es decir lo qué esperan de él. 7. El Analista de Procesos identifica entradas, salidas, procedimientos internos, actores, y su entorno. 8. El Analista de Procesos realiza los casos de Uso del Negocio. 9. El Jefe de Proyecto elabora el Plan del Proyecto y la Lista de Riesgos con la participación de los actores

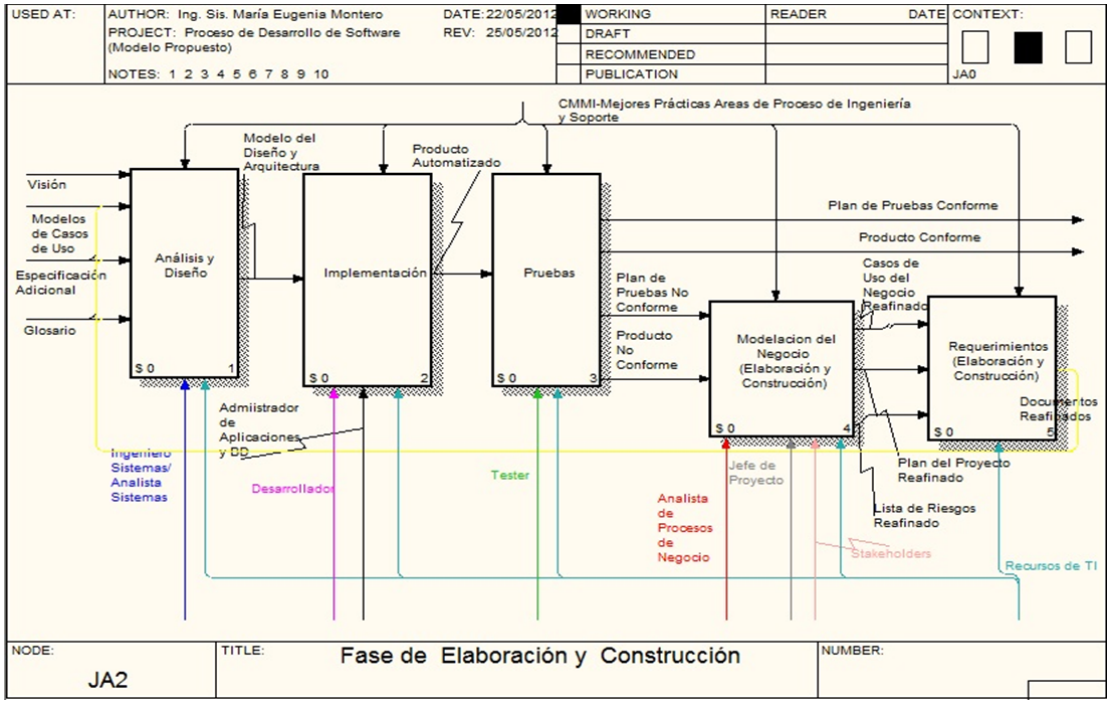
Requerimientos



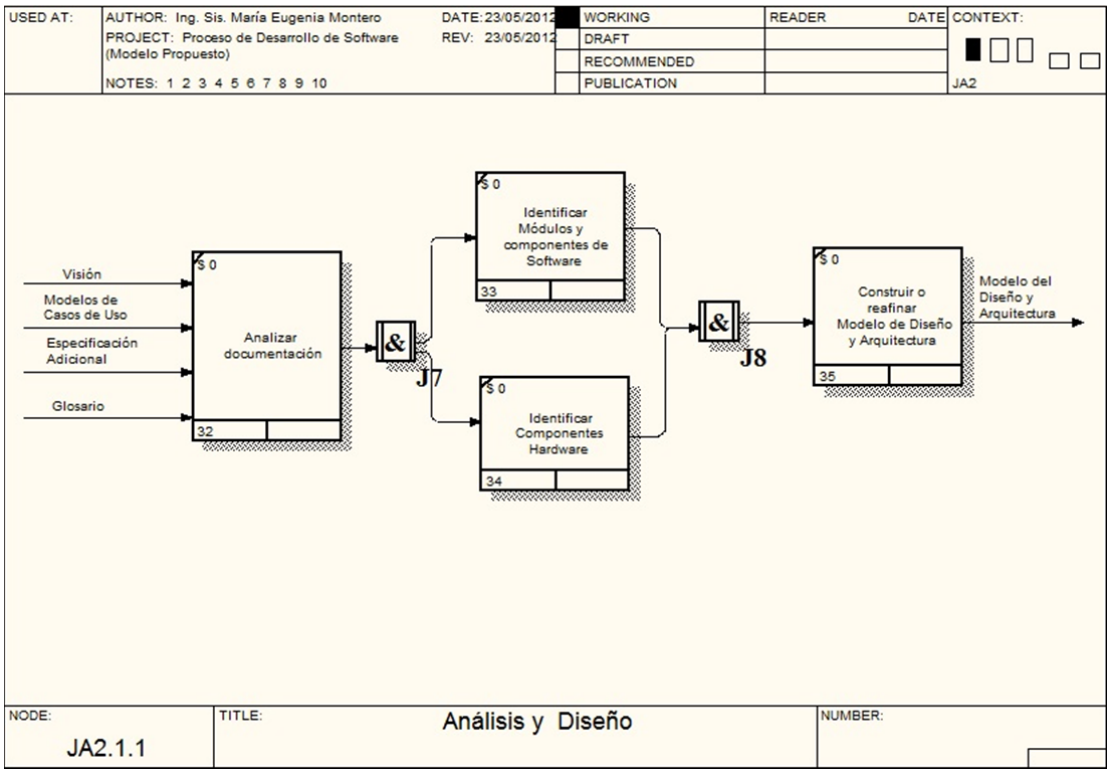
Nombre:	Especificar Requerimientos
Entrada:	Plan del Proyecto, Lista de Riesgos, Casos de Uso del Negocio,
Salida:	Documento de Visión del Sistema, Modelo de Casos de Uso, Especificaciones Adicionales, Glosario de Términos.
Actores:	Analista de Procesos del Negocio Ingeniero Sistemas/Analista de Sistemas
Procedimiento:	***1.*** El Analista de Procesos y el Ingeniero de Sistemas sincronizar los requerimientos de los *stakeholders* **2.** Establecer alcances de lo que se puede automatizar. **3.** El Analista de Procesos elabora el Documento de

	Visión del Sistema. **4.** El Ingeniero de Software/Analista de Sistemas elabora el modelo de casos de uso, el documento de especificaciones adicionales y el glosario de términos. **5.** Se pasa a la fase de “Elaboración y Construcción”.

Fase de Elaboración y Construcción



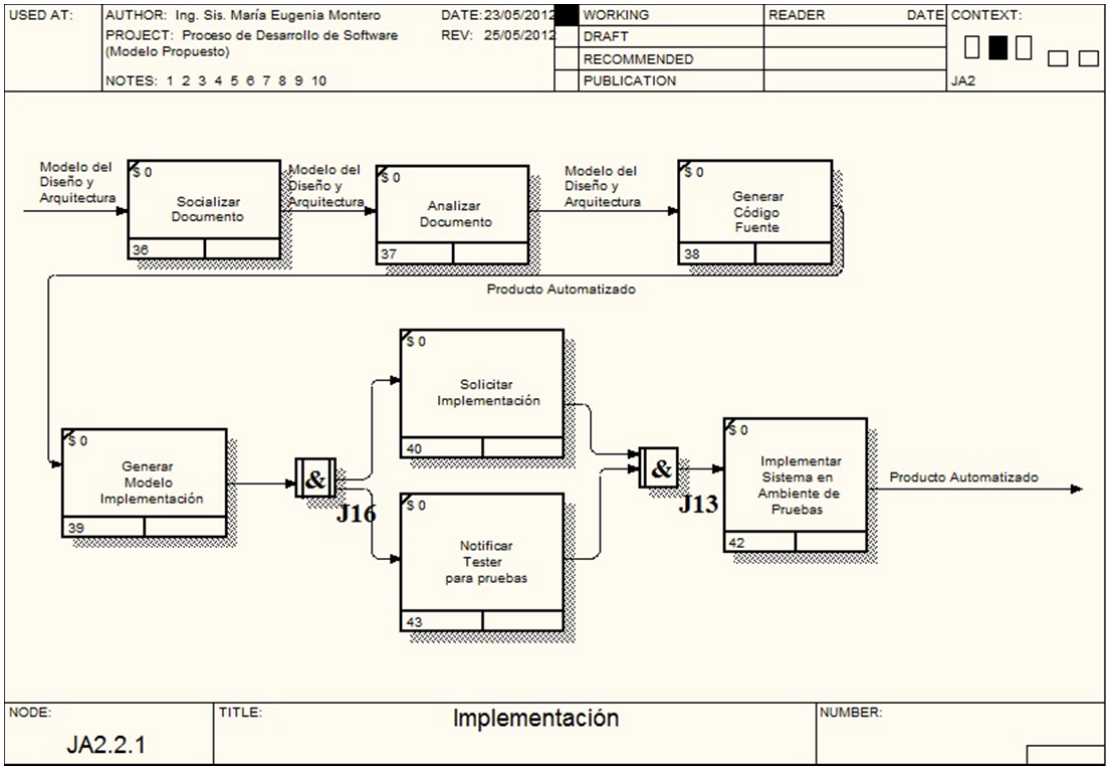
Análisis y Diseño



Nombre:	Análisis y Diseño
Entrada:	Documento de Visión del Sistema, Modelo de Casos de Uso, Especificaciones Adicionales, Glosario de Términos.

Salida:	Modelo del Diseño y Arquitectura del Software
Actores:	Ingeniero/Analista Software
Procedimiento:	1. El Ingeniero/Analista de Sistemas analiza la documentación entregada 2. Identifica los módulos, componentes de software y hardware que intervienen en el sistema. 3. Construye o reafina el Modelo de Diseño y Arquitectura del Software.

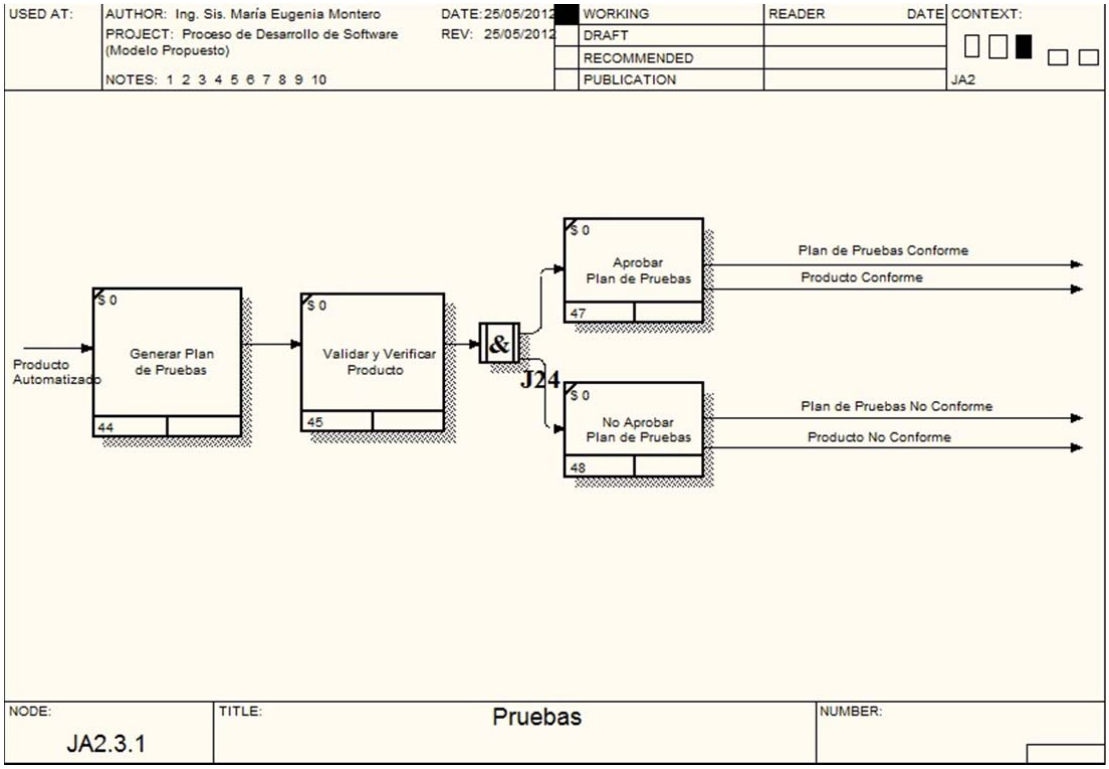
Implementación



Nombre:	**Implementación**
Entrada:	Modelo del Diseño y Arquitectura del Software
Salida:	Producto Automatizado
Actores:	Ingeniero/Analista de Sistemas Desarrollador Administrador de Aplicaciones y Base de Datos
Procedimiento:	1. El Ingeniero/Analista de Software socializa los documentos generados con el Desarrollador. 2. El Desarrollador analiza la documentación. 3. El Desarrollador genera el código fuente necesario para automatizar el requerimiento. 4. El Desarrollador genera el Modelo de Implementación. 5. El Desarrollador solicita implementar la solución al Administrador de Aplicativos y Base de Datos.

	6. El Administrador de Aplicativos y Base de Datos implementa la solución en el ambiente formal de pruebas. 7. El Desarrollador notifica al *Tester* que la versión N del producto está lista para ser probada.

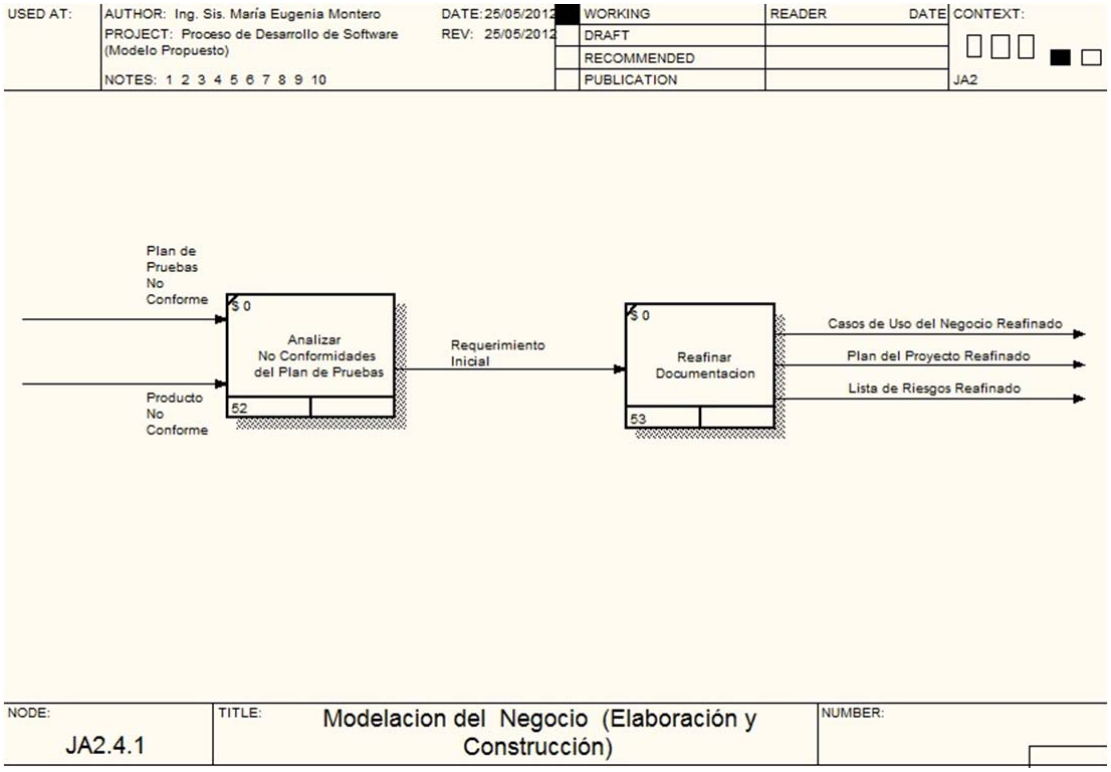
Pruebas



Nombre:	Pruebas
Entrada:	Requerimiento Automatizado.
Salida:	Plan de Pruebas, Plan de Pruebas Conforme, Producto Conforme o Plan de Pruebas inconforme, Producto Inconforme
Actores:	*Tester*
Procedimiento:	1. El *Tester* genera el Plan de Pruebas a realizar. 2. El *Tester* realiza las pruebas para verificar que el requerimiento

	automatizado se ajusta a lo ofrecido y concuerde con la Visión del Sistema que tiene el cliente. Esta verificación generalmente involucra una revisión de pares con otro ingeniero con el fin de examinar la solución planteada por el arquitecto del proyecto y proponer cambios o mejoras 3. El *Tester* realiza la validación del producto. 4. Si existen no conformidades en las pruebas se pasa a la actividad "Modelación del Negocio-Fase de Elaboración y Construcción", repitiendo las iteraciones que se establecieron en el proyecto. 5. Si las pruebas son satisfactorias se pasa a la fase de Transición.

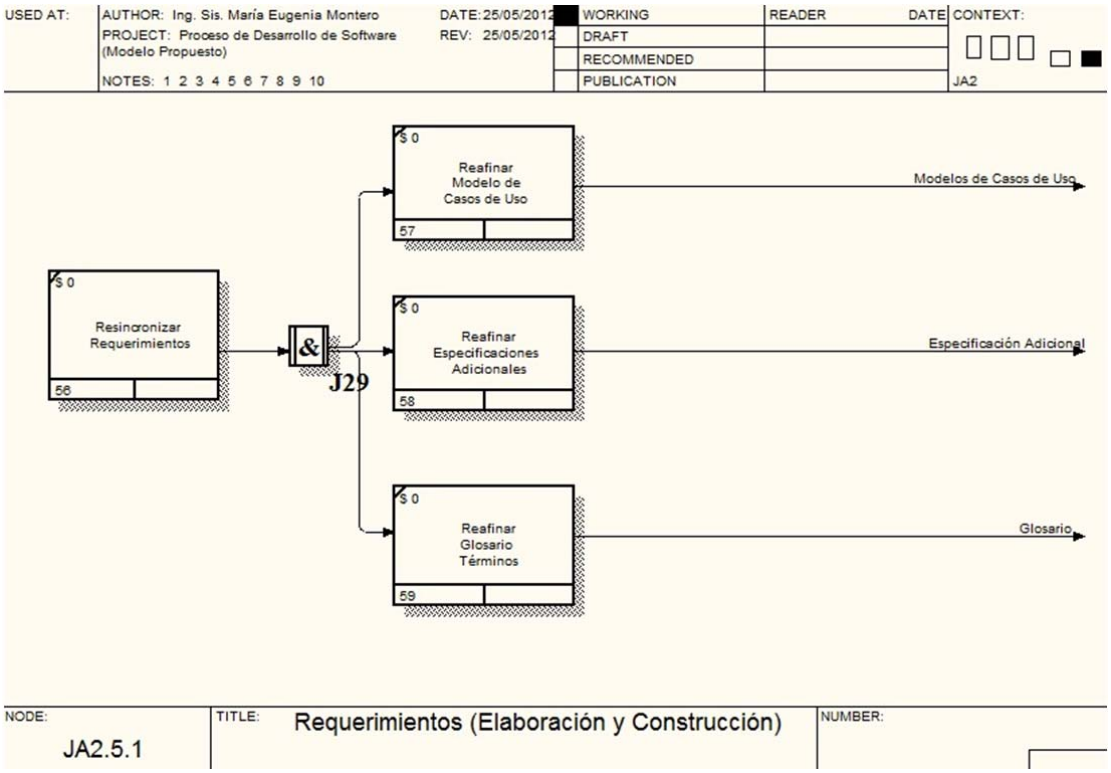
Modelación del Negocio



Nombre:	Modelación del Negocio
Entrada:	Documento de Requerimientos Iniciales, Plan del Proyecto, Casos de

	Uso del Negocio, Lista de Riesgos
Salida:	Plan del Proyecto, Casos de Uso del Negocio, Lista de Riesgos (Refinados)
Actores:	Analista de Procesos del Negocio *Stakeholders*
Procedimiento:	**1.** El Analista de Proceso y Jefe de Proyecto analizan los resultados de las pruebas. **2.** El Analista de Proceso y el Jefe de Proyecto revisan y reafinan los documentos generados por ellos.

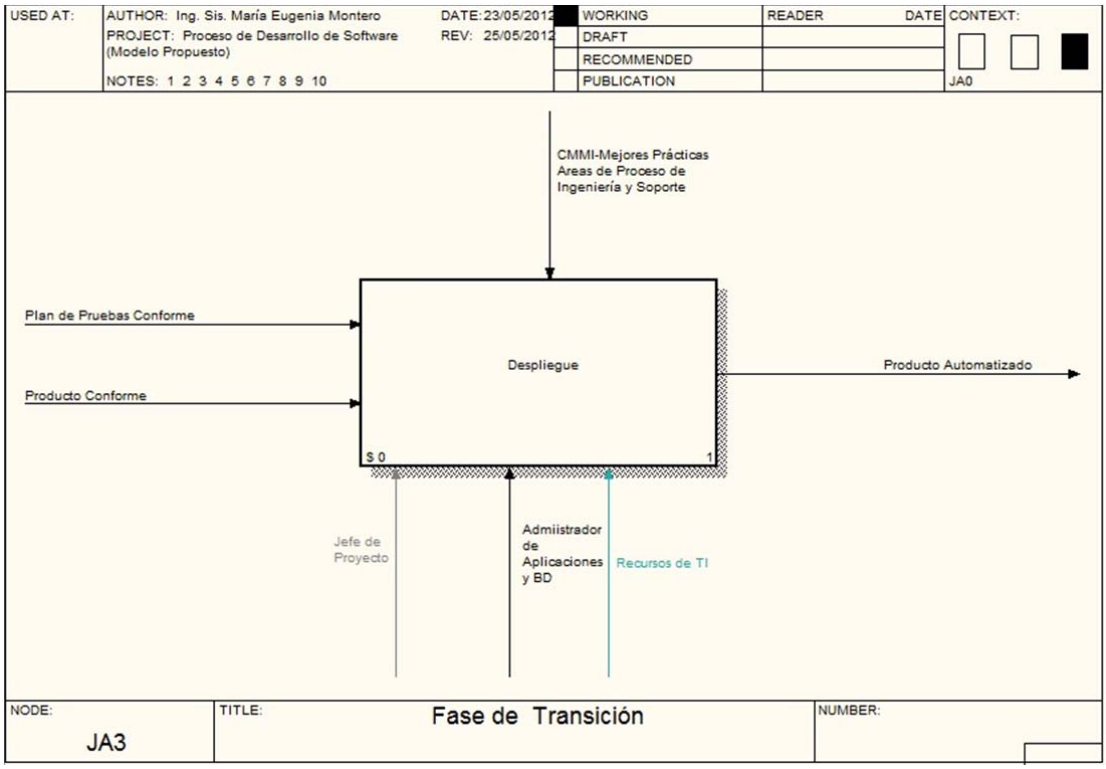
Requerimientos



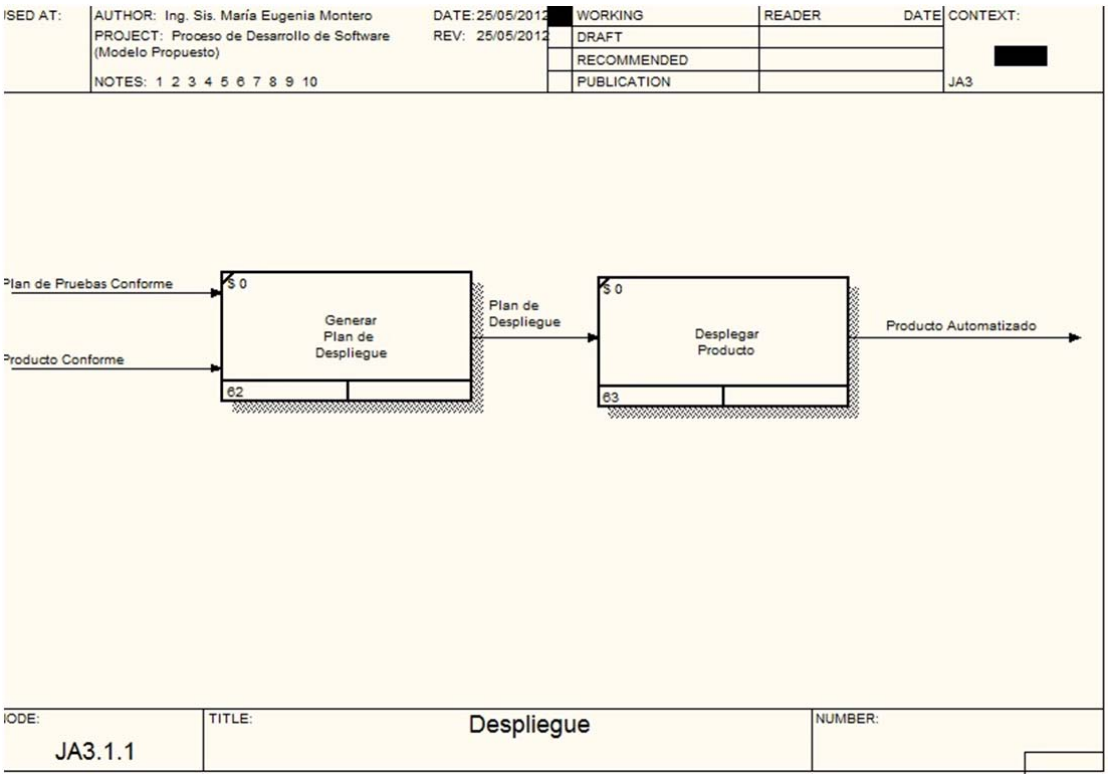
Nombre:	**Requerimientos**
Entrada:	Plan del Proyecto, Lista de Riesgos, Casos de Uso del Negocio

	refinados
Salida:	Documento de Visión del Sistema, Modelo de Casos de Uso, Especificaciones Adicionales, Glosario de Términos Refinados
Actores:	Analista de Procesos del Negocio Ingeniero Sistemas/Analista de Sistemas
Procedimiento:	1. El Analista de Procesos reafina el Documento de Visión del Sistema. 2. El Ingeniero de Software/Analista de Sistemas reafina el modelo de casos de uso, el documento de especificaciones adicionales y el glosario de términos. 3. Continuar con la actividad de "Análisis y Diseño – Fase de Elaboración y Construcción"

Fase de Transición



Despliegue



Nombre:	Despliegue
Entrada:	Plan de Pruebas aprobado, Producto aprobado
Salida:	Plan de Despliegue, Producto automatizado.

Actores:	Administrador de Aplicativos y Base de Datos
Procedimiento:	1. El Administrador de Aplicativos y Base de Datos genera el Plan de despliegue del producto tomando en cuenta la criticidad del mismo. 2. Despliega la solución en el ambiente de producción.

4.4.6 Mejores Prácticas Modelo CMMI.[12]

A continuación se describen las mejores prácticas en las Áreas de Proceso de Ingeniería y Soporte, las mismas que se propone aplicar en el modelo de proceso de desarrollo, por estar de acorde al nivel de madurez y al tipo de producto resultante del mismo.

- **Áreas de Proceso de Ingeniería**

  - **Gestión de Requerimientos. Metas específicas y prácticas específicas**.

SG 1 Gestionar los requerimientos.

SP 1.1 Obtener una comprensión de los requerimientos.

SP 1.2 Obtener el compromiso sobre los requerimientos.

SP 1.3 Gestionar los cambios de los requerimientos.

SP 1.4 Mantener la trazabilidad bidireccional de los requerimientos.

SP 1.5 Identificar las inconsistencias entre el trabajo del proyecto y los requerimientos.

- **Desarrollo de Requerimientos. Metas específicas y prácticas específicas.**

SG 1 Desarrollar los requerimientos de cliente.

SP 1.1 Obtener las necesidades.

SP 1.2 Desarrollar los requerimientos de cliente.

SG 2 Desarrollar los requerimientos de producto.

SP 2.1 Establecer los requerimientos de producto y de componentes del producto.

SP 2.2 Asignar los requerimientos de componentes del producto.

SP 2.3 Identificar los requerimientos de interfaz.

SG 3 Analizar y validar los requerimientos.

SP 3.1 Establecer los conceptos operativos y los escenarios.

SP 3.2 Establecer una definición de la funcionalidad requerida.

SP 3.3 Analizar los requerimientos.

SP 3.4 Analizar los requerimientos para alcanzar el equilibrio.

SP 3.5 Validar los requerimientos.

- **Gestión de Riesgos. Metas específicas y prácticas específicas.**

SG 1 Preparar la gestión de riesgos.

SP 1.1 Determinar las fuentes y las categorías de los riesgos.

SP 1.2 Definir los parámetros de los riesgos.

SP 1.3 Establecer una estrategia de gestión de riesgos.

SG 2 Identificar y analizar los riesgos.

SP 2.1 Identificar riesgos.

SP 2.2 Evaluar, categorizar y priorizar los riesgos.

SG 3 Mitigar los riesgos.

SP 3.1 Desarrollar los planes de mitigación de riesgo.

SP 3.2 Implementar los planes de mitigación de riesgo.

- **Solución Técnica. Metas específicas y prácticas específicas**.

SG 1 Seleccionar las soluciones de componentes de producto.

SP 1.1 Desarrollar las soluciones alternativas y los criterios de selección.

SP 1.2 Seleccionar las soluciones de componentes de producto.

SG 2 Desarrollar el diseño.

SP 2.1 Diseñar el producto o el componente de producto.

SP 2.2 Establecer un paquete de datos técnicos.

SP 2.3 Diseñar las interfaces usando criterios.

SP 2.4 Realizar los análisis sobre si hacer, comprar o reutilizar.

SG 3 Implementar el diseño de producto.

SP 3.1 Implementar el diseño.

SP 3.2 Desarrollar la documentación de soporte de producto.

- **Validación. Metas específicas y prácticas específicas**

SG1 Preparar la validación.

SP 1.1 Seleccionar los productos a validar.

SP 1.2 Establecer el entorno de validación.

SP 1.3 Establecer los procedimientos y los criterios de validación.

SG 2 Validar el producto o los componentes de producto.

SP 2.1 Realizar la validación.

SP 2.2 Analizar los resultados de la validación.

- **Verificación. Metas específicas y prácticas específicas**

SG 1 Preparar la verificación.

SP 1.1 Seleccionar los productos de trabajo a verificar.

SP 1.2 Establecer el entorno de verificación.

SP 1.3 Establecer los procedimientos y los criterios de verificación.

SG 2 Realizar revisiones entre pares.

SP 2.1 Preparar las revisiones entre pares.

SP 2.2 Llevar a cabo las revisiones entre pares.

SP 2.3 Analizar los datos de la revisión entre pares.

SG 3 Verificar los productos de trabajo seleccionados.

SP 3.1 Realizar la verificación.

SP 3.2 Analizar los resultados de la verificación.

- **Áreas de Proceso de Soporte**

  - **Planificación del Proyecto. Metas específicas y prácticas específicas**

SG 1 Establecer estimaciones.

SP 1.1 Estimar el alcance del proyecto.

SP 1.2 Establecer las estimaciones de los atributos del producto de trabajo y de las tareas.

SP 1.3 Definir el ciclo de vida del proyecto.

SP 1.4 Determinar las estimaciones de esfuerzo y de coste.

SG 2 Desarrollar un plan de proyecto.

SP 2.1 Establecer el presupuesto y el calendario.

SP 2.2 Identificar los riesgos del proyecto.

SP 2.3 Planificar la gestión de los datos.

SP 2.4 Planificar los recursos del proyecto.

SP 2.5 Planificar el conocimiento y las habilidades necesarias.

SP 2.6 Planificar la involucración de las partes interesadas.

SP 2.7 Establecer el plan de proyecto.

SG 3 Obtener el compromiso con el plan.

SP 3.1 Revisar los planes que afectan al proyecto.

SP 3.2 Reconciliar los niveles de trabajo y de recursos.

SP 3.3 Obtener el compromiso con el plan.

  - **Gestión de Configuración. Metas específicas y prácticas específicas**

SG 1 Establecer líneas base.

SP 1.1 Identificar elementos de configuración.

SP 1.2 Establecer un sistema de gestión de configuración.

SP 1.3 Crear o liberar líneas base.

SG 2 Seguir y controlar los cambios.

SP 2.1 Seguir las peticiones de cambio.

SP 2.2 Controlar los elementos de configuración.

SG 3 Establecer la integridad.

SP 3.1 Establecer registros de gestión de configuración.

SP 3.2 Realizar auditorías de configuración.

- **Medición y Análisis. Metas específicas y prácticas específicas**

SG 1 Alinear las actividades de medición y análisis.

SP 1.1 Establecer los objetivos de medición.

SP 1.2 Especificar las medidas.

SP 1.3 Especificar los procedimientos de recogida y de almacenamiento de datos.

SP 1.4 Especificar los procedimientos de análisis.

SG 2 Proporcionar los resultados de la medición.

SP 2.1 Recoger los datos de la medición.

SP 2.2 Analizar los datos de la medición.

SP 2.3 Almacenar los datos y los resultados.

SP 2.4 Comunicar los resultados.

- **Monitorización y Control del Proyecto. Metas específicas y prácticas específicas**.

SG 1 Monitorizar el proyecto frente al plan.

SP 1.1 Monitorizar los parámetros de planificación del proyecto.

SP 1.2 Monitorizar los compromisos.

SP 1.3 Monitorizar los riesgos del proyecto.

SP 1.4 Monitorizar la gestión de datos.

SP 1.5 Monitorizar la involucración de las partes interesadas.

SP 1.6 Llevar a cabo revisiones de progreso.

SP 1.7 Llevar a cabo revisiones de hitos.

SG 2 Gestionar las acciones correctivas hasta su cierre.

SP 2.1 Analizar problemas.

SP 2.2 Llevar a cabo las acciones correctivas.

SP 2.3 Gestionar las acciones correctivas.

- **Aseguramiento de la Calidad del Proceso y Producto. Metas específicas y prácticas específicas.**

SG 1 Evaluar objetivamente los procesos y los productos de trabajo.

SP 1.1 Evaluar objetivamente los procesos.

SP 1.2 Evaluar objetivamente los productos de trabajo y los servicios.

SG 2 Proporcionar una visión objetiva.

SP 2.1 Comunicar y asegurar la resolución de las no conformidades..

SP 2.2 Establecer registros.

Areas de Proceso de Ingeniería (CMMI)					MODELO PROPUESTO PARA EL PROCESO DE DESARROLLO DE SOFTWARE					
					PROCESO	ROL	PRODUCTOS EN CADA FASE			
							INICIO	ELABORACION Y CONSTRUCCIÓN		TRANSICION
							Generar	Generar	Reafinar	Generar
			Desarrollo de Requerimientos	Gestión de Requerimientos	Modelación del Negocio	Analista de Procesos del Negocio	Requerimiento inicial Plan de Proyecto, Casos de Uso del Negocio, Lista de Riesgos		Plan de Proyecto, Lista de Riesgos	
					Requerimientos	Analista de Procesos del Negocio	Documento de Visión		Documento de Visión	
						Ingeniero Sistemas/Analista de Sistemas	Modelo de Casos de Uso, Especificación Adicional y Glosario de Términos		Modelo de Casos de Uso, Especificación Adicional y Glosario de Términos	
					Análisis y Diseño	Desarrollador		Modelo de Diseño, Arquitectura del Software	Modelo de Diseño, Arquitectura del Software	
					Implementación	Desarrollador		Producto Modelo de Implementacion	Producto Modelo de Implementacion	
						Administrador de Aplicaciones				
					Pruebas	Tester		Plan de Pruebas	Plan de Pruebas	
					Si el producto no pasa las pruebas realizadas repetir el flujo de actividades en esta fase una vez mas y reafinar los productos necesarios			Iteración (n+1)		
					Despliegue	Tester				
						Administrador de Aplicaciones				Plan de Despliegue
					Area de Proceso de Soporte (CMMI) - Nivel de Madurez 2	Planificación del Proyecto Gestión de Configuración Monitorización y Control del Proyecto Medición y Análisis Aseguramiento de la Calidad del Producto y del Proceso				

Figura 36. Arquitectura del Modelo Propuesto

Fuente: Elaboración Propia

- **Métricas Propuestas [34]**

  - **Métricas Internas. Funcionalidad.**

Adecuación. Suficiencia Funcional

Nombre	Suficiencia Funcional
Propósito:	Cómo se calificaría las funciones controladas
Método de Aplicación:	Contar el número de funciones que están implementadas para el desempeño de las tareas, entonces medir el radio de esto hacia las funciones implementadas. Se puede medir: -Todas las partes del diseño de especificaciones. - Módulos completos /partes de producto de software las métricas comparado con el número de funciones evaluadas.
Medición, Fórmula:	X= 1-A/B A=Número de funcionalidades en donde se detectaron problemas en la evaluación B=Número de funcionalidades controladas
Interpretación:	0<=X<=1, Es correcto mientras más se acerca a 1.
Tipo de Escala:	Absoluto
Tipo de Medida:	X=Contable A=Contable B=Contable
Fuente de Medición:	Especificación de Requerimientos Diseño Código fuente Reporte de Revisión
ISO/IEC 12207: **SLCP:**	Verificación Revisión conjunta
Audiencia:	Desarrolladores Solicitantes

Adecuación. Integridad de la implementación funcional

Nombre	Integridad de la implementación funcional
Propósito:	Saber cuán completa es la implementación de las funcionalidades
Método de Aplicación:	Contar el número de funcionalidades faltantes detectadas en la evaluación y comparar con el número de funcionalidades descritas en la especificación de los requerimientos.
Medición, Fórmula:	X= 1-A/B A=Número de funcionalidades faltantes detectadas en la evaluación. B=Número de funcionalidades descritas en la especificación de requerimientos.
Interpretación:	0<=X<=1, Es correcto mientras más se acerca a 1.
Tipo de Escala:	Absoluto
Tipo de Medida:	X=Contable A=Contable B=Contable
Fuente de Medición:	Especificación de Requerimientos Diseño Código Fuente Reportes
ISO/IEC 12207:	Verificación
SLCP:	Revisión conjunta
Audiencia:	Desarrolladores Solicitantes

- **Métricas Externas. Funcionalidad.**

Adecuación. Suficiencia Funcional

Nombre:	**Suficiencia Funcional**
Propósito:	Cómo se calificaría las funciones evaluadas
Método de Aplicación:	Número de funciones que están implementadas para el desempeño de las tareas comparado con el número de funciones evaluadas.
Medición, Fórmula:	X= 1-A/B A=Número de funcionalidades en donde se detectaron problemas en la evaluación B=Número de funcionalidades evaluadas.
Interpretación:	0<=X<=1, Es correcto mientras más se acerca a 1.
Tipo de Escala:	Absoluto
Tipo de Medida:	X=Contable A=Contable B=Contable
ISO/IEC 12207: **SLCP:**	Validación Aseguramiento de Calidad Calificación de Pruebas
Fuente de Medición:	Especificación de Requerimientos Reporte de Evaluación
Audiencia:	Desarrolladores SQA

Adecuación. Integridad de la implementación funcional

Nombre:	**Integridad de la implementación funcional**
Propósito:	Saber cuán completa es la implementación de acuerdo a la especificación de requerimientos.
Método de Aplicación:	Hacer pruebas funcionales (pruebas de caja negra) del sistema de acuerdo a la especificación de requerimientos.
Medición, Fórmula:	X= 1-A/B A=Número de funcionalidades faltantes detectadas en la evaluación. B=Número de funcionalidades descritas en la especificación de requerimientos.
Interpretación:	0<=X<=1, Es correcto mientras más se acerca a 1.
Tipo de Escala:	Absoluto
Tipo de Medida:	X=Contable A=Contable B=Contable
Fuente de Medición:	Especificación de Requerimientos Reporte de Evaluación
ISO/IEC 12207:	Validación
SLCP:	Aseguramiento de Calidad Calificación de Pruebas
Audiencia:	Desarrolladores SQA

CONCLUSIONES

Con la elaboración del presente trabajo se afianzaron los conocimientos impartidos durante la Maestría de Gerencias de Sistemas de Información, en relación a los temas de Modelos de Procesos de Desarrollo y Calidad de Software, los mismos que en el entorno informático de Cuenca no tienen aún el apoyo necesario, sin embargo, queda cimentada la necesidad de motivar a las altas gerencias tanto administrativas como tecnológicas, para que analicen el costo-beneficio de iniciar mejoras en la manera en la que se desempeñan ciertas áreas del negocio como es la de Desarrollo de Software.

Utilizando dichos conocimientos se definió el Modelo Actual del Proceso de Desarrollo de Software utilizado en la Cooperativa “Jardín Azuayo”, en el cual se verificó la existencia de una serie de inconsistencias en su estructura, mismas que han generado problemas tanto en la gestión de los proyectos, como en los productos entregados por la Unidad de Desarrollo de Software evitando de esta manera que se cumpla con uno de los objetivos de la Unidad, que es la de entregar un producto eficaz, efectivo, eficiente y de calidad.

De hecho se pudo evidenciar que no existe un previo análisis de factibilidad para iniciar el desarrollo de las órdenes requeridas, dando lugar a que se inviertan ineficientemente tanto recursos tecnológicos, económicos y humanos, en proyectos que nunca salieron a producción. La falta del modelo del negocio para iniciar el desarrollo conjugado con la poca experiencia del personal del Área de Desarrollo en el *Core*, es otro factor que pasa desapercibido, pero que repercute notablemente tanto en la estimación de los tiempos del proyecto, como en el proceso de desarrollo propiamente dicho.

En relación al producto entregado, la ausencia de controles de calidad, hacen que éste no garantice el cumplimiento de los requerimientos solicitados por el usuario final, ni especificaciones técnicas necesarias, dando lugar a que en muchas ocasiones los errores o defectos sean detectados una vez que sale a producción,

o lo que es peor, antes de liberarlo, recién en ese momento el usuario final indica que el producto entregado no cumple sus expectativas.

Un modelo con demasiados reprocesos, poco sistémico y desalineado totalmente con los objetivos de la Unidad y de la Institución, con la entrega de productos con errores o defectos, es el resultado de un modelo que no se basa en ningún estándar, ni en ninguna norma de calidad.

RECOMENDACIONES

- Establecer un proceso de socialización y capacitación para concienciar a los involucrados en el proceso de desarrollo, sobre la necesidad de cambiar su metodología, de tal manera que exista el compromiso de apoyar y participar en la implementación de la propuesta realizada.
- Realizar un análisis previo de los recursos y cambios organizacionales que implican la implementación del mismo.
- Se debe realizar un piloto de 3 meses que permita completar n iteraciones del proceso propuesto con el fin de implementar paulatinamente el Modelo Propuesto de Proceso de Desarrollo de Software, en algunos proyectos, de tal manera que se pueda verificar con datos reales cuáles son las ventajas de utilizar un proceso más ordenado y controlado que el actual.
- No es necesario aplicar toda la documentación establecida en este material, lo recomendable es iniciar con la que se creyere la más relevante, para evitar que el proceso se enfoque la generación de documentación .
- Monitorear constantemente cada fase del proceso para establecer puntos críticos y aplicar acciones correctivas, preventivas o de mejora al mismo.
- Incrementar el número de métricas de calidad conforme vaya madurando el modelo. Entre las cuales se podría recomendar [9] :
    - Métrica Interna de Exactitud:

[9] ISO/IEC TR 9126-3:2003: International standard “Software Engineering – product quality – part 3”.[34]

        - *Exactitud computacional:* Mide el grado de precisión en la implementación de los requisitos.
    - Métrica Interna de Funcionalidad
        - *Conformidad con la Funcionalidad:* Mide el grado de cumplimiento de la funcionalidad del producto a las normas aplicables, estándares y convenciones.
    - Métrica Externa de Adecuación:
        - *Alcance de la Implementación Funcional:* Mide cuán correcta es la implementación funcional.
        - *Estabilidad de la Especificación Funcional:* Mide la estabilidad de la especificación funcional después del despliegue.
- Implementar un modelo de calidad en procesos como CMMI una vez que el mismo ya esté institucionalizado.
- Adquirir las herramientas que permitan gestionar el modelo efectivamente.
- Capacitar a los recursos en temas sobre CMMI, Control de Calidad de Software, BPM, de tal manera que se fortalezcan las bases para la continuidad de la mejora en el modelo propuesto.

GLOSARIO DE TÉRMINOS [10]

- **Arquitectura de Software.** Es la manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación entre ellos.
- **Artefacto.** Un artefacto es un producto tangible resultante del proceso de desarrollo de software.
- **Aseguramiento de la Calidad (QA).** Conjunto de actividades diseñadas para asegurar que el proceso de desarrollo y/o mantenimiento es adecuado para que el sistema cumpla sus objetivos.
- **Calidad de Procesos.** Consiste en aplicar la calidad al proceso de fabricación de un producto. Para ello se utilizan técnicas aplicadas sobre muestras tomadas del producto.
- **Caso de Prueba.** Conjunto de valores de entrada, precondiciones de ejecución, resultados esperados y post condiciones de ejecución desarrolladas para un objetivo o condición de prueba particular, tal como probar un determinado camino de un programa.
- **Casos de Uso.** Es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema**.**
- **Control de Calidad (QC).** Conjunto de actividades diseñadas para evaluar el producto de trabajo ya desarrollado.

[10] Tomado de Wikipedia.

- ***Core* del Negocio.** Es aquella actividad capaz de generar valor y que resulta necesaria para establecer una ventaja competitiva beneficiosa para la organización.
- **Especificaciones.** Representa un documento técnico oficial que establezca de forma clara todas las características, los materiales y los servicios necesarios para producir componentes destinados a la obtención de productos**.**
- **Estándares.** Es la redacción y aprobación de normas que se establecen para garantizar el acoplamiento de elementos construidos independientemente, así como garantizar el repuesto en caso de ser necesario, garantizar la calidad de los elementos fabricados, la seguridad de funcionamiento y trabajar con responsabilidad social.
- ***Feedback*.** Retroalimentación.
- **Gestión Total de la Calidad.** Es una estrategia de gestión creada por W. E. Deming orientada a crear conciencia de calidad en todos los procesos organizacionales**.**
- **Guía**. Tratado en que se dan directrices o consejos sobre determinadas materias.
- **Ingeniería de Software.** Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software.
- **Interacción.** Es una acción recíproca entre dos o más objetos, sustancias, personas o agentes**.**
- **Iteración**. Es la repetición de una serie de instrucciones**.**
- **Interfaz**. Representa la capacidad de comunicación entre componentes de software
- **Mejora Continua.** Es una actitud general que debe ser la base
- para asegurar la estabilización del proceso y la posibilidad de mejora.

- **Metodología.** Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.
- **Métricas**. El IEEE "*Standard Glosary of Software Engering Terms*" define como métrica como "una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado" **.**
- **Modelo.** Un modelo es una representación de una realidad compleja. Modelar es desarrollar una descripción lo más exacta posible de un sistema y de las actividades llevadas a cabo en él.
- **Norma de Calidad**. Es un documento, establecido por consenso y aprobado por un organismo reconocido (nacional o internacional), que proporciona para un uso común y repetido, una serie de reglas, directrices o características para las actividades de calidad o sus resultados, con el fin de conseguir un grado óptimo de orden en el contexto de la calidad. Las principales organizaciones internacionales, emisoras de normas de calidad son: ISO (Organización Internacional de Estándares)
- **Planificación.** La planificación es un proceso de toma de decisiones para alcanzar un futuro deseado, teniendo en cuenta la situación actual y los factores internos y externos que pueden influir en el logro de los objetivos**.**
- **Proceso.** Un proceso es un conjunto de actividades o eventos (coordinados u organizados) que se realizan o suceden (alternativa o simultáneamente) bajo ciertas circunstancias con un fin determinado
- **Prototipo.** Borrador de un producto potencial o de una parte del mismo, una simulación de los requisitos.
- **Proyecto.** Es el conjunto de las actividades que desarrolla una persona o una entidad para alcanzar un determinado objetivo**.**
- **Requerimiento.** Características que se desea que posea un sistema o un software**.**
- **Requisito.** Es una necesidad documentada sobre el contenido, forma o funcionalidad de un producto o servicio.

- **Riesgos.** Factor que puede resultar en consecuencias negativas en el futuro. Normalmente se expresa en términos de impacto y probabilidad.
- **Setear.** Configurar
- ***Stub*.** Implementación esquemática o con un propósito especial de un componente software, usado para desarrollar o probar un componente que llama o depende de él. Reemplaza al componente llamado.

ABREVIATURAS

- **CEO** . *Chief Executive Officer*
- **CMMI.** *Capability Maturity Model Integration.*
- **GUI.** *Graphical User Interface*
- **IEEE.** *Institute of Electrical and Electronics Engineers.*
- **ISO.** *International Organization for Standardization.*
- **PL.** *Project Leader.*
- **SPM.** *Software Project Manager.*
- **UML.** *Unified Modeling Language*
- **WEB.** *Word Wide Web.*
- **SEI.** *Software Engineering Institute.*

BIBLIOGRAFÍA

Libros en la WEB

[1] CAMPDERRICH FALGUERAS, Benet. "*Ingeniería del Software (Primera Edición)"* [En línea]. Barcelona. Fundación por la Universidad Abierta de Cataluña. Disponible en Web: <http://books.google.com/books?id=_tKTpr4Ah88C&printsec=frontcover&dq=ingenieria+de+software&hl=en&ei=8kZ5TpzZPMmUOrq0sagC&sa=X&oi=book_result&ct=result&resnum=2&ved=0CDQQ6AEwAQ#v=onepage&q&f=false >

[2] SOMMERVILLE, Ian. *"Ingeniería del Software (Séptima Edición)"* [En línea]. 2005. España. Pearson Educación S.A. Disponible en Web: <http://books.google.com.ec/books?id=gQWd49zSut4C&pg=PA62&dq=modelo+de+cascada&hl=es&ei=3QF4TpLeHJHDswaj6JGjCw&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCoQ6AEwAA#v=onepage&q=modelo%20de%20cascada&f=false>

[3] BARRANCO DE AREBA, Jesús. "*Metodología del Análisis Estructurado de Sistemas (Primera Edición)"* [En línea] . 2005. Universidad Pontificia Comillas de Madrid. Disponible en Web: <http://books.google.com.ec/books?id=PUqxsNVaQC8C&pg=PA51&dq=ventajas+del+modelo+en+cascada&hl=es&ei=xQd4TouED8Tvsga-qMWACw&sa=X&oi=book_result&ct=result&resnum=2&ved=0CC8Q6AEwAQ#v=onepage&q=ventajas%20del%20modelo%20en%20cascada&f=false>

[4] FOROUZAN, Behrouz A. "*Introducción a la Ciencia de la Computación"* [En línea]. 2003. México. THOMSON. Disponible en Web: <http://books.google.com.ec/books?id=xu4jcMBih8QC&pg=PT223&dq=modelos+cascada&hl=es&ei=svJ3Ts-oFcO4tgeV59H7Cw&sa=X&oi=book_result&ct=result&resnum=2&ved=0CC8Q6AEwAQ#v=onepage&q=modelos%20cascada&f=false>

[5] ALONSO F., MARTÍNEZ L., SEGOVIA J. "*Introducción a la Ingeniería del Software: Modelos de Desarrollo de Programas (Primera Edición)"* [En línea]. 2005. España. Delta Publicaciones Universitarias. Disponible en Web: <http://books.google.com.ec/books?id=rXU-WS4UatYC&pg=PA114&dq=desarrollo+de+software&hl=en&ei=yhlwToKNGov34QSz2dXBCQ&sa=X&oi=book_result&ct=result&redir_esc=y#v=onepage&q=desarrollo%20de%20software&f=false>

[6] CORTÉS MORALES, R. "*Introducción al Análisis de Sistemas y la Ingeniería de Software"* [En línea]. 2006. Disponible en Web: <http://books.google.com.ec/books?id=Y2CCT0fIxYwC&pg=PA24&dq=modelo+espiral+en+desarrollo+de+software&hl=es&sa=X&ei=x1cjT5reLabX0QGNuKWFCQ&ved=0CF0Q6AEwCDgK#v=onepage&q=modelo%20espiral%20en%20desarrollo%20de%20software&f=false>

[7] Scacchi, W. 2001. *"Process Models in software engineering"*, en J.J.Marciniak (ed.), Encyclopedia of Software Engineering, 2nd Edition

[8] MURALI CHEMUTURI, 2011. *"Mastering Software Quality Assurance. Best Practices,Tools and Tecniques for Software Developers"* . Editorial J.Ross Publishing.

Documentos

[9] JT. (2007. 7 de Octubre). *"El día en que el estruendo de La Josefina enmudeció al Ecuador"*. El Hoy. [En línea]. Disponible en Web: <http://www.hoy.com.ec/noticias-ecuador/el-dia-en-que-el-estruendo-de-la-josefina-enmudecio-al-ecuador-279096.html>

[10] Cooperativa de Ahorro y Crédito "Jardín Azuayo". 2008. "*Plan Estratégico 2009-2013".*

[11] 2009. *"Procesos de Ingeniería del Software".* [En línea]. Disponible en Web: <http://www.slideshare.net/rfsolano/procesos-de-ingenieria-del-software>.

[12] 2009. "*CMMI Guía para la Integración de Procesos y la Mejora de Productos"*. [En línea]. Disponible en Web: <http://www.sei.cmu.edu/library/assets/cmmi-dev-v12-spanish.pdf>

[13] Escuela de Cooperativismo. Programa para Empleados. EDUCOPE. Enero 2011.

[14] Material dictado por los profesores de la Maestría de Gerencia de Sistemas de Información, Segunda Edición.

[15] FILLOTRANI, Pablo R. *"Calidad en el Desarrollo de Software"* [en línea]. 2007. Disponible en Web: <http://www.cs.uns.edu.ar/~prf/teaching/SQ07/clase2.pdf>.

[16] RICO MENDEZ, Robinson. *"Calidad en los procesos de Software"* [en línea]. 2010. Disponible en Web: <http://www.slideshare.net/guest871c816/introduccionа-calidad-de-software>.

[17] Video *"Importancia del Aseguramiento de la calidad de software".* 2009. Disponible en Web:< http://www.youtube.com/watch?v=WW6vXq7ueMk>

[18] *"Calidad en Software"*. 2008. Disponible en Web: <http://www.slideshare.net/elsuse/calidad-del-software>.

[19] VEGA Lebrún, RIVERA Prieto y GARCÍA Santillán. "*Mejores prácticas para el establecimiento y aseguramiento de la calidad de software"*, Edición electrónica gratuita. (2008). Disponible en Web: <http://www.eumed.net/libros/2008a/351>.

[20] E-clases. *"Conceptos de calidad de software"*. Disponible en Web: <http://eclases.tripod.com/id18.html>.

[21] GORDON SCHULMEYER, G. *"Handbook of Software Quality Fourth Edition".*ARTECH HOUSE, Inc. 2008. Disponible en Web: <http://www.amazon.com/Handbook-Software-Quality-Assurance-Schulmeyer/dp/1596931868/ref=sr_1_4?s=books&ie=UTF8&qid=1298653312&sr =1-4#reader_1596931868>

[22] Wikipedia. *"Caso de prueba".* 2012. Disponible en Web: <http://es.wikipedia.org/wiki/Caso_de_prueba>

[23] ISO25000. 2012. Disponible en Web: <http://iso25000.com/index.php/25000.html>

[24] BOEHM, Barry W. *"A Spiral Model of Sotware Development and Enhancement"*. 1988. Disponible en Web: <http://www.dimap.ufrn.br/~jair/ES/artigos/SpiralModelBoehm.pdf>

[25] CARRETO, Julio. *"Proceso Administrativo".* 2007. Disponible en Web: <http://uproadmon.blogspot.com/2007/03/ejemplo-de-matriz-foda.html>

[26] *"Extreme Programming: A gentle introduction".* Disponible en Web: <http://www.extremeprogramming.org/>

[27] *"Ejemplo de Matriz FODA".* Disponible en Web: <http://grupos.emagister.com/documento/ejemplo_de_matriz_foda/1073-27584>

[28] Societe Generale de Surveillance –SGS. *"Norma ISO 9001: 2008"*. Disponible en Web: <http://www.utpl.edu.ec/iso9001/images/stories/NORMA_ISO_9001_2008.pdf>

[29] IBM. *"IBM Rational Requirements Composer"*. Disponible en Web: <ftp://ftp.software.ibm.com/software/rational/jazz/IBM_Rational_Requirements_Composer.pdf>

[30] IBM. *"Introducción a IBM Rational Quality Manager".* Disponible en Web: <http://www.ibm.com/developerworks/ssa/rational/library/08/1230_kelly/index.html>

[31] *Manual de Procesos "Cooperativa de Ahorro y Crédito Jardín Azuayo".* Disponible en la Intranet: <http://www.jardinazuayo.fin.ec/procesos/manual_1.htm>

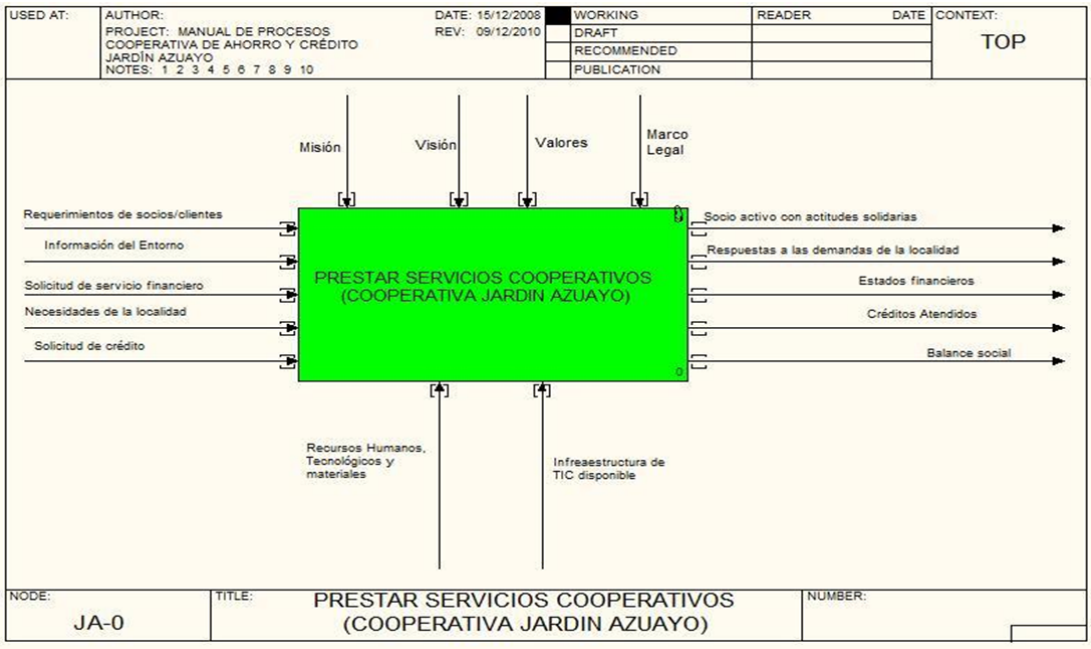
[32] MARTINEZ A., MARTINEZ R. *"Guía a Rational Unified Process".* Disponible en Web: <http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf>

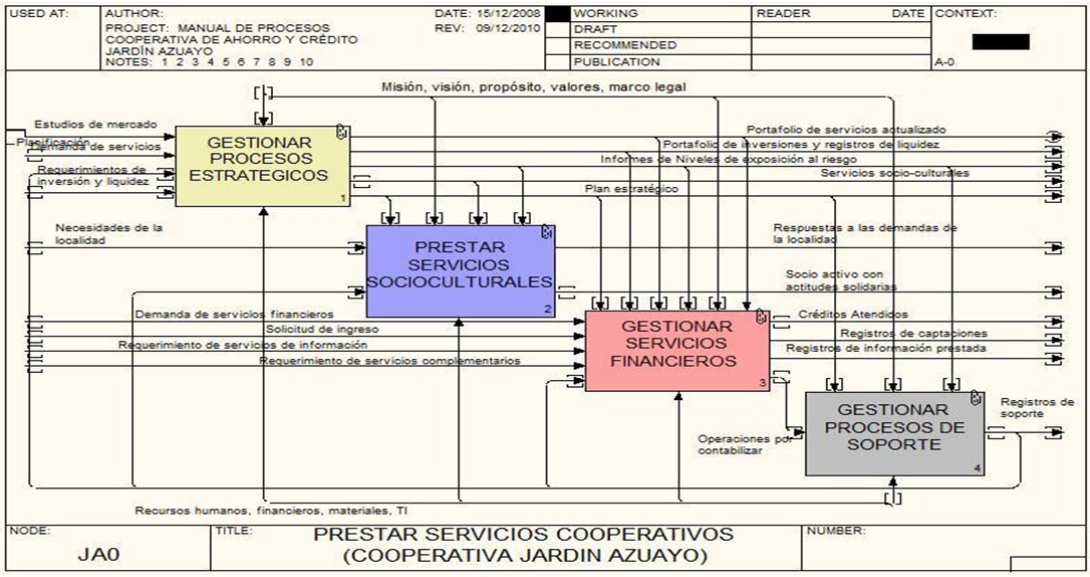
[33] VARIOS AUTORES*. "Manifiesto por el Desarrollo Ágil de Software".* Disponible en Web:< http://agilemanifesto.org/iso/es/>.

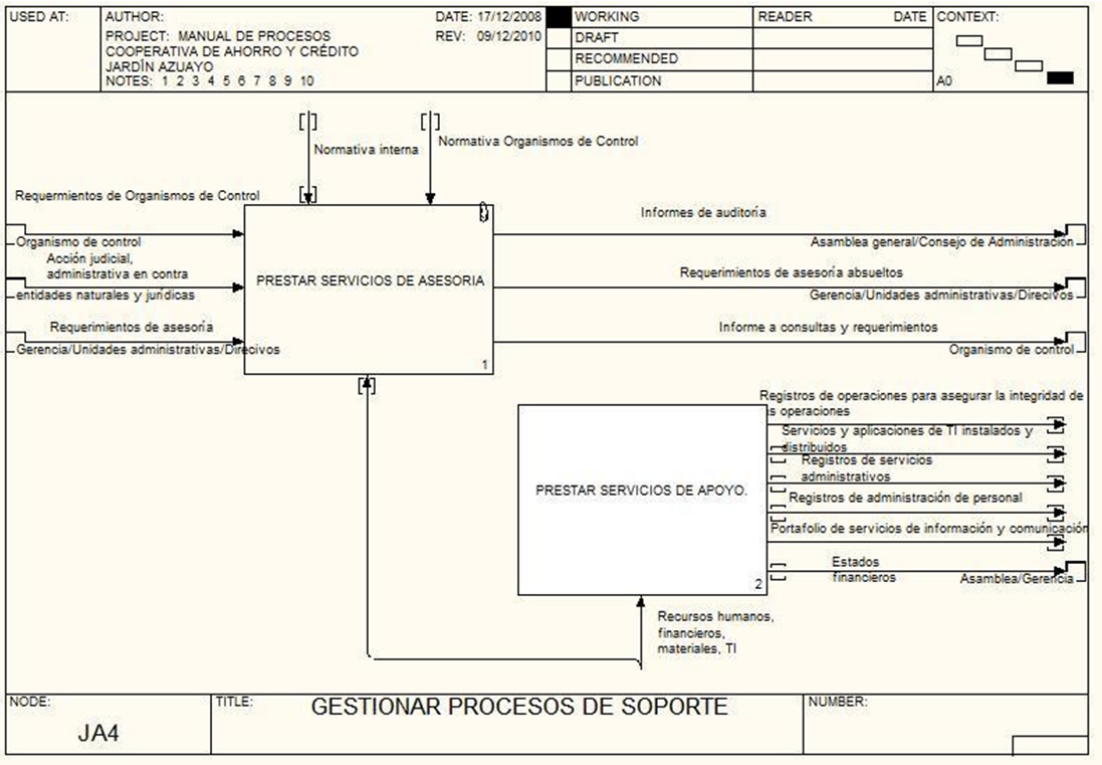
[34] "Normas Square Software Product Quality Requirements and Evaluation". Disponible en Web: <http://bdigital.eafit.edu.co/PROYECTO/P005.14CDP613/anexos.pdf>

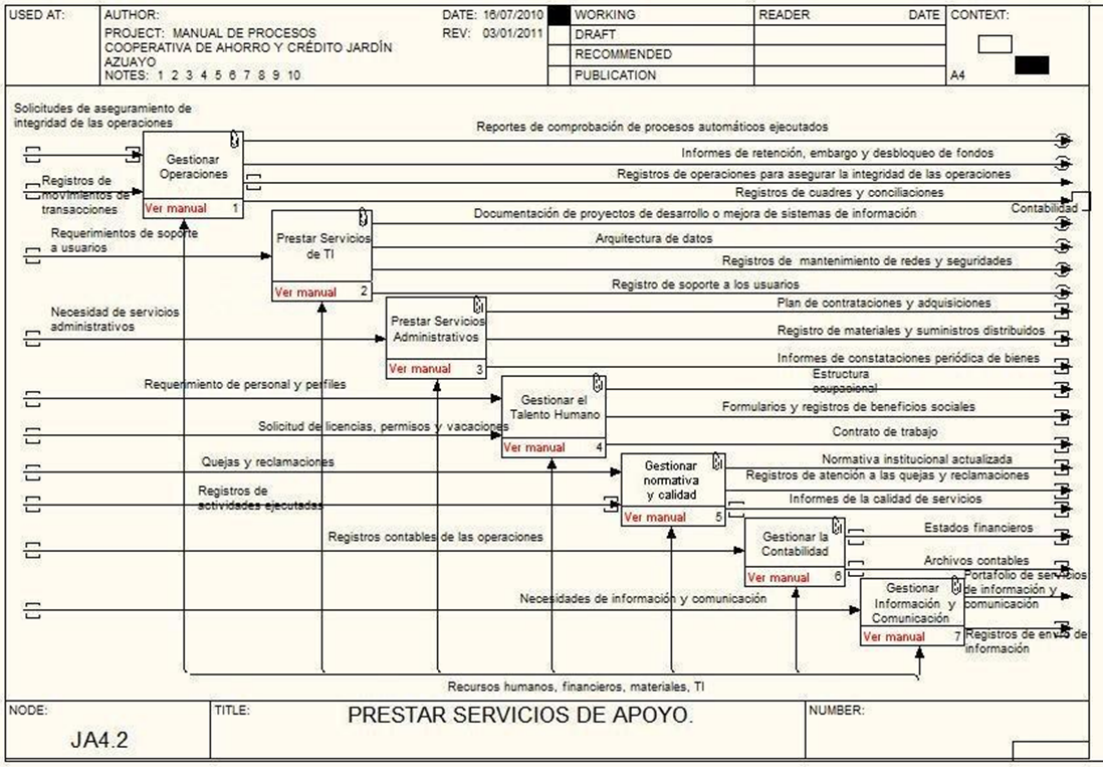
ANEXOS

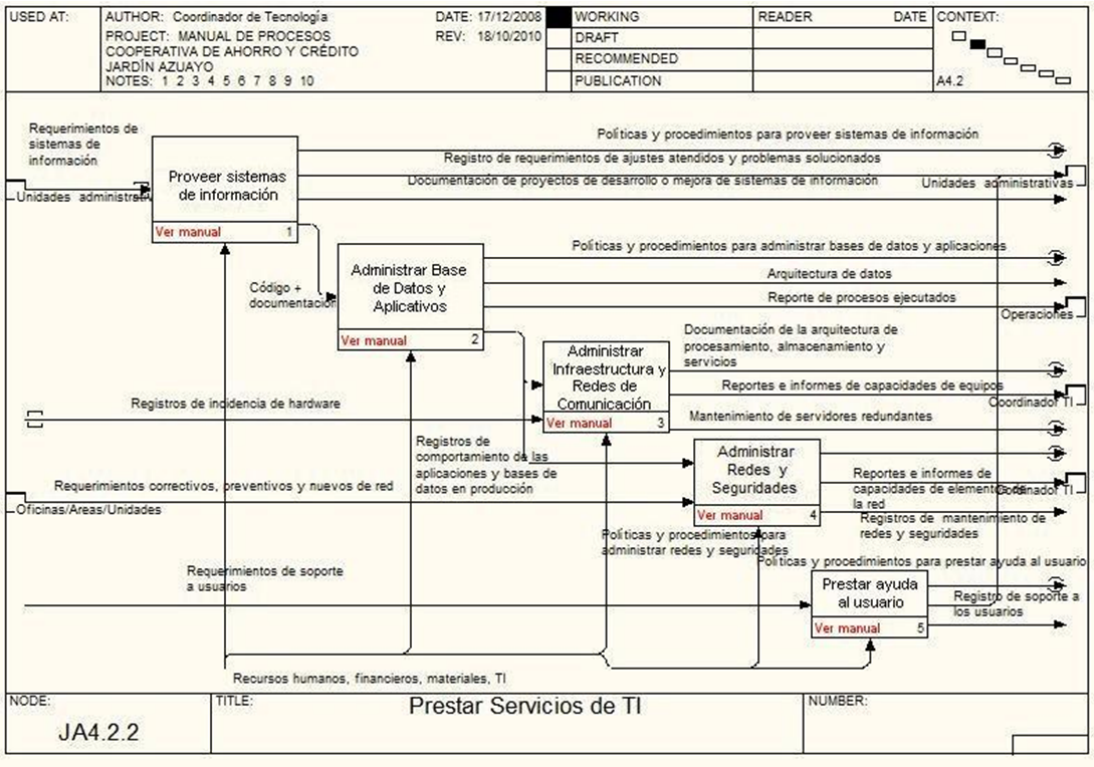
Anexo I. Manual de Procesos de la COAC "JARDÍN AZUAYO"[31]

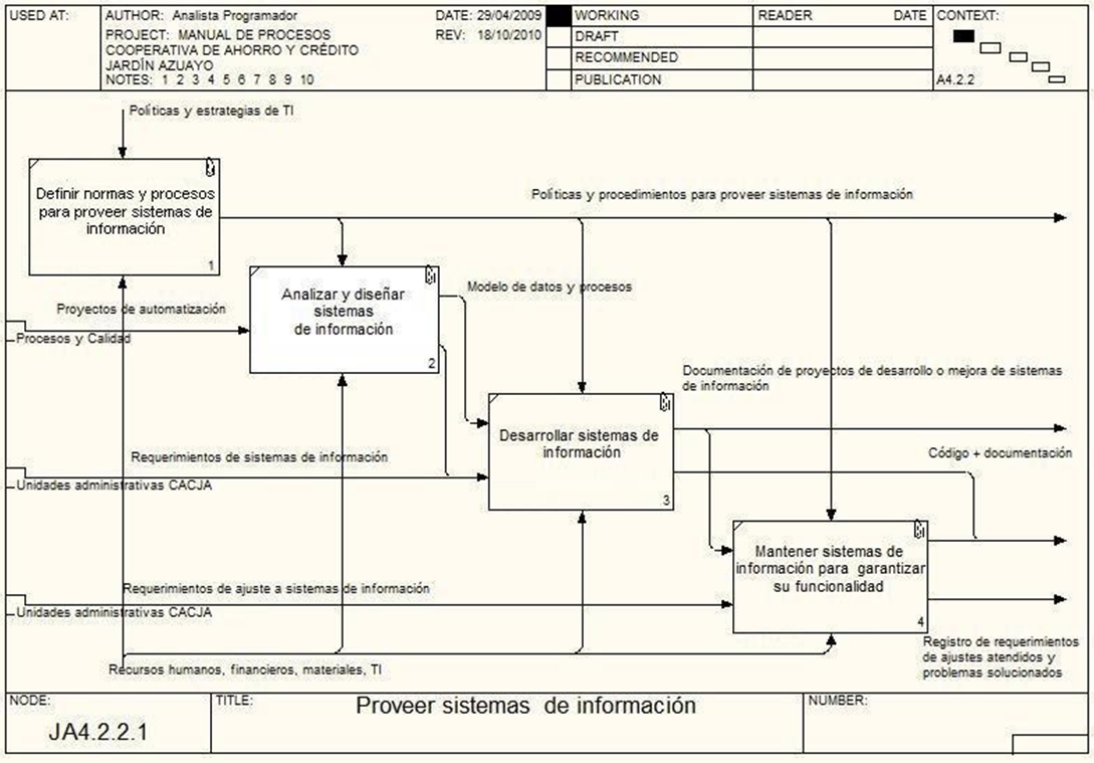





USED AT:
AUTHOR:
PROJECT: MANUAL DE PROCESOS
COOPERATIVA DE AHORRO Y CRÉDITO
JARDÍN AZUAYO
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 17/12/2008
REV: 09/12/2010
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A0
Normativa interna
Normativa Organismos de Control
Requermientos de Organismos de Control
Organismo de control
Acción judicial, administrativa en contra
entidades naturales y jurídicas
Requerimientos de asesoría
Gerencia/Unidades administrativas/Directivos
PRESTAR SERVICIOS DE ASESORIA
1
Informes de auditoría
Asamblea general/Consejo de Administración
Requerimientos de asesoría absueltos
Gerencia/Unidades administrativas/Directivos
Informe a consultas y requerimientos
Organismo de control
Registros de operaciones para asegurar la integridad de las operaciones
Servicios y aplicaciones de TI instalados y distribuidos
Registros de servicios administrativos
Registros de administración de personal
Portafolio de servicios de información y comunicación
Estados financieros
Asamblea/Gerencia
PRESTAR SERVICIOS DE APOYO.
2
Recursos humanos, financieros, materiales, TI
NODE:
JA4
TITLE:
GESTIONAR PROCESOS DE SOPORTE
NUMBER:


USED AT:
AUTHOR:
PROJECT: MANUAL DE PROCESOS COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 18/07/2010
REV: 03/01/2011
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A4
Solicitudes de aseguramiento de integridad de las operaciones
Gestionar Operaciones
Ver manual 1
Registros de movimientos de transacciones
Reportes de comprobación de procesos automáticos ejecutados
Informes de retención, embargo y desbloqueo de fondos
Registros de operaciones para asegurar la integridad de las operaciones
Registros de cuadres y conciliaciones
Contabilidad
Prestar Servicios de TI
Ver manual 2
Requerimientos de soporte a usuarios
Documentación de proyectos de desarrollo o mejora de sistemas de información
Arquitectura de datos
Registros de mantenimiento de redes y seguridades
Registro de soporte a los usuarios
Prestar Servicios Administrativos
Ver manual 3
Necesidad de servicios administrativos
Plan de contrataciones y adquisiciones
Registro de materiales y suministros distribuidos
Informes de constataciones periódica de bienes
Gestionar el Talento Humano
Ver manual 4
Requerimiento de personal y perfiles
Estructura ocupacional
Formularios y registros de beneficios sociales
Solicitud de licencias, permisos y vacaciones
Contrato de trabajo
Gestionar normativa y calidad
Ver manual 5
Quejas y reclamaciones
Normativa institucional actualizada
Registros de atención a las quejas y reclamaciones
Registros de actividades ejecutadas
Informes de la calidad de servicios
Gestionar la Contabilidad
Ver manual 6
Registros contables de las operaciones
Estados financieros
Archivos contables
Gestionar Información y Comunicación
Ver manual 7
Necesidades de información y comunicación
Portafolio de servicios de información y comunicación
Registros de envío de información
Recursos humanos, financieros, materiales, TI
NODE:
JA4.2
TITLE:
PRESTAR SERVICIOS DE APOYO.
NUMBER:


USED AT:
AUTHOR: Coordinador de Tecnología
PROJECT: MANUAL DE PROCESOS COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 17/12/2008
REV: 18/10/2010
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A4.2
Requerimientos de sistemas de información
Unidades administrativas
Proveer sistemas de información
Ver manual 1
Políticas y procedimientos para proveer sistemas de información
Registro de requerimientos de ajustes atendidos y problemas solucionados
Documentación de proyectos de desarrollo o mejora de sistemas de información
Unidades administrativas
Código + documentación
Administrar Base de Datos y Aplicativos
Ver manual 2
Políticas y procedimientos para administrar bases de datos y aplicaciones
Arquitectura de datos
Reporte de procesos ejecutados
Operaciones
Administrar Infraestructura y Redes de Comunicación
Ver manual 3
Documentación de la arquitectura de procesamiento, almacenamiento y servicios
Reportes e informes de capacidades de equipos
Mantenimiento de servidores redundantes
Coordinador TI
Registros de incidencia de hardware
Registros de comportamiento de las aplicaciones y bases de datos en producción
Administrar Redes y Seguridades
Ver manual 4
Reportes e informes de capacidades de elementos de la red
Coordinador TI
Registros de mantenimiento de redes y seguridades
Requerimientos correctivos, preventivos y nuevos de red
Oficinas/Areas/Unidades
Políticas y procedimientos para administrar redes y seguridades
Políticas y procedimientos para prestar ayuda al usuario
Requerimientos de soporte a usuarios
Prestar ayuda al usuario
Ver manual 5
Registro de soporte a los usuarios
Recursos humanos, financieros, materiales, TI
NODE:
JA4.2.2
TITLE:
Prestar Servicios de TI
NUMBER:


USED AT:
AUTHOR: Analista Programador
PROJECT: MANUAL DE PROCESOS COOPERATIVA DE AHORRO Y CRÉDITO JARDÍN AZUAYO
NOTES: 1 2 3 4 5 6 7 8 9 10
DATE: 29/04/2009
REV: 18/10/2010
WORKING
DRAFT
RECOMMENDED
PUBLICATION
READER
DATE
CONTEXT:
A4.2.2
Políticas y estrategias de TI
Definir normas y procesos para proveer sistemas de información
1
Políticas y procedimientos para proveer sistemas de información
Proyectos de automatización
Procesos y Calidad
Analizar y diseñar sistemas de información
2
Modelo de datos y procesos
Documentación de proyectos de desarrollo o mejora de sistemas de información
Desarrollar sistemas de información
3
Requerimientos de sistemas de información
Unidades administrativas CACJA
Código + documentación
Mantener sistemas de información para garantizar su funcionalidad
4
Requerimientos de ajuste a sistemas de información
Unidades administrativas CACJA
Registro de requerimientos de ajustes atendidos y problemas solucionados
Recursos humanos, financieros, materiales, TI
NODE:
JA4.2.2.1
TITLE:
Proveer sistemas de información
NUMBER:

DEFINIR NORMAS Y PROCESOS PARA PROVEER SISTEMAS DE INFORMACIÓN

Nombre	**JA4.2.2.1.1 DEFINIR NORMAS Y PROCESOS PARA PROVEER SISTEMAS DE INFORMACIÓN.**
Descripción	**Definir Normas:** El Coordinador del Departamento de Ingeniería y Desarrollo de Sistemas: 1. Redacta la propuesta de la norma sea ésta políticas, reglamentos, metodologías u otra normativa inherente para Proveer Sistemas de Información, y presenta al Coordinador y personal del área para una primera revisión y ajustes. 2. Realizados los ajustes en el Área, envía a un grupo focal para su revisión. 3. Se reúne con los miembros del grupo focal para recibir las observaciones al documento o recibe mediante correo electrónico. 4. El Coordinador de Ingeniería y Desarrollo de Sistemas actualiza el documento con las observaciones del grupo focal y envía al Coordinador de Área para su socialización ante el Comité de Coordinación General. 5. El Coordinador de Área envía el documento a los miembros del Comité de Coordinación General con 3 días de anticipación a la reunión para su revisión y análisis. 6. El Comité de Coordinación General en la reunión correspondiente realiza las observaciones a la norma planteada y aprueba su envío al Consejo de Administración. 7. El Coordinador de Área envía a prosecretaria para que se incluya en el orden del día del Consejo de Administración para su aprobación. En caso de no aprobarlo el Consejo devuelve al Coordinador de Ingeniería y Desarrollo de Sistemas para su readecuación. 8. En caso de ser aprobada la norma, la prosecretaria del Consejo Remite al Coordinador de Normas y Calidad para su publicación y difusión en la Intranet.

	9. El Coordinador de Ingeniería y Desarrollo de Sistemas realiza las acciones necesarias para implementar la norma aprobada. **Definir Procesos:** 1. El Coordinador de Ingeniería y Desarrollo de Sistemas con el apoyo del departamento de normas y calidad elabora o adecúa los procesos necesarios para el cumplimiento de sus roles y funciones de acuerdo a los formatos establecidos en el departamento de normas y calidad. 2. Socializa los procesos levantados ante su Coordinador de Área y sus compañeros de departamento, recoge las sugerencias, realiza los ajustes necesarios y envía al Coordinador de Normas y Calidad. 3. El Coordinador de Normas y Calidad revisa la coherencia del documento, la eficiencia de los procesos, asigna el número de solicitud y envía al Coordinador de Ingeniería y Desarrollo de Sistemas para que éste a través de su Coordinador de Área envíe al Comité de Coordinación General para su aprobación. 4. El Coordinador de Área solicita incluir la aprobación de los procesos en la agenda de reunión del CCG y les envía el documento a sus integrantes con 3 días de anticipación a la reunión para su revisión y posterior aprobación en la reunión. En caso de no aprobarlo el Comité devuelve al Coordinador de Ingeniería y Desarrollo de Sistemas para su readecuación. 5. En caso de ser aprobados los procesos, el Coordinador de Ingeniería y Desarrollo de Sistemas informa de su aprobación y envía el documento al Coordinador del departamento de Normas y Calidad. 6. El Coordinador de Normas y Calidad incorpora los cambios aprobados al manual correspondiente, publica en la intranet, informa a los usuarios para su aplicación y controla las versiones del manual.

Nombre	JA4.2.2.1.2 ANALIZAR Y DISEÑAR SISTEMAS DE INFORMACIÓN.
Descripción	**1. Definir herramientas de desarrollo a utilizarse.** El Coordinador de TI, con la aprobación del Comité de Informática, define las herramientas de desarrollo a utilizarse, las cuales se hallan basadas en 2 grupos de desarrollo actuales: • **Desarrollo FORMS:** Oracle Developer Suite 10g R2. Oracle PL/SQL Developer 6. • **Desarrollo JAVA:** Oracle JDeveloper 10g. Oracle SQL Developer. **2. Analizar y diseñar sistemas de información.** El Analista Programador, con la finalidad de detectar ambigüedades y contradicciones entre las diferentes ordenes de pedido, gestionar la integridad de los procesos y detectar conjuntos de acciones que son repetitivas, para poder considerarlas como una unidad de programación, plasma la parte de captura, análisis y diseño de los sistemas de información en los siguientes documentos: • Diagramas de Casos de Uso: La finalidad de este documento es tener una visión global del aplicativo, en el cual se detalle los actores y las funcionalidades prestadas. • Documento de requerimiento: La finalidad de este documento es tener un documento en el cual se establezcan claramente los requerimientos de las áreas, como segmentos de este documento se encuentran: **1.** Descripción: Presentará una breve descripción global del proceso. **2.** Pre-condiciones: Establece las condiciones previas que deberán existir. **3.** Flujos Básicos: Detalla los flujos de los procesos, como también indica los actores que intervendrán. Esta sección describirá completamente el comportamiento esperado del

	sistema, que se utilizara posteriormente para las fases de desarrollo y pruebas. **4.** Flujos Alternos: Detalla los procesos que se deberán dar en caso de presentarse algún inconveniente en el flujo básico. **5.** Pos-condiciones: Establece las condiciones posteriores que deberán ser consideradas. **3. Asignar para el desarrollo.** El Coordinador de TI asigna recursos en base al tiempo proyectado de desarrollo, prioridad e impacto que causa en la institución; para las asignaciones.

DESARROLLAR SISTEMAS DE INFORMACIÓN

Nombre	**JA4.2.2.1.3 DESARROLLAR SISTEMAS DE INFORMACIÓN.**
Descripción	**1. Define estándares de programación y Best Practice.** Para poder pasar el aplicativo hacia la etapa de pruebas, conjuntamente con la orden de pruebas, verifica que los scripts realicen los cambios de estructura en la base de datos, como también los ingresos, actualizaciones y eliminaciones de registros en el caso de ser necesarios. **2. Realiza pruebas.** Verifica que los aplicativos desarrollados cumplan con lo expuesto en los documentos de requerimientos. Este proceso dará inicio una vez que el Programador haya pasado una orden de Pruebas. **3. Entrega a producción.** Una vez que el aplicativo haya superado las pruebas correspondientes, implementa el sistema desarrollado en el ambiente de producción.

MANTENER SISTEMAS DE INFORMACIÓN PARA GARANTIZAR SU FUNCIONALIDAD

Nombre	**JA4.2.2.1.4 MANTENER SISTEMAS DE INFORMACIÓN PARA GARANTIZAR SU FUNCIONALIDAD.**

	1. En el caso de necesitar un cambio en los procesos para mejorar su funcionalidad, el Analista Programador da inicio nuevamente con la actualización del documento de requerimientos. **2.** En el caso de encontrarse un problema en la parte operativa debido a un problema de programación el Analista Programador documenta dicho problema y realiza la(s) corrección(es) necesaria(s).

Anexo II. Documentación generada en el Proceso de Desarrollo (AS-IS) de la COAC "JARDÍN AZUAYO"

Orden de Pedido

ORDEN DE PEDIDO No. 43

Fecha:	27-09-2011	
Proyecto	Nombre:	No.:
Área Usuaria:	Desarrollo Cooperativo – Talento Humano.	
Objetivo: (Propósito del sistema o mantenimiento requerido)	• Disminuir el tiempo que toma el realizar las operaciones necesarias para el manejo de las providencias judiciales.	
Problema a resolver:	Actualmente existe un retraso en las operaciones que se deben realizar para el manejo de las providencias judiciales lo cual puede producir una multa a la Cooperativa.	
Justificación: (Razón de ser, porque se necesita?)	Existe un retraso en la subida de la información correspondiente a providencias judiciales por parte del departamento de operaciones por lo cual se solicita la implementación de una herramienta que permita agilitar esta actividad, para evitar sanciones futuras de los organismos de control.	
Definición de implicaciones: Contables, riesgos, cumplimiento, etc.	N/A.	
Alcance:	Modificación en el sistema	
Descripción: (Breve descripción de lo que se necesita)	• Se requiere que los datos del archivo de Excel recibido en el departamento de operaciones referente a providencias judiciales sea cargado a la pantalla de "Personas inhabilitadas". • Validar los datos cargados con la base de datos de socios de la Cooperativa y verificar que personas de este listado son o no son socias. • De las personas que son socias se debe obtener un listado de todas las cuentas de ahorros a la vista que posee, los certificados de depósito y los ahorros Alcancía Segura que tengan.	

Firmas

Solicitante	Normas y Calidad	Ingeniería de Software
Desarrollo	Producción	

Red de Proyecto

https://mail.jardinazuayo.fin.ec/service/home/~/Op%20No.43%20Providencias%20Judiciales.pdf?auth=co&loc=es&id=4175&part=2

ID	WBS	Name	Likely start date	Duration	Low Risk Duration
11	1	**Ingeniería del Software**	NA	**3 days**	**0 days**
21	1.1	Revisión de requerimiento por Ingeniería del	Tue 27/09/11	0.5 days	1 day
31	2	Preparación de documento técnico	Tue 27/09/11	2 days	4 days
41	3	Socialización y aprobación de documento téc	Thu 29/09/11	0.5 days	1 day
52		**Desarrollo de Software**	NA	**3 days**	**0 days**
62	1	Desarrollo	Fri 30/09/11	2 days	4 days
72	2	Pruebas preliminares	Tue 04/10/11	1 day	2 days
83		Puesta en pruebas	Wed 05/10/11	1 day	2 days
94		Pruebas con usuarios	Thu 06/10/11	1 day	3 days
105		Enmiendas y ajustes	Fri 07/10/11	1 day	2 days
116		Pruebas con usuarios	Mon 10/10/11	1 day	2 days
127		Puesta en producción	Tue 11/10/11	1 day	5 days
139		Project Buffer	NA	7.5 days?	15 days
148		Fin	NA	0 days	0 days

Documento de Ingeniería

ORDEN Nº 43

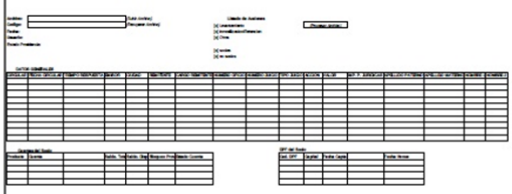
MODULO: PROVIDENCIAS JUDICIALES
FECHA: 13/10/2011
VERSION: 1
RESPONSABLE: ROMEL CANDO
DESARROLLADOR: MARIA EUGENIA MONTERO

OBJETOS QUE INTERVIENEN

Tabla/Procedimiento/Funcion/Otros	Nombre de objeto
Tabla	CLIENTES
Tabla	PERSONAS_INHABILITADAS
Tabla	CUENTAS
Tabla	PLAZOS_FIJOS
Tabla	CUENTAS_BLOQUEADAS
Tabla	CUENTAS_INACTIVAS
Tabla	CABECERA_PROVIDENCIAS
Tabla	DETALLE_PROVIDENCIAS
Paquete	FISJA.PK_PERSONAS_INHABILITADAS
Paquete	FISJA.PK_CUENTAS.P_OBTIENE_BLOQUEO_PROVIDENCIAS
Paquete	FISJA.PK_CAMARA_COMPENSACION.P_VALIDA_CUENTA
Paquete	PK_PROVIDENCIAS.P_REVISA_REGISTRO_PERSONA

PANTALLA

NOMBRE: PERSONAS_INHABILITADAS
OPCION MENU: MENU DE OPERACIONES - PROVIDENCIAS
ÁREAS QUE ACCEDEN: OPERACIONES
PERMISOS: OPERACIONES(INSERT-DELETE-UPDATE)



LISTAS DE VALORES

CAMPO DE LA PANTALLA:
codigo_providencia

CAMPOS QUE SE MUESTRAN:
código_providencia
fecha_carga
nombre del archivo
usuario que actualizo
estado_providencia

ORDEN:
codigo descendente

PROCEDIMIENTOS GENERALES

VALIDACIONES DEL REGISTRO

El archivo excel la columna ACCION podrá contener los siguientes valores:

RETENCION	RET
INMOVILIZACION	INM
LEVANTAMIENTO	LEV
CERTIFICACION	CER
INFORMACION	INF
NO INDICA	NO

si no esta registrado presentar el mensaje: "El tipo de acción {tipo acción archivo} no es valido"

Procedimiento de recuperación de información de los socios.

si en el registro seleccionado en el campo código de socio esta vació el proceso termina
recuperar los datos de las cuentas del socio
producto, numero cuenta, saldo total, saldo disponible,
saldo bloqueado por providencias: utilizando el procedimiento
pk_cuentas.p_obtiene_bloqueo_providencias
estado cuenta: utilizando pk_camara_compensacion.p_valida_cuenta
recuperar los datos de los DPF(Deposito a plazo fijo) del socio :
codigo plazo, capital, fecha captacion, fecha vence.
Recuperar el estado del registro en la tabla de personas inhabilitadas utilizando
pk_providencias.p_revisa_registro_persona

Procedimiento dentro de la forma para RECUPERAR_ARCHIVO
Valida que exista el archivo definido en el campo Archivo
Si no Existe
Presenta el mensaje "El Archivo:{Archivo} no existe"
El proceso termina
Si existe verificar que el archivo no haya sido subido antes.
Blanquear los campos de la pantalla.
Leer y validar los registros del archivo y los colocar en la pantalla.

Los campos de equivalencia son los siguientes:

FORMA ACTUAL	ARCHIVO
INSTITUCION	EMISOR
TIP. PROC	TIPO DE JUICIO
NO. PROC	NUMERO JUICIO
NO. OFIC	OFICIO
CIUDAD	CIUDAD
TIPO	NATURAL O JURIDICA
CEDULA/RUC	NUMERO DOCUMENTO
APELLIDO PATERNO	APELLIDO_PATERNO
APELLIDO MATERNO	APELLIDO_MATERNO
NOMBRE	NOMBRE
OSERVACIONES	CIRCULAR + SYSDATE
VALOR RETENCION	VALOR
ESTADO	SI NO TIENE DINERO PENDIENTE/BLOQUEADO
ELIMINADO	CUANDO YA SE LEVANTA
RAZON SOCIAL	SI ES JURIDICA SOLO SI ES SOCIO RECUPERA DE CLIENTES

FUNCIONALIDAD.

CAMPO/BOTON:
Subir Archivo.

CASO DE USO :
Al presionar el botón presentara la ventana para la selección del archivo.
Si selecciona un archivo en esta ventana colocara el nombre del archivo en el campo Archivo:
Llama al procedimiento de recuperación del archivo.
Dentro de este proceso se debe de validar cada registro,
si existe alguna validación o error se debe de presentar el mensaje "Error en la linea {numero_linea} Descripción: {detalle del error}"

Luego de cargado el registro, se debe de validar si la persona a la que se hace referencia es socio:
buscar en la base de clientes si el numero de identificación esta registrado
si esta registrado colocar en el campo código de socio
si no se encontró un registro
buscar en la base de clientes utilizando el apellido paterno,apellido materno y los nombres conjuntamente del campo del archivo ("apellido paterno,apellido materno, nombre 1, nombre 2")
si esta registrado colocar en el campo código de socio
si no se encontró un registro
buscar en la base de clientes utilizando la razón social conjuntamente, el campo del archivo ("Implicados personas jurídicas")
si esta registrado colocar en el campo código de socio

Llama al proceso de grabar/Ctrl+s

CAMPO:
casilla socios.

CASO DE USO :
si la casilla esta activa, la grilla presentara los registros que no tienen vació el código del socio
si la casilla no esta activa, la grilla no presentara estos registros.

CAMPO:
casilla no socios.

CASO DE USO :
si la casilla esta activa, la grilla presentara los registros que tienen vació el código del socio
si la casilla no esta activa, la grilla no presentara estos registros

CAMPO:
casilla levantamiento

CASO DE USO :
si la casilla esta activa, la grilla presentara los registros que tienen en la acción definida como levantamiento
si la casilla no esta activa, la grilla no presentara estos registros

CAMPO:
casilla Inmovilizacion/Retencion

CASO DE USO :
si la casilla esta activa, la grilla presentara los registros que tienen en la acción definida como inmovilización o retención
si la casilla no esta activa, la grilla no presentara estos registros

CAMPO:
casilla Otros

CASO DE USO :
si la casilla esta activa, la grilla presentara los registros que tienen en las acciones definidas como certificación, información, no indica.
si la casilla no esta activa, la grilla no presentara estos registros

CAMPO:
estado_providencia

CASO DE USO :
muestra información si el estado esta en Pendiente o procesada

CAMPO:
Estado

CASO DE USO :
Si el estado esta en SI al recuperar el registro indica que ya esta bloqueado por providencias anteriores.
Si el estado esta en NO indica que no esta bloqueado.

CAMPO/BOTON:
No bloquear por

CASO DE USO :
Si campo is null no existen observaciones para ese registro
Si campo is not null existen observaciones para ese registro

BOTON:
Grabar/ctrl+s

CASO DE USO :
si se presiona ctrl+s o botón grabar
si el campo codigo_providencia esta vació(esto se va a dar únicamente el rato de cargar el archivo)
generar un nuevo codigo, (el máximo del código, registrado en la tabla) + 1
llamar a los **DATOS INICIALES.**
validar que estado este en pendiente

Si no esta en estado pendiente, presentar el mensaje "El archivo seleccionado esta en estado {estado}, solo puede actualizar archivos en estado pendiente", y el proceso termina

Grabar los datos de la pantalla en la tablas de cabecera y detalle definidas.
ir al primer registro
Llamar al Procedimiento de recuperación de información de los socios

BOTON:
Procesar Archivo

CASO DE USO :
Al presionar el botón
validar que es estado este en pendiente
Si no esta en estado pendiente, presentar el mensaje "El archivo seleccionado esta en estado {estado}, solo puede procesar archivos en estado pendiente", y el proceso termina

procesar los registros que tengan codigo socio y acción levantamiento, Inmovilizacion o retencion y que ademas en el campo "no procesar por" esta vació.

Llamar al **pk_providencias.p_revisa_registro_persona**

si encontró un registro actualizarlo con la nueva información de la pantalla si no encontró lo ingresa
si al momento de procesar existe algún error en el registro presentar el mensaje;
"Error al procesar el registro {numero de registro} Error:{detalle del error}" y ubicarse sobre el registro del problema

CAMPO/BOTON:
Recuperar Archivo.

CASO DE USO :
al presionar el botón se presentara la lista de valores **PROVIDENCIAS JUDICIALES**
al seleccionar una opción de la lista
cargara los registros correspondientes
va al primer registro
Llamar al Procedimiento de recuperación de información de los socios

no permite realizar modificaciones si estado esta en procesado

CASO DE USO EN LA PANTALLA:

Selección de registro en el detalle:
cada que se cambie de registro se deberá de llamar al Procedimiento de recuperación de información de los socios.

DATOS INICIALES:
Estado_providencia = Pendiente
Socios = Casilla Activa
No Socios = Casilla Activa
Código de usuario = logeado
Fecha = sysdate
Levantamiento = Casilla Activa
Inmovilización/retención = Casilla Activa
Otros = Casilla Activa

OBSERVACIONES:
Este desarrollo podrá ser utilizado solo en Windows.
El archivo se puede subir de cualquier path.
Todos los campos que hacen referencia al archivo son modificables. No se puede modificar los datos de las cuentas y DPF son únicamente informativos.

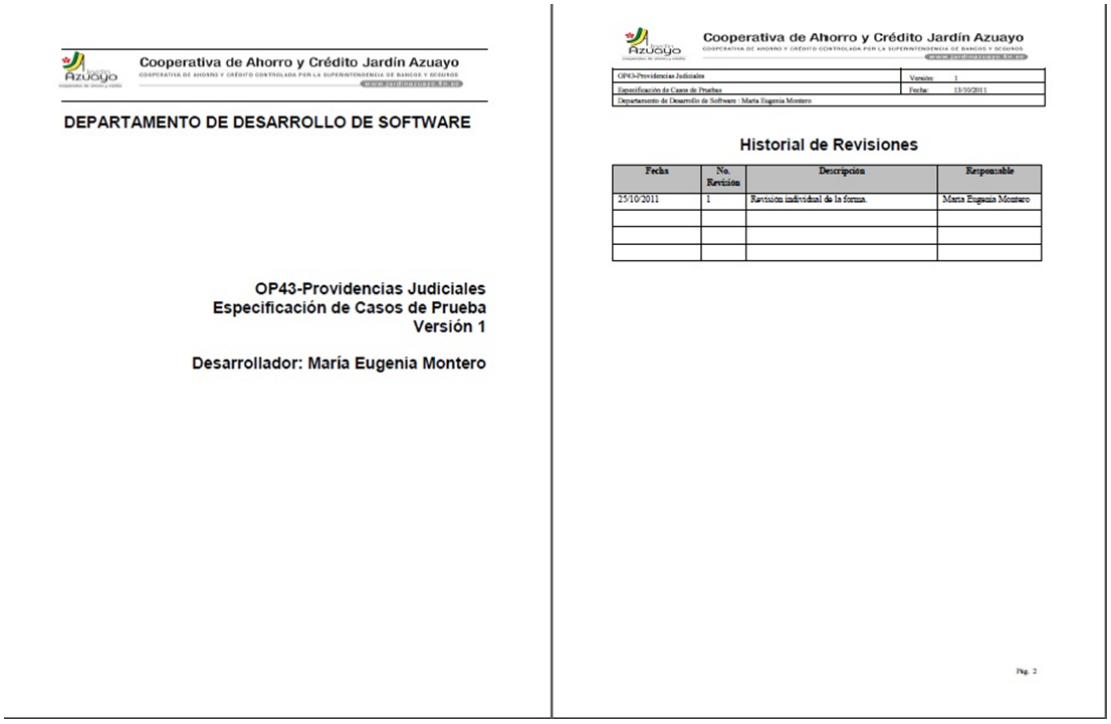
La forma se llama PERSONAS_INHABILITADAS en la dirección:
http://172.17.0.87:8890/forms/frmservlet?form=C:/10Gr2APPL/FORMS/CA_MENUCONSEP.fmx
En el servidor : 172.17.0.87

Firmas

Solicitante	Normas y Calidad	Ingeniería de Software

Desarrollo	Producción

Documento de Desarrollo



Cooperativa de Ahorro y Crédito Jardín Azuayo

DEPARTAMENTO DE DESARROLLO DE SOFTWARE

OP43-Providencias Judiciales
Especificación de Casos de Prueba
Versión 1

Desarrollador: María Eugenia Montero

Cooperativa de Ahorro y Crédito Jardín Azuayo

OP43-Providencias Judiciales	Versión: 1
Especificación de Casos de Prueba	Fecha: 13/10/2011
Departamento de Desarrollo de Software : María Eugenia Montero	

Historial de Revisiones

Fecha	No. Revición	Descripcion	Responsable
25/10/2011	1	Revision individual de la forma.	Maria Eugenia Montero

Pag. 2

Tabla de Contenidos

1. Descripción

Este documento cubre el conjunto de pruebas realizadas sobre la funcionalidad de la Pantalla de Providencias Judiciales.

Precondiciones:

- Ingresar al Sistema Caja-Credito/Mant Operaciones/Opcion Providencias.
- El sistema sera probado unicamente bajo el Sistema Operativo Windows

2. Datos Iniciales

2.1 Descripcion

Visualizacion de datos al momento de ingresar en la pantalla.

2.2 Resultado esperado

- El campo "Archivo" debe estar vacio
- El campo "Codigo" debe estar vacio
- El campo "Fecha" debe contener la fecha del dia
- El campo "Usuario" debe contener el codigo del usuario que se logea
- El campo "Estado Providencia" debe estar en estado "Pendiente"
- La casilla "Socios" debe estar activa
- La casilla "No Socios" debe estar activa
- La casilla "Levantamiento" debe estar activa
- La casilla "Inmovilizacion/Retencion" debe estar activa.
- La casilla "Otros" debe estar activa.

2.3 Evaluación de la Prueba

1. Los resultados son mostrados satisfactoriamente.

3. <Subir Archivo>

3.1 Descripcion

Activar un campo a traves del cual se pueda seleccionar el archivo en formato Excel desde cualquier ubicacion del equipo y cargar los datos del archivo a la grilla principal.

3.2 Condiciones de ejecución

1. Logearse con un usuario que tenga el rol de Operaciones.
2. Disponer del archivo en Excel (.xls) con el formato preestablecido en una ubicacion cualquiera del equipo.
3. Los datos deben estar en una hoja del libro etiquetada con el nombre "HOJA".
4. Sin columnas combinadas
5. Una sola fila para los titulos.

3.3 Entrada

- Ubicarse en el campo con la etiqueta "Archivo".
- Dar click en el boton [...]
- Se presentara un cuadro de dialogo en el cual podra seleccionar el archivo tipo .xls desde cualquier ubicacion del equipo.
- Dar click en el botnn "Abrir" del cuadro de dialogo.
- El cuadro de dialogo desaparecera.
- El campo "Archivo" se llenara con la informacion del path y el nombre del archivo seleccionado.
- Dar click en el botnn "Subir Archivo"
- Se cargaran los datos del archivo .xls a la grilla principal de la pantalla.

3.4 Excepciones

- Si la hoja no esta etiquetada con el nombre "HOJA" presenta el mensaje "Error al seleccionar hoja de excel.. revise si el nombre de la hoja es "HOJA".
- Si el archivo esta vacio se presenta el mensaje "Archivo (Archivo) no contiene registros para cargar".
- Si el archivo contiene errores en el formato se presenta el mensaje "Error en la linea (numero_linea). Descripcion: (Detalle del Error).
- Si el archivo con fecha del dia esta grabado en la base de datos presenta el mensaje "Existen datos cargados con la fecha (fecha de carga).
- Si la columna de "ACCION" del archivo de Excel no contiene los siguientes valores:
  - Retencion - RET
  - Inmovilizacion - INM
  - Levantamiento - LEV
  - Certificacion - CER
  - Information - INF
  - No Indica - NO

  Se presenta el mensaje "Error al recuperar archivo en la columna ACCION fila (numero_fila). El Tipo de Accion (Tipo accion) no es valido".

3.5 Resultado esperado

- La grilla principal debe cargarse con los mismos datos de la hoja seleccionada.
- El campo "Codigo Socio" debe estar llenado con el codigo del socio si este efectivamente es socio de la cooperativa.

- Las grilla "Cuentas de Socio" deben contener las cuentas que posee en caso de que el registro de la grilla principal contenga el codigo de un socio.
- En la grilla "Cuentas de Socio" deben mostrarse los siguientes campos:
  - Cuenta del Socio
  - Producto
  - Numero de cuenta
  - Saldo Total
  - Saldo Disponible
  - Saldo Bloqueado por providencias
  - Estado de Cuenta.
- La grilla "DPF del Socio" debe contener los plazos fijos que posee en caso de que el registro de la grilla principal contenga el codigo de un socio.
- En la grilla "DPF del Socio" deben mostrarse los siguientes campos:
  - Codigo de Plazo
  - Capital
  - Fecha de Captacion
  - Fecha de Vencimiento

3.6 Evaluación de la Prueba

1.Los datos de las cuentas y de los plazos fijos se muestran una vez que se graban los datos

4. <Casilla Socios>

4.1 Descripción

Casillero que sirve para filtrar la información de la grilla principal mostrando unicamente los registros que correspondan a socios.

4.2 Condiciones de ejecución

1.Debe recuperarse una providencia ya grabada anteriormente.

4.3 Entrada

- Seleccionar una providencia grabada
- Activar el casillero "Socios"

4.4 Excepciones

- Si la casilla no esta activa no presenta estos registros

Cooperativa de Ahorro y Crédito Jardín Azuayo
COOPERATIVA DE AHORRO Y CRÉDITO CONTROLADA POR LA SUPERINTENDENCIA DE BANCOS Y SEGUROS

CPO3-Providencias Judiciales	Versión: 1
Especificación de Casos de Pruebas	Fecha: 13/10/2011
Departamento de Desarrollo de Software : Maria Eugenia Montero	

4.5 Resultado esperado

- Al activar la casilla de "Socios" se presenta en la grilla principal unicamente los datos de los socios.

4.6 Evaluación de la Prueba

1.Resultado correcto.

5. <Casilla No Socios>

5.1 Descripción

Casillero que sirve para filtrar la información de la grilla principal mostrando unicamente los registros que correspondan a no socios.

5.2 Condiciones de ejecución

1.Debe recuperarse una providencia ya grabada anteriormente.

5.3 Entrada

- Seleccionar una providencia grabada
- Activar el casillero "No Socios"

5.4 Excepciones

- Si la casilla no esta activa no presenta estos registros

5.5 Resultado esperado

- Al activar la casilla de "No Socios" se presenta en la grilla principal unicamente los datos de los no socios.

5.6 Evaluación de la Prueba

1. Resultado esperado correcto.

6. <Casilla Levantamiento>

6.1 Descripción

Casillero que sirve para filtrar la información de la grilla principal mostrando unicamente los registros que correspondan a la acción "Levantamiento".

6.2 Condiciones de ejecución

1.Debe recuperarse una providencia ya grabada anteriormente.

6.3 Entrada

- Seleccionar una providencia grabada
- Activar el casillero "Levantamiento"

6.4 Excepciones

- Si la casilla no esta activa no presenta estos registros

6.5 Resultado esperado

- Al activar la casilla de "Levantamiento" se presenta en la grilla principal unicamente los datos de los registros cuya acción sea "Levantamiento".

6.6 Evaluación de la Prueba

1. Resultado esperado correcto.

7. <Casilla Inmovilización/Retención>

7.1 Descripción

Casillero que sirve para filtrar la información de la grilla principal mostrando unicamente los registros que correspondan a la acción "Inmovilización y Retención".

7.2 Condiciones de ejecución

1.Debe recuperarse una providencia ya grabada anteriormente.

7.3 Entrada.

- Seleccionar una providencia grabada

- Activar el casillero "Inmovilización/Retención"

7.4 Excepciones

- Si la casilla no esta activa no presenta estos registros

7.5 Resultado esperado

- Al activar la casilla de "Inmovilización/Retención" se presenta en la grilla principal unicamente los datos de los registros cuya acción sea "Inmovilización/Retención"

7.6 Evaluación de la Prueba

1. Resultado esperado correcto.

8. <Casilla Otros>

8.1 Descripción

Casillero que sirve para filtrar la información de la grilla principal mostrando unicamente los registros que correspondan a las acciones "Certificación, Información, No indica".

8.2 Condiciones de ejecución

1.Debe recuperarse una providencia ya grabada anteriormente.

8.3 Entrada

- Seleccionar una providencia grabada
- Activar el casillero "Otros"

8.4 Excepciones

- Si la casilla no esta activa no presenta estos registros

8.5 Resultado esperado

- Al activar la casilla de "Otros" se presenta en la grilla principal unicamente los datos de los registros cuya acción sea *"Certificación, Información, No indica"*.

8.6 Evaluación de la Prueba

1. Resultado esperado correcto.

9. <Validaciones de campos de grilla>

9.1 Descripción

Los campos de la grilla "Datos Generales" podran ser modificados.

9.2 Condiciones de ejecución

1.Debe existir datos en la grilla cargados o recuperados.

9.3 Entrada

- Modificar los campos de la grilla

9.4 Excepciones

- Si la fecha de la circular excede de dos meses atras de la fecha actual se presenta el mensaje "La fecha de la circular debe estar entre la [fecha_actual – 2 mes) y fecha_actual] "
- Si la fecha de la circular es mayor a la fecha actual se presenta el mensaje "La fecha de la circular no puede ser mayor a la fecha actual".

9.5 Resultado esperado

- Modificar los campos de la grilla tomando en cuenta las excepciones indicadas anteriormente.

9.6 Evaluación de la Prueba

1.Resultado esperado satisfactorio

10. <Procesar Archivo>

10.1 Descripción

Boton que actualizara el estado de la providencia

10.2 Condiciones de ejecución

1. La providencia debe estar en estado "Pendiente"

10.3 Entrada

- Dar click en el boton "Procesar Archivo".

10.4 Excepciones

- Si el estado de la providencia no esta en "Pendiente" presentar el mensaje "El archivo seleccionado esta en estado (estado), solo puede procesar archivos en estado Pendiente".
- Si el codigo de providencia esta vacio se debe presentar el siguiente mensaje 'La providencia no se encuentra grabada.. Por Favor grabe los datos antes de procesarlos...'.
- Si existe algun error al momento de procesar el archivo debe presentar el mensaje "Error al Procesar el registro (numero de registro), Error: (Detalle del Error).

10.5 Resultado esperado

- Los registros que se deben procesar son los que contemplan las siguientes consideraciones
  - Pertenecan a socios
  - Accion este en Levantamiento,Inmovilizacion o Retencion
  - El campo "No procesar por" este vacio.
- Al emitir un mensaje de error el cursor debe posicionarse en el registro que origino el mismo.

10.6 Evaluación de la Prueba

1. Resultados esperados satisfactorios

11. <Recuperar Providencia Judicial>

11.1 Descripción

Se puede recuperar un archivo de providencia grabado anteriormente.

11.2 Condiciones de ejecución

1.Debe existir grabado anteriormente un archivo de providencia.

11.3 Entrada

- Ubicarse en el campo "Codigo"
- Presionar CTRL+L
- Se listan las providencias judiciales grabadas
- Se selecciona una providencia presionando el boton "Aceptar"

11.4 Resultado esperado

- Al presionar CTRL+L los campos que deben mostrarse en ORDEN DESCENDENTE en la lista de valores son:
  - Codigo de Providencia
  - Fecha de Carga

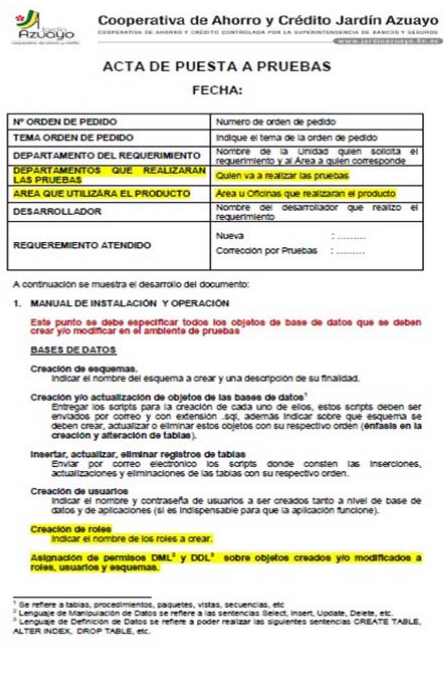
- Nombre del Archivo
- Usuario que actualizo
- Estado de la providencia

- La grilla principal debe cargarse con los datos grabados en la base.
- El campo "Codigo Socio" debe estar llenado con el codigo del socio si este efectivamente es socio de la cooperativa.
- Las grilla "Cuentas de Socio" deben contener las cuentas que posee en caso de que el registro de la grilla principal contenga el código de un socio.
- En la grilla "Cuentas de Socio" deben mostrarse los siguientes campos:
  - Cuenta del Socio
  - Producto
  - Número de cuenta
  - Saldo Total
  - Saldo Disponible
  - Saldo Bloqueado por providencias
  - Estado de Cuenta.
- La grilla "DPF del Socio" debe contener los plazos fijos que posee en caso de que el registro de la grilla principal contenga el código de un socio.
- En la grilla "DPF del Socio" deben mostrarse los siguientes campos:
  - Codigo de Plazo
  - Capital
  - Fecha de Captacion
  - Fecha de Vencimiento

11.5 Evaluación de la Prueba

1. Resultados esperados satisfactorios

Acta Puesta en Pruebas



Cooperativa de Ahorro y Crédito Jardín Azuayo

COOPERATIVA DE AHORRO Y CRÉDITO CONTROLADA POR LA SUPERINTENDENCIA DE BANCOS Y SEGUROS

ACTA DE PUESTA A PRUEBAS

FECHA:

Nº ORDEN DE PEDIDO	Numero de orden de pedido
TEMA ORDEN DE PEDIDO	Indique el tema de la orden de pedido
DEPARTAMENTO DEL REQUERIMIENTO	Nombre de la Unidad quien solicita el requerimiento y al Área a quien corresponde
DEPARTAMENTOS QUE REALIZARAN LAS PRUEBAS	Quien va a realizar las pruebas
AREA QUE UTILIZARA EL PRODUCTO	Area u Oficinas que realizaran el producto
DESARROLLADOR	Nombre del desarrollador que realizo el requerimiento
REQUEREMIENTO ATENDIDO	Nueva : ..... Corrección por Pruebas : .....

A continuación se muestra el desarrollo del documento:

1. **MANUAL DE INSTALACIÓN Y OPERACIÓN**

Este punto se debe especificar todos los objetos de base de datos que se deben crear y/o modificar en el ambiente de pruebas

BASES DE DATOS

Creación de esquemas.
Indicar el nombre del esquema a crear y una descripción de su finalidad.

Creación y/o actualización de objetos de las bases de datos[1]
Entregar los scripts para la creación de cada uno de ellos, estos scripts deben ser enviados por correo y con extensión .sql, además indicar sobre que esquema se deben crear, actualizar o eliminar estos objetos con su respectivo orden (**énfasis en la creación y alteración de tablas**).

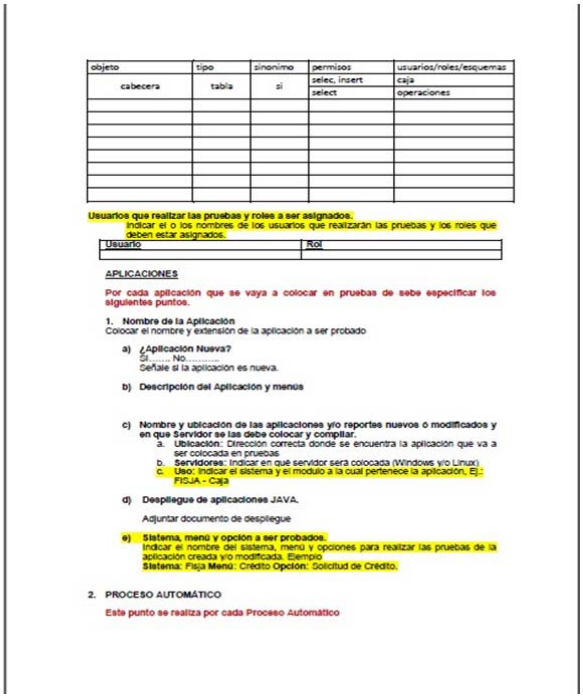
Insertar, actualizar, eliminar registros de tablas
Enviar por correo electrónico los scripts donde consten las inserciones, actualizaciones y eliminaciones de las tablas con su respectivo orden.

Creación de usuarios
Indicar el nombre y contraseña de usuarios a ser creados tanto a nivel de base de datos y de aplicaciones (si es indispensable para que la aplicación funcione).

Creación de roles
Indicar el nombre de los roles a crear.

Asignación de permisos DML[2] y DDL[3] sobre objetos creados y/o modificados a roles, usuarios y esquemas.

[1] Se refiere a tablas, procedimientos, paquetes, vistas, secuencias, etc
[2] Lenguaje de Manipulación de Datos se refiere a las sentencias Select, Insert, Update, Delete, etc.
[3] Lenguaje de Definición de Datos se refiere a poder realizar las siguientes sentencias CREATE TABLE, ALTER INDEX, DROP TABLE, etc.



objeto	tipo	sinonimo	permisos	usuarios/roles/esquemas
cabecera	tabla	si	selec, insert select	caja operaciones

Usuarios que realizar las pruebas y roles a ser asignados.
Indicar el o los nombres de los usuarios que realizaran las pruebas y los roles que deben estar asignados.

Usuario	Rol

APLICACIONES

Por cada aplicación que se vaya a colocar en pruebas de sebe especificar los siguientes puntos.

1. **Nombre de la Aplicación**
Colocar el nombre y extensión de la aplicación a ser probado

a) **¿Aplicación Nueva?**
Si..... No.....
Señale si la aplicación es nueva.

b) **Descripción del Aplicación y menús**

c) **Nombre y ubicación de las aplicaciones y/o reportes nuevos ó modificados y en que Servidor se las debe colocar y compilar.**
a. **Ubicación:** Dirección correcta donde se encuentra la aplicación que va a ser colocada en pruebas
b. **Servidoras:** Indicar en qué servidor será colocada (Windows y/o Linux)
c. **Uso:** Indicar el sistema y el modulo a la cual pertenece la aplicación, Ej.: FISJA - Caja

d) **Despliegue de aplicaciones** JAVA.
Adjuntar documento de despliegue

e) **Sistema, menú y opción a ser probados.**
Indicar el nombre del sistema, menú y opciones para realizar las pruebas de la aplicación creada y/o modificada. Ejemplo
Sistema: Fisja **Menú:** Crédito **Opción:** Solicitud de Crédito.

2. **PROCESO AUTOMÁTICO**

Este punto se realiza por cada Proceso Automático

Nombre
Nombre del procedimiento almacenado y a que esquema pertenece, si esta dentro de un paquete adicionar este nombre del paquete.

Descripción
Redacte una breve descripción del proceso automático, indicando que es lo que realiza al ser ejecutado.

Objetos que intervienen y su acción[4]
Indicar los objetos que intervienen y las acciones que se realizan sobre ellos

Servicios Externos
Si el proceso necesita de servicios externos para su funcionamiento como VPN, Internet, etc.

Periodicidad y Hora de Ejecución
Indicar la periodicidad de la ejecución del proceso automático y una hora estimada para su ejecución.

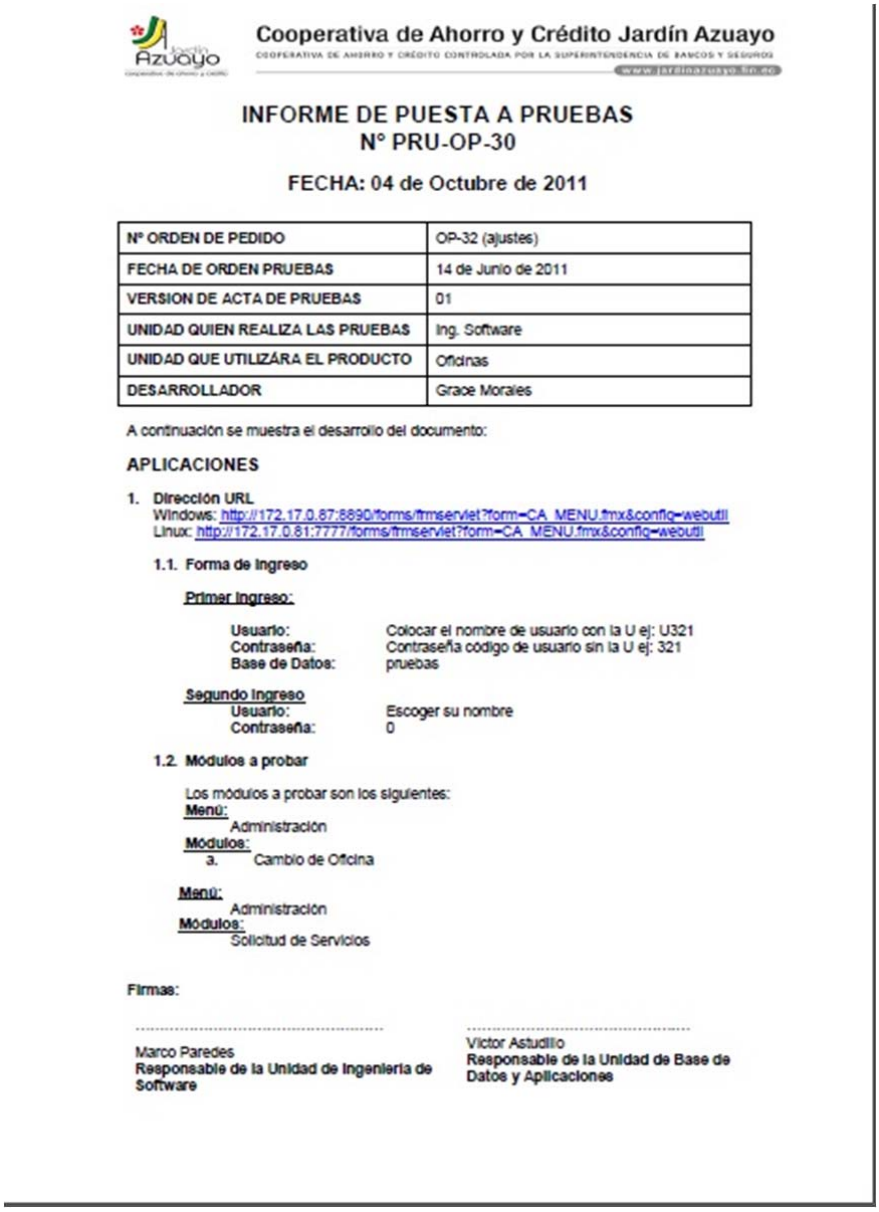
Importante: Coloquen los nombres de cada una de las personas que van en los firmantes según su rol

Firmas:

.....	.....	.....
Coo. de la Unidad de Desarrollo de Software	Desarrollador quien atendio el requerimiento.	Coo. de la Unidad de Base de Datos y Aplicaciones

[4] Tablas, procedimientos, funciones u otros procesos automáticos. Acciones: Select, Insert, Execute, etc

Informe Puesta en Pruebas



Cooperativa de Ahorro y Crédito Jardín Azuayo

COOPERATIVA DE AHORRO Y CRÉDITO CONTROLADA POR LA SUPERINTENDENCIA DE BANCOS Y SEGUROS

www.jardinazuayo.fin.ec

INFORME DE PUESTA A PRUEBAS
N° PRU-OP-30

FECHA: 04 de Octubre de 2011

Nº ORDEN DE PEDIDO	OP-32 (ajustes)
FECHA DE ORDEN PRUEBAS	14 de Junio de 2011
VERSION DE ACTA DE PRUEBAS	01
UNIDAD QUIEN REALIZA LAS PRUEBAS	Ing. Software
UNIDAD QUE UTILIZÁRA EL PRODUCTO	Oficinas
DESARROLLADOR	Grace Morales

A continuación se muestra el desarrollo del documento:

APLICACIONES

1. **Dirección URL**
   Windows: http://172.17.0.87:8890/forms/frmservlet?form=CA_MENU.fmx&config=webutil
   Linux: http://172.17.0.81:7777/forms/frmservlet?form=CA_MENU.fmx&config=webutil

   1.1. Forma de Ingreso

   Primer Ingreso:

   Usuario: Colocar el nombre de usuario con la U ej: U321
   Contraseña: Contraseña código de usuario sin la U ej: 321
   Base de Datos: pruebas

   Segundo Ingreso
   Usuario: Escoger su nombre
   Contraseña: 0

   1.2. Módulos a probar

   Los módulos a probar son los siguientes:
   Menú:
   Administración
   Módulos:
   a. Cambio de Oficina

   Menú:
   Administración
   Módulos:
   Solicitud de Servicios

Firmas:

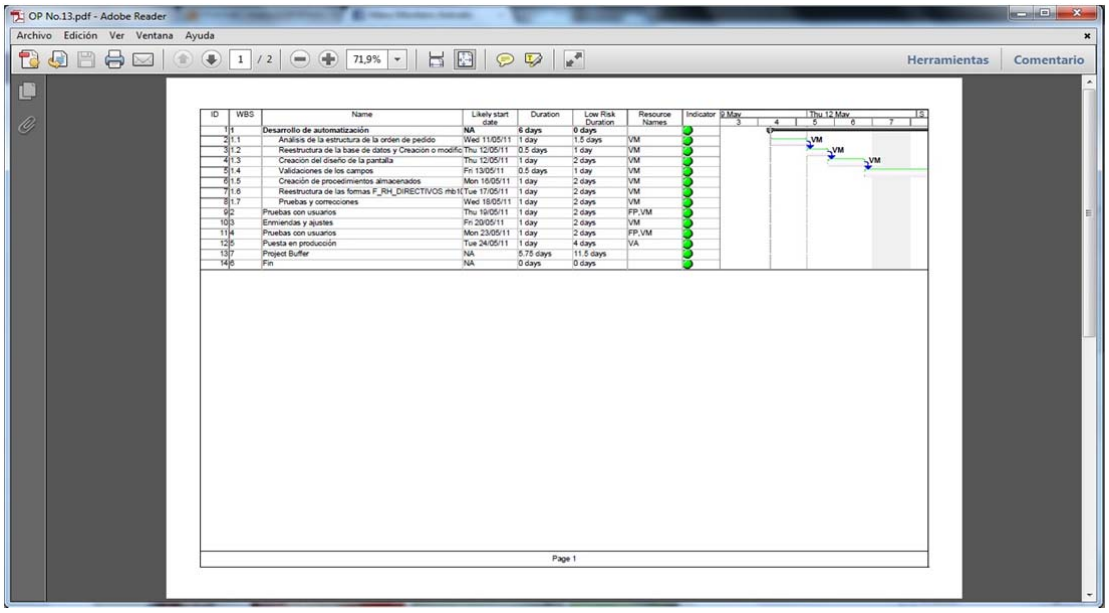
Marco Paredes
Responsable de la Unidad de Ingeniería de Software

Victor Astudillo
Responsable de la Unidad de Base de Datos y Aplicaciones

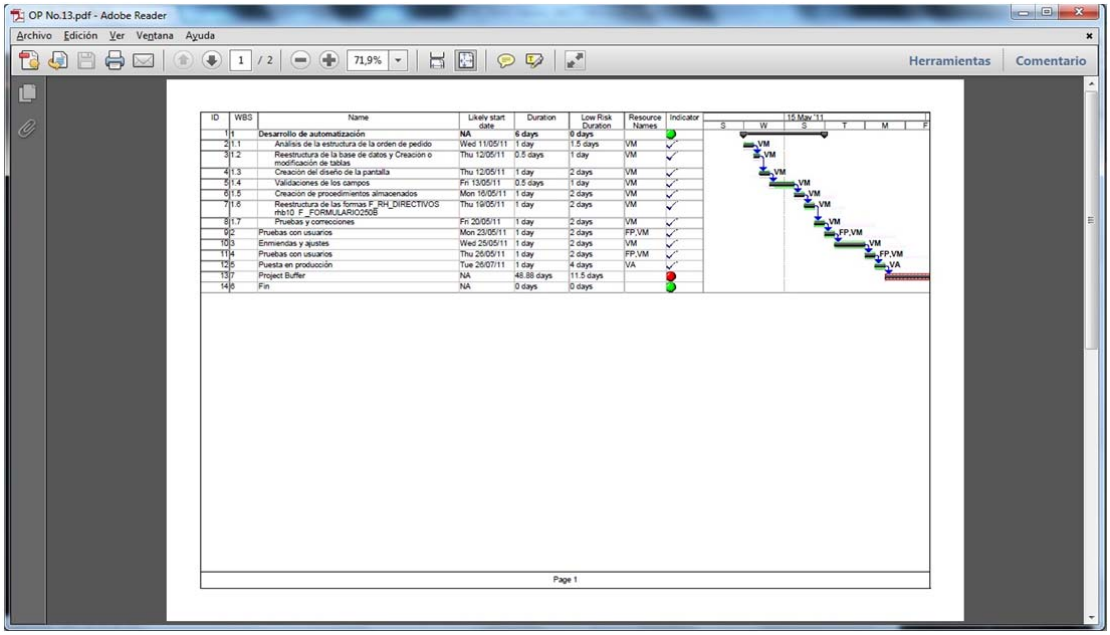
Anexo III. Redes de Proyectos de las Órdenes (Proyectadas y Reales)
ORDEN No. 13

Vinculados

Red Proyectada



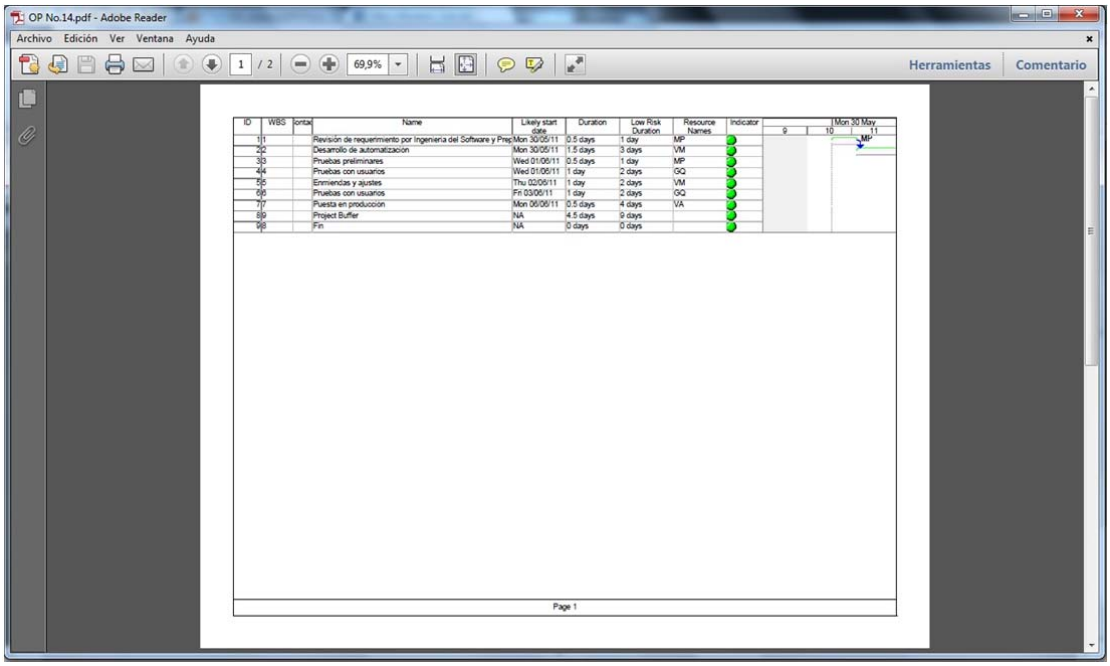
Red Real



ORDEN No. 14

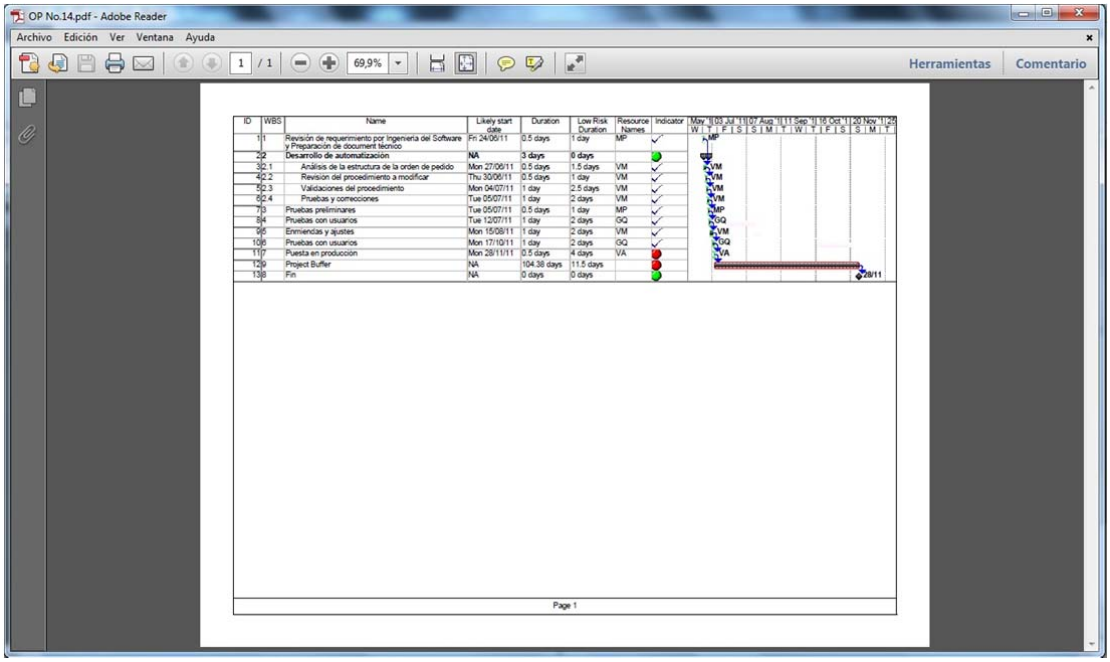
Alcancía Segura

Red Proyectada



ID	WBS	Name	Likely start date	Duration	Low Risk Duration	Resource Names	Indicator
1	1	Revisión de requerimiento por Ingeniería del Software y Prep	Mon 30/05/11	0.5 days	1 day	MP	
2	2	Desarrollo de automatizacion	Mon 30/05/11	1.5 days	3 days	VM	
3	3	Pruebas preliminares	Wed 01/06/11	0.5 days	1 day	MP	
4	4	Pruebas con usuarios	Wed 01/06/11	1 day	2 days	GQ	
5	5	Enmiendas y ajustes	Thu 02/06/11	1 day	2 days	VM	
6	6	Pruebas con usuarios	Fri 03/06/11	1 day	2 days	GQ	
7	7	Puesta en producción	Mon 06/06/11	0.5 days	4 days	VA	
8	9	Project Buffer	NA	4.5 days	9 days		
9	8	Fin	NA	0 days	0 days		

Red Real

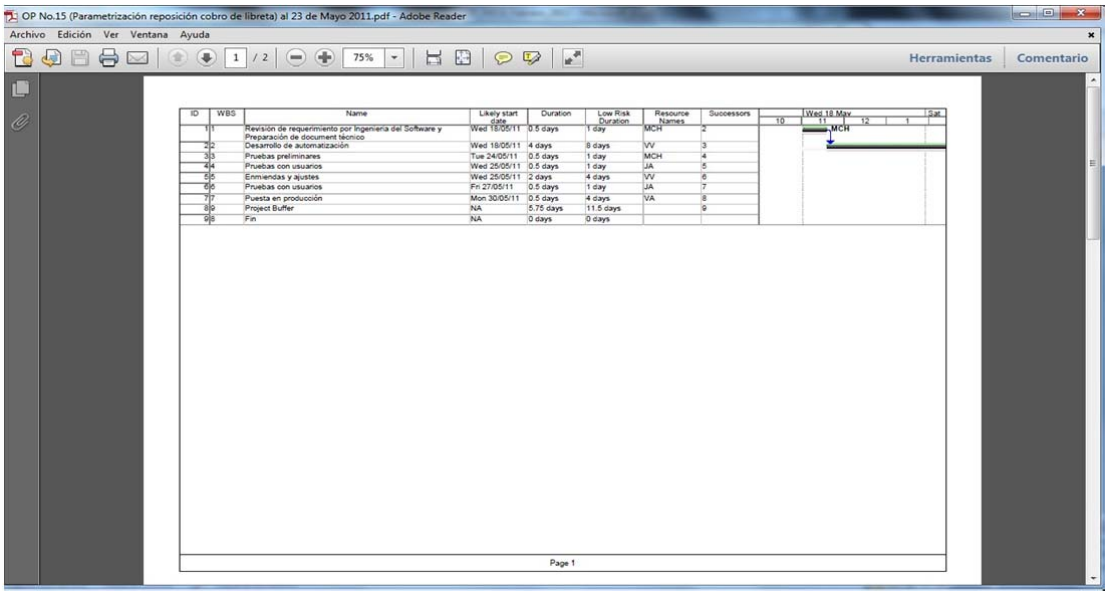


ID	WBS	Name	Likely start date	Duration	Low Risk Duration	Resource Names	Indicator
1	1	Revisión de requerimiento por Ingeniería del Software y Preparación de document técnico	Fri 24/06/11	0.5 days	1 day	MP	
2	2	Desarrollo de automatización	NA	3 days	6 days		
3	2.1	Análisis de la estructura de la orden de pedido	Mon 27/06/11	0.5 days	1.5 days	VM	
4	2.2	Revision del procedimiento a modificar	Thu 30/06/11	0.5 days	1 day	VM	
5	2.3	Validaciones del procedimiento	Mon 04/07/11	1 day	2.5 days	VM	
6	2.4	Pruebas y correcciones	Tue 05/07/11	1 day	2 days	VM	
7	3	Pruebas preliminares	Tue 05/07/11	0.5 days	1 day	MP	
8	4	Pruebas con usuarios	Tue 12/07/11	1 day	2 days	GQ	
9	5	Enmiendas y ajustes	Mon 15/08/11	1 day	2 days	VM	
10	6	Pruebas con usuarios	Mon 17/10/11	1 day	2 days	GQ	
11	7	Puesta en producción	Mon 28/11/11	0.5 days	4 days	VA	
12	9	Project Buffer	NA	104.38 days	11.5 days		
13	8	Fin	NA	0 days	0 days		

ORDEN No. 15

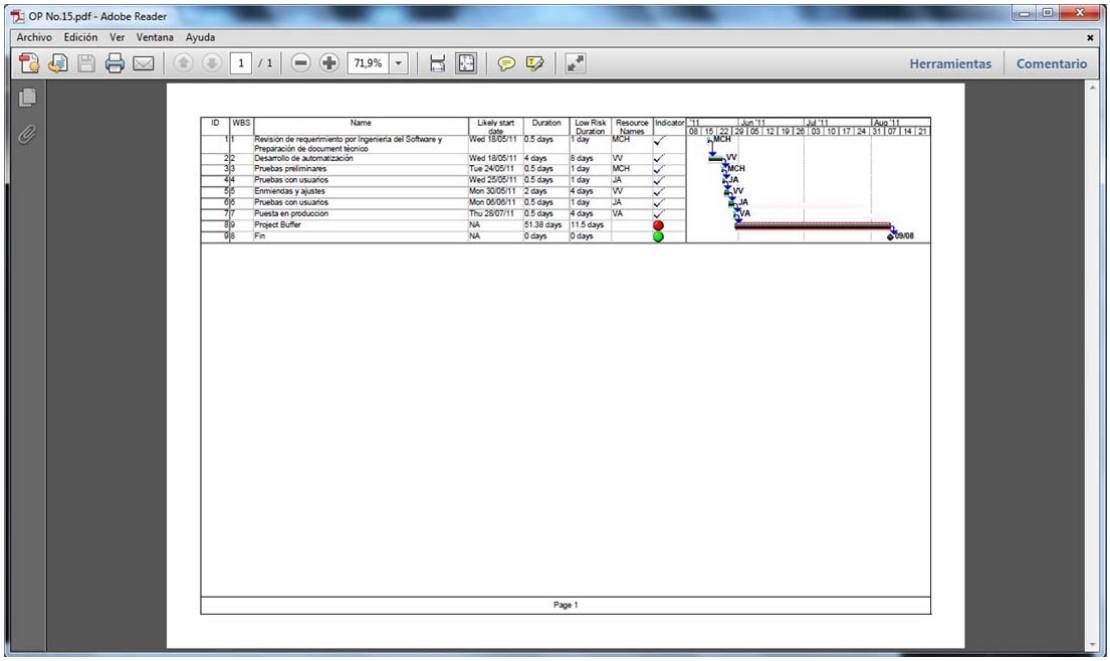
Parametrización reposición cobro de libreta

Red Proyectada



ID	WBS	Name	Likely start date	Duration	Low Risk Duration	Resource Names	Successors
1	1	Revisión de requerimiento por Ingeniería del Software y Preparación de document técnico	Wed 18/05/11	0.5 days	1 day	MCH	2
2	2	Desarrollo de automatización	Wed 18/05/11	4 days	8 days	VV	3
3	3	Pruebas preliminares	Tue 24/05/11	0.5 days	1 day	MCH	4
4	4	Pruebas con usuarios	Wed 25/05/11	0.5 days	1 day	JA	5
5	5	Enmiendas y ajustes	Wed 25/05/11	2 days	4 days	VV	6
6	6	Pruebas con usuarios	Fri 27/05/11	0.5 days	1 day	JA	7
7	7	Puesta en producción	Mon 30/05/11	0.5 days	4 days	VA	8
8	9	Project Buffer	NA	5.75 days	11.5 days		9
9	8	Fin	NA	0 days	0 days		

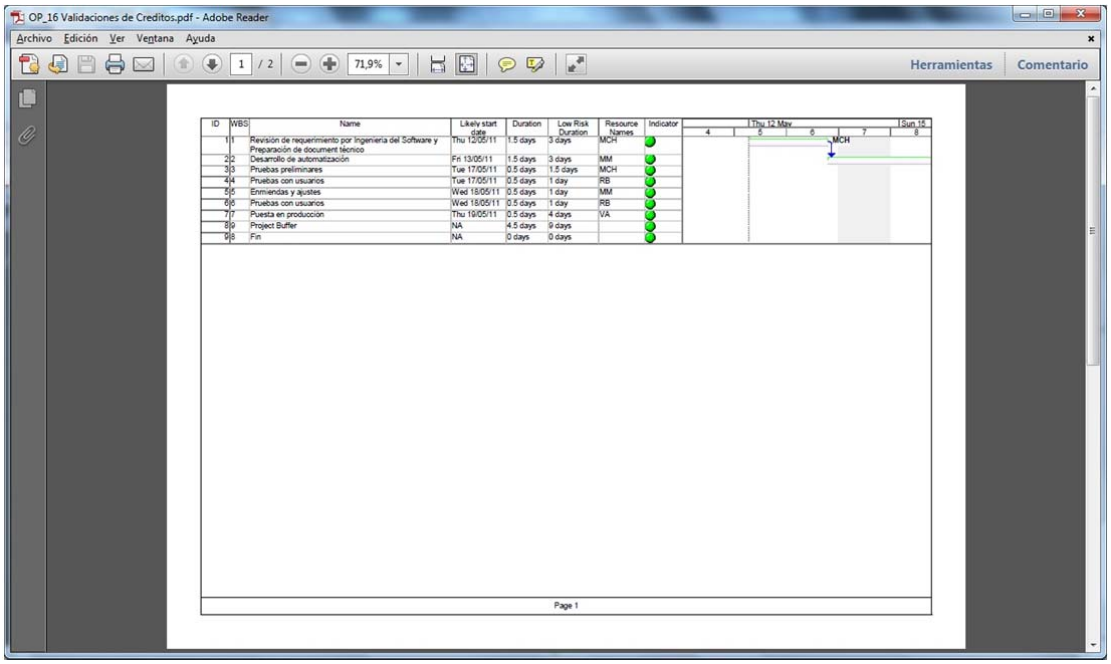
Red Real



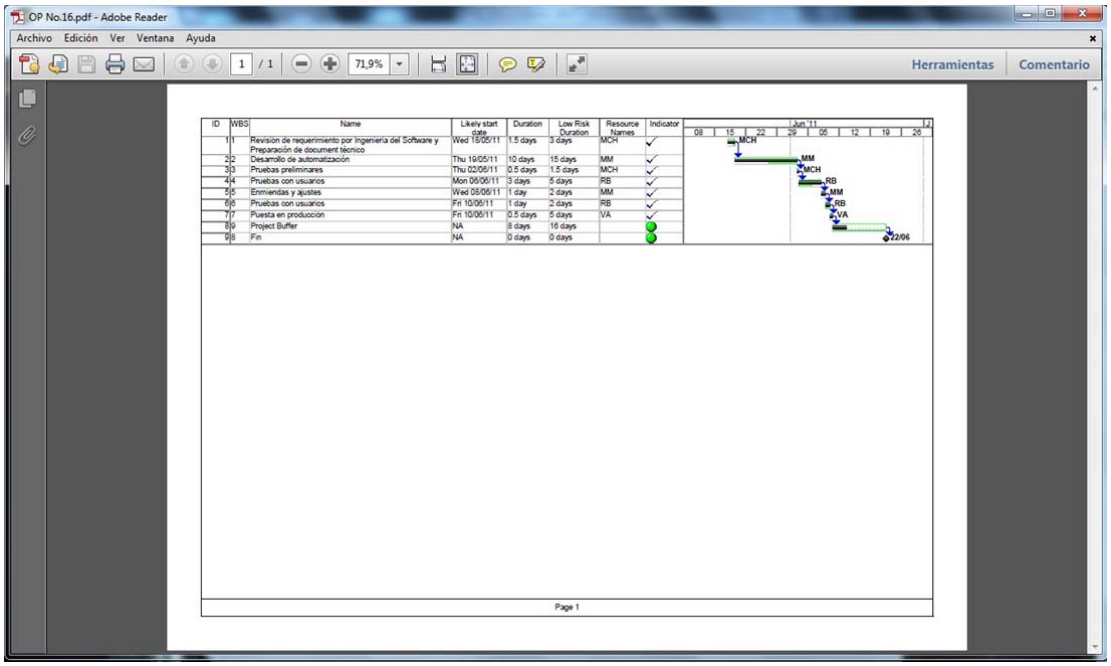
ORDEN No. 16

Validaciones de Créditos

Red Proyectada


OP_16 Validaciones de Creditos.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
ID WBS Name Likely start date Duration Low Risk Duration Resource Names Indicator
1 1 Revisión de requerimiento por Ingeniería del Software y Preparación de document técnico Thu 12/05/11 1.5 days 3 days MCH
2 2 Desarrollo de automatización Fri 13/05/11 1.5 days 3 days MM
3 3 Pruebas preliminares Tue 17/05/11 0.5 days 1.5 days MCH
4 4 Pruebas con usuarios Tue 17/05/11 0.5 days 1 day RB
5 5 Enmiendas y ajustes Wed 18/05/11 0.5 days 1 day MM
6 6 Pruebas con usuarios Wed 18/05/11 0.5 days 1 day RB
7 7 Puesta en produccion Thu 19/05/11 0.5 days 4 days VA
8 9 Project Buffer NA 4.5 days 9 days
9 8 Fin NA 0 days 0 days
Thu 12 May
Sun 15
Page 1

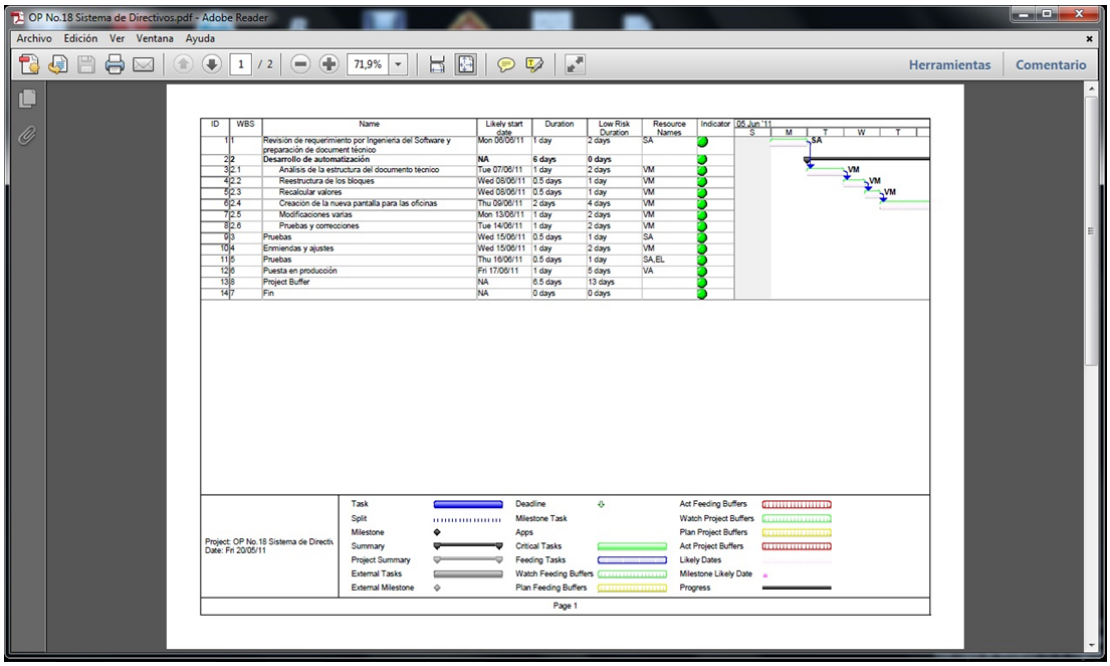
Red Real


OP No.16.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
ID WBS Name Likely start date Duration Low Risk Duration Resource Names Indicator
1 1 Revisión de requerimiento por Ingeniería del Software y Preparación de document técnico Wed 18/05/11 1.5 days 3 days MCH
2 2 Desarrollo de automatización Thu 19/05/11 10 days 15 days MM
3 3 Pruebas preliminares Thu 02/06/11 0.5 days 1.5 days MCH
4 4 Pruebas con usuarios Mon 06/06/11 3 days 5 days RB
5 5 Enmiendas y ajustes Wed 08/06/11 1 day 2 days MM
6 6 Pruebas con usuarios Fri 10/06/11 1 day 2 days RB
7 7 Puesta en produccion Fri 10/06/11 0.5 days 5 days VA
8 9 Project Buffer NA 8 days 16 days
9 8 Fin NA 0 days 0 days
Jun '11
08 15 22 29 05 12 19 26
MCH
MM
MCH
RB
MM
RB
VA
22/06
Page 1

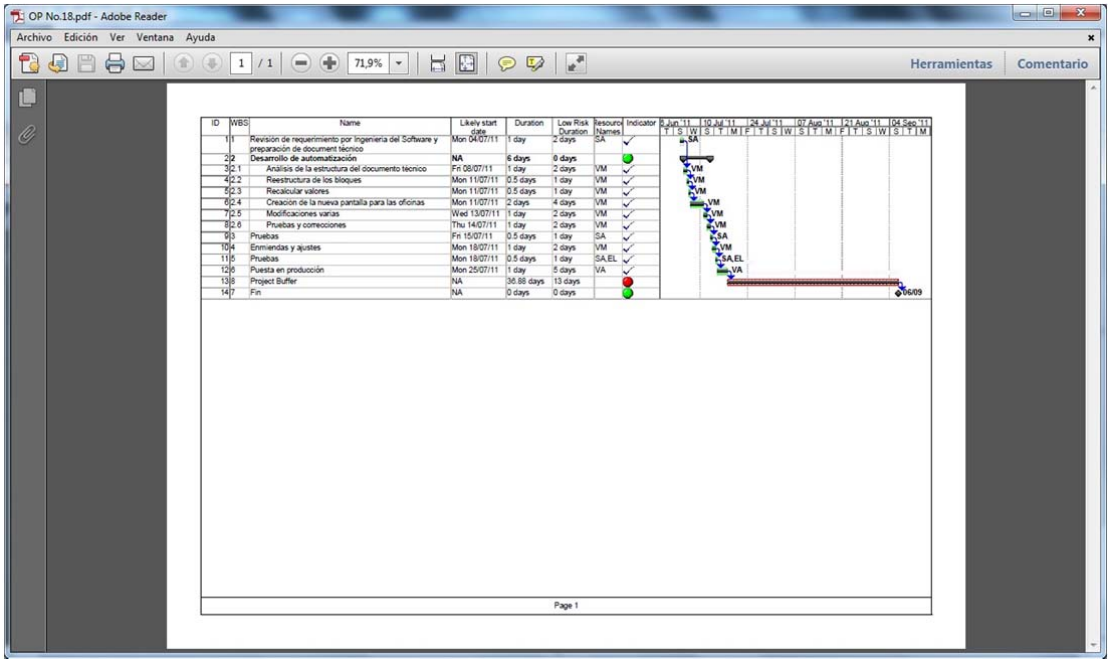
ORDEN No. 18

Sistema Directivos

Red Proyectada



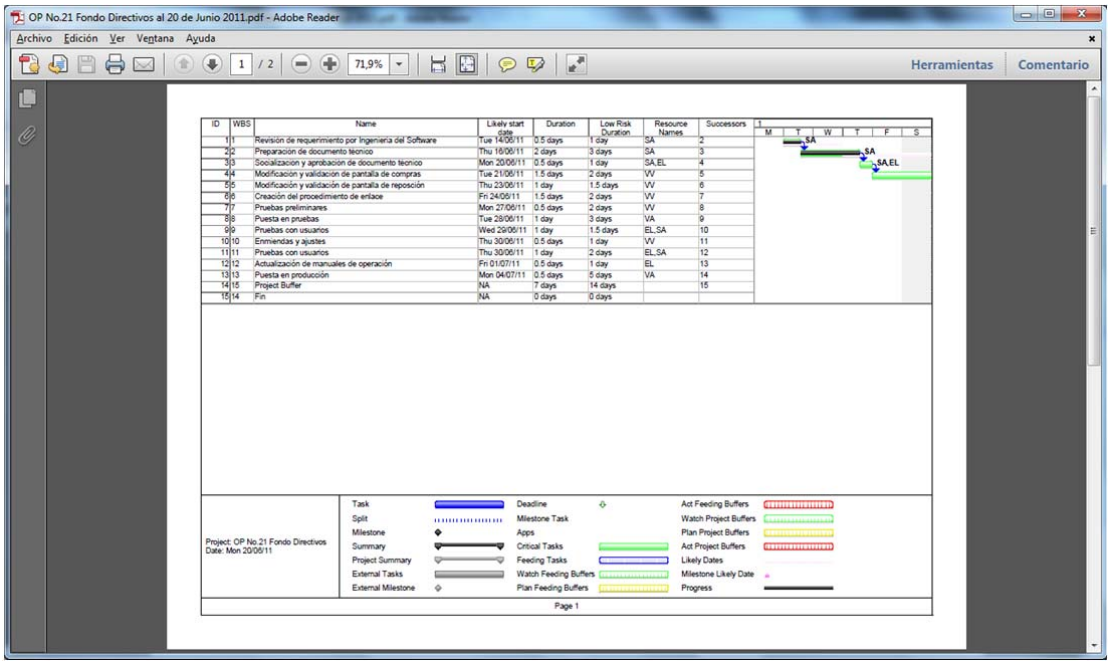
Red Real



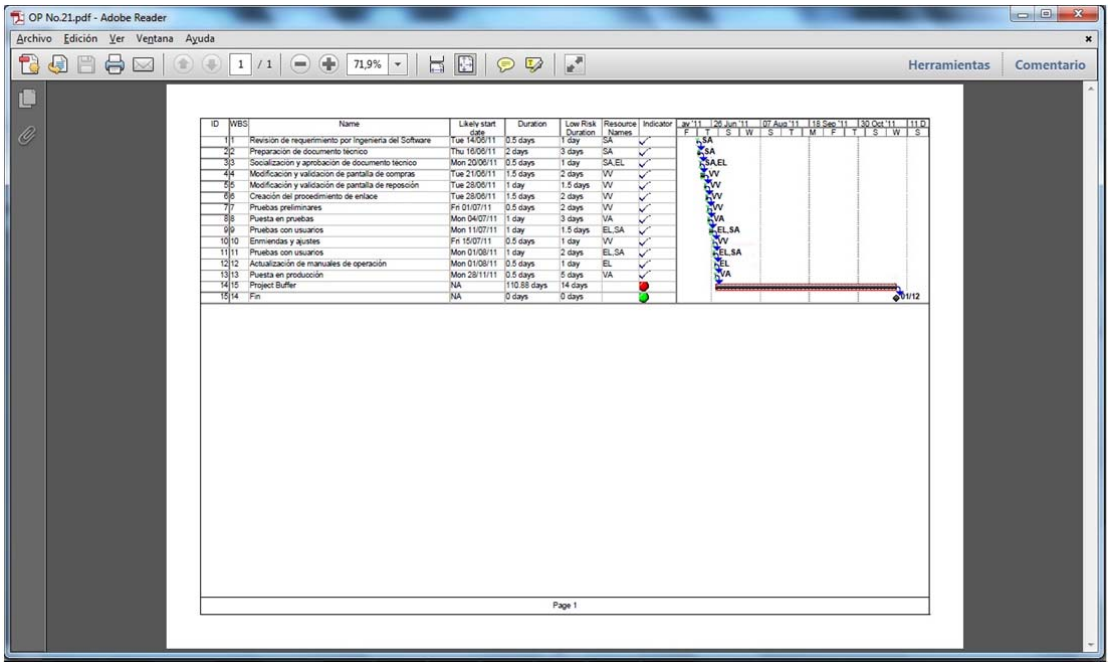
ORDEN No. 21

Fondo Directivos

Red Proyectada


OP No.21 Fondo Directivos al 20 de Junio 2011.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
Project: OP No.21 Fondo Directivos
Date: Mon 20/05/11
Page 1

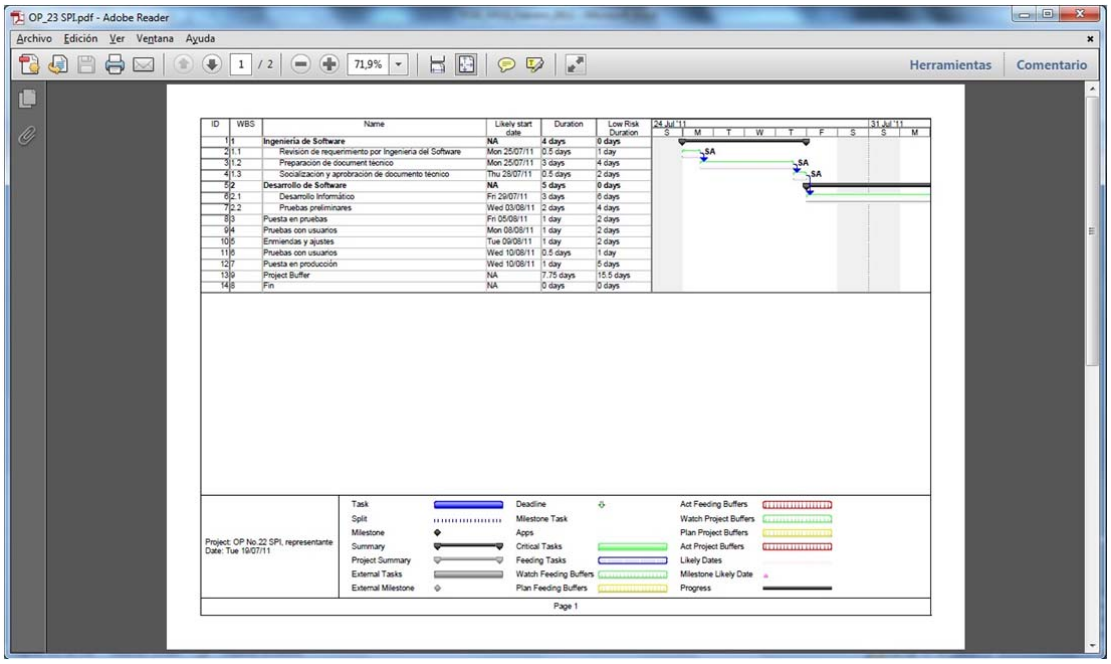
Red Real


OP No.21.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
Page 1

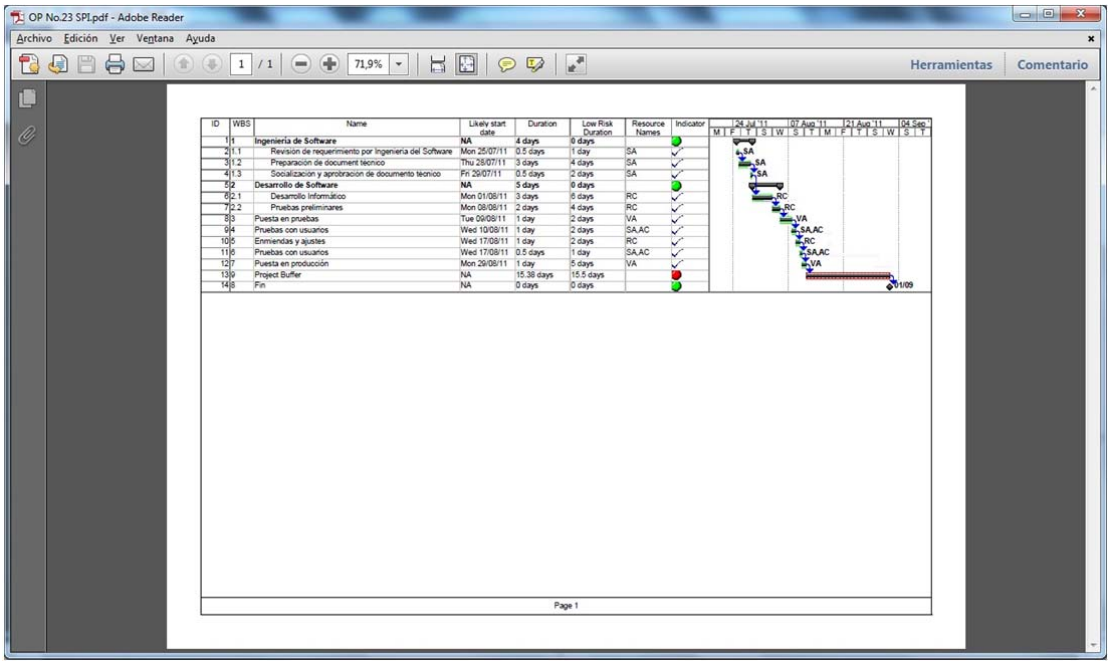
ORDEN No. 23

SPI

Red Proyectada



Red Real



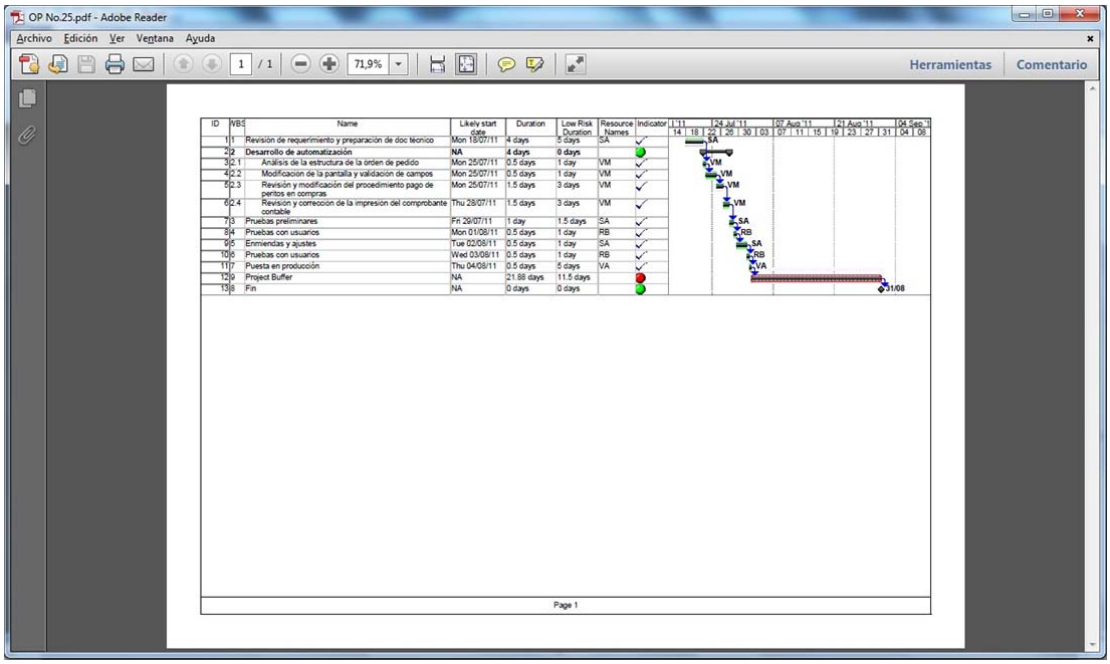
ORDEN No. 25

Liquidaciones a Peritos

Red Proyectada

NO DISPONIBLE

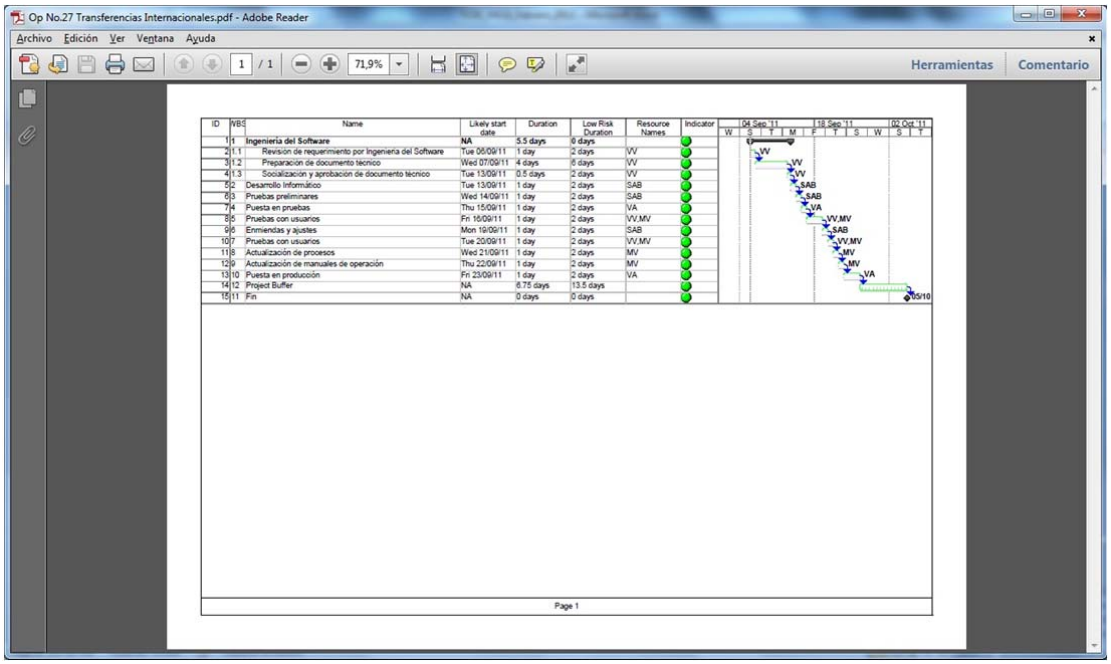
Red Real



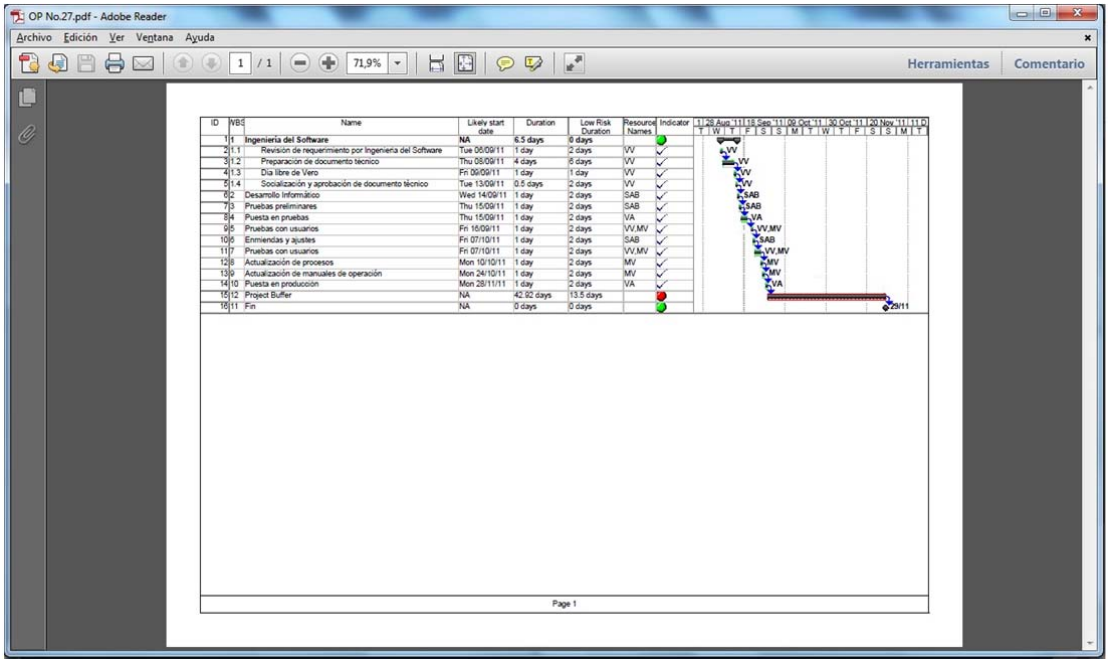
ORDEN No. 27

Transferencias Internacionales

Red Proyectada


Op No.27 Transferencias Internacionales.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
Page 1

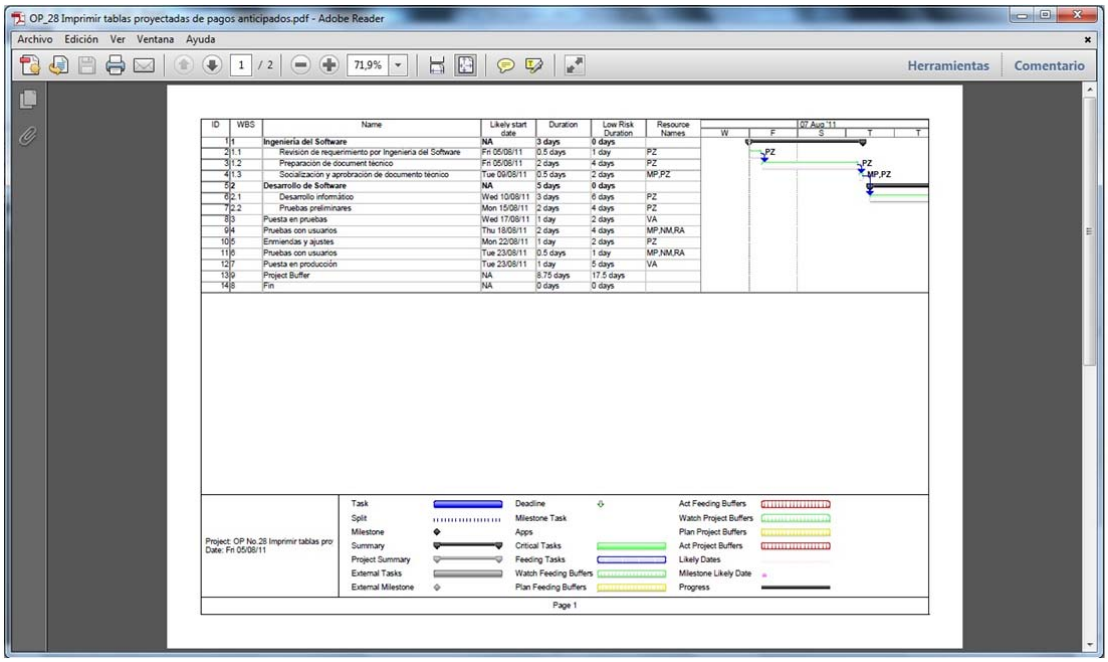
Red Real


OP No.27.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas Comentario
Page 1

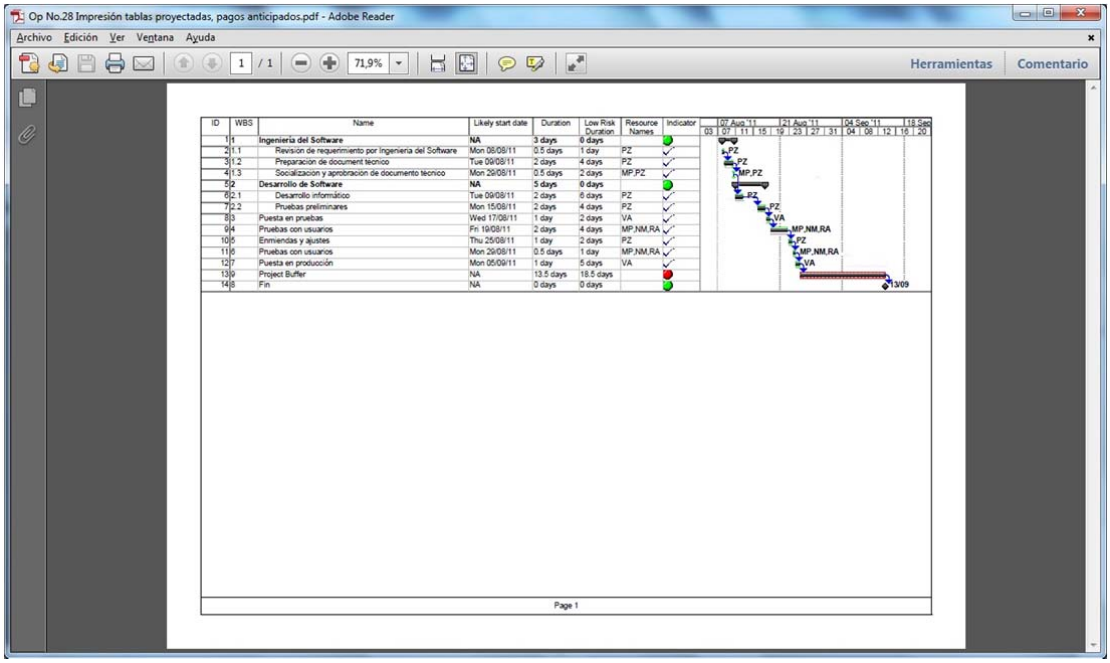
ORDEN No. 28

Imprimir Tablas Proyectadas, Pagos Anticipados

Red Proyectada



Red Real

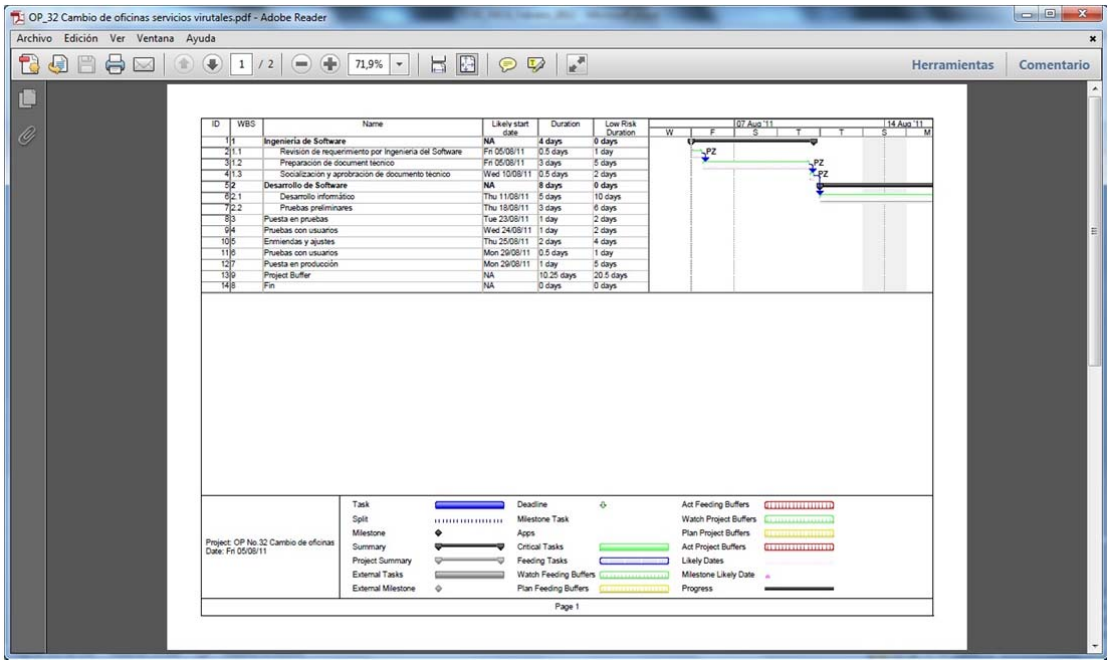


ID	WBS	Name	Likely start date	Duration	Low Risk Duration	Resource Names	Indicator
1	1	Ingeniería del Software	NA	3 days	6 days		
2	1.1	Revisión de requerimiento por Ingeniería del Software	Mon 08/08/11	0.5 days	1 day	PZ	
3	1.2	Preparacion de document técnico	Tue 09/08/11	2 days	4 days	PZ	
4	1.3	Socializacion y aprobracion de documento técnico	Mon 29/08/11	0.5 days	2 days	MP,PZ	
5	2	Desarrollo de Software	NA	5 days	9 days		
6	2.1	Desarrollo informático	Tue 09/08/11	2 days	6 days	PZ	
7	2.2	Pruebas preliminares	Mon 15/08/11	2 days	4 days	PZ	
8	3	Puesta en pruebas	Wed 17/08/11	1 day	2 days	VA	
9	4	Pruebas con usuarios	Fri 19/08/11	2 days	4 days	MP,NM,RA	
10	5	Enmiendas y ajustes	Thu 25/08/11	1 day	2 days	PZ	
11	6	Pruebas con usuarios	Mon 29/08/11	0.5 days	1 day	MP,NM,RA	
12	7	Puesta en producción	Mon 05/09/11	1 day	5 days	VA	
13	9	Project Buffer	NA	13.5 days	18.5 days		
14	8	Fin	NA	0 days	0 days		

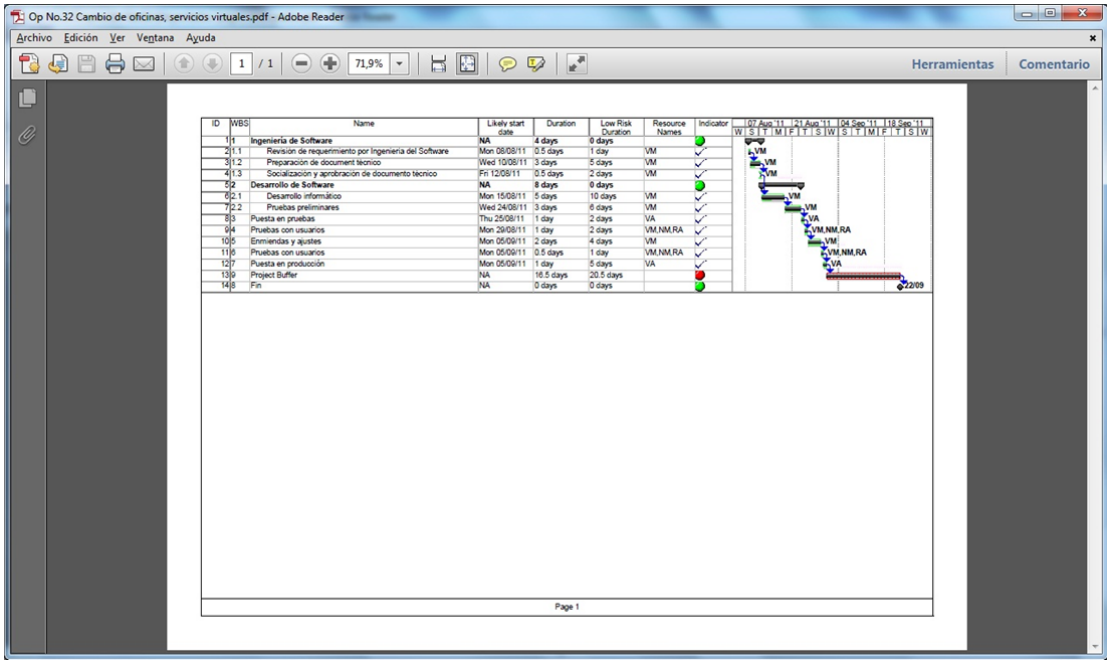
ORDEN No. 32

Cambio de Oficinas, Servicios Virtuales

Red Proyectada



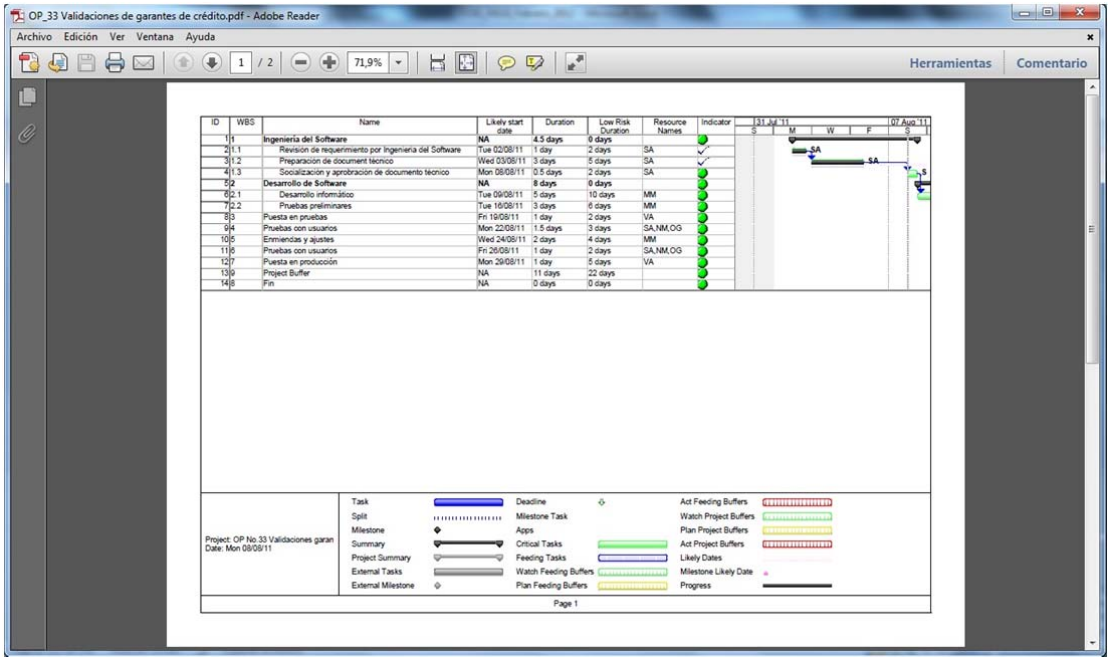
Red Real



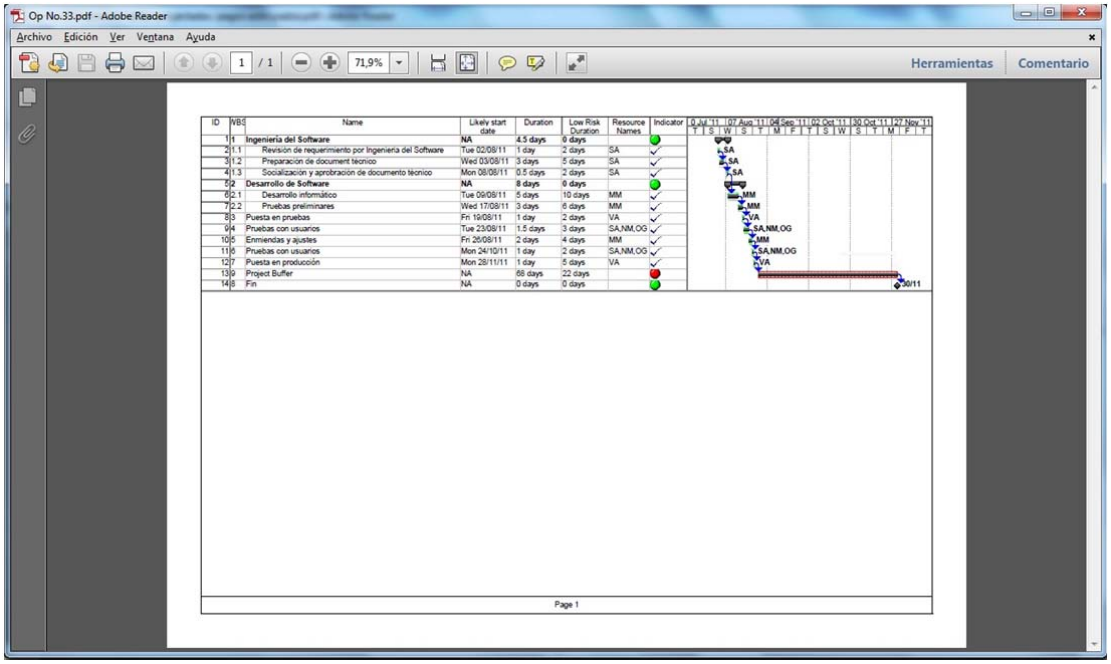
ORDEN No. 33

Validaciones Garantes de Crédito

Red Proyectada



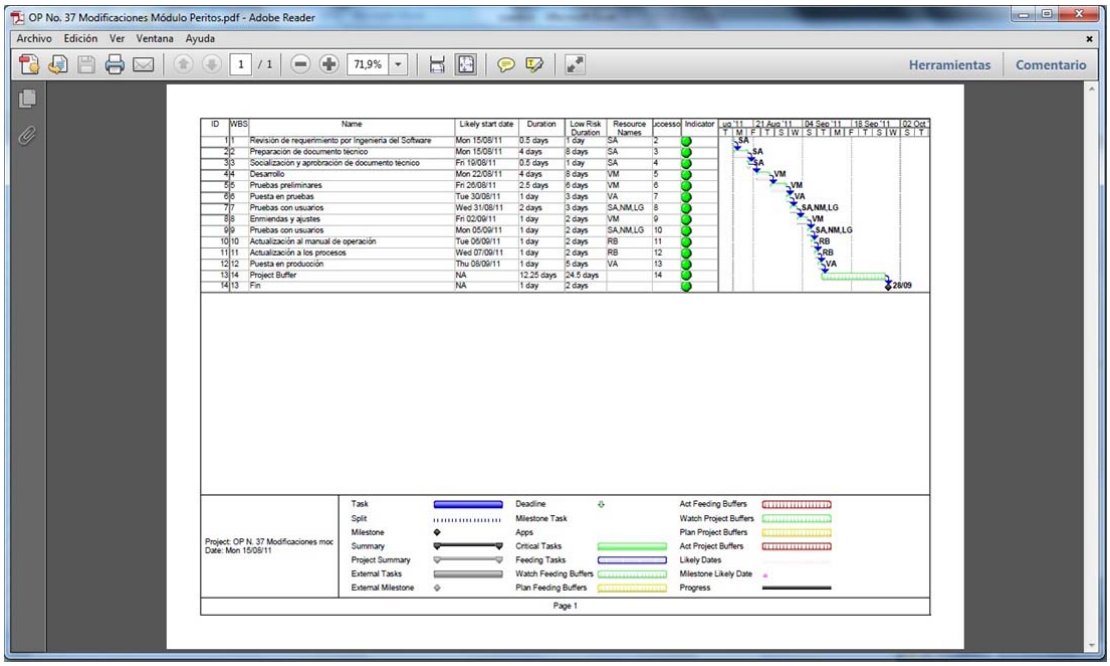
Red Real



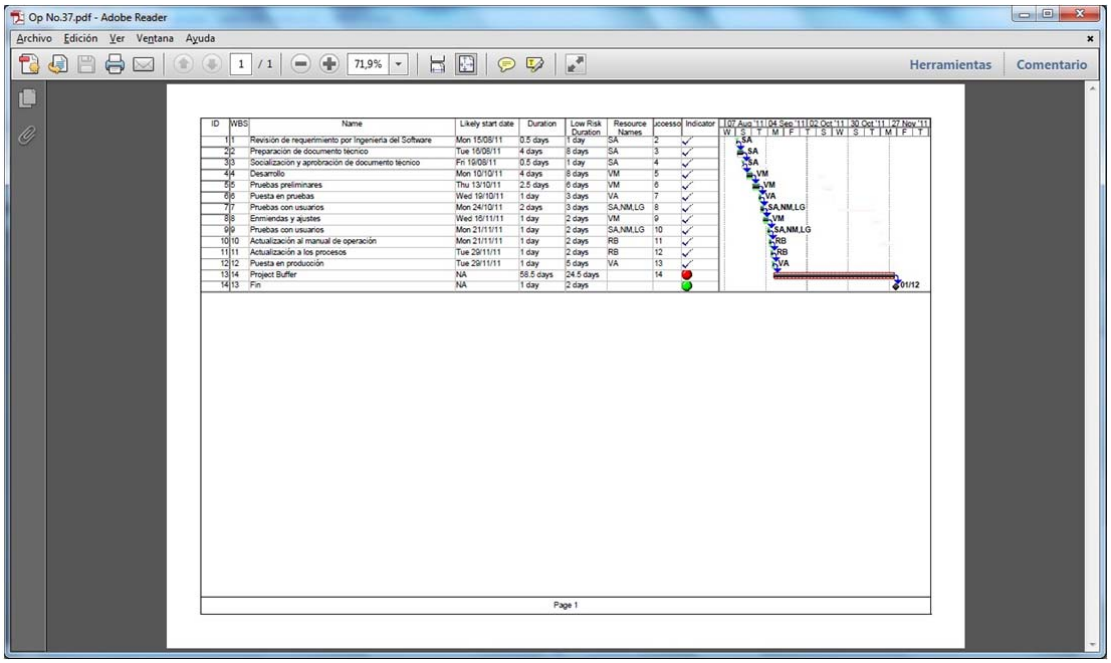
ORDEN No. 37

Modificaciones al Módulo de Peritos

Red Proyectada



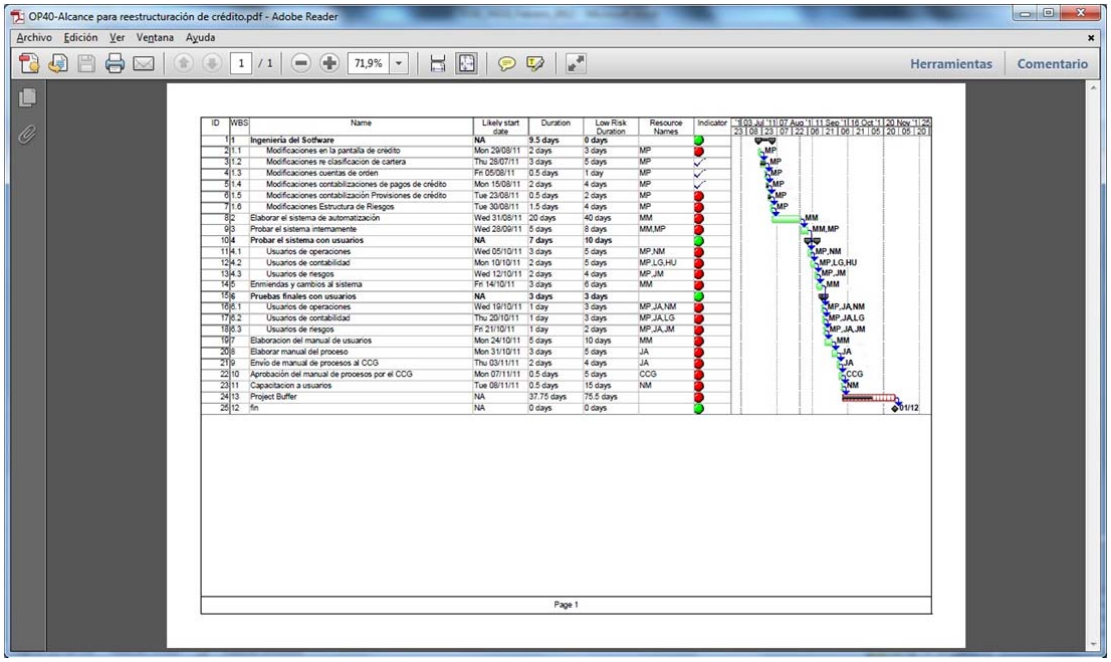
Red Real



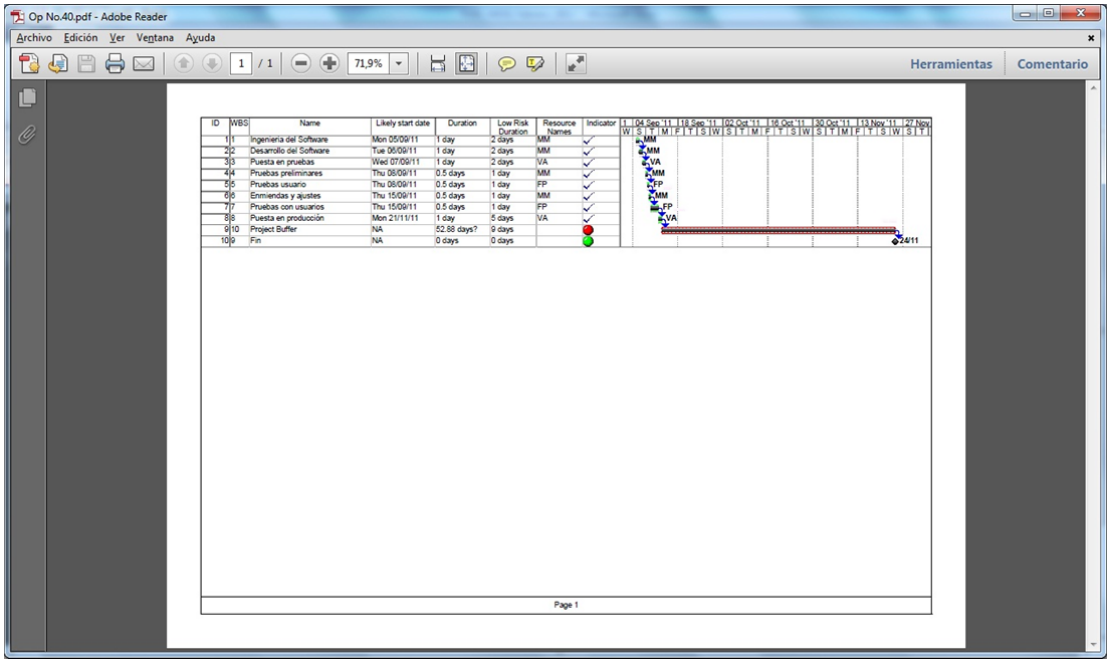
ORDEN No. 40

Modificación al Módulo de Vinculados

Red Proyectada



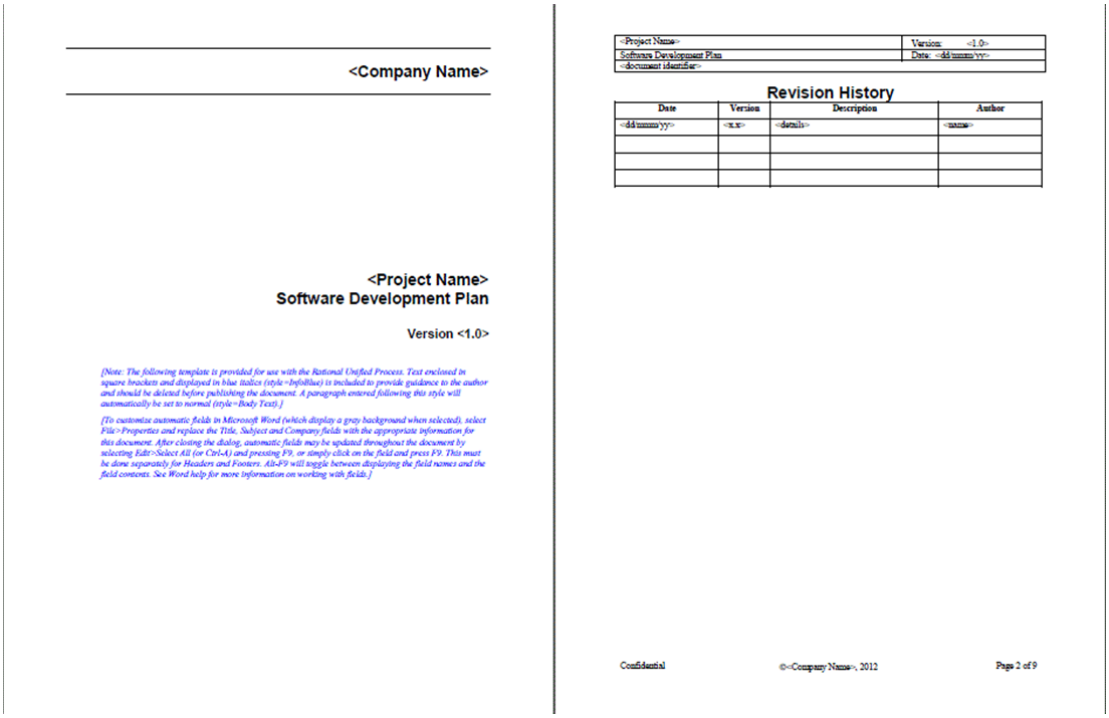
Red Real


Op No.40.pdf - Adobe Reader
Archivo Edición Ver Ventana Ayuda
Herramientas
Comentario
Page 1

Anexo IV. Plantillas de Flujo de Actividades según el Modelo de Proceso de Desarrollo RUP.

MODELADO DEL NEGOCIO

Plan de Desarrollo



<Company Name>

<Project Name>
Software Development Plan

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Software Development Plan	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2 of 9

Table of Contents

Software Development Plan

1. Introduction

*[The introduction of the **Software Development Plan** should provide an overview of the entire document. It should include the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Software Development Plan**.]*

1.1 Purpose

*[Specify the purpose of this **Software Development Plan**.]*

1.2 Scope

*[A brief description of the scope of this **Software Development Plan**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Development Plan**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Software Development Plan**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.*

*For the **Software Development Plan**, the list of referenced artifacts includes:*

- *Iteration Plans*
- *Requirements Management Plan*
- *Measurement Plan*
- *Risk Management Plan*
- *Development Case*
- *Business Modeling Guidelines*
- *User Interfaces Guidelines*
- *Use-Case-Modeling Guidelines*
- *Design Guidelines*
- *Programming Guidelines*
- *Test Guidelines*
- *Manual Style Guide*
- *Infrastructure Plan*
- *Product Acceptance Plan*
- *Configuration Management Plan*
- *Evaluation Plan (only if this is a separate plan—normally this is addressed in Section 6.2 of the **Software Development Plan**)*

- *Documentation Plan*
- *Quality Assurance Plan*
- *Problem Resolution Plan*
- *Subcontractor Management Plan*
- *Process Improvement Plan]*

1.5 Overview

*[This subsection describes what the rest of the **Software Development Plan** contains and explains how the document is organized.]*

2. Project Overview

2.1 Project Purpose, Scope, and Objectives

[A brief description of the purpose and objectives of this project and a brief description of what deliverables the project is expected to deliver.]

2.2 Assumptions and Constraints

[A list of assumptions that this plan is based and any constraints, for example. budget, staff, equipment, schedule, that apply to the project.]

2.3 Project Deliverables

[A tabular list of the artifacts to be created during the project, including target delivery dates.]

2.4 Evolution of the Software Development Plan

*[A table of proposed versions of the **Software Development Plan**, and the criteria for the unscheduled revision and release of this plan.]*

3. Project Organization

3.1 Organizational Structure

[Describe the organizational structure of the project team, including management and other review authorities.]

3.2 External Interfaces

[Describe how the project interfaces with external groups. For each external group, identify the internal and external contact names.]

3.3 Roles and Responsibilities

[Identify the project organizational units that will be responsible for each of the disciplines, workflow details, and supporting processes.]

4. Management Process

4.1 Project Estimates

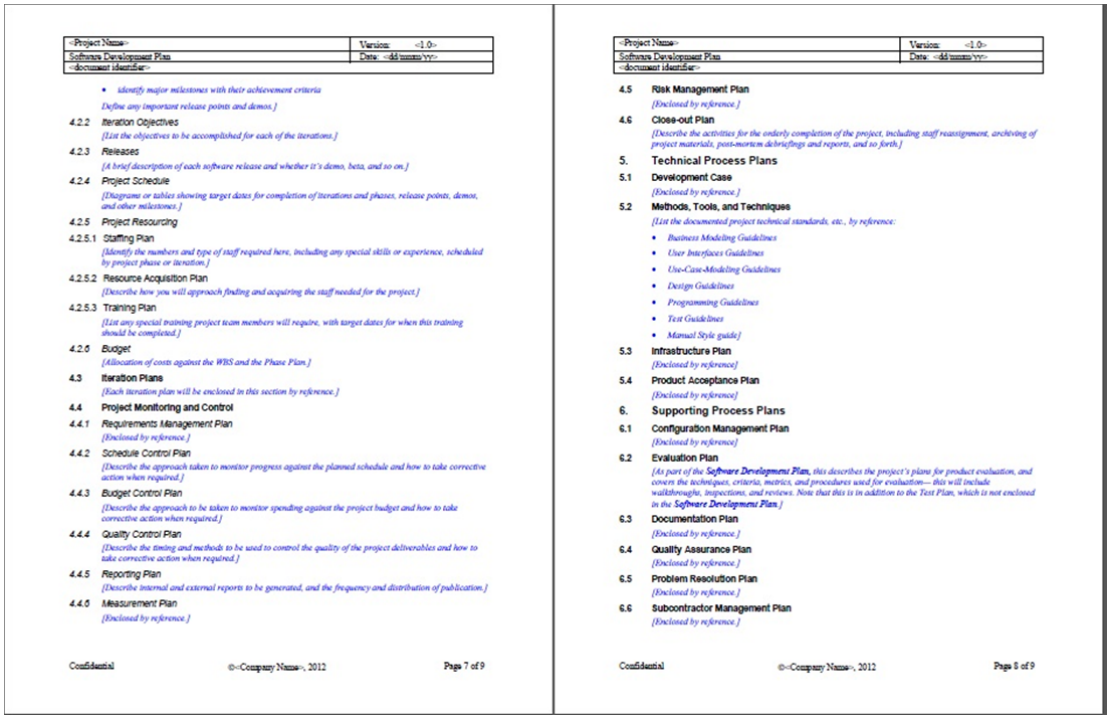
[Provide the estimated cost and schedule for the project, as well as the basis for those estimates, and the points and circumstances in the project when re-estimation will occur.]

4.2 Project Plan

4.2.1 Phase Plan

[Include the following:

- *Work Breakdown Structure (WBS)*
- *a timeline or Gantt chart showing the allocation of time to the project phases or iterations*



- *identify major milestones with their achievement criteria*

Define any important release points and demos.]

4.2.2 *Iteration Objectives*

[List the objectives to be accomplished for each of the iterations.]

4.2.3 *Releases*

[A brief description of each software release and whether it's demo, beta, and so on.]

4.2.4 *Project Schedule*

[Diagrams or tables showing target dates for completion of iterations and phases, release points, demos, and other milestones.]

4.2.5 *Project Resourcing*

4.2.5.1 Staffing Plan

[Identify the numbers and type of staff required here, including any special skills or experience, scheduled by project phase or iteration.]

4.2.5.2 Resource Acquisition Plan

[Describe how you will approach finding and acquiring the staff needed for the project.]

4.2.5.3 Training Plan

[List any special training project team members will require, with target dates for when this training should be completed.]

4.2.6 *Budget*

[Allocation of costs against the WBS and the Phase Plan.]

4.3 Iteration Plans

[Each iteration plan will be enclosed in this section by reference.]

4.4 Project Monitoring and Control

4.4.1 *Requirements Management Plan*

[Enclosed by reference.]

4.4.2 *Schedule Control Plan*

[Describe the approach taken to monitor progress against the planned schedule and how to take corrective action when required.]

4.4.3 *Budget Control Plan*

[Describe the approach to be taken to monitor spending against the project budget and how to take corrective action when required.]

4.4.4 *Quality Control Plan*

[Describe the timing and methods to be used to control the quality of the project deliverables and how to take corrective action when required.]

4.4.5 *Reporting Plan*

[Describe internal and external reports to be generated, and the frequency and distribution of publication.]

4.4.6 *Measurement Plan*

[Enclosed by reference.]

4.5 Risk Management Plan

[Enclosed by reference.]

4.6 Close-out Plan

[Describe the activities for the orderly completion of the project, including staff reassignment, archiving of project materials, post-mortem debriefings and reports, and so forth.]

5. Technical Process Plans

5.1 Development Case

[Enclosed by reference.]

5.2 Methods, Tools, and Techniques

[List the documented project technical standards, etc., by reference:

- *Business Modeling Guidelines*
- *User Interfaces Guidelines*
- *Use-Case-Modeling Guidelines*
- *Design Guidelines*
- *Programming Guidelines*
- *Test Guidelines*
- *Manual Style guide]*

5.3 Infrastructure Plan

[Enclosed by reference]

5.4 Product Acceptance Plan

[Enclosed by reference]

6. Supporting Process Plans

6.1 Configuration Management Plan

[Enclosed by reference]

6.2 Evaluation Plan

*[As part of the **Software Development Plan**, this describes the project's plans for product evaluation, and covers the techniques, criteria, metrics, and procedures used for evaluation— this will include walkthroughs, inspections, and reviews. Note that this is in addition to the Test Plan, which is not enclosed in the **Software Development Plan**.]*

6.3 Documentation Plan

[Enclosed by reference.]

6.4 Quality Assurance Plan

[Enclosed by reference.]

6.5 Problem Resolution Plan

[Enclosed by reference.]

6.6 Subcontractor Management Plan

[Enclosed by reference.]

6.7 **Process Improvement Plan**

[Enclosed by reference.]

7. **Additional Plans**

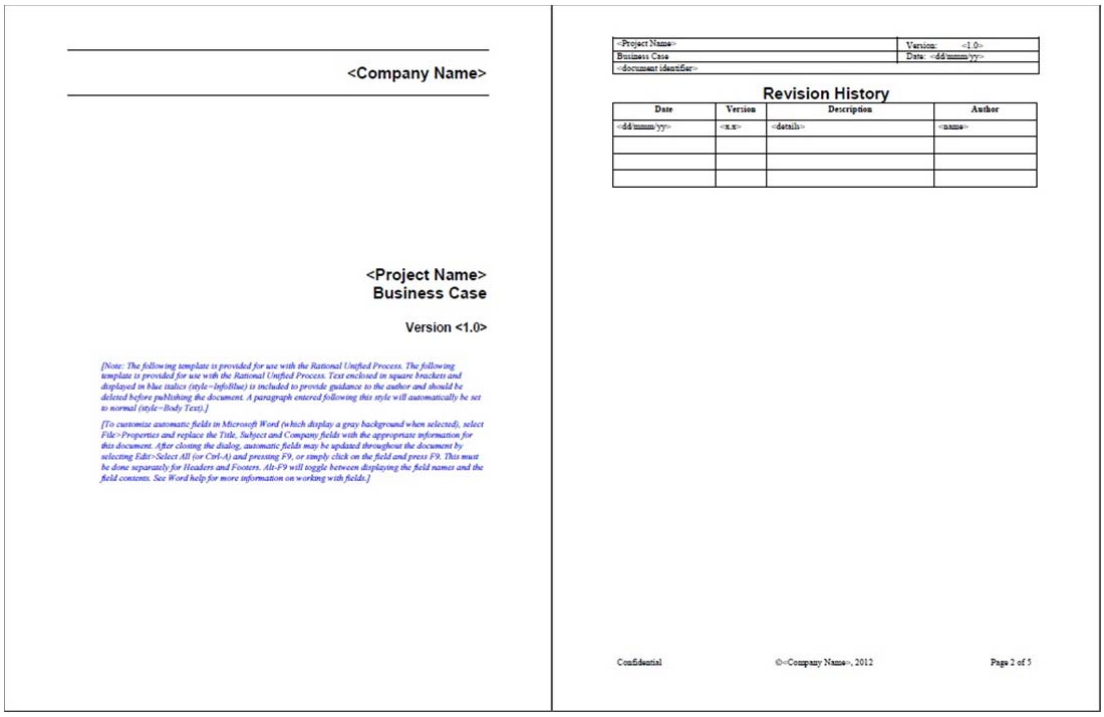
[Additional plans if required by contract or regulations.]

8. **Annexes**

*[Additional material of use to the reader of the **Software Development Plan**.]*

9. **Index**

Caso De Negocio



<Company Name>

<Project Name>
Business Case

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Business Case	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2 of 5

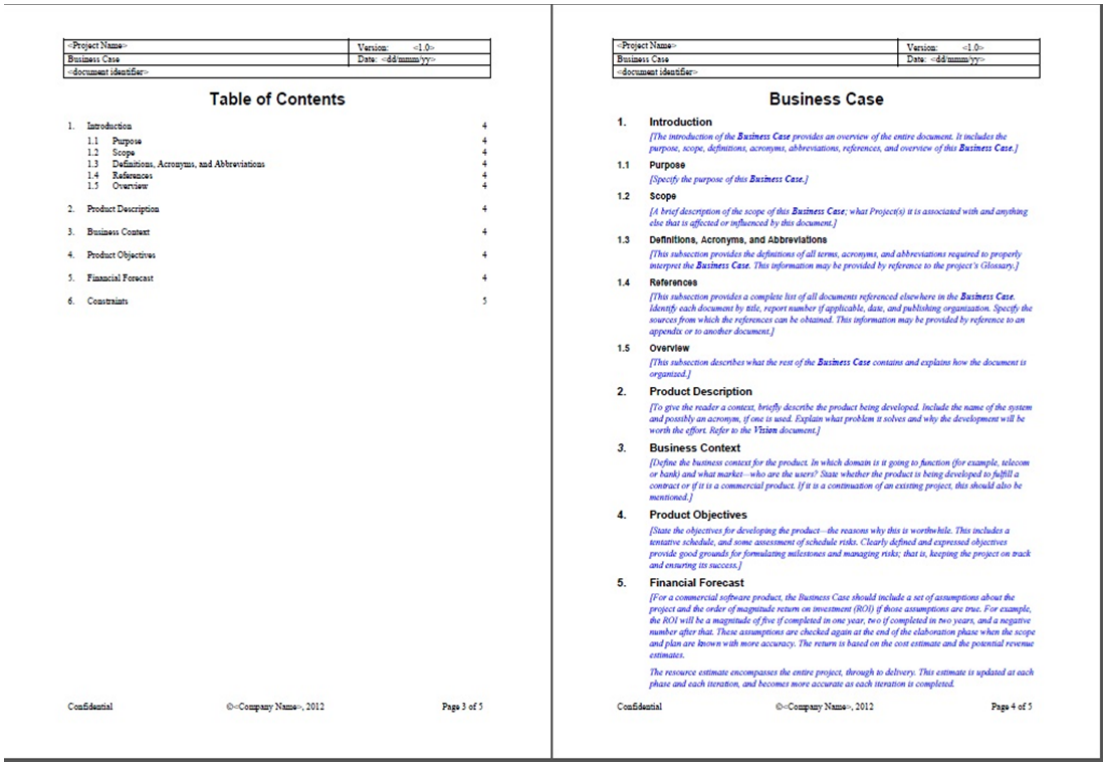


Table of Contents

Business Case

1. Introduction

*[The introduction of the **Business Case** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Business Case**.]*

1.1 Purpose

*[Specify the purpose of this **Business Case**.]*

1.2 Scope

*[A brief description of the scope of this **Business Case**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Business Case**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Business Case**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Business Case** contains and explains how the document is organized.]*

2. Product Description

*[To give the reader a context, briefly describe the product being developed. Include the name of the system and possibly an acronym, if one is used. Explain what problem it solves and why the development will be worth the effort. Refer to the **Vision** document.]*

3. Business Context

[Define the business context for the product. In which domain is it going to function (for example, telecom or bank) and what market—who are the users? State whether the product is being developed to fulfill a contract or if it is a commercial product. If it is a continuation of an existing project, this should also be mentioned.]

4. Product Objectives

[State the objectives for developing the product—the reasons why this is worthwhile. This includes a tentative schedule, and some assessment of schedule risks. Clearly defined and expressed objectives provide good grounds for formulating milestones and managing risks; that is, keeping the project on track and ensuring its success.]

5. Financial Forecast

[For a commercial software product, the Business Case should include a set of assumptions about the project and the order of magnitude return on investment (ROI) if those assumptions are true. For example, the ROI will be a magnitude of five if completed in one year, two if completed in two years, and a negative number after that. These assumptions are checked again at the end of the elaboration phase when the scope and plan are known with more accuracy. The return is based on the cost estimate and the potential revenue estimates.

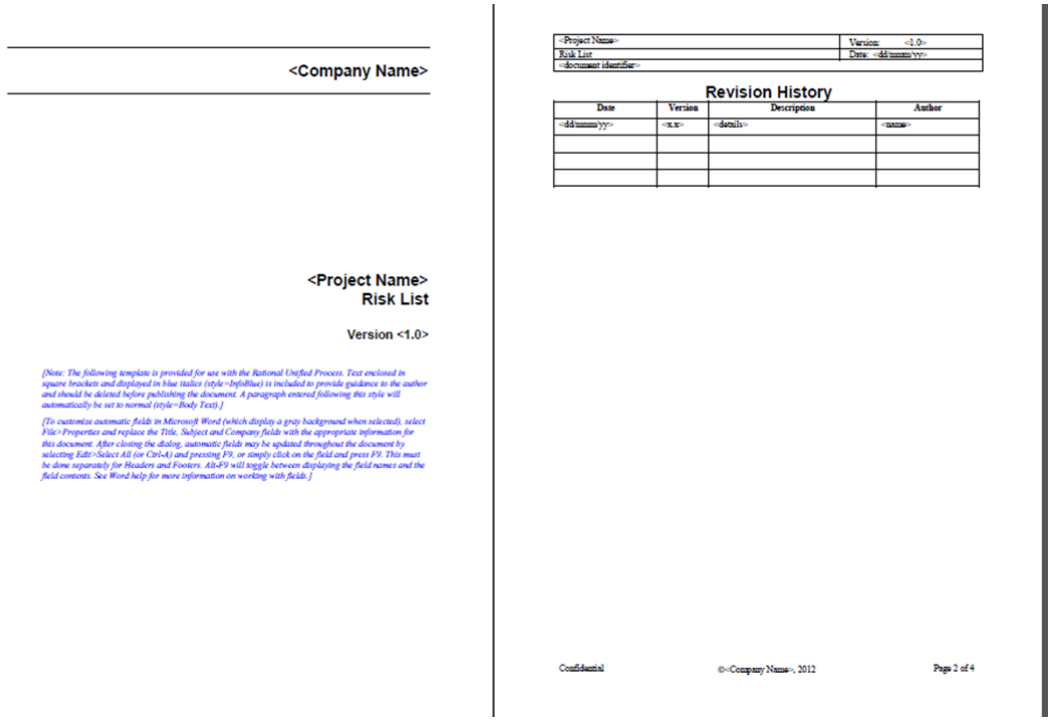
The resource estimate encompasses the entire project, through to delivery. This estimate is updated at each phase and each iteration, and becomes more accurate as each iteration is completed.

An explanation of the basis of estimates should be included.]

6. Constraints

[Express the constraints under which the project is undertaken. These constraints impact risk and cost. They could be things like external interfaces that the system must adhere to, standards, certifications or a technical approach employed for strategic reasons, such as using a certain database technology or distribution mechanism.]

Lista De Riesgos



<Company Name>

<Project Name>
Risk List

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

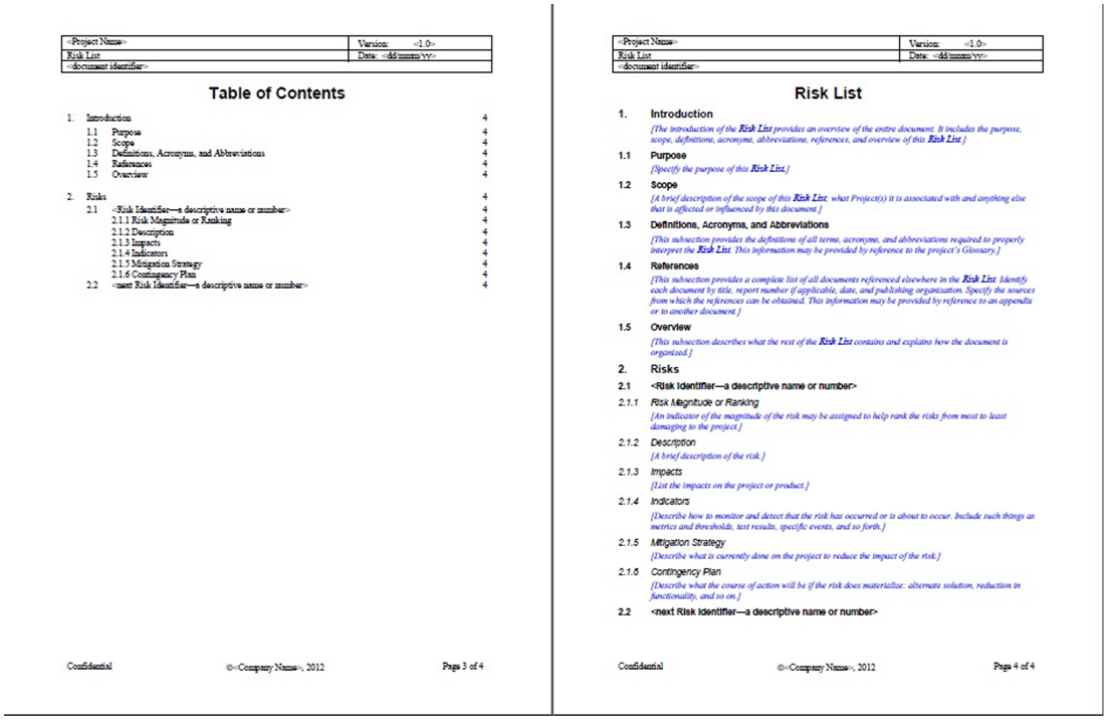
[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl+A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Risk List	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2 of 4



<Project Name>	Version: <1.0>
Risk List	Date: <dd/mmm/yy>
<document identifier>	

Table of Contents

Confidential ©<Company Name>, 2012 Page 3 of 4

<Project Name>	Version: <1.0>
Risk List	Date: <dd/mmm/yy>
<document identifier>	

Risk List

1. Introduction

*[The introduction of the **Risk List** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Risk List**.]*

1.1 Purpose

*[Specify the purpose of this **Risk List**.]*

1.2 Scope

*[A brief description of the scope of this **Risk List**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Risk List**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Risk List**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Risk List** contains and explains how the document is organized.]*

2. Risks

2.1 <Risk Identifier—a descriptive name or number>

2.1.1 Risk Magnitude or Ranking

[An indicator of the magnitude of the risk may be assigned to help rank the risks from most to least damaging to the project.]

2.1.2 Description

[A brief description of the risk.]

2.1.3 Impacts

[List the impacts on the project or product.]

2.1.4 Indicators

[Describe how to monitor and detect that the risk has occurred or is about to occur. Include such things as metrics and thresholds, test results, specific events, and so forth.]

2.1.5 Mitigation Strategy

[Describe what is currently done on the project to reduce the impact of the risk.]

2.1.6 Contingency Plan

[Describe what the course of action will be if the risk does materialize: alternate solution, reduction in functionality, and so on.]

2.2 <next Risk Identifier—a descriptive name or number>

Confidential ©<Company Name>, 2012 Page 4 of 4

REQUISITOS

Modelo De Casos De Uso

<Company Name>

<Project Name>
Use-Case Specification: <Use-Case Name>

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Use-Case Specification: <Use-Case Name>

[The following template is provided for a Use-Case Specification, which contains the textual properties of the use case. This document is used with a requirements management tool, such as Rational RequisitePro, for specifying and marking the requirements within the use-case properties.

The use-case diagrams can be developed in a visual modeling tool, such as Rational Rose. A use-case report, with all properties, may be generated with Rational SoDA. For more information, see the tool mentors in the Rational Unified Process.]

1. Brief Description

[The description briefly conveys the role and purpose of the use case. A single paragraph will suffice for this description.]

2. Basic Flow of Events

[This use case starts when the actor does something. An actor always initiates use cases. The use case describes what the actor does and what the system does in response. It is phrased in the form of a dialog between the actor and the system.

The use case describes what happens inside the system, but not how or why. If information is exchanged, be specific about what is passed back and forth. For example, it is not very illuminating to say that the actor enters customer information [if it is not defined. It is better to say the actor enters the customer's name and address. A Glossary of Terms (or a more formal Domain Model) is essential to keep the complexity of the use case manageable—you may want to define things like customer information there to keep the use case from drowning in details.

*Simple alternatives may be presented within the text of the flow of events. If it only takes a few sentences to describe what happens when there is an alternative, do it directly within the flow. If the alternative flow is more complex, use a separate section to describe it. For example, an **Alternative Flow** subsection explains how to describe more complex alternatives.*

Complex flow of events should be further structured into sub-flows. In doing this, the main goal should be improving the readability of the text. Subflows can be invoked many times from many places. Remember that the use case can perform subflows in optional sequences or in loops or even several at the same time..

A picture is sometimes worth a thousand words, though there is no substitute for clean, clear prose. If it improves clarity, feel free to paste flow charts, activity diagrams or other figures into the use case. If a flow chart is useful to present a complex decision process, by all means use it! Similarly for state-dependent behavior, a state-transition diagram often clarifies the behavior of a system better than pages upon pages of text. Use the right presentation medium for your problem, but be wary of using terminology, notations or figures that your audience may not understand. Remember that your purpose is to clarify, not obscure.]

3. Alternative Flows

*[More complex alternatives are described in a separate section, referred to in the **Basic Flow** subsection of **Flow of Events** section. Think of the **Alternative Flow** subsections like alternative behavior—each alternative flow represents alternative behavior usually due to exceptions that occur in the main flow. They may be as long as necessary to describe the events associated with the alternative behavior.*

Start each alternative flow with an initial line clearly stating where the alternative flow can occur and the conditions under which it is performed.

End each alternative flow with a line that clearly states where the events of the main flow of events are resumed. This must be explicitly stated.

Using alternative flows improves the readability of the use case. Keep in mind that use cases are just

textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise, and understandable way.]

3.1 <Area of Functionality>

[Often there are multiple alternative flows related to a single area of functionality (for example specialist withdrawal facilities, card handling or receipt handling for the Withdraw Cash use case of an Automated Teller Machine). It improves readability if these conceptually related sets of flows are grouped into their own clearly named sub-section.]

3.1.1 *< A1 First Alternative Flow >*

[Describe the alternative flow, just like any other flow of events.]

3.1.1.1 < An Alternative Subflow >

[Alternative flows may, in turn, be divided into subsections if it improves clarity. Only place subflows here is they are only applicable to a single alternative flow.]

3.1.2 *< A2 Second Alternative Flow >*

[There may be, and most likely will be, a number of alternative flows in each area of functionality. Keep each alternative flow separate to improve clarity.]

3.2 <Another Area of Functionality>

[There may be, and most likely will be, a number of areas of functionality giving rise to sets of alternative flows. Keep each set of alternative flow separate to improve clarity.]

3.2.1 *< AN Another Alternative Flow >*

4. Subflows

4.1 <S1 First Subflow >

A subflow should be a segment of behavior within the use case that has a clear purpose, and is "atomic" in the sense that you do either all or none of the actions described. You may need to have several levels of sub-flows, but if you can you should avoid this as it makes the text more complex and harder to understand.

4.2 < S2 Second Subflow >

[There may be, and most likely will be, a number of subflows in a use case. Keep each sub flow separate to improve clarity. Using sub flows improves the readability of the use case, as well as preventing use cases from being decomposed into hierarchies of use cases. Keep in mind that use cases are just textual descriptions, and their main purpose is to document the behavior of a system in a clear, concise, and understandable way.]

5. Key Scenarios

[List the most important scenarios of the use case. Simply provide a short name and accompanying description to uniquely identify each key scenario. There will potentially be many scenarios possible with this use-case specification; it is important to focus on the most important or frequently discussed scenario's that are either exemplars of this use case or are of concern or specific importance to the actor stakeholders.]

6. Preconditions

[A precondition of a use case is the state of the system that must be present prior to a use case being performed.]

6.1 < Precondition One >

7. Postconditions

[A postcondition of a use case is a list of possible states the system can be in immediately after a use case has finished.]

7.1 < Postcondition One >

8. Extension Points

[Extension points of the use case.]

8.1 <Name of Extension Point>

[Definition of the location of the extension point in the flow of events.]

9. Special Requirements

[A special requirement is typically a nonfunctional requirement that is specific to a use case, but is not easily or naturally specified in the text of the use case's event flow. Examples of special requirements include legal and regulatory requirements, application standards, and quality attributes of the system to be built including usability, reliability, performance or supportability requirements. Additionally, other requirements—such as operating systems and environments, compatibility requirements, and design constraints—should be captured in this section.]

9.1 < First Special Requirement >

10. Additional Information

[Include, or provide references to, any additional information required to clarify the use case. This could include overview diagrams, examples or any thing else you fancy.]

Visión

<Company Name>

<Project Name>
Vision

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Vision	Date: <dd/mmm/yy>
<document identifier>	

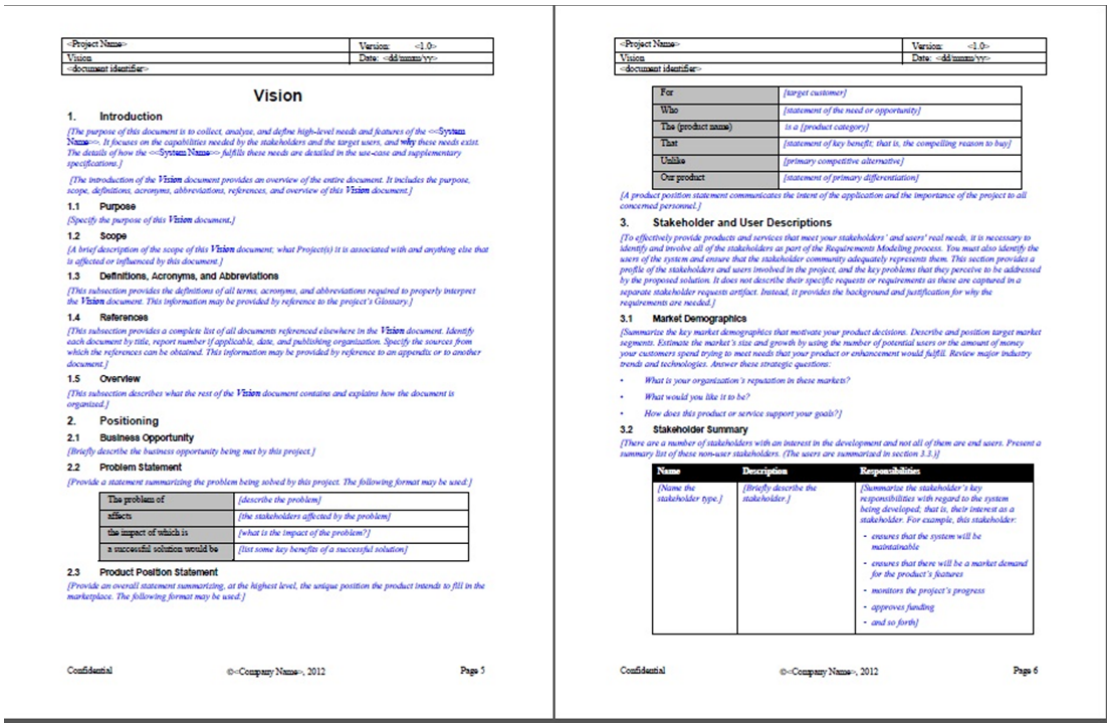
Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

<Project Name>	Version: <1.0>
Vision	Date: <dd/mmm/yy>
<document identifier>	

Table of Contents

<Project Name>	Version: <1.0>
Vision	Date: <dd/mmm/yy>
<document identifier>	



Vision

1. Introduction

*[The purpose of this document is to collect, analyze, and define high-level needs and features of the <<System Name>>. It focuses on the capabilities needed by the stakeholders and the target users, and **why** these needs exist. The details of how the <<System Name>> fulfills these needs are detailed in the use-case and supplementary specifications.]*

*[The introduction of the **Vision** document provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Vision** document.]*

1.1 Purpose

*[Specify the purpose of this **Vision** document.]*

1.2 Scope

*[A brief description of the scope of this **Vision** document; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Vision** document. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Vision** document. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Vision** document contains and explains how the document is organized.]*

2. Positioning

2.1 Business Opportunity

[Briefly describe the business opportunity being met by this project.]

2.2 Problem Statement

[Provide a statement summarizing the problem being solved by this project. The following format may be used.]

The problem of	*[describe the problem]*
affects	*[the stakeholders affected by the problem]*
the impact of which is	*[what is the impact of the problem?]*
a successful solution would be	*[list some key benefits of a successful solution]*

2.3 Product Position Statement

[Provide an overall statement summarizing, at the highest level, the unique position the product intends to fill in the marketplace. The following format may be used:]

For	*[target customer]*
Who	*[statement of the need or opportunity]*
The (product name)	*is a [product category]*
That	*[statement of key benefit; that is, the compelling reason to buy]*
Unlike	*[primary competitive alternative]*
Our product	*[statement of primary differentiation]*

[A product position statement communicates the intent of the application and the importance of the project to all concerned personnel.]

3. Stakeholder and User Descriptions

[To effectively provide products and services that meet your stakeholders' and users' real needs, it is necessary to identify and involve all of the stakeholders as part of the Requirements Modeling process. You must also identify the users of the system and ensure that the stakeholder community adequately represents them. This section provides a profile of the stakeholders and users involved in the project, and the key problems that they perceive to be addressed by the proposed solution. It does not describe their specific requests or requirements as these are captured in a separate stakeholder requests artifact. Instead, it provides the background and justification for why the requirements are needed.]

3.1 Market Demographics

[Summarize the key market demographics that motivate your product decisions. Describe and position target market segments. Estimate the market's size and growth by using the number of potential users or the amount of money your customers spend trying to meet needs that your product or enhancement would fulfill. Review major industry trends and technologies. Answer these strategic questions:

- *What is your organization's reputation in these markets?*
- *What would you like it to be?*
- *How does this product or service support your goals?]*

3.2 Stakeholder Summary

[There are a number of stakeholders with an interest in the development and not all of them are end users. Present a summary list of these non-user stakeholders. (The users are summarized in section 3.3.)]

Name	Description	Responsibilities
[Name the stakeholder type.]	*[Briefly describe the stakeholder.]*	*[Summarize the stakeholder's key responsibilities with regard to the system being developed; that is, their interest as a stakeholder. For example, this stakeholder* • *ensures that the system will be maintainable* • *ensures that there will be a market demand for the product's features* • *monitors the project's progress* • *approves funding* • *and so forth]*

3.3 User Summary

[Present a summary list of all identified users.]

Name	Description	Responsibilities	Stakeholder
[Name the user type.]	*[Briefly describe what they represent with respect to the system.]*	*[List the user's key responsibilities with regard to the system being developed; for example:* - *captures details* - *produces reports* - *coordinates work* - *and so on]*	*[If the user is not directly represented, identify which stakeholder is responsible for representing the user's interest.]*

3.4 User Environment

[Detail the working environment of the target user. Here are some suggestions:

- *Number of people involved in completing the task? Is this changing?*
- *How long is a task cycle? Amount of time spent in each activity? Is this changing?*
- *Any unique environmental constraints: mobile, outdoors, in-flight, and so on?*
- *Which systems platforms are in use today? Future platforms?*
- *What other applications are in use? Does your application need to integrate with them?*

This is where extracts from the Business Model could be included to outline the task and roles involved and so on.]

3.5 Stakeholder Profiles

[Describe each stakeholder in the system here by filling in the following table for each stakeholder. Remember that stakeholder types can be as divergent as users, departments, and technical developers. A thorough profile would cover the following topics for each type of stakeholder.]

3.5.1 <Stakeholder Name>

Representative	*[Who is the stakeholder representative to the project? (Optional if documented elsewhere.) What we want here is names.]*
Description	*[A brief description of the stakeholder type.]*
Type	*[Qualify the stakeholder's expertise, technical background, and degree of sophistication—that is, guru, business, expert, casual user, and so on.]*
Responsibilities	*[List the stakeholder's key responsibilities with regard to the system being developed—that is, their interest as a stakeholder.]*
Success Criteria	*[How does the stakeholder define success?* *How is the stakeholder rewarded?]*
Involvement	*[How is the stakeholder involved in the project? Relate where possible to Rational Unified Process roles—that is, Requirements Reviewer and so on.]*
Deliverables	*[Are there any additional deliverables required by the stakeholder? These could be project deliverables or outputs from the system under development.]*
Comments / Issues	*[Problems that interfere with success and any other relevant information go here.]*

3.6 User Profiles

[Describe each unique user of the system here by filling in the following table for each user type. Remember user types can be as divergent as gurus and novices. For example, a guru might need a sophisticated, flexible tool with cross-platform support, while a novice might need a tool that is easy to use and user-friendly. A thorough profile needs to cover the following topics for each type of user.]

3.6.1 <User Name>

Representative	*[Who is the user representative to the project? (Optional if documented elsewhere.) This often refers to the Stakeholder that represents the set of users, for example, Stakeholder: Stakeholder1.]*
Description	*[A brief description of the user type.]*
Type	*[Qualify the user's expertise, technical background, and degree of sophistication—that is, guru, casual user, and so on.]*
Responsibilities	*[List the user's key responsibilities with regard to the system being developed—that is, captures details, produces reports, coordinates work, and so forth.]*
Success Criteria	*[How does the user define success?* *How is the user rewarded?]*
Involvement	*[How is the user involved in the project? Relate where possible to Rational Unified Process roles—that is, Requirements Reviewer, and so on.]*
Deliverables	*[Are there any deliverables the user produces and, if so, for whom?]*
Comments / Issues	*[Problems that interfere with success and any other relevant information go here. These would include trends that make the user's job easier or harder.]*

3.7 Key Stakeholder or User Needs

[List the key problems with existing solutions as perceived by the stakeholder or user. Clarify the following issues for each problem:

- *What are the reasons for this problem?*
- *How is it solved now?*
- *What solutions does the stakeholder or user want?]*

*[It is important to understand the **relative** importance the stakeholder or user places on solving each problem. Ranking and cumulative voting techniques indicate problems that **must** be solved versus issues they would like addressed.*

Fill in the following table—if using Rational RequisitePro to capture the Needs, this could be an extract or report from that tool.]

Need	Priority	Concerns	Current Solution	Proposed Solutions
Broadcast messages				

3.8 Alternatives and Competition

[Identify alternatives the stakeholder perceives as available. These can include buying a competitor's product,

building a homegrown solution or simply maintaining the status quo. List any known competitive choices that exist or may become available. Include the major strengths and weaknesses of each competitor as perceived by the stakeholder or end user.]

3.8.1 <aCompetitor>

3.8.2 <anotherCompetitor>

4. Product Overview

[This section provides a high level view of the product capabilities, interfaces to other applications, and system configurations. This section usually consists of three subsections, as follows:

- *Product perspective*
- *Product functions*
- *Assumptions and dependencies]*

4.1 Product Perspective

*[This subsection of the **Vision** document puts the product in perspective to other related products and the user's environment. If the product is independent and totally self-contained, state it here. If the product is a component of a larger system, then this subsection needs to relate how these systems interact and needs to identify the relevant interfaces between the systems. One easy way to display the major components of the larger system, interconnections, and external interfaces is with a block diagram.]*

4.2 Summary of Capabilities

*[Summarize the major benefits and features the product will provide. For example, a **Vision** document for a customer support system may use this part to address problem documentation, routing, and status reporting without mentioning the amount of detail each of these functions requires.*

Organize the functions so the list is understandable to the customer or to anyone else reading the document for the first time. A simple table listing the key benefits and their supporting features might suffice. For example:]

Table 4-1 Customer Support System

Customer Benefit	Supporting Features
New support staff can quickly get up to speed.	Knowledge base assists support personnel in quickly identifying known fixes and workarounds.
Customer satisfaction is improved because nothing falls through the cracks.	Problems are uniquely numbered, classified and tracked throughout the resolution process. Automatic notification occurs for any aging issues.
Management can identify problem areas and ensure staff workload.	Trend and distribution reports allow high level review of problem status.
Distributed support teams can work together to solve problems.	Replication server allows current database information to be shared across the enterprise.
Customers can help themselves, lowering support costs and improving response time.	Knowledge base can be made available over the Internet. Includes hypertext search capabilities and graphical query engine.

4.3 Assumptions and Dependencies

*[List each of the factors that affect the features stated in the **Vision** document. List assumptions that, if changed, will alter the **Vision** document. For example, an assumption may state that a specific operating system will be available for the hardware designated for the software product. If the operating system is not available, the **Vision** document*

will need to change.]

4.4 Cost and Pricing

[For products sold to external customers and for many in-house applications, cost and pricing issues can directly impact the application's definition and implementation. In this section, record any cost and pricing constraints that are relevant. For example, distribution costs, (8 of diskettes, 8 of CD-ROMs, CD mastering) or other cost of goods sold constraints (manuals, packaging) may be material to the projects success, or irrelevant, depending on the nature of the application.]

4.5 Licensing and Installation

[Licensing and installation issues can also directly impact the development effort. For example, the need to support serializing, password security or network licensing will create additional requirements of the system that must be considered in the development effort.

Installation requirements may also affect coding or create the need for separate installation software.]

5. Product Features

[List and briefly describe the product features. Features are the high-level capabilities of the system that are necessary to deliver benefits to the users. Each feature is an externally desired service that typically requires a series of inputs to achieve the desired result. For example, a feature of a problem tracking system might be the ability to provide trending reports. As the use-case model takes shape, update the description to refer to the use cases.

*Because the **Vision** document is reviewed by a wide variety of involved personnel, the level of detail needs to be general enough for everyone to understand. However, enough detail must be available to provide the team with the information they need to create a use-case model.*

To effectively manage application complexity, we recommend for any new system, or an increment to an existing system, capabilities are abstracted to a high enough level so 25-99 features result. These features provide the fundamental basis for product definition, scope management, and project management. Each feature will be expanded in greater detail in the use-case model.

Throughout this section, each feature will be externally perceivable by users, operators or other external systems. These features need to include a description of functionality and any relevant usability issues that must be addressed. The following guidelines apply:

- *Avoid design. Keep feature descriptions at a general level. Focus on capabilities needed and why (not how) they should be implemented.*
- *If you are using the Rational RequisitePro toolkit, all need to be selected as requirements of type for easy reference and tracking.]*

5.1 <aFeature>

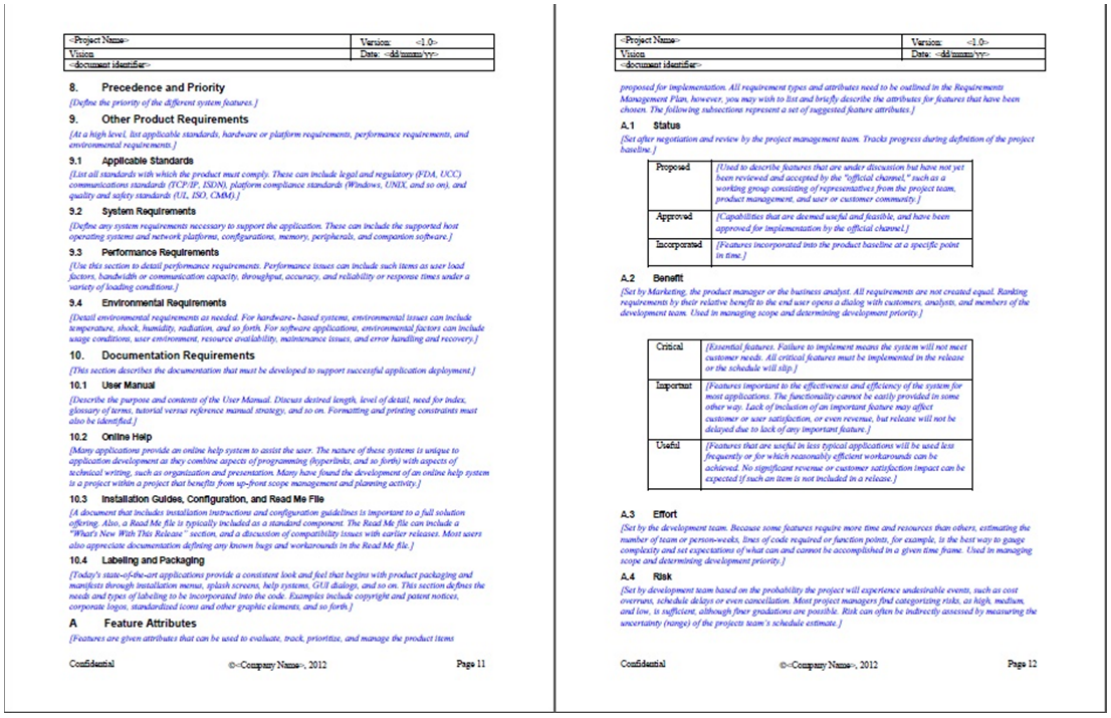
5.2 <anotherFeature>

6. Constraints

[Note any design constraints, external constraints or other dependencies.]

7. Quality Ranges

[Define the quality ranges for performance, robustness, fault tolerance, usability, and similar characteristics that are not captured in the Feature Set.]



8. Precedence and Priority

[Define the priority of the different system features.]

9. Other Product Requirements

[At a high level, list applicable standards, hardware or platform requirements, performance requirements, and environmental requirements.]

9.1 Applicable Standards

[List all standards with which the product must comply. These can include legal and regulatory (FDA, UCC) communications standards (TCP/IP, ISDN), platform compliance standards (Windows, UNIX, and so on), and quality and safety standards (UL, ISO, CMM).]

9.2 System Requirements

[Define any system requirements necessary to support the application. These can include the supported host operating systems and network platforms, configurations, memory, peripherals, and companion software.]

9.3 Performance Requirements

[Use this section to detail performance requirements. Performance issues can include such items as user load factors, bandwidth or communication capacity, throughput, accuracy, and reliability or response times under a variety of loading conditions.]

9.4 Environmental Requirements

[Detail environmental requirements as needed. For hardware- based systems, environmental issues can include temperature, shock, humidity, radiation, and so forth. For software applications, environmental factors can include usage conditions, user environment, resource availability, maintenance issues, and error handling and recovery.]

10. Documentation Requirements

[This section describes the documentation that must be developed to support successful application deployment.]

10.1 User Manual

[Describe the purpose and contents of the User Manual. Discuss desired length, level of detail, need for index, glossary of terms, tutorial versus reference manual strategy, and so on. Formatting and printing constraints must also be identified.]

10.2 Online Help

[Many applications provide an online help system to assist the user. The nature of these systems is unique to application development as they combine aspects of programming (hyperlinks, and so forth) with aspects of technical writing, such as organization and presentation. Many have found the development of an online help system is a project within a project that benefits from up-front scope management and planning activity.]

10.3 Installation Guides, Configuration, and Read Me File

[A document that includes installation instructions and configuration guidelines is important to a full solution offering. Also, a Read Me file is typically included as a standard component. The Read Me file can include a "What's New With This Release" section, and a discussion of compatibility issues with earlier releases. Most users also appreciate documentation defining any known bugs and workarounds in the Read Me file.]

10.4 Labeling and Packaging

[Today's state-of-the-art applications provide a consistent look and feel that begins with product packaging and manifests through installation menus, splash screens, help systems, GUI dialogs, and so on. This section defines the needs and types of labeling to be incorporated into the code. Examples include copyright and patent notices, corporate logos, standardized icons and other graphic elements, and so forth.]

A Feature Attributes

[Features are given attributes that can be used to evaluate, track, prioritize, and manage the product items

proposed for implementation. All requirement types and attributes need to be outlined in the Requirements Management Plan, however, you may wish to list and briefly describe the attributes for features that have been chosen. The following subsections represent a set of suggested feature attributes.]

A.1 Status

[Set after negotiation and review by the project management team. Tracks progress during definition of the project baseline.]

Proposed	*[Used to describe features that are under discussion but have not yet been reviewed and accepted by the "official channel," such as a working group consisting of representatives from the project team, product management, and user or customer community.]*
Approved	*[Capabilities that are deemed useful and feasible, and have been approved for implementation by the official channel.]*
Incorporated	*[Features incorporated into the product baseline at a specific point in time.]*

A.2 Benefit

[Set by Marketing, the product manager or the business analyst. All requirements are not created equal. Ranking requirements by their relative benefit to the end user opens a dialog with customers, analysts, and members of the development team. Used in managing scope and determining development priority.]

Critical	*[Essential features. Failure to implement means the system will not meet customer needs. All critical features must be implemented in the release or the schedule will slip.]*
Important	*[Features important to the effectiveness and efficiency of the system for most applications. The functionality cannot be easily provided in some other way. Lack of inclusion of an important feature may affect customer or user satisfaction, or even revenue, but release will not be delayed due to lack of any important feature.]*
Useful	*[Features that are useful in less typical applications will be used less frequently or for which reasonably efficient workarounds can be achieved. No significant revenue or customer satisfaction impact can be expected if such an item is not included in a release.]*

A.3 Effort

[Set by the development team. Because some features require more time and resources than others, estimating the number of team or person-weeks, lines of code required or function points, for example, is the best way to gauge complexity and set expectations of what can and cannot be accomplished in a given time frame. Used in managing scope and determining development priority.]

A.4 Risk

[Set by development team based on the probability the project will experience undesirable events, such as cost overruns, schedule delays or even cancellation. Most project managers find categorizing risks, as high, medium, and low, is sufficient, although finer gradations are possible. Risk can often be indirectly assessed by measuring the uncertainty (range) of the project team's schedule estimate.]

<Project Name>	Version: <1.0>
Vision	Date: <dd/mmm/yy>
<document identifier>	

A.5 Stability

[Set by the analyst and development team, this is based on the probability that features will change or the team's understanding of the feature will change. Used to help establish development priorities and determine those items for which additional elicitation is the appropriate next action.]

A.6 Target Release

*[Records the intended product version in which the feature will first appear. This field can be used to allocate features from a **Vision** document into a particular baseline release. When combined with the status field, your team can propose, record, and discuss various features of the release without committing them to development. Only features whose Status is set to Incorporated and whose Target Release is defined will be implemented. When scope management occurs, the Target Release Version Number can be increased so the item will remain in the **Vision** document but will be scheduled for a later release.]*

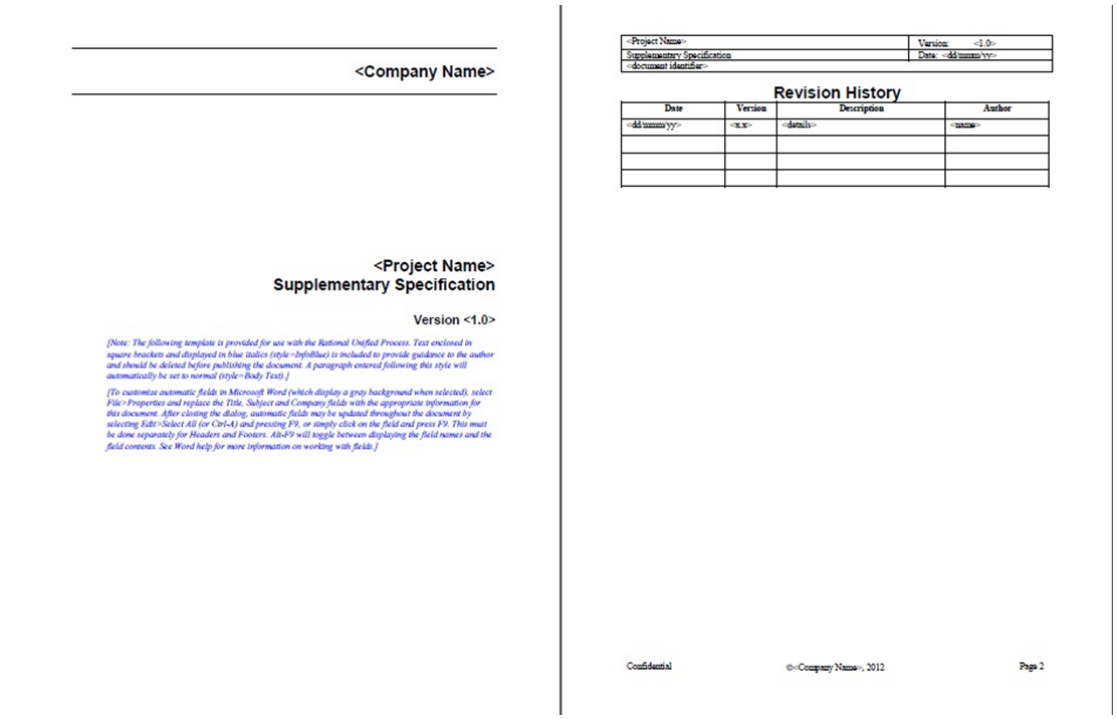
A.7 Assigned To

[In many projects, features will be assigned to "feature teams" responsible for further elicitation, writing the software requirements, and implementation. This simple pull-down list will help everyone on the project team to understand responsibilities better.]

A.8 Reason

[This text field is used to track the source of the requested feature. Requirements exist for specific reasons. This field records an explanation or a reference to an explanation. For example, the reference might be to a page and line number of a product requirement specification or to a minute marker on a video of an important customer review.]

Especificación Adicional



<Company Name>

<Project Name>
Supplementary Specification

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

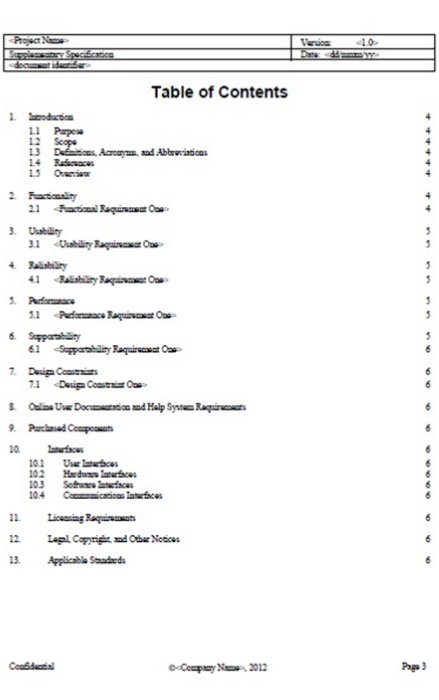
[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Supplementary Specification	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

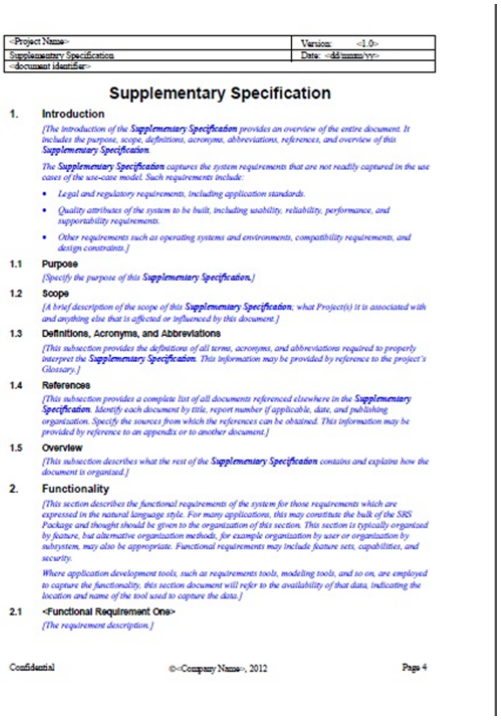
Confidential ©<Company Name>, 2012 Page 2



<Project Name>	Version: <1.0>
Supplementary Specification	Date: <dd/mmm/yy>
<document identifier>	

Table of Contents

Confidential ©<Company Name>, 2012 Page 3



<Project Name>	Version: <1.0>
Supplementary Specification	Date: <dd/mmm/yy>
<document identifier>	

Supplementary Specification

1. Introduction

*[The Introduction of the **Supplementary Specification** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Supplementary Specification**.*

*The **Supplementary Specification** captures the system requirements that are not readily captured in the use cases of the use-case model. Such requirements include:*

- *Legal and regulatory requirements, including application standards.*
- *Quality attributes of the system to be built, including usability, reliability, performance, and supportability requirements.*
- *Other requirements such as operating systems and environments, compatibility requirements, and design constraints.]*

1.1 Purpose

*[Specify the purpose of this **Supplementary Specification.**]*

1.2 Scope

*[A brief description of the scope of this **Supplementary Specification**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Supplementary Specification**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Supplementary Specification**. Identify each document by title, report number if applicable, date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Supplementary Specification** contains and explains how the document is organized.]*

2. Functionality

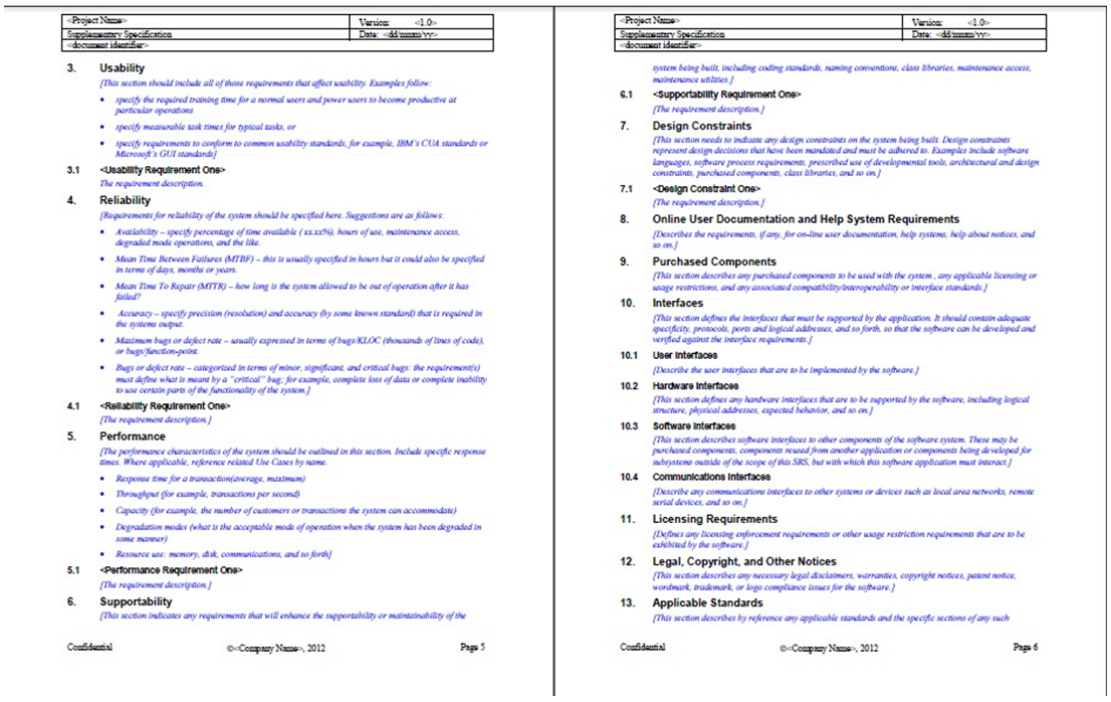
[This section describes the functional requirements of the system for those requirements which are expressed in the natural language style. For many applications, this may constitute the bulk of the SRS Package and thought should be given to the organization of this section. This section is typically organized by feature, but alternative organization methods, for example organization by user or organization by subsystem, may also be appropriate. Functional requirements may include feature sets, capabilities, and security.

Where application development tools, such as requirements tools, modeling tools, and so on, are employed to capture the functionality, this section document will refer to the availability of that data, indicating the location and name of the tool used to capture the data.]

2.1 <Functional Requirement One>

[The requirement description.]

Confidential ©<Company Name>, 2012 Page 4



3. Usability

[This section should include all of those requirements that affect usability. Examples follow:

- *specify the required training time for a normal users and power users to become productive at particular operations*
- *specify measurable task times for typical tasks, or*
- *specify requirements to conform to common usability standards, for example, IBM's CUA standards or Microsoft's GUI standards]*

3.1 <Usability Requirement One>

The requirement description.

4. Reliability

[Requirements for reliability of the system should be specified here. Suggestions are as follows:

- *Availability – specify percentage of time available (xx.xx%), hours of use, maintenance access, degraded mode operations, and the like.*
- *Mean Time Between Failures (MTBF) – this is usually specified in hours but it could also be specified in terms of days, months or years.*
- *Mean Time To Repair (MTTR) – how long is the system allowed to be out of operation after it has failed?*
- *Accuracy – specify precision (resolution) and accuracy (by some known standard) that is required in the systems output.*
- *Maximum bugs or defect rate – usually expressed in terms of bugs/KLOC (thousands of lines of code), or bugs/function-point.*
- *Bugs or defect rate – categorized in terms of minor, significant, and critical bugs: the requirement(s) must define what is meant by a "critical" bug; for example, complete loss of data or complete inability to use certain parts of the functionality of the system.]*

4.1 <Reliability Requirement One>

[The requirement description.]

5. Performance

[The performance characteristics of the system should be outlined in this section. Include specific response times. Where applicable, reference related Use Cases by name.

- *Response time for a transaction(average, maximum)*
- *Throughput (for example, transactions per second)*
- *Capacity (for example, the number of customers or transactions the system can accommodate)*
- *Degradation modes (what is the acceptable mode of operation when the system has been degraded in some manner)*
- *Resource use: memory, disk, communications, and so forth]*

5.1 <Performance Requirement One>

[The requirement description.]

6. Supportability

[This section indicates any requirements that will enhance the supportability or maintainability of the

system being built, including coding standards, naming conventions, class libraries, maintenance access, maintenance utilities.]

6.1 <Supportability Requirement One>

[The requirement description.]

7. Design Constraints

[This section needs to indicate any design constraints on the system being built. Design constraints represent design decisions that have been mandated and must be adhered to. Examples include software languages, software process requirements, prescribed use of developmental tools, architectural and design constraints, purchased components, class libraries, and so on.]

7.1 <Design Constraint One>

[The requirement description.]

8. Online User Documentation and Help System Requirements

[Describes the requirements, if any, for on-line user documentation, help systems, help about notices, and so on.]

9. Purchased Components

[This section describes any purchased components to be used with the system , any applicable licensing or usage restrictions, and any associated compatibility/interoperability or interface standards.]

10. Interfaces

[This section defines the interfaces that must be supported by the application. It should contain adequate specificity, protocols, ports and logical addresses, and so forth, so that the software can be developed and verified against the interface requirements.]

10.1 User Interfaces

[Describe the user interfaces that are to be implemented by the software.]

10.2 Hardware Interfaces

[This section defines any hardware interfaces that are to be supported by the software, including logical structure, physical addresses, expected behavior, and so on.]

10.3 Software Interfaces

[This section describes software interfaces to other components of the software system. These may be purchased components, components reused from another application or components being developed for subsystems outside of the scope of this SRS, but with which this software application must interact.]

10.4 Communications Interfaces

[Describe any communications interfaces to other systems or devices such as local area networks, remote serial devices, and so on.]

11. Licensing Requirements

[Defines any licensing enforcement requirements or other usage restriction requirements that are to be exhibited by the software.]

12. Legal, Copyright, and Other Notices

[This section describes any necessary legal disclaimers, warranties, copyright notices, patent notice, wordmark, trademark, or logo compliance issues for the software.]

13. Applicable Standards

[This section describes by reference any applicable standards and the specific sections of any such

standards that apply to the system being described. For example, this could include legal, quality and regulatory standards, industry standards for usability, interoperability, internationalization, operating system compliance, and so forth.]

Glosario

<Company Name>

<Project Name>
Glossary

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Glossary

1. Introduction

*[The introduction of the **Glossary** provides an overview of the entire document. Present any information the reader might need to understand the document in this section. This document is used to define terminology specific to the problem domain, explaining terms that may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal data dictionary, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information. This document should be saved in a file called Glossary.]*

1.1 Purpose

*[Specify the purpose of this **Glossary**.]*

1.2 Scope

*[A brief description of the scope of this **Glossary**; what Project(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Glossary**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.4 Overview

*[This subsection describes what the rest of the **Glossary** contains and explains how the document is organized.]*

2. Definitions

[The terms defined here form the essential substance of the document. They can be defined in any order desired, but generally alphabetical order provides the greatest accessibility.]

2.1 <aTerm>

[The definition for <aTerm> is presented here. As much information as the reader needs to understand the concept should be presented.]

2.2 <anotherTerm>

The definition for <anotherTerm> is presented here. As much information as the reader needs to understand the concept should be presented.

2.3 <aGroupofTerms>

[Sometimes it is useful to organize terms into groups to improve readability. For example, if the problem domain contains terms related to both accounting and building construction (as would be the case if we were developing a system to manage construction projects), presenting the terms from the two different sub-domains might prove confusing to the reader. To solve this problem, we use groupings of terms. In presenting the grouping of terms, provide a short description that helps the reader understand what <aGroupofTerms> represents. Terms presented within the group should be organized alphabetically for easy access.]

2.3.1 <aGroupTerm>

[The definition for <aGroupTerm> is presented here. Present as much information as the reader needs to understand the concept.]

2.3.2 <anotherGroupTerm>

[The definition for <anotherGroupTerm> is presented here. Present as much information as the reader

needs to understand the concept.]

2.4 <aSecondGroupofTerms>

2.4.1 <yetAnotherGroupTerm>

[The definition for the term is presented here. Present as much information as the reader needs to understand the concept.]

2.4.2 <andAnotherGroupTerm>

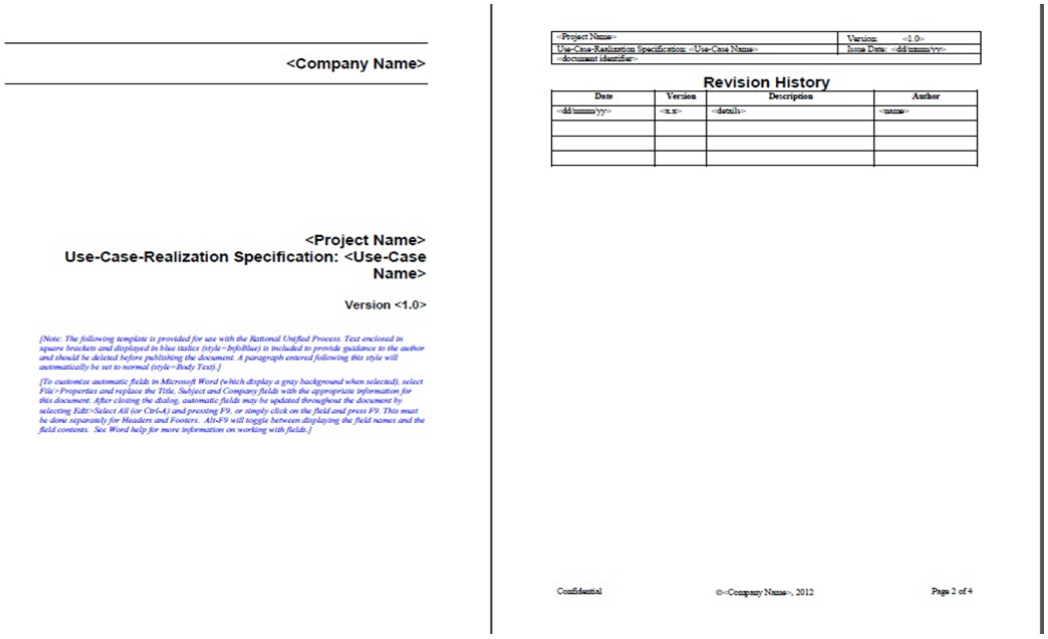
[The definition for the term is presented here. Present as much information as the reader needs to understand the concept.]

3. UML Stereotypes

[This section contains or references specifications of Unified Modeling Language (UML) stereotypes and their semantic implications—a textual description of the meaning and significance of the stereotype and any limitations on its use—for stereotypes already known or discovered to be important for the system being modeled. The use of these stereotypes may be simply recommended or perhaps even made mandatory, for example, when their use is required by an imposed standard or when it is felt that their use makes models significantly easier to understand. This section may be empty if no additional stereotypes, other than those predefined by the UML and the Rational Unified Process, are considered necessary.]

ANALISIS Y DISEÑO

Modelo De Diseño



<Company Name>

<Project Name>
Use-Case-Realization Specification: <Use-Case Name>

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

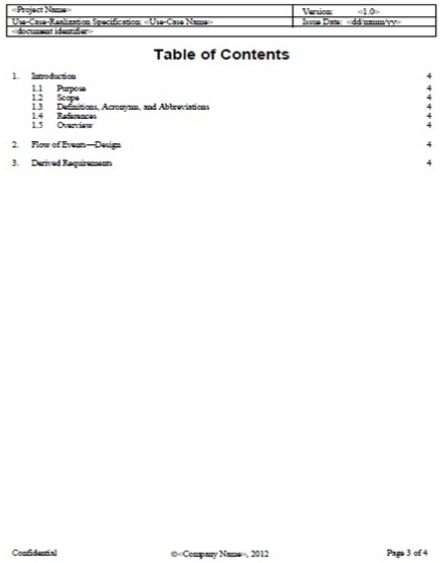
[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Use-Case-Realization Specification: <Use-Case Name>	Issue Date: <dd/mmm/yy>
<document identifier>	

Revision History

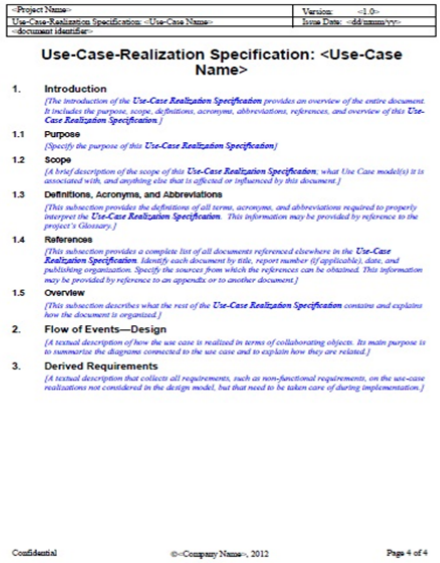
Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2 of 4



<Project Name>	Version: <1.0>
Use-Case-Realization Specification: <Use-Case Name>	Issue Date: <dd/mmm/yy>
<document identifier>	

Table of Contents



<Project Name>	Version: <1.0>
Use-Case-Realization Specification: <Use-Case Name>	Issue Date: <dd/mmm/yy>
<document identifier>	

Use-Case-Realization Specification: <Use-Case Name>

1. Introduction

*[The introduction of the **Use-Case Realization Specification** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Use-Case Realization Specification**.]*

1.1 Purpose

*[Specify the purpose of this **Use-Case Realization Specification**.]*

1.2 Scope

*[A brief description of the scope of this **Use-Case Realization Specification**; what Use Case model(s) it is associated with, and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Use-Case Realization Specification**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Use-Case Realization Specification**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Use-Case Realization Specification** contains and explains how the document is organized.]*

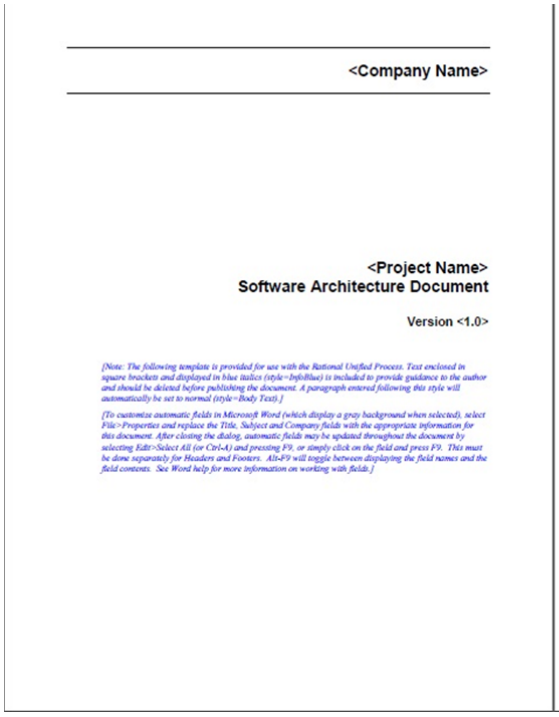
2. Flow of Events—Design

[A textual description of how the use case is realized in terms of collaborating objects. Its main purpose is to summarize the diagrams connected to the use case and to explain how they are related.]

3. Derived Requirements

[A textual description that collects all requirements, such as non-functional requirements, on the use-case realizations not considered in the design model, but that need to be taken care of during implementation.]

Documentación De La Arquitectura De Software



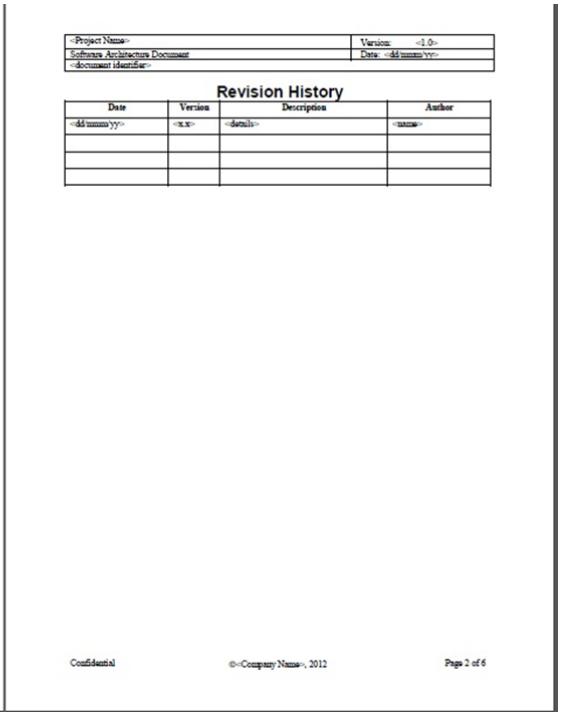
<Company Name>

<Project Name>
Software Architecture Document

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]



<Project Name>	Version: <1.0>
Software Architecture Document	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Software Architecture Document

1. Introduction

*[The introduction of the **Software Architecture Document** provides an overview of the entire **Software Architecture Document**. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of the **Software Architecture Document**.]*

1.1 Purpose

This document provides a comprehensive architectural overview of the system, using a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

*[This section defines the role or purpose of the **Software Architecture Document** in the overall project documentation, and briefly describes the structure of the document. The specific audiences for the document is identified, with an indication of how they are expected to use the document.]*

1.2 Scope

[A brief description of what the Software Architecture Document applies to; what is affected or influenced by this document.]

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Software Architecture Document**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Software Architecture Document**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Software Architecture Document** contains and explains how the **Software Architecture Document** is organized.]*

2. Architectural Representation

*[This section describes what software architecture is for the current system, and how it is represented. Of the **Use-Case**, **Logical**, **Process**, **Deployment**, and **Implementation Views**, it enumerates the views that are necessary, and for each view, explains what types of model elements it contains.]*

3. Architectural Goals and Constraints

[This section describes the software requirements and objectives that have some significant impact on the architecture; for example, safety, security, privacy, use of an off-the-shelf product, portability, distribution, and reuse. It also captures the special constraints that may apply: design and implementation strategy, development tools, team structure, schedule, legacy code, and so on.]

4. Use-Case View

[This section lists use cases or scenarios from the use-case model if they represent some significant, central functionality of the final system, or if they have a large architectural coverage—they exercise many architectural elements or if they stress or illustrate a specific, delicate point of the architecture.]

5. Logical View

[This section describes the architecturally significant parts of the design model, such as its decomposition

into subsystems and packages. And for each significant package, its decomposition into classes and class utilities. You should introduce architecturally significant classes and describe their responsibilities, as well as a few very important relationships, operations, and attributes.]

5.1 Overview

[This subsection describes the overall decomposition of the design model in terms of its package hierarchy and layers.]

5.2 Architecturally Significant Design Packages

[For each significant package, include a subsection with its name, its brief description, and a diagram with all significant classes and packages contained within the package.

For each significant class in the package, include its name, brief description, and, optionally, a description of some of its major responsibilities, operations, and attributes.]

5.3 Use-Case Realizations

[This section illustrates how the software actually works by giving a few selected use-case (or scenario) realizations, and explains how the various design model elements contribute to their functionality.]

6. Process View

[This section describes the system's decomposition into lightweight processes (single threads of control) and heavyweight processes (groupings of lightweight processes). Organize the section by groups of processes that communicate or interact. Describe the main modes of communication between processes, such as message passing, interrupts, and rendezvous.]

7. Deployment View

*[This section describes one or more physical network (hardware) configurations on which the software is deployed and run. It is a view of the Deployment Model. At a minimum for each configuration it should indicate the physical nodes (computers, CPUs) that execute the software and their interconnections (bus, LAN, point-to-point, and so on.) Also include a mapping of the processes of the **Process View** onto the physical nodes.]*

8. Implementation View

[This section describes the overall structure of the implementation model, the decomposition of the software into layers and subsystems in the implementation model, and any architecturally significant components.]

8.1 Overview

[This subsection names and defines the various layers and their contents, the rules that govern the inclusion to a given layer, and the boundaries between layers. Include a component diagram that shows the relations between layers.]

8.2 Layers

[For each layer, include a subsection with its name, an enumeration of the subsystems located in the layer, and a component diagram.]

9. Data View (optional)

[A description of the persistent data storage perspective of the system. This section is optional if there is little or no persistent data, or the translation between the Design Model and the Data Model is trivial.]

10. Size and Performance

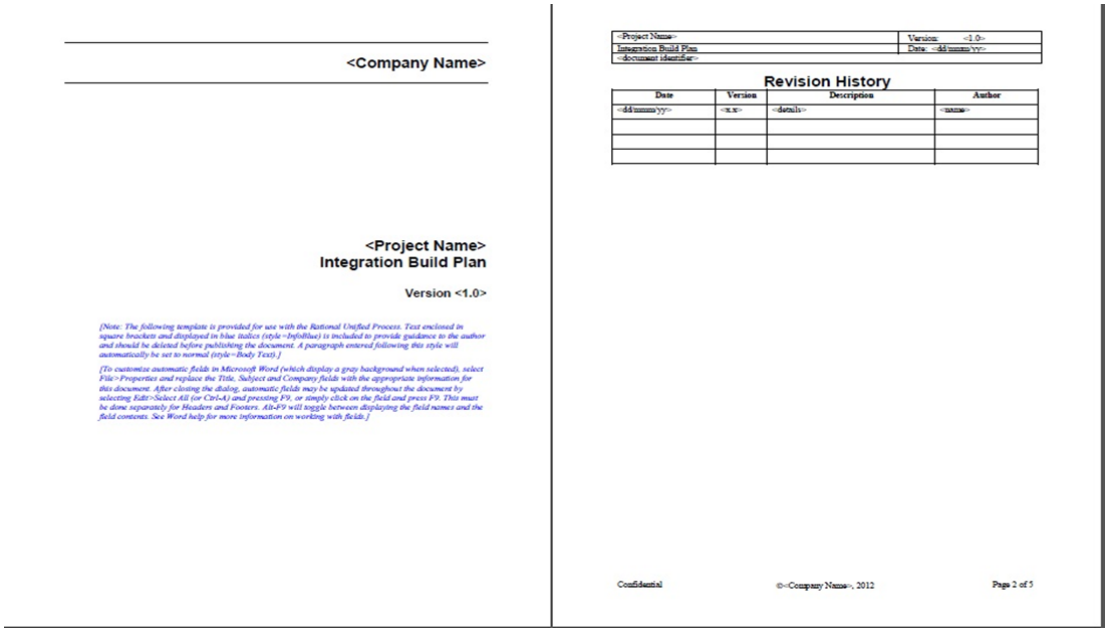
[A description of the major dimensioning characteristics of the software that impact the architecture, as well as the target performance constraints.]

11. Quality

[A description of how the software architecture contributes to all capabilities (other than functionality) of the system: extensibility, reliability, portability, and so on. If these characteristics have special significance, such as safety, security or privacy implications, they must be clearly delineated.]

IMPLEMENTACIÓN

Modelo De La Implementación



<Company Name>

<Project Name>
Integration Build Plan

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Integration Build Plan	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2 of 5

Table of Contents

Integration Build Plan

1. Introduction

*[The Introduction of the **Integration Build Plan** provides an overview of the entire document. It includes the purpose, scope, definitions, acronyms, abbreviations, references, and overview of this **Integration Build Plan**.]*

1.1 Purpose

*[Specify the purpose of this **Integration Build Plan**.]*

1.2 Scope

*[A brief description of the scope of this **Integration Build Plan**; what model(s) it is associated with and anything else that is affected or influenced by this document.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Integration Build Plan**. This information may be provided by reference to the project's Glossary.]*

1.4 References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Integration Build Plan**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

1.5 Overview

*[This subsection describes what the rest of the **Integration Build Plan** contains and explains how the document is organized.]*

2. Subsystems

[State which subsystems to implement in this iteration. Also state the preferred order in which the subsystems will be implemented to be ready in time for integration.]

3. Builds

[The integration, in the iteration, is divided into a number of increments, each resulting in a build, which is integration-tested. This section specifies which builds to create and which subsystems will be part of each build. For each build, this section needs to specify how the build is constructed, the criteria for its assessment and how it is to be tested, in particular:

- *Construction*

  Build scripts and any other instructions that describe how the build is constructed.

  Baseline records that define the versions of the configuration items used to construct the build.

- *Evaluation and Test*

<Project Name>	Version: <1.0>
Integration Build Plan	Date: <dd/mmm/yy>
<document identifier>	

Evaluation criteria—a description of the capabilities against which the build will be judged. This may contain a subset of the evaluation criteria in the corresponding Iteration Plan and other build specific evaluation criteria (particularly when, for example, the build is an architecture build which does not deliver much, if any capability that is visible to the end-user.

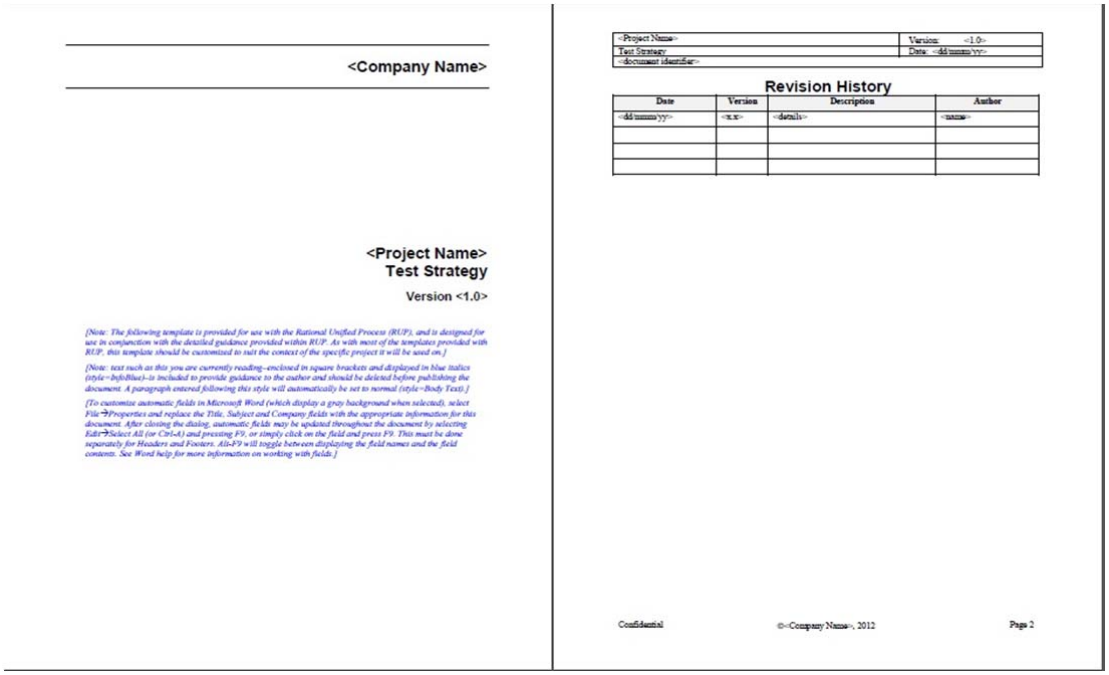
Installation and setup instructions to execute and test the build.

Test cases, test procedures, test scripts, and test results.

Note: In all cases, there is no requirement to replicate material in this plan—references will suffice if the material exists in other artifacts; for example, the Artifact: Iteration Test Plan.]

PRUEBAS

Plan De Pruebas



<Company Name>

<Project Name>
Test Strategy
Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process (RUP), and is designed for use in conjunction with the detailed guidance provided within RUP. As with most of the templates provided with RUP, this template should be customized to suit the context of the specific project it will be used on.]

[Note: text such as this you are currently reading–enclosed in square brackets and displayed in blue italics (style=InfoBlue)–is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File→Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit→Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

<Project Name>	Version: <1.0>
Test Strategy	Date: <dd/mmm/yy>
<document identifier>	

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Confidential ©<Company Name>, 2012 Page 2

Table of Contents

Test Strategy

1. Introduction

1.1 Purpose

The purpose of the Test Strategy for the <complete lifecycle, specific-phase> of the <Project Name> is to:

- Provide a central artifact to govern the strategic approach to the test effort. It defines the general approach that will be employed to test the software and to evaluate the results of that testing, and is the artifact that planning artifacts will refer to in terms of governing the detailed testing work.
- Provide visibility to stakeholders in the testing effort that adequate consideration has been given to various aspects of governing the testing effort, and where appropriate to have those stakeholders approve the strategy.

This *Test Strategy* also supports the following specific objectives:

- *[Identifies the items that should be targeted by the tests.*
- *Identifies the motivation for and ideas behind the test areas to be covered.*
- *Outlines the testing approach that will be used.*
- *Identifies the required resources and provides an estimate of the test efforts.*
- *Lists the deliverable elements of the test project.]*

1.2 Scope

*[Defines the types of testing—such as Functionality, Usability, Reliability, Performance, and Supportability—and if necessary the levels of testing—for example, Integration or System—that will be addressed by this **Test Strategy**. It is also important to provide a general indication of significant elements that will be **excluded** from scope, especially where the intended audience might otherwise reasonably assume the inclusion of those elements.*

Note**: Be careful to avoid repeating detail here that you will define in sections* ***¡Error! No se encuentra el origen de la referencia.*,* ***¡Error! No se encuentra el origen de la referencia.****, and* ***¡Error! No se encuentra el origen de la referencia.****. ¡Error! No se encuentra el origen de la referencia.]*

1.3 Intended Audience

*[Provide a brief description of the audience for whom you are writing the **Test Strategy**. This helps readers of your document identify whether it is a document intended for their use, and helps prevent the document from being used inappropriately.*

***Note**: Document style and content often alters in relation to the intended audience.*

This section should only be about three to five paragraphs in length.]

1.4 Document Terminology and Acronyms

*[This subsection provides the definitions of any terms, acronyms, and abbreviations required to properly interpret the **Test Strategy**. Avoid listing items that are generally applicable to the project as a whole and that are already defined in the project's Glossary. Include a reference to the project's Glossary in the References section.]*

1.5 References

*[This subsection provides a list of the documents referenced elsewhere within the **Test Strategy**. Identify each document by title, version (or report number if applicable), date, and publishing organization or original author. Avoid listing documents that are influential but not directly referenced. Specify the sources from which the "official versions" of the references can be obtained, such as intranet UNC names or document reference codes. This information may be provided by reference to an appendix or to another document.]*

1.6 Document Structure

*[This subsection outlines what the rest of the **Test Strategy** contains and gives an introduction to how the rest of the document is organized. This section may be eliminated if a Table of Contents is used.]*

2. Governing Evaluation Mission

[Provide an overview of the mission(s) that will govern the detailed testing within the iterations.]

2.1 Project Context and Background

[Provide a brief description of the background surrounding the project with specific reference or focus on important implications for the test effort. Include information such as the key problem being solved, the major benefits of the solution, the planned architecture of the solution, and a brief history of the project. Note that where this information is defined sufficiently in other documents, you might simply include a reference to those other documents if appropriate; however, it may save readers of the test strategy time and effort if a limited amount of information is duplicated here: so you should use your judgement. As a general rule, this section should only be about three to five paragraphs in length.]

2.2 Evaluation Mission

[Provide a brief statement that defines the mission(s) for the test and evaluation effort over the scope of the plan. The governing mission statement(s) might incorporate one or more concerns including:

- *find as many bugs as possible*
- *find important problems, assess perceived quality risks*
- *advise about perceived project risks*
- *certify to a standard*
- *verify a specification (requirements, design or claims)*
- *advise about product quality, satisfy stakeholders*
- *advise about testing*
- *fulfill process mandates*
- *and so forth*

Each mission provides a different context to the test effort and changes the way in which testing should be approached.]

2.3 Test Motivators

[Provide an outline of the key elements that will motivate the testing effort in this iteration. Testing will be motivated by many things—quality risks, technical risks, project risks, use cases, functional requirements, non-functional requirements, design elements, suspected failures or faults, change requests, and so forth.]

DESPLIEGUE

Plan De Despliegue

<Company Name>

<Project Name>
Deployment Plan

Version <1.0>

[Note: The following template is provided for use with the Rational Unified Process. Text enclosed in square brackets and displayed in blue italics (style=InfoBlue) is included to provide guidance to the author and should be deleted before publishing the document. A paragraph entered following this style will automatically be set to normal (style=Body Text).]

[To customize automatic fields in Microsoft Word (which display a gray background when selected), select File>Properties and replace the Title, Subject and Company fields with the appropriate information for this document. After closing the dialog, automatic fields may be updated throughout the document by selecting Edit>Select All (or Ctrl-A) and pressing F9, or simply click on the field and press F9. This must be done separately for Headers and Footers. Alt-F9 will toggle between displaying the field names and the field contents. See Word help for more information on working with fields.]

Revision History

Date	Version	Description	Author
<dd/mmm/yy>	<x.x>	<details>	<name>

Table of Contents

Deployment Plan

1. Introduction

[Provide an overview of the entire document.]

1.1 Purpose

[Describe the purpose of the software to which this document applies.]

1.2 Scope

*[Identify the recipients for the items identified in the **Deployment Plan**.]*

1.3 Definitions, Acronyms, and Abbreviations

*[This subsection provides the definitions of all terms, acronyms, and abbreviations required to properly interpret the **Deployment Plan**. This information may be provided by reference to the project's Glossary.]*

1.4 Overview

[Explain how this document is organized.]

2. References

*[This subsection provides a complete list of all documents referenced elsewhere in the **Deployment Plan**. Identify each document by title, report number (if applicable), date, and publishing organization. Specify the sources from which the references can be obtained. This information may be provided by reference to an appendix or to another document.]*

3. Deployment Planning

[Describe all activities performed in deploying the product to the customer. Activities include planning, beta testing, preparing items to be delivered, packaging, "shipping", installing the product, training, and support.]

3.1 Responsibilities

[Identify the responsibilities of both the customer and the development team in preparing for deployment. Of particular relevance in this section is the description of the customer's involvement in acceptance tests and the process to handle any discrepancies.]

3.2 Schedule

[Describe the schedule and milestones to conduct the deployment activities. Deployment milestones need to conform to the project milestones.

Take into account the following Deployment workflow details:

- *Planning the Deployment*
- *Developing Support Material*
- *Managing Acceptance Tests*
  - *Acceptance Testing at the Development Site*
  - *Acceptance Testing at the Deployment Site*
- *Producing the Deployment Unit*
- *Managing the Beta Program*
- *Managing Product Mass Production and Packaging*
- *Making the Product Accessible over the Internet]*

4. Resources

[List the resources and their sources required to carry out the planned deployment activities.]

4.1 Facilities

[As applicable, describe the facilities required to test and deploy the software. Facilities may include special buildings or rooms with raised flooring, power requirements, and special features to support privacy and security requirements.]

4.2 Hardware

[Identify the hardware required to run and support the software, as required. Specify model, versions, and configurations. Provide information about manufacturer support and licensing.]

4.3 The Deployment Unit

[List the software and documentation provided as part of the deliverable product.]

4.3.1 Support Software

[As applicable, describe all software needed to support the deliverable product, such as tools, compilers, test tools, test data, utilities, CM tools, databases, data files, and so on.]

4.3.2 Support Documentation

[As applicable, describe the documentation required to support the deliverable product, such as design descriptions, test cases and procedures, user manuals, and so on.]

4.3.3 Support Personnel

[As applicable, describe the personnel, and their skill levels, required to support the deliverable product.]

5. Training

[Describe the plan and inputs for training the end users so they can use and adapt the product as required.]