ANÁLISIS, RE-EVALUACIÓN Y RECOMENDACIONES FRENTE AL DOCUMENTO <<1-IN-030 ESTÁNDARES DE ANÁLISIS, DISEÑO Y DESARROLLO DE SISTEMAS DE INFORMACIÓN>>

ALEJANDRO RAMÍREZ MEDINA

UNIVERSIDAD CATÓLICA POPULAR DEL RISARALDA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
PRACTICAS PROFESIONALES
PEREIRA
2010

ANÁLISIS, RE-EVALUACIÓN Y RECOMENDACIONES FRENTE AL DOCUMENTO <<1-IN-030 ESTÁNDARES DE ANÁLISIS, DISEÑO Y DESARROLLO DE SISTEMAS DE INFORMACIÓN>>

ALEJANDRO RAMÍREZ MEDINA

Informe de Práctica Profesional

Tutor
ANDRÉS VARGAS GARCÍA
Ingeniero de Sistemas y Computación

UNIVERSIDAD CATÓLICA POPULAR DEL RISARALDA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y TELECOMUNICACIONES
PRACTICAS PROFESIONALES
PEREIRA
2010

CONTENIDO

Pág.

LISTA DE TABLAS

Pág.

LISTA DE FIGURAS

Pág.

LISTA DE ANEXOS

Pág.

Resumen	Abstract
Como es bien sabido todo software tiene un ciclo de vida, el cual desde su concepción como idea, tiene que estar bajo unos parámetros y debe estar funcionando dentro de una metodología que siendo bien escogida podrá ser la clave de un producto exitoso y de calidad. Ahora bien, las empresas actualmente buscan que sus productos sean de calidad y su desarrollo sea lo suficientemente ágil y eficiente, de tal manera que la productividad como los costos se mejoren. Comfamiliar Risaralda se encamina a un proceso de estandarización y calidad bajo la norma ISO 9001/2008 por eso este informe es desarrollado como justificante del cambio de metodología hacia una de mayor eficacia al momento de desarrollos de sistemas de información y acomodada a las necesidades de la institución, de acuerdo con las tendencias del contexto actual en el que se hace este trabajo dentro del proceso administrativo de sistemas.	As it´s well known every software has a life cycle, which since its conception as an idea, must be under some parameters and must also be functioning within a methodology that being well chosen will be the key to a successful and good quality product. However nowadays companies are looking for their products to be of a good quality and for their development to be sufficiently agile and efficient, so that productivity as well as cost improve. Comfamiliar Risaralda is directed to a standardization and quality process under the standards of ISO 9001/2008, that's why this report is developed as proof of the changes in the methodology towards one that's more efficient at the moment of developing systems of information and accommodated to the needs of the institution, according to the trends of the current context in which this job is done within the administrative process of systems.
Descriptores: Proyectos agiles, programación extrema, metodologías, sistemas de información, ciclo de vida, iteración, SCRUM, RUP, ingeniería de software, métodos.	Descriptors: Agile Project, extreme programming, methodologies, information system, software life cycle, iteration, SCRUM, RUP, software engineering, methods.

INTRODUCCIÓN

En el contexto actual de la globalización se dice popularmente que: quien tiene la información, es quien tiene el poder; y es verdad, pero para que esto se cumpla se debe entender que no solo es poseer la información, sino saberla manejar, y procesarla de una manera adecuada; es decir debe estar disponible para quienes están autorizados para utilizarla, además de mantenerla de manera íntegra y segura. Para esto el hombre desde hace muchos años viene desarrollando sistemas de información, que hoy en día se han convertido en la parte fundamental de las empresas.

Ahora bien, con cumplir lo dicho anteriormente no basta, pues la competencia permanente en la que viven las empresas exige que cada vez sus procesos sean más productivos, tanto en tiempo como económicamente, para poder invertir de manera adecuada todos los recursos disponibles y así lograr un desarrollo rápido y sostenible durante sus años de vida. Para esto las empresas vienen buscando certificarse como una de calidad, y para lograr ese objetivo es necesario que todas sus dependencias, departamentos y procesos se optimicen. En este caso es el proceso de creación de sistemas de información el que se trabajará; pues es importante que los desarrollos de software que se hacen en la compañía tengan una metodología adecuada para aumentar todos los factores que indican que se hacen productos de calidad.

Durante este trabajo se analizara la metodología que se utiliza actualmente en el proceso administrativo de Sistemas, para la creación de sistemas de información, y de esta manera dejar la visión clásica de centrarse en la planeación de un proyecto de software, para identificar las actividades de adaptabilidad que dicho proceso tiene que tener durante el tiempo de producción e implementación. Todo esto se hace utilizando conocimientos en ingeniería del software tanto: literarios como experiencias aplicadas por el equipo de trabajo del proceso de sistemas.

1. PRESENTACIÓN DE LA ORGANIZACIÓN O SITIO DE PRÁCTICA

1.1 RESEÑA HISTÓRICA

La historia de COMFAMILIAR RISARALDA comienza con el decreto 118 del 21 de junio de 1957. En él se establecen los argumentos con los cuales el Gobierno colombiano considera de vital importancia: “Atender las necesidades de las clases menos favorecidas económicamente y fomentar su mejoramiento”.

En Pereira y en general en el Eje Cafetero, la vida comercial e industrial estaba en plena etapa de desarrollo: alimentos, bebidas, textiles, confecciones, vivienda, metalmecánica, forman parte de las grandes transformaciones de la región a partir de los sesenta.

Al igual que en otras ciudades del país, se presentan carencias de servicios esenciales en la comunidad (analfabetismo, desnutrición, vivienda, etc.) y cuyas soluciones son imperiosas antes de acceder a los retos de la modernización económica, comercial, industrial y financiera.

- Antecedentes históricos

1957 Decreto 118 de 21 Junio 1957 Cajas País.

1957 Acta 001 ANDI 29 Agosto 1957 Comfamiliar Risaralda

1966 Servicios Sociales Salud Pediátrico

1971 Mercadeo Social

1974 Educación

1976 Centro Recreacional

1991 Vivienda

1992 Centro clínico Odontológico

1994 Clínica Materno Infantil – Socio SOS

1997 Clínica Comfamiliar

2003 Granja de Noé – Alianza Estratégica Mercadeo

2004 Cardiología Invasiva

2005 Construcción de la Bolera Comfamiliar
Ampliación locativa de la Clínica Comfamiliar Risaralda
Aprobación de la participación de Comfamiliar en el manejo y administración del parque de los nevados

2006 Se aprueba la creación del servicio de hospitalización domiciliaria o en casa, el retiro de Comfamiliar del manejo directo del Régimen Subsidiado, cierre ARS y el ingreso al Programa Nacional de Interrelación social originado en remesas que se envían desde el exterior por personas del eje cafetero, conjuntamente con Comfama, las tres Cajas del Eje Cafetero, Comfamiliar del atlántico, Cafam, Asocajas y Bancolombia. Además se aprueba la opción de ingreso de la Caja en el manejo del aseguramiento de los actuales afiliados al ISS, en asocio con otras Cajas.

1.2 INFORMACIÓN INSTITUCIONAL

-Misión

Es una entidad de servicios dentro del campo de la protección social, que promueve el desarrollo integral de la familia, como núcleo básico de la sociedad.

-Visión

Permanecer como actores dinámicos en el campo de la protección social y del bienestar de la comunidad, conservando el liderazgo y proyección institucional.

1.3 ACTIVIDAD A LA CUAL SE DEDICA LA ORGANIZACIÓN Y LÍNEAS QUE PRODUCE O SERVICIOS QUE PRESTA

Aportes y Subsidio
Crédito Social
Vivienda
Salud
Gerontología
Educación
Capacitación
Área Cultural
Recreación
Deportes
Agencia de Viajes
Atención Integral a la Niñez
Jornadas Escolares Complementarias
Centros Culturales Y Bibliotecas
Bolera

1.4 NÚMERO DE TRABAJADORES

- Al 26 de febrero del 2010, Comfamiliar Risaralda tiene: ~ 1600 trabajadores.

1.5 ORGANIGRAMA

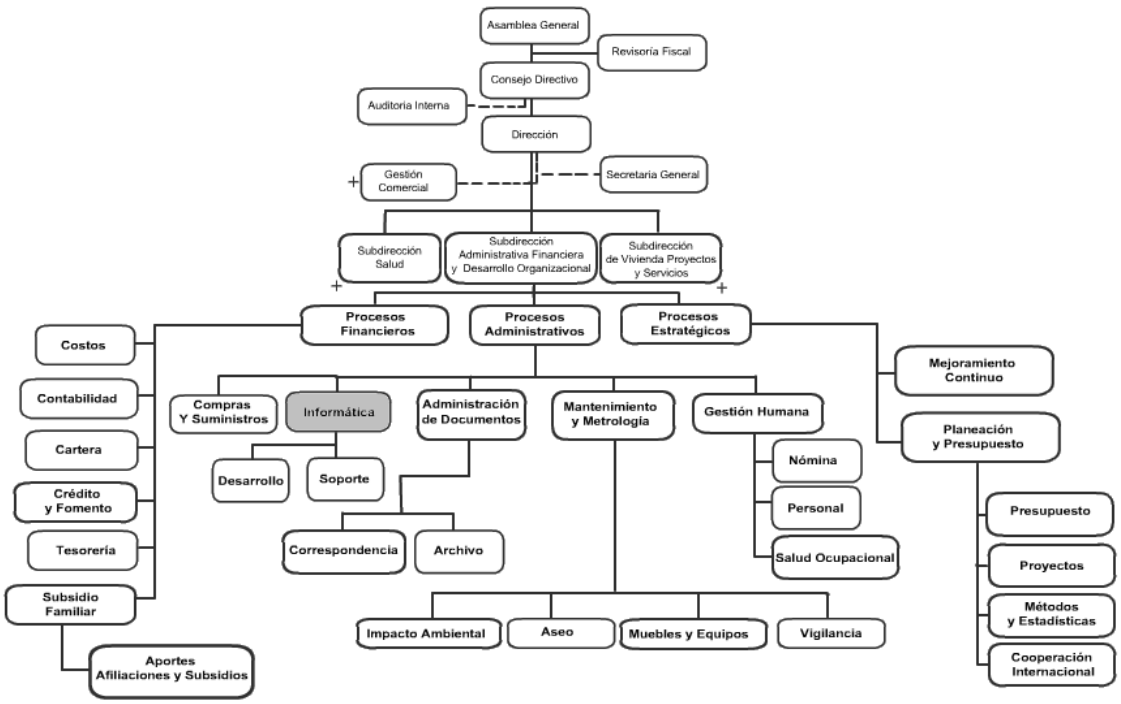


Figura 1 - Organigrama Comfamiliar

2. DEFINICIÓN DE LAS LÍNEAS DE INTERVENCIÓN

Para la realización del proyecto interventor en la empresa, es necesario que el practicante asuma la posición de Asistente de Procesos, el cual para hacerse efectivo, es necesario que se involucre en las siguientes líneas:

- Desarrollo de Software

- Sistemas de Información

3. DIAGNÓSTICO DEL ÁREA DE INTERVENCIÓN O IDENTIFICACIÓN DE LAS NECESIDADES

En Comfamiliar Risaralda, dentro de la rama administrativa, más específicamente en el proceso de Sistemas, se destaca en el interior de su objetivo “desarrollar e implementar tecnologías en software”, el cual para ser cumplido a cabalidad se deben llevar a cabo una serie de protocolos establecidos en el manual de calidad interno de la organización. Para la relevancia de esta propuesta, uno de los protocolos o pasos a seguir al momento de desarrollar e implementar software es el documento instructivo: <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>> el cual debe ser efectuado en la totalidad de los proyectos relacionados con el diseño, desarrollo e implementación de software, pero hasta Noviembre del año 2009 había sido ejecutado en 21 de 30 proyectos puestos en marcha y finalizados, aproximadamente. El efecto de este documento mencionado es que se pueda presentar al software como un producto de calidad dentro de la compañía, y al no cumplirse esto no se está logrando el objetivo del manual de calidad.

4. EJE DE INTERVENCIÓN

Como se mencionó anteriormente, en el manual de calidad de la empresa Comfamiliar Risaralda, se presenta el documento instructivo correspondiente a los estándares relacionados con los sistemas de información, precisamente está en el manual de calidad porque se espera que con éste se logre un producto completo en todos los aspectos, y que cumpla con los requerimientos de la organización, pero en muchas ocasiones este documento no se ve implementado en un proyecto, y las razones varían, entre las cuales se destacan: lo dispendioso que se hace para los programadores algunos procesos de documentación, la fabricación de software a la medida con eventual rapidez en la demanda de este mismo, etc.

5. JUSTIFICACIÓN DEL EJE DE INTERVENCIÓN

Uno de los objetivos del presente año 2010 en el proceso Administrativo Sistemas, es re-evaluar la metodología que se viene implementando hasta el momento para la documentación de los sistemas de información, y es entonces en donde entra este proyecto, pues es necesario que esto se haga ya que se tienen que analizar las causas del porqué el documento instructivo mencionado anteriormente no se está tomando en cuenta en algunos casos, y revisar que tan relevantes son todos los puntos del estándar, además de generar una serie de recomendaciones a tener en cuenta para ser evaluadas dentro del proceso.

6. OBJETIVO GENERAL

Analizar, re-evaluar y hacer recomendaciones frente al documento <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>>.

7. OBJETIVOS ESPECÍFICOS

- Identificar los Procesos de documentación establecidos en el instructivo <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>>.

- Estudiar alternativas de documentación vigentes actualmente en el mercado.

- Levantar información con los respectivos implicados en cada uno de los procesos relacionados con el documento mencionado.

- Estructurar resultados de la información adquirida y compararla con las alternativas estudiadas.

- Generar documento con observaciones y conclusiones a tener en cuenta para ser incluidas en el Proceso de documentación.

- Este documento debe servir como justificante del cambio de metodología de desarrollo de software en el proceso de sistemas.

8. REFERENTE CONCEPTUAL

Extractos del libro: “Ingeniería del Software un Enfoque Practico”

El estado del arte de la ingeniería del software

La Ingeniería del Software es una disciplina o área de la Informática o Ciencias de la Computación, que ofrece métodos y técnicas para desarrollar y mantener software de calidad que resuelven problemas de todo tipo. Hoy día es cada vez más frecuente la consideración de la Ingeniería del Software como una nueva área de la ingeniería, y el ingeniero del software comienza a ser una profesión implantada en el mundo laboral internacional, con derechos, deberes y responsabilidades que cumplir, junto a una, ya, reconocida consideración social en el mundo empresarial y, por suerte, para esas personas con brillante futuro.
La Ingeniería del Software trata con áreas muy diversas de la informática y de las ciencias de la computación, tales como construcción de compiladores, sistemas operativos o desarrollos en Intranet, Internet, abordando todas las fases del ciclo de vida del desarrollo de cualquier tipo de sistemas de información y aplicables a una infinidad de áreas tales como: negocios, investigación científica, medicina, producción, logística, banca, control de tráfico, meteorología, el mundo del derecho, la red de redes Internet, redes Intranet y Extranet, etc.

Definición del término «Ingeniería del Software»

El término Ingeniería se define en el DRAE (Diccionario de la Real Academia de la lengua Española) como: «1. Conjunto de conocimientos y técnicas que permiten aplicar el saber científico a la utilización de la materia y de las fuentes de energía. 2. Profesión y ejercicio del ingeniero» y el término ingeniero se define como «1. Persona que profesa o ejerce la ingeniería». De igual modo la Real Academia de Ciencias Exactas, Físicas y Naturales de España define el término Ingeniería como: «Conjunto de conocimientos y técnicas cuya aplicación permite la utilización racional de los materiales y de los recursos naturales, mediante invenciones, construcciones u otras realizaciones provechosas para el hombre».

Evidentemente, si la Ingeniería del Software es una nueva ingeniería, parece lógico que reúna las propiedades citadas en las definiciones anteriores. Sin embargo, ni el DRAE ni la Real Academia Española de Ciencias han incluido todavía el término en sus Últimas ediciones; en consecuencia vamos a recurrir para su definición más precisa a algunos de los autores más acreditados que comenzaron en su momento a utilizar el término o bien en las definiciones dadas por organismos internacionales profesionales de prestigio tales como IEEE o ACM. Así, hemos seleccionado las siguientes definiciones de Ingeniería del Software:

Definición 1
Ingeniería de Software es el estudio de los principios y metodologías para desarrollo y mantenimiento de sistemas de software [Zelkovitz, 1978].

Definición 2
Ingeniería del Software es la aplicación práctica del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar (funcionar) y mantenerlos. Se conoce también como desarrollo de software o producción de software [Bohem, 1976].

Definición 3
Ingeniería del Software trata del establecimiento de los principios y métodos de la ingeniería a fin de obtener software de modo rentable que sea fiable y trabaje en máquinas reales [Bauer, 1972].

Definición 4
La aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación (funcionamiento) y mantenimiento del software; es decir, la aplicación de ingeniería al software.

El proceso de software

Un proceso de software se establece un marco común del proceso definiendo un pequeño número de actividades del marco de trabajo que son aplicables a todos

los proyectos del software, con independencia de su tamaño o complejidad. Un número de conjuntos de tareas –cada uno es una colección de tareas de trabajo de ingeniería del software, hitos de proyectos, productos de trabajo, y puntos de garantía de calidad- que permiten que las actividades del marco de trabajo se adapten a las características del proyecto del software y a los requisitos del equipo del proyecto. Finalmente, las actividades de protección -tales como garantía de calidad del software, gestión de configuración del software y medición- abarcan el modelo de procesos. Las actividades de protección son independientes de cualquier actividad del marco de trabajo y aparecen durante todo el proceso.

En los Últimos años, se ha hecho mucho énfasis en la «madurez del proceso>> El Software Engineering Institute (SEI) ha desarrollado un modelo completo que se basa en un conjunto de funciones de ingeniería del software que deberían estar presentes conforme organizaciones alcanzan diferentes niveles de madurez del proceso. Para determinar el estado actual de madurez del proceso de una organización, el SEI utiliza un cuestionario de evaluación y un esquema de cinco grados.

El esquema de grados determina la conformidad con un modelo de capacidad de madurez que define las actividades clave que se requieren en los diferentes niveles de madurez del proceso. El enfoque del SEI proporciona una medida de la efectividad global de las prácticas de ingeniería del software de una compañía y establece cinco niveles de madurez del proceso, que se definen de la forma siguiente:

Nivel 1: Inicial. El proceso del software se caracteriza según el caso, y ocasionalmente incluso de forma caótica. Se definen pocos procesos, y el éxito depende del esfuerzo individual.

Nivel 2: Repetible. Se establecen los procesos de gestión del proyecto para hacer seguimiento del coste, de la planificación y de la funcionalidad. Para repetir éxitos anteriores en proyectos con aplicaciones similares se aplica la disciplina necesaria para el proceso.

Nivel 3: Definido. El proceso del software de las actividades de gestión y de ingeniería se documenta, se estandariza y se integra dentro de un proceso de

software de toda una organización. Todos los proyectos utilizan una versión documentada y aprobada del proceso de la organización para el desarrollo y mantenimiento del software. En este nivel se incluyen todas las características definidas para el nivel 2.

Nivel 4: Gestionado. Se recopilan medidas detalladas del proceso del software y de la calidad del producto. Mediante la utilización de medidas detalladas, se comprenden y se controlan cuantitativamente tanto los productos como el proceso del software. En este nivel se incluyen todas las características definidas para el nivel 3.

Nivel 5: Optimización. Mediante una retroalimentación cuantitativa del proceso, ideas y tecnologías innovadoras se posibilita una mejora del proceso. En este nivel se incluyen todas las características definidas para el nivel 4.

Merece la pena destacar que cada nivel superior es la suma de los niveles anteriores, por ejemplo, un desarrollador de software en el nivel 3 tiene que haber alcanzado el estado nivel 2 para poder disponer de sus procesos en el nivel 3.

El nivel 1 representa una situación sin ningún esfuerzo en la garantía de calidad y gestión del proyecto, donde cada equipo del proyecto puede desarrollar software de cualquier forma eligiendo los métodos, estándares y procedimientos a utilizar que podrán variar desde lo mejor hasta lo peor.

El nivel 2 representa el hecho de que un desarrollador de software ha definido ciertas actividades tales como el informe del esfuerzo y del tiempo empleado, y el informe de las tareas realizadas.

El nivel 3 representa el hecho de que un desarrollador de software ha definido tanto procesos técnicos como de gestión, por ejemplo un estándar para la programación ha sido detallado y se hace cumplir por medio de procedimientos tales como auditorias. Este nivel es aquel en el que la mayoría de los desarrolladores de software, pretenden conseguir con estándares como el ISO 9001, y existen pocos casos de desarrolladores de software que superan este nivel.

El nivel 4 comprende el concepto de medición y el uso de métricas. Sin embargo, cabe destacar el concepto de métrica para comprender la importancia que tiene que el desarrollador del software alcance el nivel 4 o el nivel 5. Una métrica es una cantidad insignificante que puede extraerse de algún documento o código dentro de un proyecto de software. Un ejemplo de métrica es el número de ramas condicionales en una sección de código de un programa. Esta métrica es significativa en el sentido de que proporciona alguna indicación del esfuerzo necesario para probar el código: está directamente relacionado con el número de caminos de prueba dentro del código.

Una organización del nivel 4 maneja numerosas métricas. Estas métricas se utilizan entonces para supervisar y controlar un proyecto de software, por ejemplo: Una métrica de prueba puede usarse para determinar cuándo finalizar la prueba de un elemento del código.

Una métrica de legibilidad puede usarse para juzgar la legibilidad de algún documento en lenguaje natural. Una métrica de comprensión del programa puede utilizarse para proporcionar algún umbral numérico que los programadores no pueden cruzar.

Para que estas métricas alcancen este nivel es necesario que todos los componentes del nivel 3 CMM, en castellano MCM (Modelo de Capacidad de Madurez), estén conseguidos, por ejemplo notaciones bien definidas para actividades como la especificación del diseño de requisitos, por lo que estas métricas pueden ser fácilmente extraídas de modo automático.

El nivel 5 es el nivel más alto a alcanzar. Hasta ahora, muy pocos desarrolladores de software han alcanzado esta fase. Representa la analogía del software con los mecanismos de control de calidad que existen en otras industrias de mayor madurez. Por ejemplo el fabricante de un producto industrial como un cojinete de bolas (rodamiento) puede supervisar y controlar la calidad de los rodamientos producidos y puede predecir esta calidad basándose en los procesos y máquinas utilizados para desarrollar los rodamientos. Del mismo modo que el desarrollador del software en el nivel 5 puede predecir resultados como el número de errores latentes en un producto basado en la medición tomada durante la ejecución de un proyecto. Además, dicho desarrollador puede cuantificar el efecto que un proceso nuevo o herramienta de manufacturación ha tenido en un proyecto examinando métricas para ese proyecto y comparándolas

con proyectos anteriores que no utilizaron ese proceso o herramienta.

En este orden debe destacarse que para que un desarrollador de software alcance el nivel 5 tiene que tener cada proceso definido rigurosamente y seguirlo al pie de la letra; esto es una consecuencia de estar en el nivel 3. Si el desarrollador del software no tiene definidos rigurosamente los procesos pueden ocurrir una gran cantidad de cambios en el proceso de desarrollo y no se podrán utilizar las estadísticas para estas actividades.

Los cinco niveles definidos por el SEI se obtienen como consecuencia de evaluar las respuestas del cuestionario de evaluación basado en el MCM (Modelo de capacidad de madurez). Los resultados del cuestionario se refinan en un único grado numérico que proporciona una indicación de la madurez del proceso de una organización.

El SEI ha asociado áreas claves del proceso (ACPs) a cada uno de los niveles de madurez. Las ACPs describen esas funciones de la ingeniería del software (por ejemplo: planificación del proyecto de software, gestión de requisitos) que se deben presentar para satisfacer una buena práctica a un nivel en particular. Cada ACP se describe identificando las características siguientes:

Objetivos - los objetivos globales que debe alcanzar la ACP

Compromisos - requisitos (impuestos en la organización) que se deben cumplir para lograr los objetivos y que proporcionan una prueba del intento por ajustarse a los objetivos.

Capacidades - aquellos elementos que deben encontrarse (organizacional y técnicamente) para permitir que la organización cumpla los objetivos.

Actividades - las tareas específicas que se requieren para lograr la función ACP.

Métodos para supervisar la implementación-la manera en que las actividades son supervisadas conforme se aplican.

Métodos para verificar la implementación- la forma en que se puede verificar la práctica adecuada para la ACP.

Se definen dieciocho ACPs (descritas mediante la estructura destacada anteriormente) en el modelo de madurez y se distribuyen en niveles diferentes de madurez del proceso. Las ACPs se deberían lograr en cada nivel de madurez de proceso:

Nivel 2 de Madurez del Proceso
-Gestión de configuración del software
-Garantía de calidad del software
-Gestión de subcontratación del software
-Seguimiento y supervisión del proyecto del software
-Planificación del proyecto del software
-Gestión de requisitos

Nivel 3 de Madurez del Proceso
-Revisiones periódicas
-Coordinación entre grupos
-Ingeniería de productos de software
-Gestión de integración del software
-Programa de formación
-Definición del proceso de la organización
-Enfoque del proceso de la organización

Nivel 4 de Madurez del Proceso
-Gestión de calidad del software
-Gestión cuantitativa del proceso

Nivel 5 de Madurez del Proceso
-Gestión de cambios del proceso
-Gestión de cambios de tecnología
-Prevención de defectos

Cada una de las ACPs se define con un conjunto de prácticas clave que contribuyen a cumplir estos objetivos. Las prácticas clave son normas,

procedimientos y actividades que deben ocurrir antes de que se haya instituido completamente un área clave de proceso. El SEI define a los indicadores clave como «aquellas prácticas clave o componentes de prácticas clave que ofrecen una visión mejor para lograr los objetivos de un área clave de proceso». Las cuestiones de valoración se diseñan para averiguar la existencia (o falta) de un indicador clave.

Modelos de proceso del software

Para resolver los problemas reales de una industria, un ingeniero del software o un equipo de ingenieros deben incorporar una estrategia de desarrollo que acompañe al proceso, métodos y capas de herramientas. Esta estrategia a menudo se llama modelo de proceso o paradigma de ingeniería del software. Se selecciona un modelo de proceso para la ingeniería del software según la naturaleza del proyecto y de la aplicación, los métodos y las herramientas a utilizarse, y los controles y entregas que se requieren.

Todo el desarrollo del software se puede caracterizar como bucle de resolución de problemas en el que se encuentran cuatro etapas distintas: «status quo», definición de problemas, desarrollo técnico e integración de soluciones. Status quo «representa el estado actual de sucesos»; la definición de problemas identifica el problema específico a resolverse; el desarrollo técnico resuelve el problema a través de la aplicación de alguna tecnología y la integración de soluciones ofrece los resultados (por ejemplo: documentos, programas, datos, nueva función comercial, nuevo producto) a los que solicitan la solución en primer lugar.

Extractos de: “Swebok”

Los instrumentos de desarrollo de software son los instrumentos asistidos por ordenador que son requeridos para ayudar a los procesos de ciclo de vida de software. Los instrumentos permiten a acciones repetidas, bien definidas para ser automatizadas, reduciendo la carga cognoscitiva sobre el ingeniero de software que es entonces libre de concentrarse en los aspectos creativos del proceso. Los instrumentos a menudo son diseñados para apoyar el software particular métodos de la ingeniería, reduciendo cualquier carga administrativa asociada con la aplicación del método a mano. Como los métodos de la

ingeniería de software, ellos son queridos para hacer el software que trama más sistemático, varían en el alcance de apoyar tareas individuales que abarcan el ciclo de vida completo.

Los métodos de la ingeniería de software imponen la estructura a la actividad de la ingeniería de software con el objetivo de hacer la actividad sistemática y en última instancia más probablemente de ser acertado. Los métodos por lo general proporcionan la notación y el vocabulario, procedimientos para realizar tareas identificables, y directrices para comprobar tanto el proceso como el producto. Ellos varían extensamente en el alcance, de una fase única del ciclo de vida al ciclo de vida completo. El énfasis en esta Área de Conocimiento está sobre los métodos de la ingeniería de software que abarcan múltiples fases de ciclo de vida, ya que métodos específicos de fase son cubiertos por otras áreas de conocimiento. Mientras hay manuales detallados sobre instrumentos específicos y numerosos papeles de investigación sobre instrumentos innovadores, escrituras genéricas técnicas sobre instrumentos de la ingeniería de software son relativamente escasas. Una dificultad es la alta tarifa de cambio de instrumentos de software en general. Detalles específicos cambian con regularidad, haciendo difícil de proporcionar ejemplos concretos y actualizados.

Los Instrumentos de Ingeniería de Software y los Métodos del Área de Conocimiento cubren los procesos de ciclo de vida completos, y por lo tanto son relacionados con cada área de conocimiento en la Guía.

Los métodos de la Ingeniería de Software

Los Métodos de la Ingeniería de Software están dividido en tres temas: métodos heurísticos que tratan con accesos informales, métodos formales que tratan con accesos matemáticamente basados, y métodos de prototipado que tratan con software que trama accesos basados en varias formas de prototipado.
Estos tres temas no son inconexos; más bien representan preocupaciones distintas. Por ejemplo, un método orientado por objeto puede incorporar técnicas formales y confiar en prototipado para la verificación y la validación. Como los instrumentos de la Ingeniería de Software, las metodologías continuamente se desarrollan. Por consiguiente, la descripción del área de conocimiento evita en la medida de lo posible llamar metodologías particulares.

Métodos heurísticos

Este tema contienen cuatro categorías: estructurado, orientado a datos, orientado a objetos, y específico de dominio. La categoría específica de dominio incluye métodos especializados para desarrollar los sistemas que implican en tiempo real, de seguridad, o aspectos de seguridad.

♦ Métodos Estructurados. El sistema es construido de un punto de vista funcional, que comienza con una vista de alto nivel y cada vez más la refinación de esto en un diseño más detallado.

♦ Métodos Orientados a datos. Aquí, los puntos de partida son las estructuras de datos que un programa manipula más que la función que esto realiza.

♦ Métodos Orientados a objetos. El sistema es visto como una colección de objetos más que de funciones.

Métodos Formales

Esta subdivisión trata con el software matemáticamente basado métodos de la ingeniería, y es subdividida según varios aspectos de métodos formales.

♦ Especificación del lenguaje y notaciones. Este tema concierne la notación de especificación o la lengua usada. Las lenguas de especificación pueden ser clasificadas como orientado por modelo, orientado por característica, u orientado por comportamiento.

♦ Refinamiento. Este tema trata como el método refina (o transforma) la especificación en una forma que es más cercana a la forma deseada final de un programa ejecutable.

♦ Propiedades de Verificación/confirmación. Este tema cubre las propiedades de verificación que son específicas el acercamiento formal, incluyendo tanto confirmación de teorema como la comprobación del modelo.

Métodos de prototipado

Esta subdivisión cubre métodos que implican el prototipado de software y es subdividida en estilos de prototipado, objetivos, y técnicas de evaluación.

♦ Estilos de prototipado. El tema de estilos de prototipado identifica varios accesos: especificación desechable, evolutiva, y ejecutable.

♦ Objetivo del prototipado. Los Ejemplos de los objetivos de un método prototipado pueden ser exigencias, el diseño arquitectónico, o el interfaz de

usuario.
♦ Técnicas de evaluación del prototipado. Este tema cubre las razones por las cuales los resultados de un ejercicio de prototipo son usados.

Modelos de ciclo de vida en desarrollo de software

En el contexto de la industria colombiana de software.

La evolución de la disciplina de ingeniería de software ha traído consigo propuestas diferentes para mejorar los resultados del proceso de construcción. Las metodologías tradicionales haciendo énfasis en la planeación, y las metodologías ágiles haciendo énfasis en la adaptabilidad del proceso, delinean las principales propuestas presentes en la literatura. De manera paralela, el tema de modelos para el mejoramiento de los procesos de desarrollo ocupa un lugar importante en la búsqueda de la metodología adecuada para producir software de calidad en cualquier contexto de desarrollo.

Metodologías tradicionales en el desarrollo

Se caracterizan por exponer procesos basados en planeación exhaustiva. Esta planeación se realiza esperando que el resultado de cada proceso sea determinante y predecible. La experiencia ha mostrado que, como consecuencia de las características del software, los resultados de los procesos no son siempre predecibles y sobre todo, es difícil predecir desde el comienzo del proyecto cada resultado. Sin embargo, es posible por medio de la recolección y estudio de métricas de desarrollo lograr realizar estimaciones acertadas en contextos de desarrollo repetibles.

Remontándose a la historia, el modelo de cascada fue uno de los primeros modelos de ciclo de vida (MCV) que formalizó un conjunto de procesos de desarrollo de software. Este MCV describe un orden secuencial en la ejecución de los procesos asociados. El modelo espiral se postuló como una alternativa al modelo de cascada. La ventaja de este modelo radica en el perfeccionamiento de las soluciones encontradas con cada ciclo de desarrollo, en términos de dar respuesta a los requerimientos inicialmente analizados. El modelo de cascada y el modelo espiral suponen, de manera general, que los requerimientos del

cliente no cambian radicalmente en el transcurso del desarrollo del sistema.
Por otro lado, la realización de prototipos es una herramienta en la que se apoyan diferentes MCV. Un prototipo debe tener el objetivo de mostrar al cliente o a la gerencia del proyecto el resultado que se obtendrá de la implementación de cada uno de los requerimientos del cliente una vez terminado el desarrollo. Con los prototipos se tiene la posibilidad de obtener retroalimentación de manera temprana.

La solución a algunos de los problemas presentados por las metodologías tradicionales se logra con una gran evolución del modelo espiral. El proceso unificado propone la elaboración de varios ciclos de desarrollo, donde cada uno finaliza con la entrega al cliente de un producto terminado. Este se enmarca entre los conocidos modelos iterativo-incremental.

Metodologías ágiles

Grupos de desarrollo han experimentado soluciones que basan su fundamento en la adaptabilidad de los procesos de desarrollo, en lugar de seguir esperando lograr resultados predecibles de un proceso que no evoluciona. Esta comunidad de desarrolladores e investigadores han nombrado su trabajo bajo lo que conocemos como metodologías ágiles. Las metodologías ágiles como puede entenderse mal, no están en contra de administrar procesos de desarrollo. Por el contrario promueve la formalización de procesos adaptables.

La compilación de los principios y valores que resaltan las metodologías ágiles fue formalizada en el manifiesto para el desarrollo de software ágil. Este documento desarrollado por los representantes de cada una de las metodologías que en el momento se presentaban como ágiles, logra resumir en un conjunto de ideas las prácticas que una metodología de este estilo debe llevar a cabo. Como característica fundamental, la habilidad de responder al cambio es la principal característica de las metodologías ágiles.

XP, una de las más difundidas, es una metodología de desarrollo de software ágil que define pocas reglas y pocas prácticas. XP promueve la adaptabilidad de los procesos de desarrollo basándose en los principios y prácticas que presenta. Quienes trabajan usando XP deben seguir procesos disciplinados, pero más que eso, deben combinar la disciplina con la adaptabilidad necesaria del proceso.

Las metodologías de Cristal se basan en el principio de que tipos diferentes de proyectos requieren tipos diferentes de metodologías. La metodología escogida debe depender de dos factores: el número de personas en el proyecto, y las consecuencias de los errores. Conforme al principio de las metodologías ágiles, Scrum recalca la imposibilidad de encontrar procesos definidos y repetibles cuando no existen problemas, personas, ni ambientes definidos y repetibles.

¿Metodologías ágiles o metodologías tradicionales?

En las metodologías tradicionales el principal problema es que nunca se logra planear bien el esfuerzo requerido para seguir la metodología. Pero entonces, si logramos definir métricas que apoyen la estimación de las actividades de desarrollo, muchas prácticas de metodologías tradicionales podrían ser apropiadas. El no poder predecir siempre los resultados de cada proceso no significa que estamos frente a una disciplina de azar. Lo que significa es que estamos frente a la necesidad de adaptación de los procesos de desarrollo que son llevados por parte de los equipos que desarrollan software.

Tener metodologías diferentes para aplicar de acuerdo con el proyecto que se desarrolle resulta una idea interesante. Estas metodologías pueden involucrar prácticas tanto de metodologías ágiles como de metodologías tradicionales. De esta manera podríamos tener una metodología por cada proyecto, la problemática sería definir cada una de las prácticas, y en el momento preciso definir parámetros para saber cual usar.

Es importante tener en cuenta que el uso de un método ágil no es para todos. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero y por eso las personas que no estén acostumbradas a seguir procesos encuentran estas metodologías bastante agradables.

En el contexto colombiano

Los primeros desarrollos de software en Colombia iniciaron de manera artesanal. Incrementalmente, y con la llegada de nuevas tecnología, plataformas de desarrollo, y programas de formación superior bien estructurados, se inició un

proceso de mejoramiento de procesos entre los que se incluye el tema de la planeación y seguimiento de los proyectos de software.

El modelo de cascada fue en ese entonces, y tal vez lo sigue siendo, el modelo más usado por los desarrolladores de software que recién constituyen sus empresas de desarrollo. Modelos de desarrollo sin alto nivel de complejidad como el espiral, o el orientado a prototipos, siguen siendo los más usados. Sin embargo, estos modelos siguen siendo interpretados en algunos casos de manera errónea. Es el caso de equipos de desarrollo que por tener ciclos de desarrollo iterativo y/o incremental, aseguran seguir un modelo espiral, sin realizar un análisis de riesgo bien soportado.

Extracto de: ProyectosAgiles.ORG

Qué es SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

El proceso

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos (iteraciones de un mes natural y hasta de dos semanas, si así se necesita). Cada iteración tiene que proporcionar un resultado completo, un incremento de producto final que sea susceptible de ser entregado con el mínimo esfuerzo al cliente cuando lo solicite.

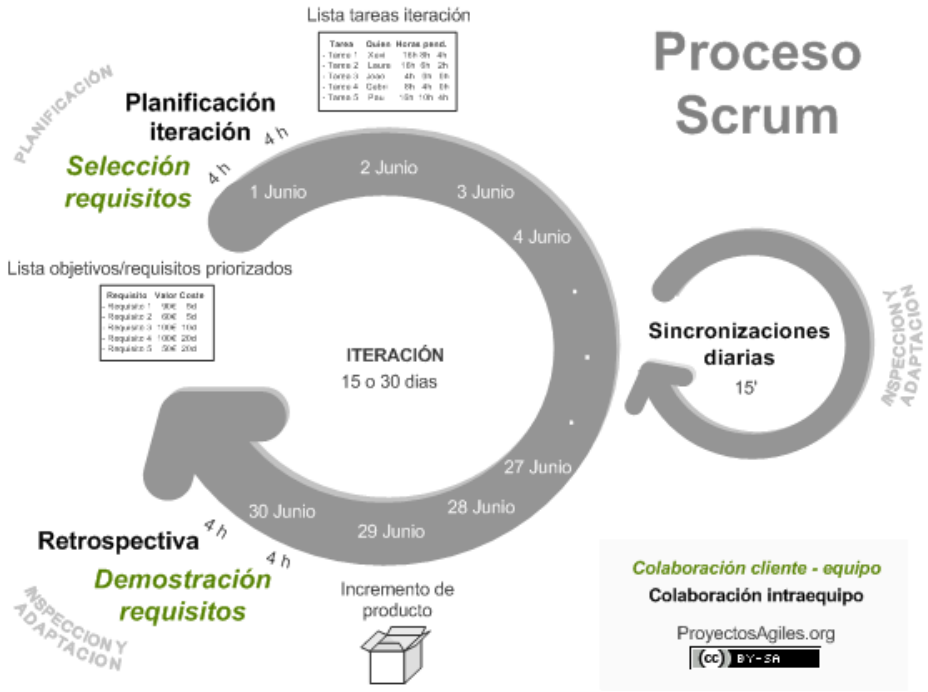


Figura 2 – Proceso Scrum – Tomado de: www.proyectosagiles.org

El proceso parte de la lista de objetivos/requisitos priorizada del producto, que actúa como plan del proyecto. En esta lista el cliente prioriza los objetivos balanceando el valor que le aportan respecto a su coste y quedan repartidos en iteraciones y entregas. De manera regular el cliente puede maximizar la utilidad de lo que se desarrolla y el retorno de inversión mediante la re-planificación de objetivos que realiza al inicio de cada iteración.

Las actividades que se llevan a cabo en Scrum son las siguientes:

Planificación de la iteración

El primer día de la iteración se realiza la reunión de planificación de la iteración. Tiene dos partes:

Selección de requisitos (4 horas máximo). El cliente presenta al equipo la lista de requisitos priorizada del producto o proyecto. El equipo pregunta al cliente las dudas que surgen y selecciona los requisitos más prioritarios que se compromete a completar en la iteración, de manera que puedan ser entregados si el cliente lo solicita.

Planificación de la iteración (4 horas máximo). El equipo elabora la lista de tareas de la iteración necesarias para desarrollar los requisitos a que se ha comprometido. La estimación de esfuerzo se hace de manera conjunta y los miembros del equipo se auto asignan las tareas.

Ejecución de la iteración

Cada día el equipo realiza una reunión de sincronización (15 minutos máximos). Cada miembro del equipo inspecciona el trabajo que el resto está realizando (dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo) para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido. En la reunión cada miembro del equipo responde a tres preguntas:
¿Qué he hecho desde la última reunión de sincronización?
¿Qué voy a hacer a partir de este momento?
¿Qué impedimentos tengo o voy a tener?
Durante la iteración el Facilitador se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad.
Elimina los obstáculos que el equipo no puede resolver por sí mismo.
Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

Inspección y adaptación

El último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes:

Demostración (4 horas máximo). El equipo presenta al cliente los requisitos completados en la iteración, en forma de incremento de producto preparado para ser entregado con el mínimo esfuerzo. En función de los resultados mostrados y de los cambios que haya habido en el contexto del proyecto, el cliente realiza las adaptaciones necesarias de manera objetiva, ya desde la primera iteración, re-planificando el proyecto.

Retrospectiva (4 horas máximo). El equipo analiza cómo ha sido su manera de trabajar y cuáles son los problemas que podrían impedirle progresar adecuadamente, mejorando de manera continua su productividad. El Facilitador se encargará de ir eliminando los obstáculos identificados.

Extractos de: Programación Extrema y el Software Libre

La Programación Extrema

La programación extrema se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la ingeniería del software. Usadas conjuntamente proporcionan una nueva metodología de desarrollo software que se puede englobar dentro de las metodologías ligeras, que son aquéllas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible (pero no más). La programación extrema, dentro de las metodologías ágiles, se puede clasificar dentro de las evolutivas.

Una de las características de extreme Programming es que muchos de, si no todos, sus ingredientes son de sobra conocidos dentro de la rama de la ingeniería del software desde hace tiempo, incluso desde sus comienzos. Los autores de han seleccionado los que han considerados como los mejores y han profundizado en sus relaciones y en cómo se refuerzan unos a otros. El

resultado ha sido una metodología única y compacta. Por eso, aunque se pueda alegar que la programación extrema no se base en principios nada nuevos, se ha de aclarar que, en conjunto, es una nueva forma de ver el desarrollo de software.

Aunque, como ya se ha comentado, la programación extrema se basa en unos valores, unos principios fundamentales y unas prácticas, en este artículo no se van a enumerar así de primeras, ya que el autor considera que no es la mejor forma de presentarlos. Los principios y prácticas no se han hecho a priori o porque sí, sino que tienen un porqué a partir de una forma global de desarrollar software que, al menos en teoría, parece ser más eficiente.

Por tanto, en este artículo se presentará la programación extrema desde un punto de vista práctico para luego dar paso a enunciar los valores y principios que se han extraído y las prácticas que hacen que se lleven a buen fin. La idea es seguir que el lector pueda seguir en los siguientes párrafos un proceso de desarrollo extremo tal y como debería darse en un equipo de desarrollo que siguiera la metodología XP. De esta forma se irán detallando y explicando las diferentes técnicas utilizadas, así como su razón de ser.

Una vez que hayamos visto el proceso de desarrollo extremo, los valores, principios y prácticas serán evidentes y no requerirán mucho detenimiento.
El proceso de desarrollo extremo

La programación extrema parte del caso habitual de una compañía que desarrolla software, generalmente software a medida, en la que hay diferentes roles: un equipo de gestión, un equipo de desarrolladores y los clientes. La relación con el cliente es totalmente diferente a lo que se ha venido haciendo en las metodologías tradicionales que se basan fundamentalmente en una fase de captura de requisitos previa al desarrollo y una fase de validación posterior al mismo.

Interacción con el cliente

En la programación extrema al cliente no sólo se le pide que apoye al equipo de desarrollo, en realidad podríamos decir que es parte de él. Su importancia es

capital a la hora de abordar las historias de los usuarios y las reuniones de planificación, como veremos más adelante. Además, será tarea suya realimentar al equipo de desarrolladores después de cada iteración con los problemas con los que se ha encontrado, mostrando sus prioridades, expresando sus sensaciones... Existirán métodos como pruebas de aceptación que ayudarán a que la labor del cliente sea lo más fructífera posible.

En resumen, el cliente se encuentra mucho más cercano al proceso de desarrollo. Se elimina la fase inicial de captura de requisitos y se permite que éstos se vayan definiendo de una forma ordenada durante el tiempo que dura el proyecto.

El cliente puede cambiar de opinión sobre la marcha y a cambio debe encontrarse siempre disponible para resolver dudas del equipo de desarrollo y para detallar los requisitos especificados cuando sea necesario.

El proceso de captura de requisitos de XP gira en torno a una lista de características que el cliente desea que existan en el sistema final. Cada una de estas características recibe el nombre de historias de usuarios y su definición consta de dos fases:

En la primera fase el cliente describe con sus propias palabras las características y el responsable del equipo de desarrollo le informa de la dificultad técnica de cada una de ellas y por lo tanto de su coste. A través del diálogo resultante el cliente deja por escrito un conjunto de historias y las ordena en función de la prioridad que tienen para él.

En este momento ya es posible definir unos hitos y unas fechas aproximadas para ellos.

La segunda fase consiste en coger las primeras historias que serán implementadas (primera iteración) y dividirlas en las tareas necesarias para llevarlas a cabo. El cliente también participa, pero hay más peso del equipo de desarrollo, que dará como resultado una planificación más exacta. En cada iteración se repetirá esta segunda fase para las historias planificadas para ella.

Este proceso es una de las principales diferencias con las metodologías tradicionales. Aunque las historias de usuarios guardan cierta relación con otras técnicas como los casos de uso de UML, su proceso de creación es muy diferente.

En lo que al cliente se refiere no se le exige que especifique exactamente lo que quiere al principio con un documento de requisitos de usuario. La parte que se mantiene con este documento es que es el cliente el que tiene que escribir lo que quiere, no se permite que alguien del equipo de desarrolladores lo escriba por él.

Como se ha comentado, son los desarrolladores los que se encargan de catalogar las historias de los usuarios y asignarles una duración. Para ello se sigue una norma simple: las historias de usuarios deberían poder ser abordables en un espacio de tiempo de entre una y tres semanas de programación ideal. Historias de los usuarios que requieran menos tiempo de implementación son agrupadas, mientras que aquéllas que necesiten más tiempo deben ser modificadas o divididas. Una semana de programación ideal es una semana (cinco días de trabajo) de desarrollo por parte de un desarrollador sin interferencias de otras partes del proyecto. Al hacer la planificación se aplica un factor de corrección medido de proyectos anteriores para ajustar este tiempo ideal al real.

Las historias de los usuarios se plasmarán en tarjetas, lo que facilitará que el cliente pueda especificar la importancia relativa entre las diferentes historias de usuario, así como la tarea de los desarrolladores que podrán catalogarlas convenientemente. El formato de tarjeta además es muy provechoso a la hora de realizar pruebas de aceptación.

9. DEFINICIÓN OPERACIONAL DE TÉRMINOS

Compiladores: Programa que convierte el lenguaje informático empleado por el usuario en lenguaje propio del computador.

Modelo de Ciclo de vida: Es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito.

Software Engineering Institute (SEI): es un instituto federal estadounidense de investigación y desarrollo, fundado por el Congreso de los Estados Unidos en 1984 para desarrollar modelos de evaluación y mejora en el desarrollo de software.

Métrica: es una cantidad insignificante que puede extraerse de algún documento o código dentro de un proyecto de software

Software a la medida: producto no genérico, que resuelve una serie de problemas en particular para determinada empresa o usuario.

Requerimientos: son especificaciones para un producto del software en particular, programa, o juego de programas que realizan ciertas funciones en un ambiente específico.

Diagrama de transición de estado: indica cómo se comporta el sistema como consecuencia de sucesos externos. Para lograr esto, el DTE representa los diferentes modos de comportamiento (llamados estados) del sistema y la manera en que se hacen las transiciones de estado a estado. El DTE sirve como la base del modelado de comportamiento. Dentro de la especificación de control (EC) se encuentra más información sobre los aspectos de control del software.

Iteración: Las iteraciones se pueden entender como mini proyectos: en todas las iteraciones se repite un proceso de trabajo similar (de ahí el nombre “iterativo”) para proporcionar un resultado completo sobre producto final, de manera que el cliente pueda obtener los beneficios del proyecto de forma incremental

10. CRONOGRAMA

CRONOGRAMA				
Actividades	**Días**	**Féc. Inicio**	**Féc. Fin**	**OBSERVACIONES**
1. Identificar los Procesos de documentación establecidos en el instructivo <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>>	**19**	**15/03/2010**	**13/04/2010**	
1.1 Analizar evolución del documento instructivo	5	15/03/2010	19/03/2010	
1.2 Identificar actuales actividades del documento instructivo	4	23/03/2010	26/03/2010	
1.3 Identificar Modelo de Ciclo de Vida, Paradigmas y Lenguajes de Programación	2	29/03/2010	30/03/2010	
1.4 Identificar actores responsables de cada proceso de documentación	2	31/03/2010	05/04/2010	
1.5 Estructurar con diagramas la información adquirida	6	06/04/2010	13/04/2010	
2. Estudiar alternativas de documentación vigentes actualmente en el mercado	**20**	**14/04/2010**	**11/05/2010**	
2.1 Identificar posibles Modelos de Ciclo de Vida	6	14/04/2010	21/04/2010	
2.2 Identificar Metodologías afines con el proceso	14	22/04/2010	11/05/2010	
3. Levantar información con los respectivos implicados en cada uno de los procesos relacionados con el documento instructivo	**9**	**12/05/2010**	**25/05/2010**	
3.1 elaborar herramienta para el levantamiento de información	5	12/05/2010	19/05/2010	
3.2 Entrevista con los actores relacionados con los procesos	4	20/05/2010	25/05/2010	
4. Estructurar resultados de la información adquirida y compararla con las alternativas estudiadas	**12**	**26/05/2010**	**11/06/2010**	
4.1 Identificar que procesos están realizando con respecto al documento instructivo	4	26/05/2010	31/05/2010	
4.2 Establecer la metodología y Modelo de ciclo de vida, apropiados para los proyectos del proceso	8	01/06/2010	11/06/2010	
5. Generar documento con observaciones y conclusiones a tener en cuenta para ser incluidas en el Proceso de documentación	**8**	**15/06/2010**	**24/06/2010**	
5.1 Recomendaciones	4	15/06/2010	18/06/2010	
5.2 Conclusiones	4	21/06/2010	24/06/2010	
TOTAL DÍAS	68			

Tabla 1 - Cronograma

10.1 DIAGRAMA DE GANTT

	FECHA DE INICIO: 15 de Marzo del 2010																
	MARZO				ABRIL				MAYO				JUNIO				OBSERVACIONES
No ACTIVIDAD A DESARROLLAR	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	
1. Identificar los Procesos de documentación establecidos en el instructivo <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>> 1.1 Analizar evolución del documento instructivo 1.2 Identificar actuales actividades del documento instructivo 1.3 Identificar Modelo de Ciclo de Vida, Paradigmas y Lenguajes de Programación 1.4 Identificar actores responsables de cada proceso de documentación 1.5 Estructurar con diagramas la información adquirida																	
2. Estudiar alternativas de documentación vigentes actualmente en el mercado 2.1 Identificar posibles Modelos de Ciclo de Vida 2.2 Identificar Metodologías afines con el proceso																	
3. Levantar información con los respectivos implicados en cada uno de los procesos relacionados con el documento instructivo 3.1 elaborar herramienta para el levantamiento de información 3.2 Entrevista con los actores relacionados con los procesos																	
4. Estructurar resultados de la información adquirida y compararla con las alternativas estudiadas 4.1 Identificar que procesos están realizando con respecto al documento instructivo 4.2 Establecer la metodología y Modelo de ciclo de vida, apropiados para los proyectos del proceso																	
5. Generar documento con observaciones y conclusiones a tener en cuenta para ser incluidas en el Proceso de documentación 5.1Recomendaciones 5.2 Conclusiones																	

Tabla 2 – Diagrama de Gantt

11. PRESENTACIÓN Y ANÁLISIS DE LOS RESULTADOS

11.1 IDENTIFICACIÓN DE LOS PROCESOS DE DOCUMENTACIÓN ESTABLECIDOS EN EL INSTRUCTIVO <<1-IN-030 ESTÁNDARES DE ANÁLISIS, DISEÑO Y DESARROLLO DE SISTEMAS DE INFORMACIÓN>>

11.1.1 Análisis de la evolución del documento instructivo. Esta es una tarea simplemente de reconocimiento de campo, pues al momento de estudiar la evolución del instructivo no se espera llegar a una conclusión final, ni mucho menos descubrir los problemas que tenga dicho instructivo, pues el tiempo en el cual se implementó no es el mismo contexto al actual, pues a simple vista es un documento muy rígido que requiere ser diligenciado completamente para ser aprobado. Durante las averiguaciones no se encontró referente conceptual de la evolución de dicho instructivo, pues fue creado sustrayendo practicas idóneas para el desarrollo de sistemas de información de otras metodologías ya establecidas y aceptadas mundialmente, al no existir el referente conceptual de estas, solo se concluye que es un documento que evolucionó a partir de las necesidades organizacionales de estandarización para ser una empresa de calidad, y la necesidad de que procesos internos requerían unas metodologías de trabajo estabilizadas para llegar a una comunicación fluida entre todos sus miembros.

11.1.2 Identificación de actuales actividades del instructivo <<Estándares de Análisis, Diseño y Desarrollo de Sistemas de información>> (Anexo A). Una vez definida la evolución que durante el tiempo ha tenido el instructivo en el proceso de Sistemas de Comfamiliar Risaralda, es válido establecer que este instructivo es modificado con periodicidad buscando acomodarlo a las tendencias, no solo del contexto en el que se encuentre la ingeniería del software en un momento determinado, sino también de las necesidades de la empresa.

En su primera instancia, el documento ubica al lector en las dimensiones en las que se debe mover, le menciona el objetivo del instructivo, y le explica conceptos técnicos básicos necesarios para poder desarrollar un proyecto de software utilizando este recurso en busca de la calidad. Al explicar los conceptos básicos, no se espera que cualquier lector pueda implementar un proyecto con este instructivo, pues es necesario tener unos conocimientos previamente adquiridos que se relacionan con el desarrollo del software para poder ejecutarlo. El

instructivo únicamente proporciona unos parámetros de documentación para aplicar conocimiento a nivel de ingeniería. Se establecen nuevamente otros parámetros, unas condiciones generales, dentro de las cuales se indica quienes deben participar en el proyecto y en casos especiales de modulo que es lo que se debe documentar. Luego se especifica que se debe seguir un estándar concerniente a la entidad como los colores, iconos, visualizaciones, distribuciones, botones, barras, entre otros elementos utilizados al momento del diseño e implementación de la interfaz grafica de los sistemas de información desarrollados en la empresa.

En este momento se procede al desarrollo del documento instructivo, el cual está subdividido principalmente en tres fases: Análisis, Diseño y desarrollo; luego se divide en: registro de modificación para sistemas de información, y en estándares para desarrollo. A continuación se detallan cada una de estas:

- Análisis

Las tareas y actividades aquí plasmadas son responsabilidad de los Analistas de procesos y de los Analistas de sistemas. Es importante ver que se muestra el conducto regular que se debe seguir al momento de inicio de un proyecto de software. Y es aquí mismo en donde se determina que tan pertinente es el desarrollo de este mismo, o si hay otras opciones más factibles para la solución del problema propuesto.

El camino a seguir es sencillo: se especifica qué área realiza el requerimiento y se hace un cronograma para el análisis, se especifica quienes y cuáles fueron las fuentes de información y se procede a examinar la situación actual del proceso, hay un levantamiento de información relacionado con el problema a solucionar teniendo en cuenta los parámetros establecidos por el sistema de gestión de calidad, y se recomienda construir un diagrama de flujo de cómo es el comportamiento del proceso a sistematizar, una descripción de las limitantes que tiene el proceso con respecto al futuro sistema, unos beneficios, hay una identificación de la complejidad del problema, una solución tentativa y unos objetivos. Finalmente se determina un diagnostico del problema, una vez analizadas las alternativas se elige cual solución es más conveniente: adquirir por medio de proveedores externos un nuevo sistema de información, el proceso no tiene una solución informática, creación de un nuevo sistema de información, o si es un modulo; para los dos primeros casos el documento termina aquí.

Este establecimiento de la viabilidad del sistema es importante al momento de aplicar ingeniería, pues evita esfuerzos innecesarios en procesos, tal vez “quemados” y subutilizados. Permite en caso de ser viable tener una base de arranque y fundamentada de los requerimientos del usuario para continuar con la siguiente fase del proceso.

- Diseño

En esta fase del proyecto es donde convergen el grupo de Analistas con los desarrolladores para aplicar técnicas y principios propios de Ingeniería de Software, definiendo los aspectos y parámetros del sistema de información próximo a construir a partir del documento de Análisis. Por medio de gráficos, diagramas y cuadros modelan cada uno de los requerimientos que el usuario hizo, para permitir la interpretación y realización lógica del sistema.

Inicialmente se define el alcance que tendrá el desarrollo plasmando unos objetivos que luego se desglosaran en un cronograma para su cumplimiento, en las restricciones como su nombre lo indica se hace referencia a las limitantes y reglas del negocio que tendrá el sistema de información.

Para modelar el comportamiento del nuevo sistema se acude a Casos de Uso en los niveles que sean necesarios para que el equipo de desarrollo comprenda completamente los requerimientos, posteriormente se listaran los perfiles, diagramas de motivos o diagramas de transición de estado, el modelo entidad relación, diccionario de datos y diagramas de funciones y navegación.

Una vez terminado este documento se tienen los argumentos suficientes para construir un sistema de información de manera ordenada y eficiente, es decir, los planos del proyecto están listos para empezar a edificar, y el grupo de ingenieros ya solucionaron el problema temporalmente de forma teórica.

- Desarrollo/ Manual Técnico

Ahora el grupo de analistas ha terminado gran parte de su trabajo quedando como apoyo, y es función de los desarrolladores este documento.

Se le pide a cada programador que especifique cual tarea se le fue asignada, y comience a documentar su desarrollo de la siguiente forma: en caso de que el modelo entidad-relación haya sido modificado con respecto al original se anexa el nuevo; luego describe detalladamente las restricciones en el funcionamiento; hace un listado de clases y funciones; un glosario de términos; convenciones pactadas en caso de que existan y finalmente el cronograma de desarrollo.

Todo esto con el fin de que no se restrinja a un solo programador el entendimiento del proyecto, haciendo fácil las actividades de mantenimiento y de corrección de errores, de igual forma haciéndolo reutilizable en cualquier momento.

- Registro de Modificación para Sistemas de Información

Este documento es implementado como su nombre lo indica al momento de una modificación en un sistema de información, el responsable del mismo es de cada Interventor del sistema. Éste se reúne con todo el personal relacionado con el sistema para comunicarles el cambio y ver la factibilidad de él.

Una vez aprobado, se completa un formato en el cual se documentan: Nombre, Responsable, Tipo de Cambio, Creación modificación y eliminación de funciones, Creación de un modulo, Justificación, vigencia y se anexa el diseño y desarrollo.

Lo anterior a manera de ilustración de los cambios para no perder la calidad del sistema ya en funcionamiento.

- Estándares de Desarrollo

Finalmente se pacta la nomenclatura que se utilizará para el desarrollo de las aplicaciones, los parámetros que se deben seguir para programar de una manera ordenada y entendible para otros, haciendo legible en todo momento la estructura de las líneas de código, de las tablas, los campos, las carpetas raíz.

Con esta etapa concluye la identificación de actividades, que en conclusión son actividades propias de Ingeniería de Software, quedando por identificar parcialmente la documentación de la implantación y mantenimientos del sistema.

Por el momento no se pretende evaluar el desempeño ni la capacidad del instructivo para desarrollar un producto de calidad, y no es tampoco un objetivo del presente proyecto, simplemente se identifican y se describe globalmente las actividades para una mejor contextualización del proceso que actualmente se lleva al momento de desarrollar software.

11.1.3 Identificación de Modelo de Ciclo de Vida, Paradigmas y Lenguajes de Programación. Luego de un proceso de análisis e identificación de las tareas del documento instructivo (Ver Anexo A), se procede a identificar otras prácticas que a nivel de ingeniería son relevantes para el proceso de la creación de software que actualmente se llevan a cabo en el proceso de sistemas de Comfamiliar Risaralda.

Para la identificación del primer punto, es necesario recordar que la entidad ha optado por el camino de que la mayoría de sus desarrollos se haga en "Casa", es decir, el proceso de sistemas está capacitado completamente para desarrollar software y aplicarlo en la empresa, como se dice comúnmente "Software a la Medida". Ahora bien, por la robustez de la compañía y la gran variedad de servicios que ofrece, el proceso de sistemas obtiene un número significativo de solicitudes para diferentes sistemas de información, por lo cual necesita un acompañamiento por parte del proceso al cual se le desarrolla el aplicativo. Con el pasar del tiempo el grupo de ingenieros logra establecer un modo de operación que le ha resultado eficaz al momento de desarrollar proyectos de sistemas de información, este modo de operación funciona de la siguiente manera: se obtienen los requisitos, se analiza y se diseña, se hace una versión la cual es mostrada al Cliente (proceso solicitante del aplicativo) quien da muestra de una conformidad haciendo nuevos requerimientos, estos requerimientos generan una nueva versión del software la cual es puesta a prueba nuevamente…; repitiendo el ciclo se llega al refinamiento del aplicativo, que avanza desde forma primitiva hasta el estado avanzado que se solicitó.

"Un cliente, a menudo, define un conjunto de objetivos generales para el software, pero no identifica los requisitos detallados de entrada, proceso o salida. En otros casos, el responsable del desarrollo del software puede no estar seguro de la eficacia de un algoritmo, de la capacidad de adaptación de un sistema operativo, o de la forma en que debería tomarse la interacción hombre-máquina".[1] Por esto, por lo dicho anteriormente y por otras razones que define el modelo se puede

concluir que el proceso de sistemas trabaja actualmente con un ciclo de vida de construcción de prototipos.

** Para la identificación del segundo y tercer punto de esta actividad se estableció contacto directo con el equipo de sistemas a través de encuentros informales*.

Segundo punto*. Actualmente para el desarrollo de software se utilizan dos paradigmas: Estructurado y Orientado a Objetos, este último se viene implementando y posicionando cada vez más por todos los beneficios que posee al momento de programar software robusto, "…a medida que los programas se hacen más grandes y complejos, el paradigma estructurado comienza a dar señales de debilidad y resultando muy difícil terminar los programas de un modo eficiente. Existen varias razones de la debilidad de los programas estructurados… …Al contrario que la programación procedimental que enfatiza en los algoritmos, la POO enfatiza en los datos. En lugar de intentar ajustar un problema al enfoque procedimental de un lenguaje, POO intenta ajustar el lenguaje al problema. La idea es diseñar formatos de datos que se correspondan con las características esenciales de un problema.".[2]

Tercer punto*. En los actuales proyectos de software se destacan tres únicos lenguajes de programación, que obedeciendo con los paradigmas utilizados son: Visual Basic, VB.Net y Java. En muchas ocasiones estos lenguajes convergen en un mismo sistema de información, por lo cual la claridad y legibilidad al momento de programar se hace una labor importante.

11.1.4 Identificación de los actores responsables de cada proceso de documentación. Actualmente el área de desarrollo del proceso administrativo de sistemas encargado de todos los proyectos de sistemas de información de la entidad está compuesto por: Coordinador de Desarrollo, Analistas de Sistemas, Programadores de Sistemas y Analistas de Procesos. Todos estos tienen un papel dentro del proceso de documentación de software, pues durante todo el transcurso debe existir unos productos de salida, y en este caso unos documentos que modelan el comportamiento inicialmente planteado, luego implementado y finalmente probado de un aplicativo durante su ciclo de vida.

1. PRESSMAN, Roger S. Ingeniería del Software. Un enfoque práctico.

2. PROGRAMACIÓN EN C++. Algoritmos, estructuras de datos y objetos (2ª ed.)

Para el instructivo que se está trabajando <<Estándares de Análisis, Diseño y Desarrollo de Sistemas de información>> su intervención se da de la siguiente manera:

El Coordinador de Procesos, Los Analistas de Sistemas Y Analistas de procesos intervienen durante el documento de análisis, pues son ellos los encargados de la primera etapa de documentación de la aplicación que será efectiva a través del grupo de desarrollo, los analistas se apoyan en el coordinador para evaluar todas las eventualidades que surjan a partir de este análisis del nuevo sistema de información partiendo de los requisitos que se levanten.

Durante la realización del diseño del sistema de información, convergen los Analistas de Sistemas, Analistas de Procesos y los Programadores de Sistemas para moldear el comportamiento que este tendrá al momento de ser implementado, apoyándose durante todo este ciclo en el Coordinador de procesos como en la etapa anterior.

En esta etapa los programadores de sistemas realizan su tarea correspondiente, documentándola y apoyándose tanto en los analistas como en el coordinador de desarrollo para garantizar un producto de calidad.

En caso de que se necesite alguna modificación, se reúnen todos los actores involucrados en la creación y desarrollo del sistema de información en particular.
Cabe referenciar al cliente (Proceso solicitante del aplicativo) como un actor involucrado, pues por el ciclo de vida en el que se desarrollan los sistemas de información es parte fundamental para su éxito.

11.1.5 Diagramas de Información Adquirida

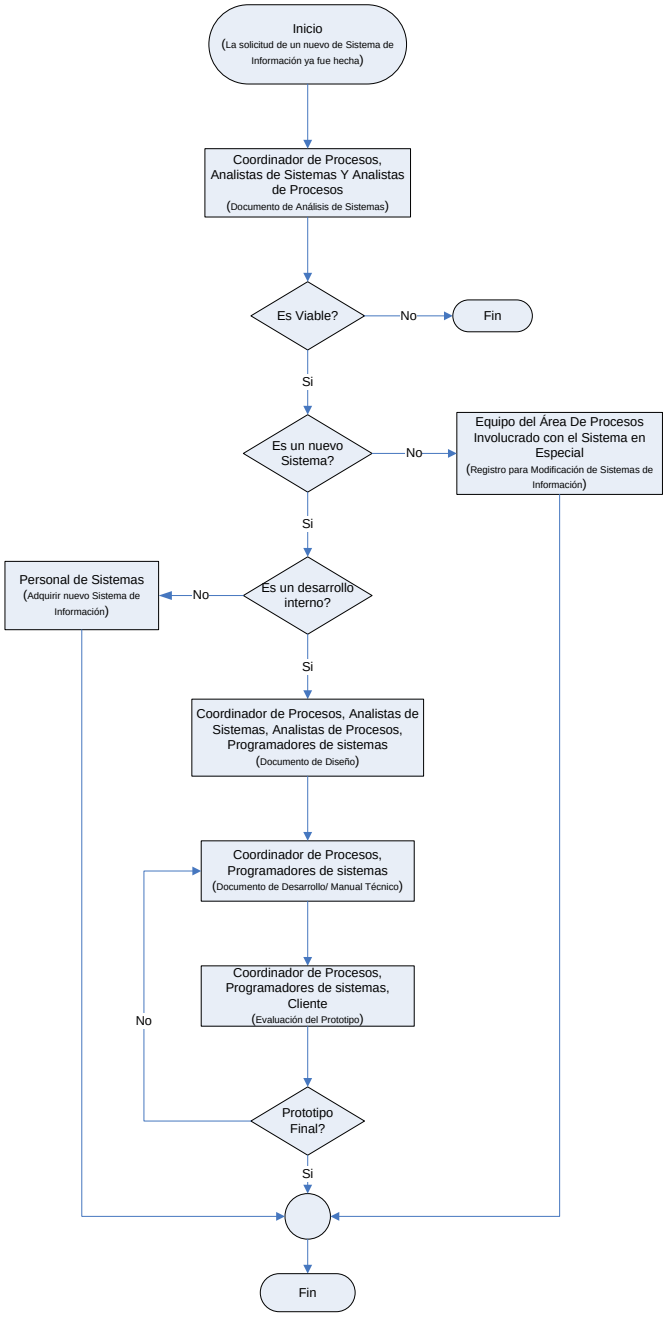


Figura 3 - Diagrama de flujo de un sistema de información

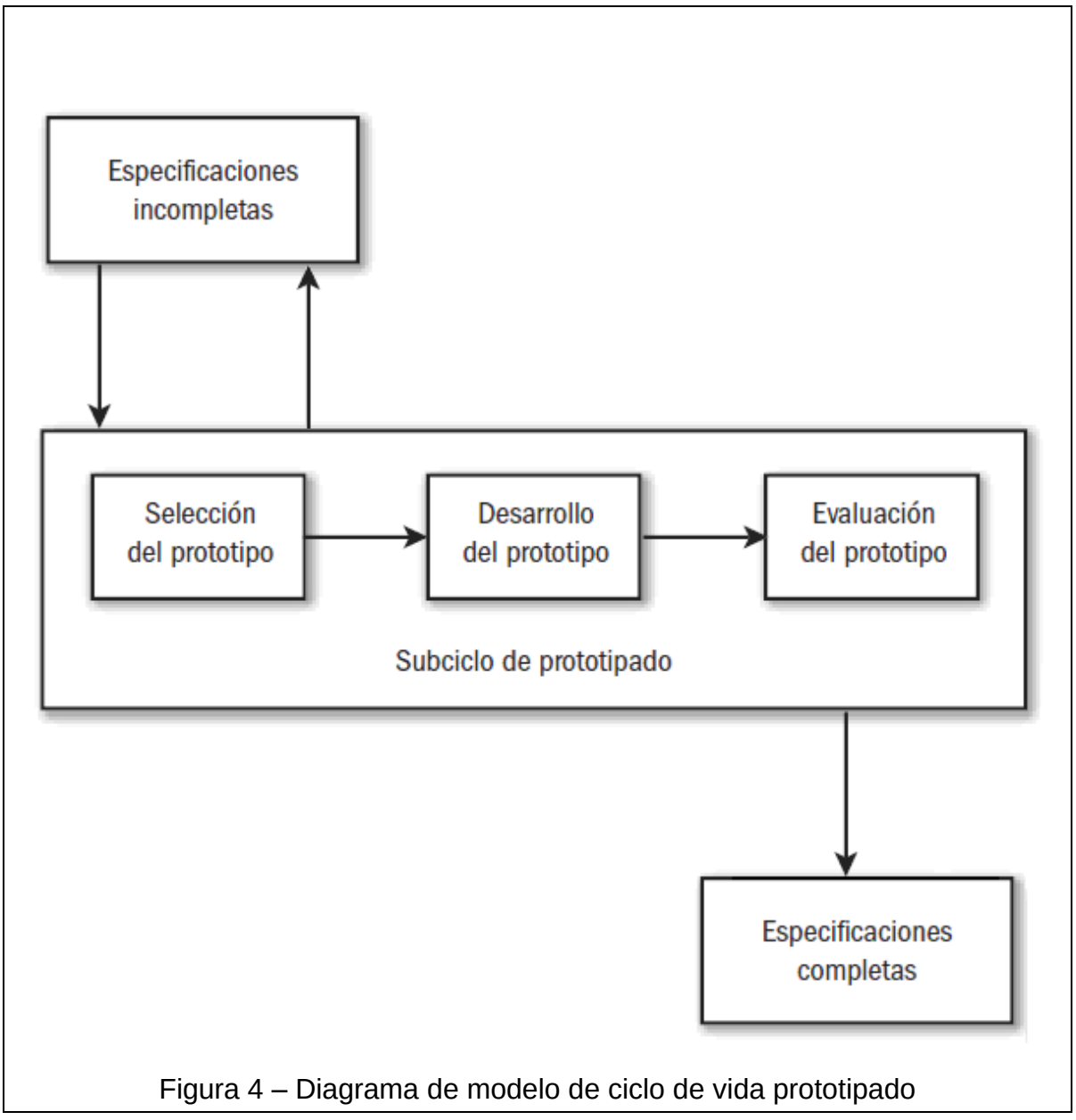


Figura 4 – Diagrama de modelo de ciclo de vida prototipado

11.2. ESTUDIO DE ALTERNATIVAS DE DOCUMENTACIÓN VIGENTES ACTUALMENTE EN EL MERCADO

Una vez analizado completamente el instructivo en la etapa anterior de este proyecto, en donde se logra: contextualizar el ambiente en el que este se desarrolla, ver como es utilizado por los ejecutantes del proceso de sistemas al momento de elaborar un software requerido por la entidad, y sin concluir que tan pertinente es el documento se modela el comportamiento actual del mismo. A continuación se disponen una serie de alternativas que se acomodan a la forma de trabajar del equipo, sin decir cuál es la más adecuada para el actual proceso de sistemas, solo se enuncian en búsqueda de una solución que se dará más adelante como objetivo principal de este proyecto

11.2.1 Identificación de posibles Modelos de Ciclo de Vida. Siendo consecuente con el modelo de desarrollo que ha evolucionado en el proceso de sistemas de Comfamiliar Risaralda se podría establecer que las alternativas en cuanto al modelo de ciclo de vida se enmarcan dentro de las categorías de: Modelos de ciclo de vida evolutivos. Es importante entender que se busca un modelo que involucre tareas más sencillas pero sin despreciar la complejidad al momento del desarrollo para obtener cada vez un producto de mejor calidad. Por ello los ciclos de vida que se analizarán son propiamente desarrollados a partir del modelo de construcción de prototipos. Para la identificación de los mismos se hará una breve descripción de los más afines con el proceso, y finalmente se procede a evidenciar un cuadro comparativo entre cada uno de ellos:

- Modelo incremental

Lo que caracteriza este ciclo de vida al igual que el actual usado en la compañía, es que los desarrolladores conocen los requisitos pero también tienen conocimiento que con el tiempo estos mismos van cambiando, por lo cual el sistema planteado inicialmente no es el real que se ejecuta finalmente.

Combina el modelo lineal con el prototipado, dando como resultado un modelo lineal reiterado: cada evolución es un pequeño pedazo del proyecto utilizable construido sobre el pedazo anterior.

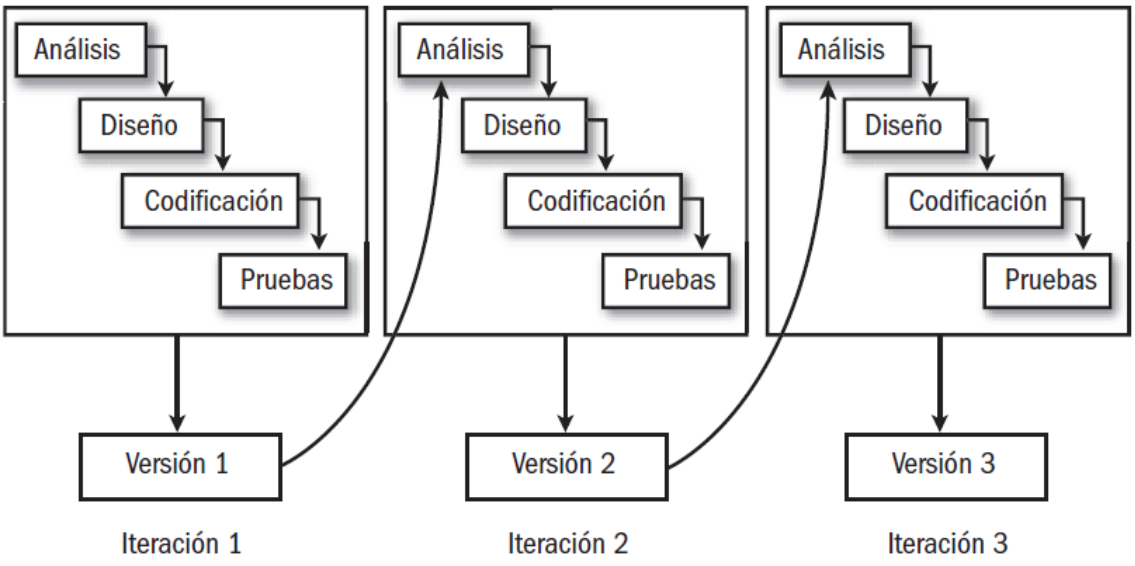


Figura 5 – Modelo de ciclo de vida incremental

-Modelo Espiral

Para este modelo se emplean tareas más específicas que en el incremental, se detallan aun mejor las actividades de cada etapa y como se debería elaborar cada una para lograr una disminución notable en el riesgo de no lograr el software deseado. Estos riesgos se detallan y se especifica que tan viable es la solución de ellos.

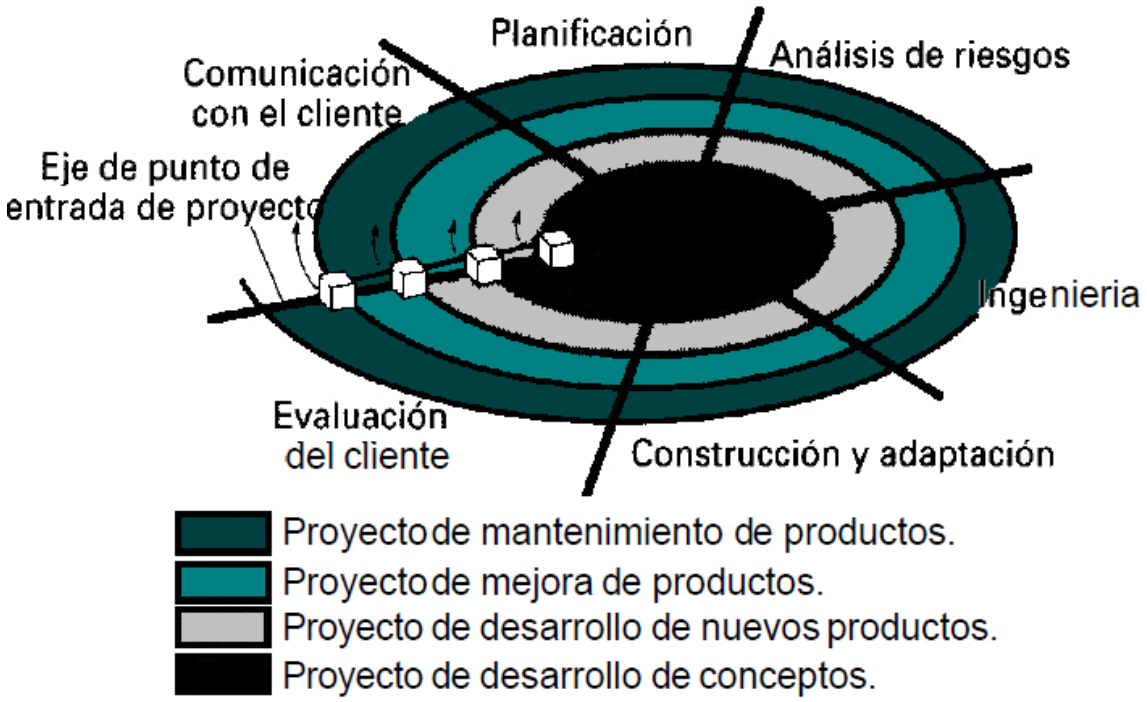


Figura 6 – Modelo de ciclo de vida en espiral – extraído de Ingeniería del Software. Un enfoque práctico

-Modelo basado en Componentes

Enfatiza la creación de clases encapsulando los datos o algoritmos que se utilizan para manejar los datos, la actividad de ingeniería da inicio cuando se identifican una serie de clases candidatas las cuales se depuran por medio de una biblioteca de clases creada con clases de proyectos anteriores y se decide que clases son pertinentes, cuales se tienen que crear y cuales se pueden extraer de la biblioteca, el análisis previo del sistema de información es necesario y el ensamblaje posterior de las clases seleccionadas junto con la aprobación del cliente conforman la primera iteración del ciclo, sucesivamente por medio de prototipos que evolucionan con el tiempo se llega al producto final.

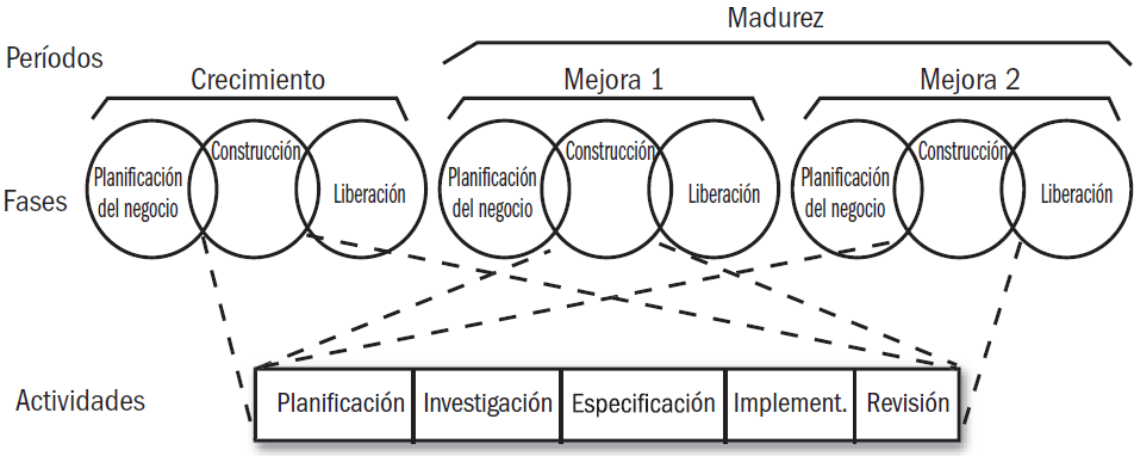


Figura 7 – Modelo de ciclo de vida basado en componente

- Cuadro comparativo de Modelos de Ciclo de Vida

Ciclo de vida	Características
Modelo Incremental	• Se repiten las etapas de requisitos, desarrollo y evaluación. • Se pueden hacer varios incrementos en paralelo, eso depende de la complejidad y el tipo de desarrollo que se esté haciendo • Forma en cascada escalonada • Hay alto riesgo de que el proyecto no tenga el fin deseado, por la velocidad en la que se trabaja, es decir es un modelo para desarrollos rápidos
Modelo en Espiral	• Dependiendo el nivel de complejidad del desarrollo las tareas pueden ser más o menos formales. • Puede tardar mucho tiempo un desarrollo • Es difícil de implementar y controlar
Modelo basado en Componentes	• Es propiamente evolutivo y está muy relacionado con el modelo en espiral. • Es un enfoque basado en escenarios, es actualmente efectivo para la construcción de grandes sistemas y aplicaciones de software pues se modela de forma parecida al paradigma de programación orientada a objetos. • Altamente reutilizable

Tabla 3 – Cuadro Comparativo

11.2.2 Identificación de Metodologías afines con el proceso. En esta sección se pretende hacer una radiografía general de metodologías consultadas, (que a criterio y estudios en el mercado por parte del autor pueden ser identificadas como afines para el proceso de Sistemas de Comfamiliar Risaralda) verificando la importancia de las tareas en algunas etapas del proceso de construcción de software a nivel de ingeniería, y principalmente durante las tres etapas: Análisis, Diseño y Desarrollo.

- Documento de Análisis

Durante esta primera etapa, se encuentra una tarea primordial para que el sistema tenga éxito, detallando escrupulosamente lo que se pide en ésta el desarrollo del sistema se realizará con un rango de riesgo demasiado reducido, pues durante las siguientes fases la cantidad de errores se reduce, y el tiempo de desarrollo del software podrá cumplir lo pactado con el cliente. Esta primera tarea es la especificación de los requisitos del software. Esta etapa se refiere específicamente al análisis, especificación y validación de los requisitos del software. Para este punto se analizaran las propuestas de las metodologías presentadas por la IEEE, y por la metodología Métrica V3.

IEEE-STD-830-1998: Especificaciones de los requisitos del software

Este documento considera que para que un análisis de requisitos tenga éxito debe cumplir con las siguientes características: Correcto, inequívoco, completo, consistente, delineación de importancia y estabilidad, comprobable, modificable y por ultimo identificable. Da recomendaciones de cómo lograr cada una de las características para lograr una especificación de requisitos correcta.

Según las recomendaciones de este documento un buen análisis de requisitos debe estar estructurado de la siguiente manera:

1. Introducción
1.1 Propósito
1.2 Alcance
1.3 Definiciones, siglas, y abreviaciones
1.4 Referencias
1.5 Apreciación global
2. Descripción global
2.1 Perspectiva del producto
2.2 Funciones del producto
2.3 Características del usuario
2.4 Restricciones
2.5 Atención y dependencias
3. Los requisitos específicos
Apéndices
Índice

Swebok: esta guía recoge los aspectos más importantes de todas las normas ISO relacionados con el software y para el análisis de requerimientos plasma el siguiente esquema:

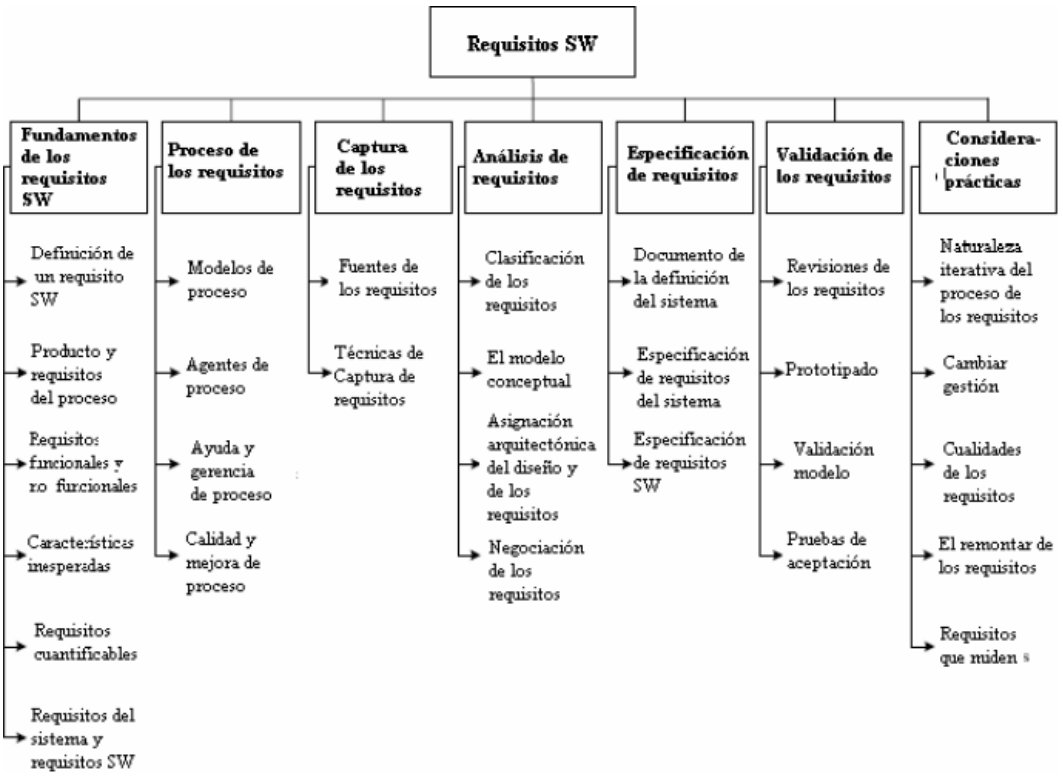


Figura 8 – Diagrama de requisitos - SWEBOK

"No toda organización tiene una cultura de documentación y manejo de requisitos. Es frecuente en las compañías de Start-up dinámicas, conducidas por una "visión fuerte del producto" y recursos limitados, para ver la documentación de los requisitos como gastos indirectos innecesarios. Más a menudo, sin embargo, según estas compañías crecen, mientras que su base de cliente crece, y como su producto comienza a desarrollarse, estas descubren que necesitan recuperar los requisitos que las características de producto motivadas para determinar el impacto de cambios propuestos. Por lo tanto, la documentación de los requisitos y la gerencia del cambio son llave al éxito de cualquier proceso de los requisitos."[3]

3 Swebok, Capitulo II, Requerimientos del Software

En Métrica V3, a diferencia de las dos metodologías anteriores en esta se examina primero el estado de los sistemas de información de la organización, la viabilidad del sistema, las opciones que se ofrecen para solucionar el problema, y luego se especifican y se analizan los requisitos, durante el análisis se especifican

inicialmente los posibles casos de uso, y otros procesos organización, la viabilidad del sistema, las opciones que se ofrecen para solucionar el problema, y luego se especifican y se analizan los requisitos, durante el análisis se especifican inicialmente los posibles casos de uso, y otros procesos relacionados con el diseño del sistema de información, en el siguiente grafico se puede observar el esquema general que propone, y hace una diferenciación de las tareas en caso de utilizar un paradigma u otro para programar.

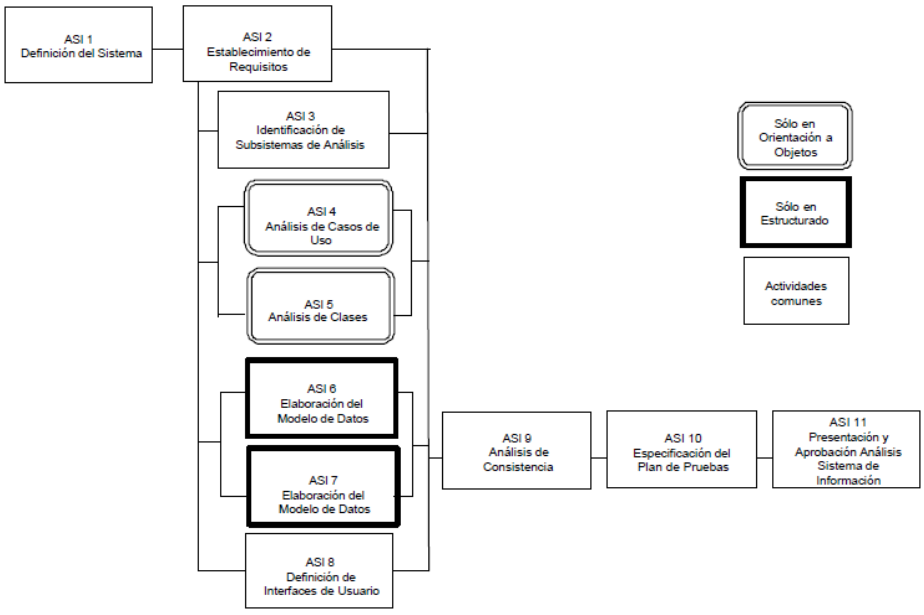


Figura 9 – Análisis de sistema de información – Métrica V3

- Documento de diseño

Durante esta etapa se modelan los requisitos obtenidos anteriormente con la finalidad de estructurar teóricamente el sistema de información a construir, con ayuda de técnicas propias de ingeniería se logra este objetivo. Se consideraron las siguientes metodologías para esta etapa: propuestas de IEEE, Métrica V3 y Moprosoft.

IEEE Std 1016-1998, Recommended Practice for Software Design, recomienda la siguiente estructura, pero no entra en detalle de las prácticas de ingeniería a seguir para lograr un diseño de calidad:

1 Introduction
Paragraph describing the context and intent of this design. What project is this part of? What is the Project trying to achieve? How does this software relate to the project?
1.1 Purpose
What is the purpose of this document? Who is the audience for this document? Different organizations will have different views of the role the SDD plays in the software development life cycle – this is the place to describe that role.
1.2 Scope
Describe the scope that this document is attempting to cover. How does this document relate to the requirements? Is the design attempting to satisfy the entire requirements or only part?
1.3 Definitions and acronyms
Include definitions of terminology used and descriptions of any acronyms or abbreviations that are used within the SDD.
2 References
Include any references. For example, this document has referenced IEEE Std 830-1998 and IEEE Std 1016-
1998.
IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications
IEEE Std 1016-1998 IEEE Recommended Practice for Software Design Descriptions
3 Decomposition description
This section describes how various aspects of the design are decomposed into smaller parts that make the design more manageable.

3.1 Module decomposition
This section describes structure of the design, or in other words, how the design is decomposed into functional modules (packages or classes).
This section should describe how the modules described in the subsections relate to each other.
3.1.1 Module 1 description
3.1.2 Module n description
3.2 Concurrent process decomposition
This section describes purpose and communication between processes or threads created by the software. For software is a single thread of execution it is sufficient just to say that.
3.2.1 Process 1 description
3.2.2 Process n description
3.3 Data decomposition
Describe the data that is manipulated or stored by the software. Does the design create any data structures that do not correspond to the modular decomposition described above? Does the design access external data stores?
3.3.1 Data entity 1 description
3.3.2 Data entity n description
4 Dependency Description
What dependencies exist between the decomposed modules; between the software's processes; and between the software's data structures?
4.1 Intermodule dependencies
4.2 Interprocess dependencies
4.3 Data dependencies
5 Interface Description
Describe the interfaces of each module or process. How does one module communicate with others? How does one process communicate with others?
5.1 Module Interfaces
Describe the module interfaces. How is a module initialized, or instantiated? What are the available operations upon each module?
5.1.1 Module 1 description
5.1.2 Module n description
5.2 Process Interfaces
Describe how the processes communicate. Do the processes communicate using shared memory, or do they invoke methods using a variant of remote procedure call?
5.2.1 Process 1 description
5.2.2 Process n description
6 Detailed Design
6.1 Module detailed design

6.1.1 Module 1 detail
6.2 Data detailed design
6.2.1 Data entity 1 detail

Swebok: “El diseño se define en [IEEE610.12-90] como “el proceso para definir la arquitectura, los componentes, los interfaces, y otras características de un sistema o un componente” y “el resultado de este proceso.””[4] en esta metodología se detalla de mejor forma los pasos a seguir para lograr un producto de calidad por medio de un buen diseño de software, su definición sería la siguiente:

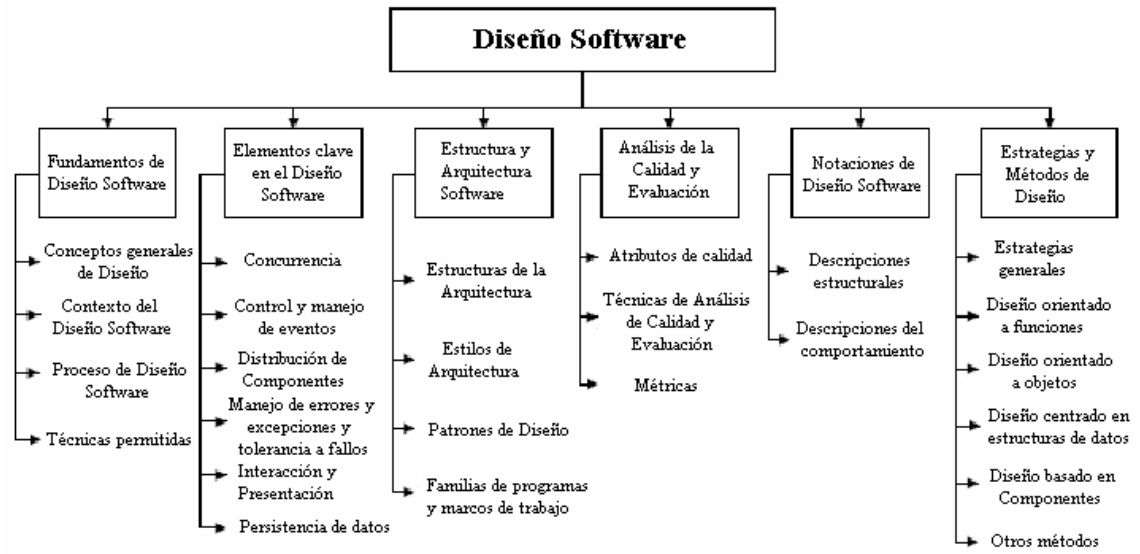


Figura 10 – Diseño de software - SWEBOK

Métrica V3, como en la etapa anterior, esta metodología pide que cada tarea sea bien planificada, estructurada y documentada, tiene en cuenta diferentes paradigmas para desarrollar esta metodología, de forma muy completa detalla explica de qué forma se tiene que desarrollar cada una de las tareas a nivel de ingeniería para lograr el objetivo de un producto de calidad.

4 Swebok Capítulo III, Diseño de Software

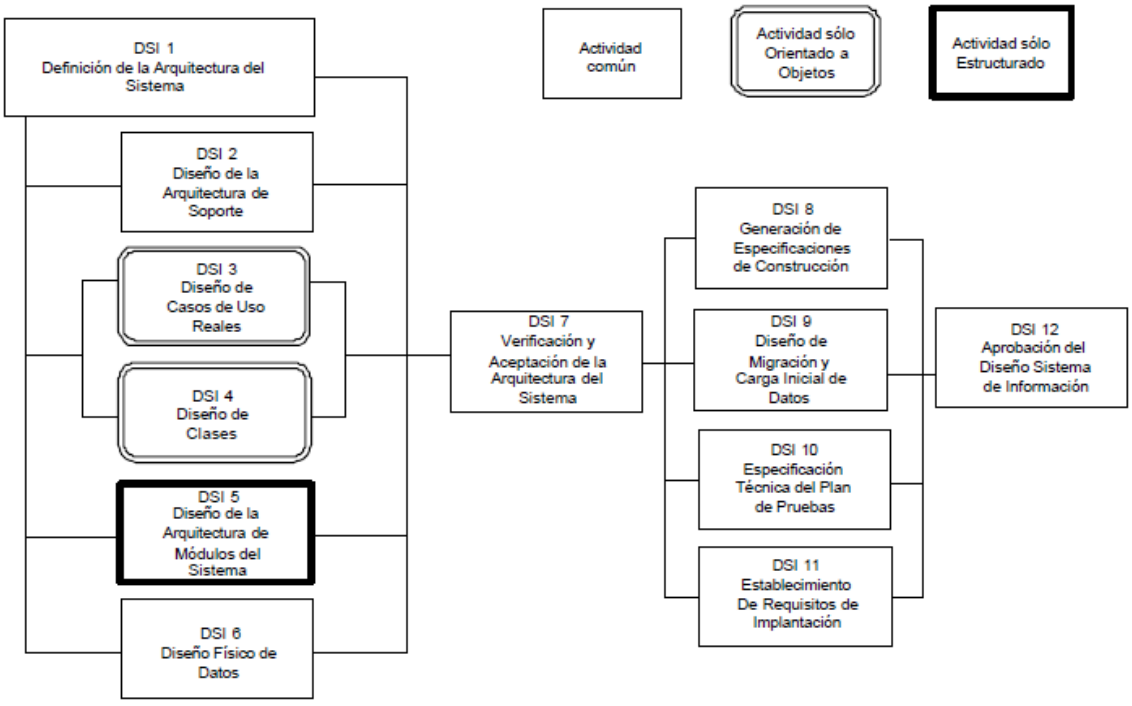


Figura 11 – Diseño del sistema de información – Métrica V3

La metodología Moprosoft, entrega una propuesta en la cual en una misma etapa cubre el análisis, diseño, desarrollo y mantenimiento del software, puede ser relevante para desarrollar un software, pero no concentra el documento esencialmente en el desarrollo de este, sino en el desarrollo de un proyecto informático en donde el software es un articulo mas. Este diagrama muestra como se propone el proceso de creación de software:

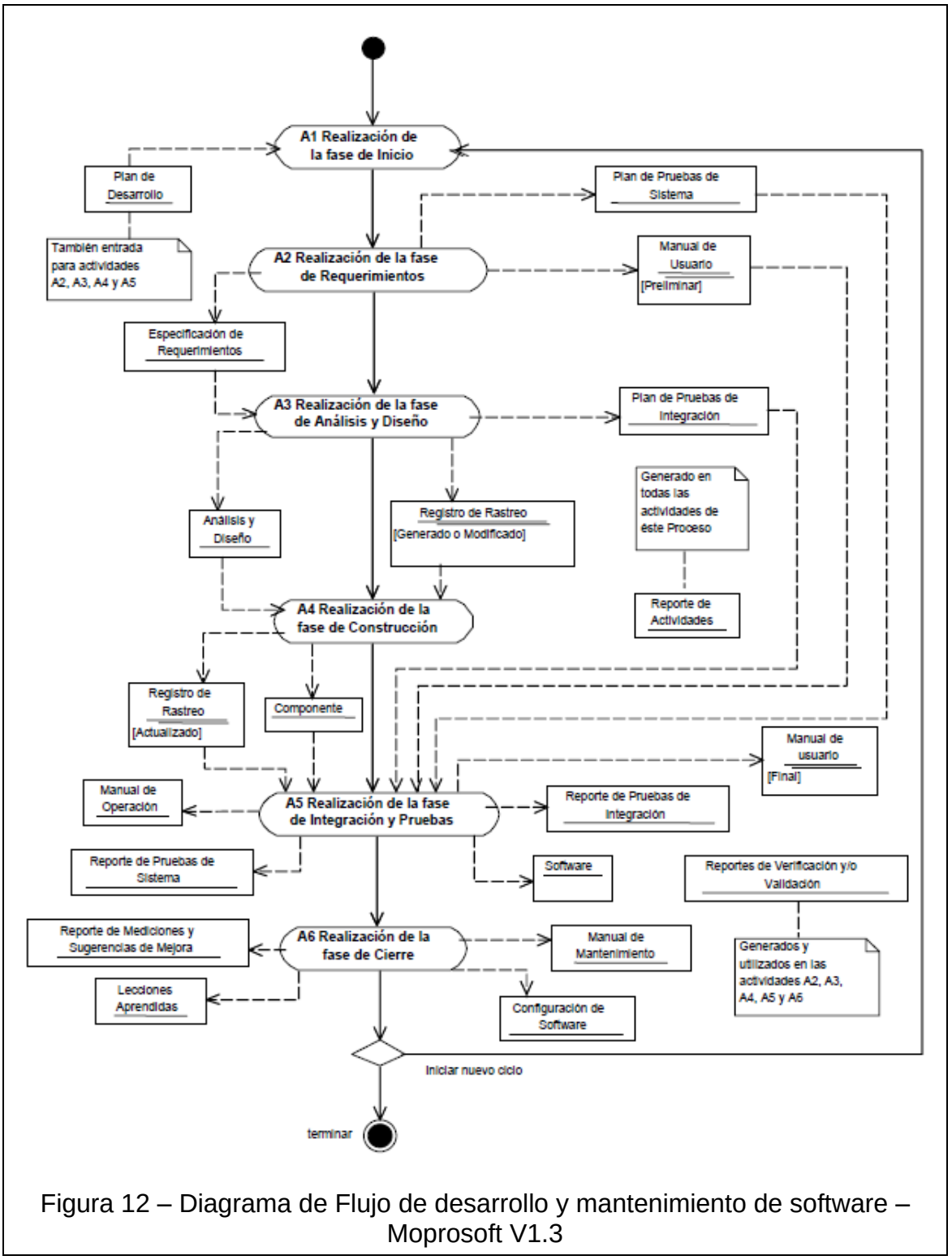


Figura 12 – Diagrama de Flujo de desarrollo y mantenimiento de software – Moprosoft V1.3

- Documento de Desarrollo

Este documento está relacionado directamente con el documento de diseño pues es aquí en donde los desarrolladores materializan, metafóricamente, la solución teórica propuesta por los analistas, en gran mayoría de las ocasiones el desarrollo genera modificaciones en el diseño, y esto no es un punto negativo, por el contrario es retroalimentación en búsqueda de un producto desarrollado con las mejores técnicas capaz de exaltarse por su calidad, en esta sección solo se verán involucrados Swebok y Métrica V3.

En Swebok se propone un modelo de construcción de software muy bien estructurado donde se descompone cada tarea y se presenta una forma de medir la calidad mientras se está construyendo el software, la siguiente es la estructura propuesta por esta metodología en donde las pruebas hacen parte fundamental del desarrollo de un sistema de información.

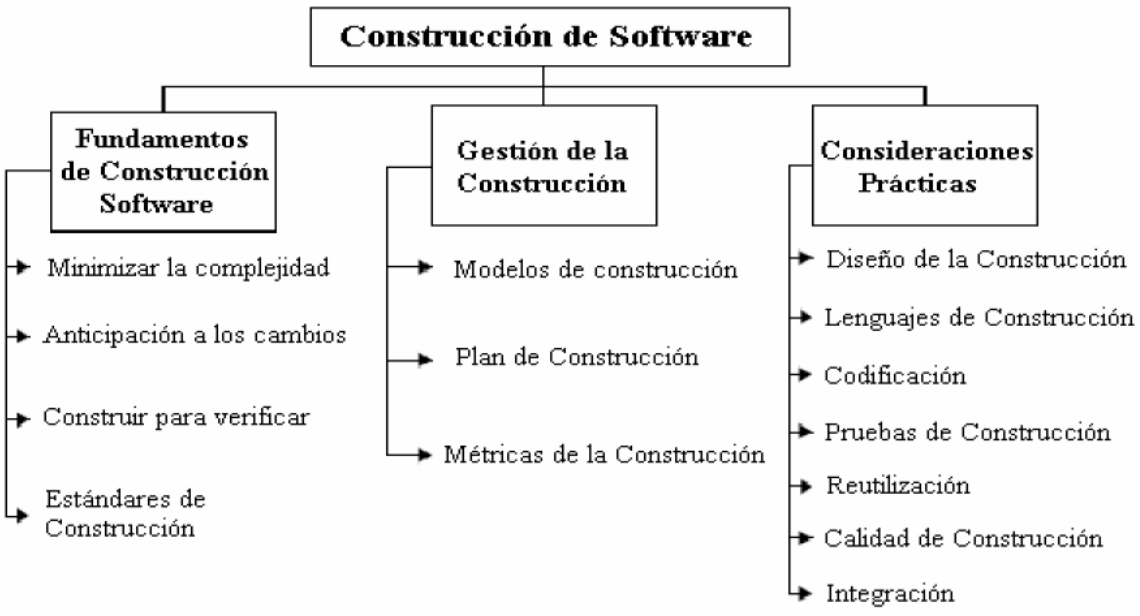


Figura 13 – Construcción del software - SWEBOK

“Entre las Disciplinas Descritas de la Ingeniería del Software el KA (Área del Conocimiento) de construcción del Software es lo más parecido a la ciencia informática en su uso del conocimiento de algoritmos y de las prácticas detalladas de codificación, ambas son consideradas, con frecuencia, como pertenecientes al dominio de la ciencia informática. También está relacionada con la gestión del proyecto en la medida en que la gestión de la construcción pueda presentar retos considerables.”[5]

En Métrica V3 dice que “En este proceso se genera el código de los componentes del Sistema de Información, se desarrollan todos los procedimientos de operación y seguridad y se elaboran todos los manuales de usuario final y de explotación con el objetivo de asegurar el correcto funcionamiento del Sistema para su posterior implantación.”[6] A continuación se presenta la estructura que está completa metodología propone:

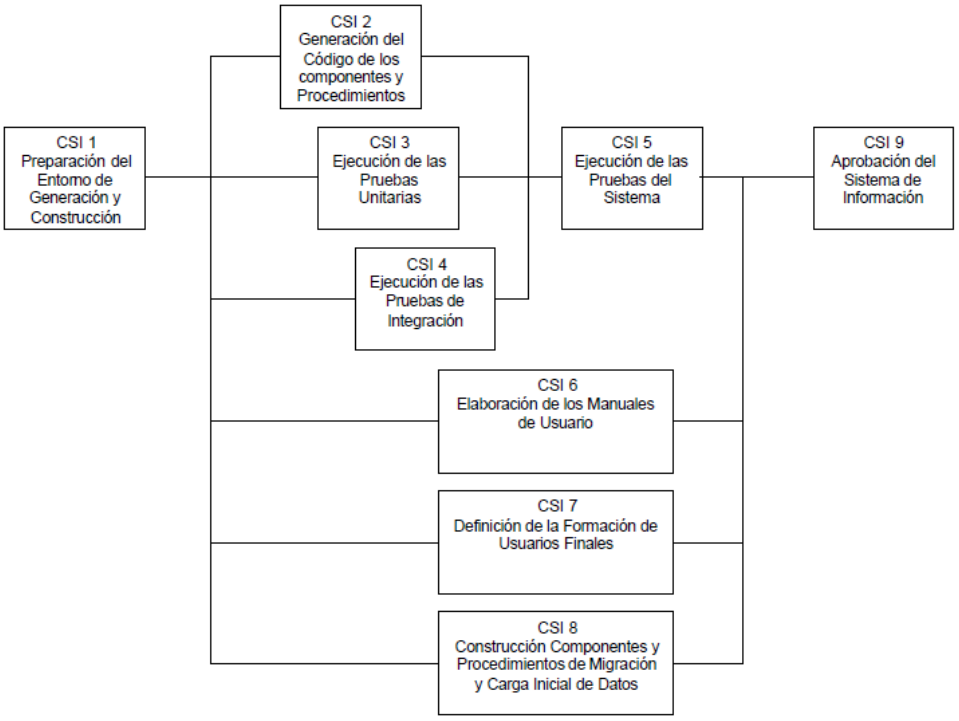


Figura 14 – Construcción del sistema de información – Métrica V3

5 Swebok Capítulo IV, Construcción de Software

6 Métrica V3, Construcción del Sistema de Información

11.3 LEVANTAMIENTO DE INFORMACIÓN CON LOS RESPECTIVOS IMPLICADOS EN CADA UNO DE LOS PROCESOS RELACIONADOS CON EL DOCUMENTO INSTRUCTIVO*.*

En esta sección se hará un levantamiento de información para detectar a través de los actores involucrados con el instructivo los posibles problemas que se presentan al momento de documentar el desarrollo de un sistema de información. El levantamiento de información se hará a un nivel general a cerca del estándar que se utiliza (Anexo A) a los ingenieros desarrolladores por medio de reuniones y se espera terminar de contextualizar el ambiente actual del proceso de ingeniería que se aplica en los proyectos de la empresa.

11.3.1 Elaboración de herramienta para el levantamiento de información. Como herramienta se usaría inicialmente una encuesta para hacer un levantamiento de la información directamente con los implicados, y así establecer algún tipo de contrastes para detectar que es lo que piensan, cuales son los problemas y que propusieran algunas posibles soluciones para el problema de documentación.

Una vez se llegó a este punto se propuso recurrir a reuniones periódicas como herramienta de comunicación y levantamiento de información reemplazando la encuesta, esto se dio en conjunto con un equipo que comparte el mismo objetivo, pues el autor no es el único trabajando en esta actualización del instructivo (Anexo A) y se apoya en los ingenieros de sistemas Carlos Pérez, Jaime Andrés Brito y la analista de sistema Jenny Silvana Ortega, supervisados por el coordinador de desarrollo Asdrúbal Gutiérrez. Durante 4 reuniones informales con más o menos una semana de diferencia entre ellas, se definirán las metodologías, técnicas, y pasos a seguir mínimos para sistemas de información ya desarrollados que no han sido documentados y para los nuevos sistemas de información que se desarrollaran a partir del término de este proyecto o de la fecha definida por el coordinador.

11.3.2 Entrevista con los actores relacionados con los procesos. Las entrevistas no fueron programadas por el proceso, sencillamente el equipo de trabajo se puso de acuerdo a hacerlas cierto determinado tiempo, el cual es más o menos con una semana de diferencia entre las mismas, en las cuales se proponían ideas, metodologías y pasos a seguir. Durante las tres primeras reuniones principalmente se definieron cronológicamente los siguientes hitos:

En la primera reunión (14 de mayo de 2010), los actores directamente implicados, es decir los programadores, expresaron su estudio previo de metodologías afines con el proceso, y la intención de utilizarlas. Se proporcionó la información necesaria para que el equipo de trabajo se documentara con respecto a estas mismas metodologías propuestas. Se propone por parte de los programadores utilizar un ciclo de vida RUP, el cual es bastante robusto y mezclarlo con un desarrollo ágil como lo es SCRUM; por otro lado utilizar conceptos relacionados con programación XP.

En el desarrollo de la segunda reunión (24 de mayo de 2010), se consolidaron las metodologías propuestas por los programadores, pues son las que más se acomodan al desarrollo de sistemas de información, ellos son los que han estado involucrados en este proceso de gestión de la calidad del software y han tenido estudios previos sobre estas metodologías. Se propuso inicialmente desarrollar un documento con los pasos mínimos a seguir al momento de documentar el desarrollo de sistemas de información (más adelante se justifica la elección de estas metodologías), y una solución mínima para los sistemas de información sin documentar.

Para la tercera reunión (28 de Mayo de 2010), se propusieron pasos mínimos para la documentación de software, tanto para nuevos de largo alcance como de corto alcance y sistemas de información ya desarrollados que no se han documentado. También se propone un cronograma grosso modo para el desarrollo de este proceso de migración hacia nuevas metodologías y prácticas como Spike Solution e Iconix.

Para una última reunión se aprobaría este documento.

11.4. RESULTADOS DE LA INFORMACIÓN ADQUIRIDA Y COMPARACIÓN CON LAS ALTERNATIVAS ESTUDIADAS

Durante este punto, se estructurarán las propuestas que surgieron de las reuniones efectuadas y se compararán con los estudios realizados con anterioridad en este trabajo, los cuales no se alejan mucho a la realidad ni a las propuestas seleccionadas. Para lograr esto más que identificar qué procesos y que tareas se realizan con respecto al documento instructivo, se analizará que es lo que se quisiera hacer y, una vez concretados estos fundamentos se procederá a establecer y justificar tanto la metodología como el modelo de ciclo de vida que es propuesta clave y objetivo de este trabajo.

11.4.1. Identificación que procesos están realizando con respecto al documento instructivo.

Los objetivos de la documentación de un proceso de software en donde se aplican técnicas propias de ingeniería son fundamentales para llegar a un producto de calidad en todos los aspectos. Pero muchas veces para los programadores se vuelve como una especie de paso obligatorio antes de ir a codificar un programa, volviendo dispendioso el proceso de documentación y perdiendo de vista que, más que un formato que se debe llenar, es una herramienta de suma importancia para procesos de desarrollo de sistemas de información de calidad que cumplen con todas las características necesarias para que el cliente quede satisfecho con el producto que se le entrega.

En otras ocasiones el problema no es de que si la documentación se hace o no, empiezan a resultar otra serie de dificultades basadas en los trabajos aislados de personas que conforman un equipo, es decir, el analista hace una parte del documento totalmente independiente de quien hace el diseño, este a su vez le entrega al desarrollador toda una serie de instrucciones teóricas que hacen que el programador tenga que sentarse a estudiar un problema con el que no está familiarizado desde el comienzo del proyecto haciendo subrutinas improductivas, las cuales muchas veces terminan cambiando completamente el diseño que se había planteado inicialmente y consumiendo tiempo valioso en actividades innecesarias.

Otro proceso que se lleva a cabo es el de las negociaciones, es de total comprensión que en la mayoría de ocasiones los requisitos van cambiando con el tiempo, y tanto el desarrollador del proyecto como el cliente deben estar

conscientes de esto, para así hacer negociaciones, es decir; se recogen algunos requisitos iníciales para hacer una estimación de costos, alcance y tiempo que va a llevar el proyecto, hasta ahí todo va bien, como los requisitos van cambiando y normalmente aparecen requisitos nuevos el tiempo empieza a apretarse y un desfase en este significaría más gastos para el cliente, pero como el cliente no quiere tener más gastos la producción se acelera y muchas veces se pierde el sentido de calidad con el que se comenzó trabajando.

11.4.2. Establecimiento la metodología y Modelo de ciclo de vida, apropiados para los proyectos del proceso. En las reuniones quedo claro que es de vital importancia que el proceso de desarrollo de sistemas de información debe pasar del caos en el que está a la organización, pero una organización que permita hacerla desde lo básico hasta lo especializado, y se necesita que la metodología por la que se opte sea lo suficiente compleja como para desarrollar un producto de excelente calidad, pero tan sencilla y amigable que los programadores no lo vean como una obligación tediosa, sino como una herramienta para lograr sus objetivos en el tiempo determinado y con la calidad necesaria, para lo cual los estudios iníciales de este proyecto encontraron algunos modelos de ciclo de vida basados en procesos iterativos pero, que al final terminaban siendo demasiado robustos, y no permiten que se desarrollen productos con agilidad sin dejar a un lado la calidad del mismo. Y cuando se intentó llegar a un proceso rápido, el modelo de ciclo de vida no era lo suficientemente complejo para los desarrollos que se hacen en la entidad. Por eso finalmente se propuso lo siguiente:

RUP (Rational Unified Process), es un proceso de software iterativo e incremental, creado por IBM que está dirigido por casos de uso, enfocándose en la arquitectura, es muy robusto y extenso pero permite hacer seguimiento de proyectos grandes y pequeños, de una forma muy detallada a un nivel de ingeniería de gran altura.

Estos tres fundamentos mencionados anteriormente son la base de RUP:

- Dirigido por casos de Uso

Los casos de uso en esta metodología no son solo la herramienta que especifica los requisitos de un sistema, también permiten a los participantes ser guiados durante su diseño, atravesando la implementación y prueba. Convirtiéndose así en

un elemento que proporciona un hilo conductor entre los artefactos que se generan durante las actividades del proceso de desarrollo.

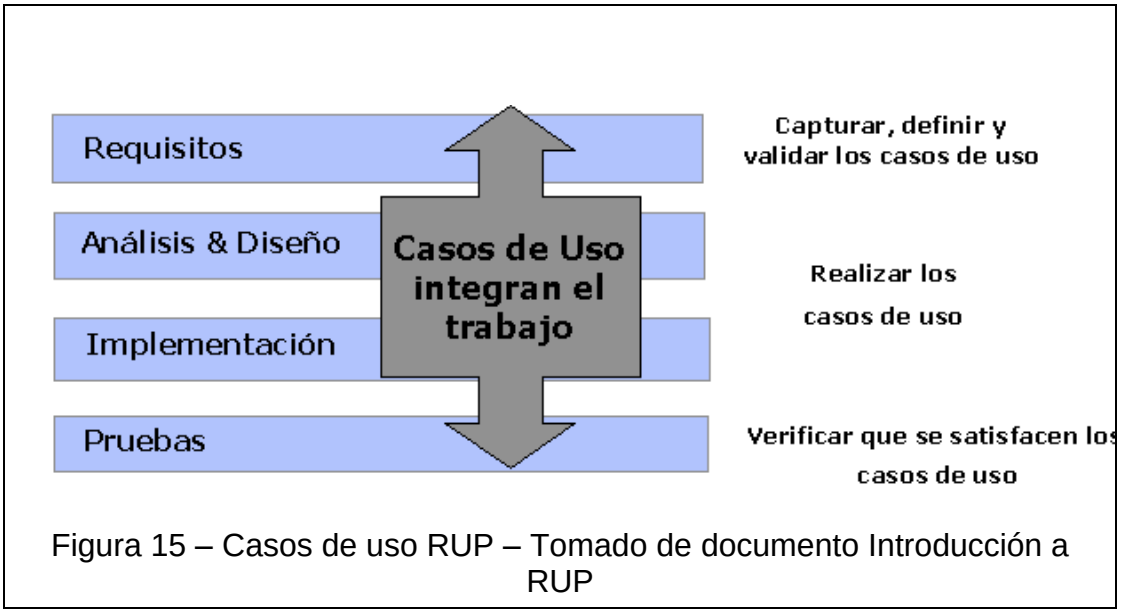


Figura 15 – Casos de uso RUP – Tomado de documento Introducción a RUP

- Proceso centrado en la arquitectura

En un sistema la arquitectura es la parte estructural y más relevante, que permite una visión en común entre los involucrados directamente en el desarrollo del sistema de información incluyendo al usuario. Esta estructura define como tiene que ser construido el sistema y en qué orden se van a ejecutar cada una de las tareas, definiendo estás mismas para llegar a un producto con características de calidad.

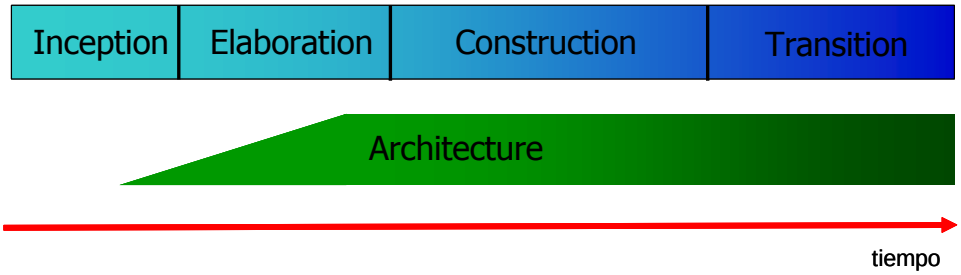


Figura 16 – Proceso centrado a la arquitectura – Tomado de documento Introducción a RUP

- Proceso iterativo e incremental

RUP propone un proceso iterativo e incremental en el cual el desarrollo total se divide en partes pequeñas permitiendo una especie de equilibrio entre los elementos anteriores, es decir entre casos de uso y la arquitectura.

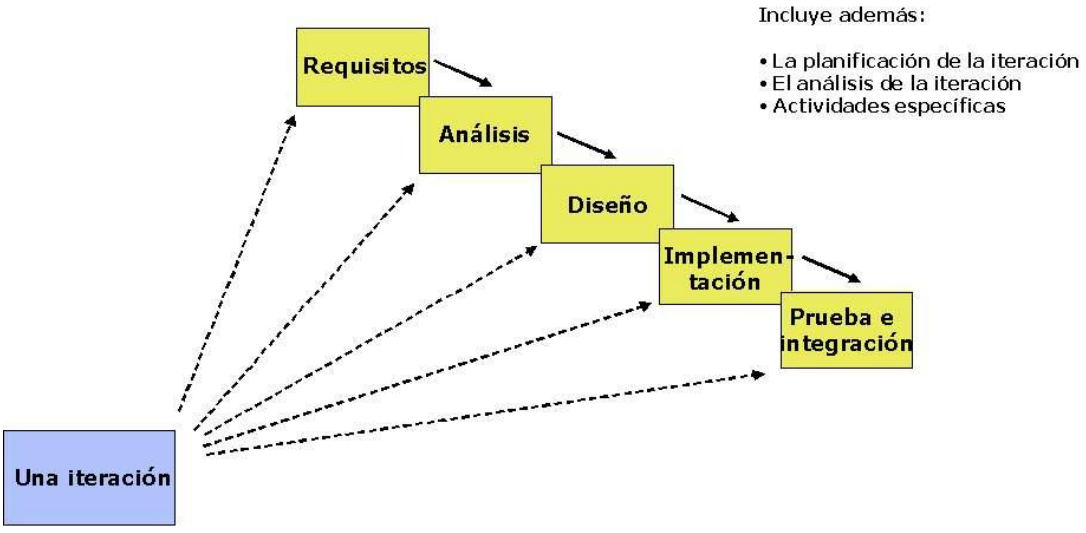


Figura 17 – Proceso Iterativo Incremental – Tomado de documento Introducción a RUP

SCRUM, método para gestión de proyectos de sistemas de información que exigen agilidad y cumplir con el tiempo, el alcance y la calidad, basado en un proceso iterativo e incremental.

Este modelo define un conjunto de prácticas y roles que los participantes del desarrollo del proyecto deben ejecutar durante el tiempo que tarde el proyecto para lograr un desarrollo sistemático y ágil en donde se solapan las etapas del proyecto permitiendo reducir el costo de tiempo de producción.

Cada Iteración es llamada Sprint, la cual normalmente dura un periodo entre 15 y 30 días, pero el equipo de trabajo es quien decide cuánto tiempo va a durar cada una, y en qué nivel se va incrementando a medida que el proyecto va subiendo de nivel de complejidad. Al finalizar cada Sprint, el equipo de trabajo deberá entregar un incremento del software potencialmente utilizable.

Es una metodología a la cual se le podría llamar como “comunicativa” en donde todos los miembros del equipo se auto-organizan y se distribuyen responsabilidades verbalmente y el proceso de estas durante cortos periodos de tiempo en donde la frecuencia podría ser diaria.

XP (Programación Extrema), es una metodología de desarrollo ágil basado en una serie de actividades y buenas prácticas de ingeniería que buscan como objetivo aumentar en gran medida la productividad al momento de desarrollar sistemas de información. Para tener en cuenta al momento de adoptar esta metodología es necesario que los desarrolladores tenga un muy buen nivel de experiencia y de conocimiento pues se espera que la comunicación se haga de forma rápida y sus soluciones de igual manera.

A groso modo la programación extrema se basa en cuatro categorías que reúnen 12 principios básicos los cuales son:

-Retroalimentación a escala fina:
El principio de pruebas
Proceso de planificación
El cliente en el sitio
Programación en parejas

-Proceso continuo en lugar de por lotes:
Integración continúa
Refactorización
Entregas pequeñas
Estándar de codificación

-Entendimiento compartido:
Diseño simple
Metáfora
Propiedad colectiva del código

-Bienestar del programador:
La semana de 40 horas

Cada iteración en esta metodología es propuesta de la siguiente manera:

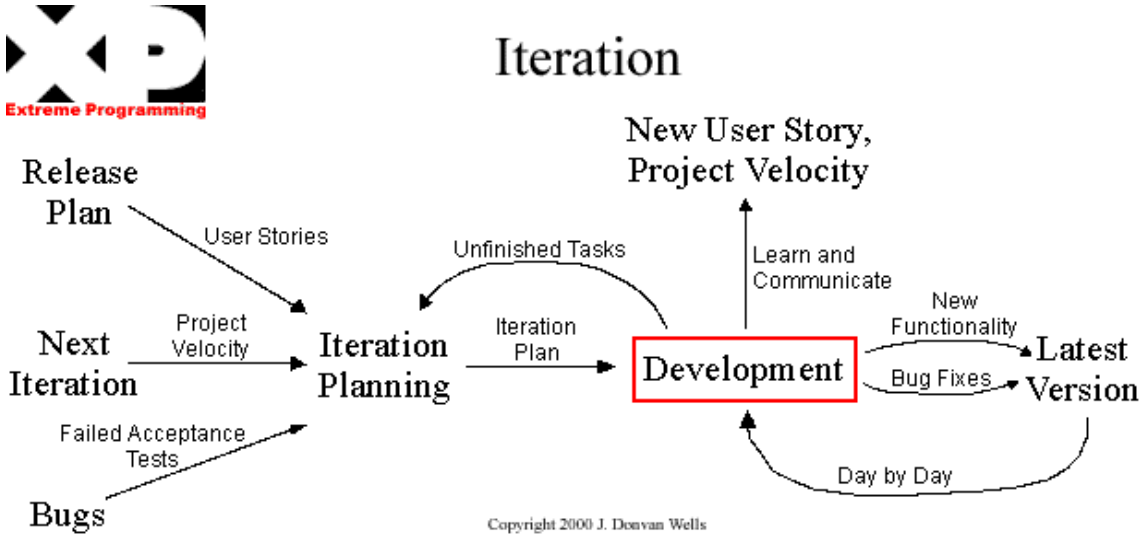


Figura 18 – Proceso de Programación Extrema – extraído de:
http://www.extremeprogramming.org/

En los primeros estudios presentados durante este trabajo, se mostraban diferentes alternativas de modelos de ciclo de vida y metodologías que se podrían consolidar para dar como resultado un documento bastante extenso y robusto, los estudios muestran que en la actualidad un proceso iterativo es la tendencia absoluta para desarrollos de sistemas de información, por eso se eligió RUP, por su calidad al detalle al proponer actividades a nivel de ingeniería de una forma muy evolucionada, pero por su misma robustez puede tardar mucho tiempo en su ejecución, por eso se enuncia SCRUM, una metodología ágil que permitiría acelerar a RUP utilizando mejores prácticas organizacionales y de gestión de proyectos. Siguiendo en línea con las metodologías de desarrollo de software se propone XP como otro método para desarrollar proyectos agiles, pues este se centra en desarrollar e ir haciendo todos los procesos en pequeñas proporciones, es decir análisis, diseño y desarrollo proporcionándole entregables al cliente en menor tiempo que son funcionales e irlos acomodando a los requerimientos cambiantes del cliente. Logrando así el producto en poco tiempo sin afectar en ningún momento la calidad del sistema de información que se le da al usuario, el cual se verá más involucrado con el equipo de trabajo para lograr concretar los requisitos y no tener virajes de ellos durante el transcurso del desarrollo del proyecto. Para lograr estos objetivos se incluyen conceptos como XP y Spike Solution, los cuales permiten agilidad y comprensión de los requisitos del usuario que son los que cambian constantemente y hacen que el tiempo del proyecto y el alcance se aprieten.

Los estudios hechos en principio no se menos preciaron, pues las metodologías como SWEBOK, Métrica V3 y Moprosoft se pueden ver integradas en el modelo establecido para que se ejecute de ahora en adelante, pues la sustracción de sus mejores actividades se ven reflejadas en RUP el cual integra el ciclo de vida con la metodología. En cuanto al modelo de ciclo de vida como ya fue mencionado por tendencias y necesidades actuales del proceso administrativo de sistemas de Comfamiliar Risaralda se utilizará un iterativo e incremental, punto en el que convergen RUP y SCRUM, y esa allí mismo en donde empalman estas metodologías haciéndola una mezcla fresca, rápida, robusta, organizada y progresiva que es lo que busca el proceso administrativo actualmente.
Los principios básicos en los que se hará hincapié para el desarrollo del manual que especificará la forma de utilizar esta convergencia de metodologías se basan en:

Requisitos no predecibles: los requisitos cambian con el tiempo y se tiende a trabajar conforme a los cambios que vayan teniendo.

Iterar: para lograr un moldeamiento de los requisitos proporcionados por el usuario para retroalimentar el desarrollo del software, se tendrá que iterar cada cierto corto tiempo y de esta manera tener una mayor comunicación con el usuario.

Pruebas: sistematizar las pruebas para obtener el mejor provecho de estas en todo momento, no solo al integrar el software en su totalidad sino al momento de hacerle cambios, y como es bien sabido para entregar un producto probado y encontrar errores básicos que los usuarios finales no deberían encontrar.

Tiempo/Costo/Alcance: como se menciono con anterioridad el tiempo y el costo no se pueden negociar, o al menos el usuario no negocia con ellos, al crecer los requisitos el tiempo se aprieta y el usuario no tiene presupuestado un costo mayor al que inicialmente se le dio, por eso es necesario negociar con el alcance de los requisitos que tiene el proyecto para no afectar la calidad del mismo.

Centrado al cliente: el cliente se tiene que acercar más al proyecto de modo que no cree falsas expectativas por la diferencia de tiempos de entrega de los ejecutables, tiempo en el cual no tiene contacto con el desarrollo y espera que después de pasado éste se le entregue un producto de alta complejidad.

No obreros: en muchas de las metodologías se propone que una persona trabaje en el análisis, otra en el diseño y esto se le entregue al programador para que ensamble, a manera de analogía el programador seria el obrero de una construcción, la tendencia y la necesidad del proceso es que se le entregue más responsabilidad al programador de manera que esté mucho mas involucrado en el análisis y en el diseño pues su trabajo como Ingeniero le exige creatividad al instante de construir software y necesita su acompañamiento desde el primer momento del proyecto.

Equipo estima: el mismo equipo que ejecutará el proyecto es quien debería estimar, pues conocen su forma de trabajo y podrían llegar a una estimación más precisa luego de utilizar algún método de estimación de costos.

Tipos de Proyecto: normalmente los proyectos no son iguales, hay unos de mayor y otros con menor complejidad. Es evidente que no se deberían utilizar una gran cantidad de métricas y actividades para desarrollos sencillos en los cuales los requisitos no cambian de manera drástica y las iteraciones son cortas. Por esto

debería haber una diferenciación de que practicas debe existir según el tipo de proyecto.

Enfoque gerencial: respecto a la gerencia del proyecto se tienen que presentar una forma de gestión organizada que aumente la productividad del personal, mostrando su disponibilidad, sus responsabilidades y así conseguir desarrollo con garantías de éxito.

Proceso claro: es necesario que el primer manual de desarrollo que se haga tiene que ser demasiado claro para todos los integrantes del proceso administrativo de sistemas, pues es obvio que los cambios generan confusión, y hay que evitar que se presenten traumatismos de alto riesgo.

Arquitectura: es otro de los enfoques que se le tienen que dar a los desarrollos pues la arquitectura en la que corren los sistemas de información es necesario que sea la adecuada y que no se le recargue con demasiados procesos, además es importante para la organización.

11.5. OBSERVACIONES

Es importante resaltar que finalmente este documento hace de referente conceptual e investigativo para justificar el cambio de metodologías para desarrollo de software en el proceso administrativo de sistemas de Comfamiliar Risaralda. Para la elaboración de este se tuvo en cuenta la manera de analizar, diseñar y desarrollar además de estudiar cómo se documenta todo esto y si el instructivo (Anexo A) es pertinente al asunto tratado. Además se tiene todo el estudio de las alternativas y tendencias que actualmente presenta el mercado para el desarrollo de proyectos de sistemas de información, se analiza cada una de ellas y se presenta un resultado final en el cual los propios actores involucrados en este desarrollo se identifican con unas propuestas y cómo estas se acomodan a lo que necesita en este momento la organización.

RECOMENDACIONES

Finalmente, este documento representa parte de un proyecto mayor, el cual es el cambio completo de metodología, tanto de documentación como de trabajo al momento de desarrollar sistemas de información en el proceso administrativo de sistemas de Comfamiliar Risaralda, el presente trabajo es el estudio que sirve como justificación para realizar el proyecto a nivel macro, se espera que luego de este documento, en otro proyecto paralelo se genere un manual en el cual se presentan los pasos mínimos que se deben seguir para desarrollar un software, y presentar la metodología de trabajo apropiada luego de este concienzudo estudio de alternativas, es recomendable que este manual mínimo sea escalable, es decir, que quede lo suficiente abierto para adicionarle mejores prácticas de ingeniería. Además también debe ser lo suficientemente flexible para moldear cierto tipo de desarrollos.

El manual tendrá que tener la adaptabilidad suficiente a las circunstancias del proyecto, pues es bien sabido que no es la misma cantidad de actividades que se le hacen a un proyecto de menos complejidad que a uno con gran complejidad; la gerencia de los equipos de trabajo tiene que ser redefinida, y precisamente eso es lo que se busca en todo este proyecto por lo cual se espera que SCRUM cumpla todos los requerimientos necesarios.

La socialización de todo este proyecto es necesaria en los integrantes de los equipos de trabajo que de ahora en adelante desarrollaran sistemas de información, pues es importante que para que tenga éxito toda esta evolución los desarrolladores deben conocer todas las practicas y actividades, y tenerlas a disposición en algún tipo de repositorio. Además de que los mismos implicados en los desarrollos de sistemas de información deberían aportar nuevas prácticas y experiencias para una retro-alimentación de todos los miembros del departamento.

CONCLUSIONES

El proceso administrativo de sistemas en este momento a adquirido una madurez suficiente, y requería que se hiciera este estudio para el cambio de metodología, pues aunque muchas de las practicas que se implementarán ya estaban siendo usadas empíricamente por muchos de los desarrolladores, se necesita establecer una metodología estándar de mejores prácticas.

Los requisitos cambian, y esa es la principal razón que se encontró durante el desarrollo de este estudio para que el instructivo anterior se quedara corto al momento de estructurar los sistemas de información. Las metodologías agiles son el entorno del estado cambiante de los requisitos.

Con el transcurso de: el tiempo, de los desarrollos de sistemas de información, de las experiencias, de las iteraciones y de la retroalimentación; los desarrolladores identifican los problemas y, casi empíricamente logran establecer mejores prácticas.

Espontáneamente los procesos exigen cambios, y con el ritmo al que se mueve la tecnología es casi impredecible esperar ciertos comportamientos como estándares, por eso las metodologías agiles han tomado los modelos clásicos y los han llevado a un plano más práctico, y abierto a cambios.

La organización en la gestión de los proyectos de software requieren un alto nivel de comprensión, no es suficiente únicamente el conocimiento teórico, la práctica y afrontar todos estos problemas que puedan traer los desarrollos generan en los equipos de trabajo fortaleza y disciplina para volverse cada vez más productivos

BIBLIOGRAFÍA

BOEHM, Barry W. A spiral model of software development and enhancement. En: ACM SIGSOFT Software Engineering, Volumen 11(Agosto, 1986). P. 14-24

PRESSMAN, Roger S. Ingeniería del Software. Un enfoque práctico. Quinta edición. Madrid: Concepción Fernández, 2002. P. 601.

IEEE, The Institute of Electrical and Electronics Engineers. **SWEBOK**: guide to the Software Engineering body of knowledge. Versión 2004. Estados Unidos: Deborah Plumer, 2004. P 200.

DRAE, Diccionario de la Real Academia de la lengua Española. Edición 22. España: 2001
Disponible en: http://buscon.rae.es/draeI/

JOYANES, Luis A. Programación en C++. Algoritmos, estructuras de datos y objetos. Segunda Edición. España: McGraw-Hill Interamericana de España, s.a. 2006.

[IEEE830-98], IEEE Std 830-1998, IEEE Recommended Practice for Software Requirements Specifications, IEEE, 1998.

[IEEE1016-98] IEEE Std 1016-1998, IEEE Recommended Practice for Software Design Descriptions, IEEE, 1998.

MÉTRICA. Metodología de Planificación, Desarrollo y Mantenimiento de sistemas de información, Versión 3. España: Ministerio de Administraciones Publicas

MoProSoft, Modelo de procesos para la industria de software. Versión 1.3. México: Secretaria de economía. 2005.

ARBOLEDA, Hugo F. Modelos de ciclo de vida en desarrollo de software. Edición N° 93 Julio - Septiembre de 2005.
Disponible en: http://www.acis.org.co/index.php?id=551

INTRODUCCIÓN A RUP. Portal de Desarrollo de Software. Universidad Politécnica de Valencia (UPV).
Disponible en:
https://pid.dsic.upv.es/C1/Material/Documentos%20Disponibles/Introducción%20a%20RUP.doc

DON WELLS. Extreme Programming: A gentle introduction. 2009.
Disponible en: http://www.extremeprogramming.org/

ROBLES, Gregorio. Programación extrema y Software Libre. Universidad Politécnica de Madrid: V Congreso Hispalinux, 2002.

Comfamiliar Risaralda. 1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información. Departamento de sistemas: Pereira 2007

Anexo A <<1-IN-030 Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información>>

Comfamiliar	**INSTRUCTIVO**	**VERSIÓN: 0**
		CÓDIGO: 1-IN-030
Estándares de Análisis, Diseño y Desarrollo de Sistemas de Información		**FECHA: 16/Nov/2007**

0. LISTA DE VERSIONES

VERSIÓN	FECHA	RAZÓN DE LA ACTUALIZACIÓN

1. OBJETIVO

Recopilar los puntos necesarios para documentar proyectos informáticos desde la fase inicial, conocimiento de la problemática, hasta la fase de entrega del producto final a la comunidad usuaria.

2. ALCANCE

Comprende la descripción de la documentación necesaria para el análisis, diseño y desarrollo (manual técnico) de Sistemas de Información, aplicada a proyectos internos de la institución, con el propósito de garantizar una adecuada estandarización de documentos.

3. DEFINICIONES

CASOS DE USO

Es una operación o tarea específica que se realiza tras una orden de algún agente externo, sea desde una petición de un actor o bien desde la invocación desde otro caso de uso.

REGLAS DEL NEGOCIO

Son temas o concepciones que representan componentes de un sistema.

DIAGRAMA DE NAVEGACIÓN

Tipo de diagrama, que representa el desplazamiento interno del sistema entre las diferentes funciones.

DIAGRAMA DE MOTIVOS

Representa la secuencia de estados por los cuales pasa un objeto en respuesta a diversos eventos.

MÉTODO

Programa procedimental escrito en un lenguaje, asociado a un objeto determinado y cuya ejecución sólo puede desencadenarse a través de parámetros recibidos por éste o por sus descendientes.

MODELO ENTIDAD RELACIÓN (MER)

Representación estática del sistema, muestra los elementos y relaciones que existen entre estos.

4. GENERALIDADES

El grupo de desarrollo se encuentra compuesto por el Coordinador de Desarrollo, Analistas de Sistemas, Programadores de Sistemas y Analistas de Procesos.

Si el nuevo proyecto informático a desarrollar no se considera como un Sistema de Información sino como un Módulo, se documentan únicamente los puntos que apliquen para éste.

Diseños de Interfaz

Creación de pantallas no estandarizadas por la entidad, en el caso de requerir una nueva pantalla para el nuevo sistema de información se debe anexar como estandarizada si es aprobada.
Las ventanas dentro de las aplicaciones estarán enlazadas de forma funcional, es decir, encadenadas de acuerdo a las funciones establecidas en el diagrama de navegación.

Estándares de interfaz a tener en cuenta:

- Colores neutros permiten una mejor visualización, recomendable máximo cuatro colores.
- El formato se debe mantener igual para todas las pantallas (comandos, botones, íconos) por tipo de función.
- Manejo de íconos adecuados para agilizar la percepción del usuario.
- Visualización de funciones y teclas deshabilitadas para una mayor comprensión.
- Distribución de encabezados, datos o campos, botones, barras de estados, entre otros elementos utilizados para mejor entendimiento y mayor organización.
- Manejo lineamientos de campos y textos.

5. DOCUMENTOS RELACIONADOS

Diseño de Sistemas de Información
Desarrollo de Sistemas de Información
Registro de Modificación para Sistemas de Información

6. METODOLOGÍA DOCUMENTACIÓN PARA PROYECTOS INFORMÁTICOS

Documentación aplicada para el desarrollo interno de Sistemas de Información de la entidad Comfamiliar Risaralda, depende de la fase en que se encuentre el proyecto informático:

6.1 DOCUMENTO ANÁLISIS DE SISTEMAS

Documento con previa disposición para el inicio de la etapa de diseño, los Analistas de Procesos y los Analistas de Sistemas son los responsables de la elaboración de este documento, en donde se clasifican los requerimientos de manera ordenada solicitados para un nuevo proyecto informático, presentado en un plan lógico:

A. Proceso
Área que realiza el requerimiento

B. Cronograma de Análisis
Anexo #

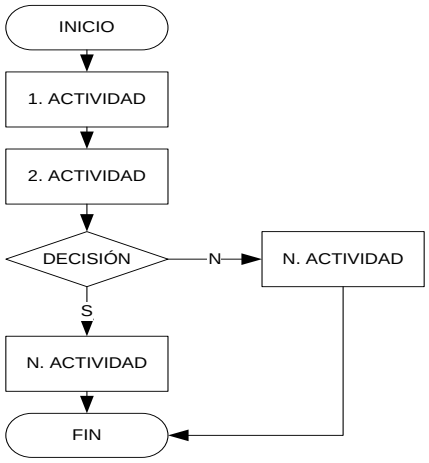
C. Fuente de Información
Cargo(s), que operará(n) el producto directamente, responsable(s) de suministrar la información pertinente para el análisis de la situación.

D. Hallazgos Situación Actual
Antecedentes o circunstancias actuales, que incluyen de manera detallada la descripción del problema por el cual se dio origen a la solicitud de una nueva herramienta.

E. Procedimientos
Levantamiento de procesos, que se relacionan con el problema a solucionar, siguiendo los parámetros establecidos por el Sistema de Gestión de Calidad. Responsables a cargo Proceso de Mejoramiento Continuo y Analistas de Procesos.

También conocidos como Diagramas de Actualidad, permiten modelar procesos o procedimientos del negocio, se construyen con los siguientes aspectos:

1. Entrevista con el(os) Responsable(s) del proceso a sistematizar,
2. Levantamiento de información referente al manejo actual del proceso,
3. Puesta de condiciones, abarcando los posibles casos que pueden darse en el proceso,
4. Recomendaciones de mejoras, a partir de eliminación o modificación de actividades para optimizar el proceso con respecto al objetivo del sistema solicitado,
5. Recursos que se utilizan para llevar a cabo el desarrollo del proceso, con el fin de sistematizarlos, si es el caso,
6. Construcción del flujo grama, a partir de la información recopilada.



Flujo grama

En caso de no existir la documentación de procedimientos, se deberá remitir al Sistema de Gestión de Calidad para obtener la información y se anexará al historial del Sistema de Información o Módulo desarrollado; de lo contrario se debe realizará el estudio correspondiente y la construcción de flujo gramas para mayor entendimiento durante el análisis, diseño y desarrollo de la herramienta.

F. Reglas del Negocio

Describir detalladamente las limitantes que se tienen en los procedimientos con respecto a la funcionalidad del futuro sistema.

G. Beneficios

Describir las mejoras adicionales obtenidas a partir del desarrollo, como por ejemplo el impacto en la competitividad del área, la reducción de tiempo en la ejecución de tareas, la optimización de los recursos, etc.

H. Tipo de problema

Se predetermina con el Coordinador de Desarrollo, la escala requerida para el funcionamiento correspondiente a un proyecto de alta, mediana o baja complejidad para su implementación.

I. Objetivos

Supuesto teórico: Propuesta tentativa a un problema, se coloca a prueba para determinar su validez

Objetivo general: Se expresa la acción general que se llevará a cabo para cumplir el supuesto teórico

Objetivos específicos: Serie de acciones o actividades particulares para cumplir el objetivo general

J. Diagnóstico

Se determina si la solución es un Sistema de Información o es un Módulo. Igualmente, se estipula de acuerdo al análisis previamente establecido desde el punto de vista de desarrollo de software, que el proceso de diseño y desarrollo de la nueva herramienta tiene una solución aplicable. Si la solución no es aplicable desde el punto de vista de desarrollo de software, se realiza un diagnóstico alternativo a una posible solución desde el campo de aplicación profesional.

6.2 DOCUMENTO DE DISEÑO

Documento de aplicación de técnicas y principios de ingeniería de software, en donde se detallan pasos que permitan describir los aspectos del Sistema de Información a construir. Permite la interpretación y realización lógica del sistema; enfocando los dominios de datos, funcionalidad y comportamiento desde el punto de vista de la implementación. La realización del documento es responsabilidad tanto del grupo de Analistas de Procesos y Analistas de Sistemas como de los Programadores. Se construye a partir del documento de análisis:

A. Alcance

Definir claramente los elementos que abarca el desarrollo.

B. Objetivos

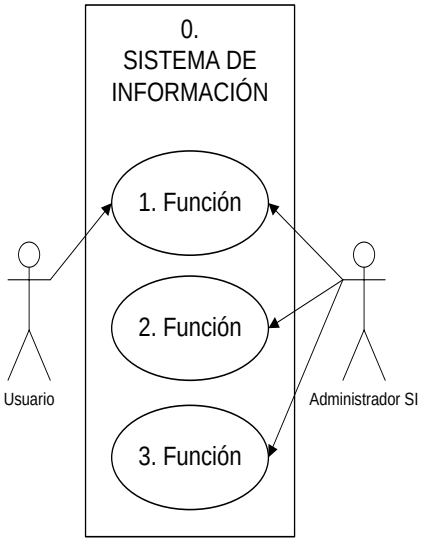
Objetivos específicos para llevar a cabo el alcance, se desglosa en un Cronograma de Diseño.

C. Restricciones

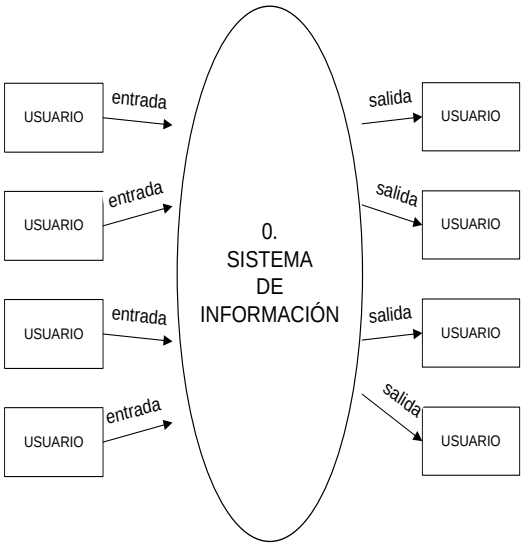
Se especifica cada limitante y reglas del negocio que tendrá el sistema de información en el momento de su implementación, por decisión del Jefe de Sistemas, Coordinador de Desarrollo y de la Comunidad Usuaria con el fin de salvaguardar futuros riesgos.

D. Casos de Uso

Diagrama contexto General o Nivel 0: Presenta la función más general del sistema. Se detallan las principales entradas y salidas.

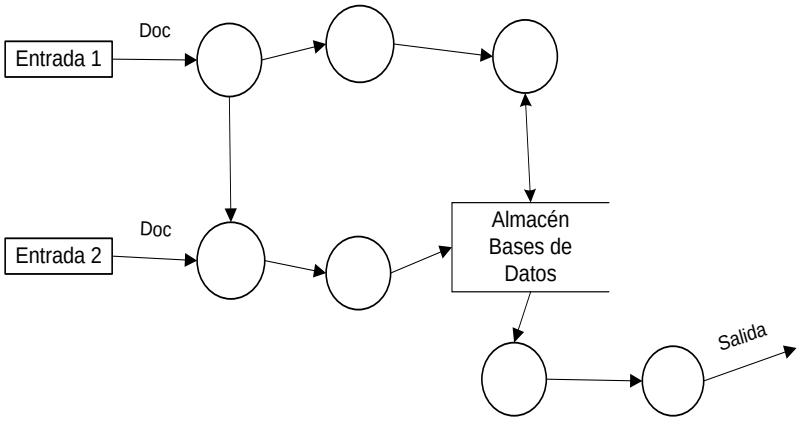


Función general



Inputs y Outputs

Diagrama de Nivel 1: Define en un diagrama las relaciones entre procesos y componentes.



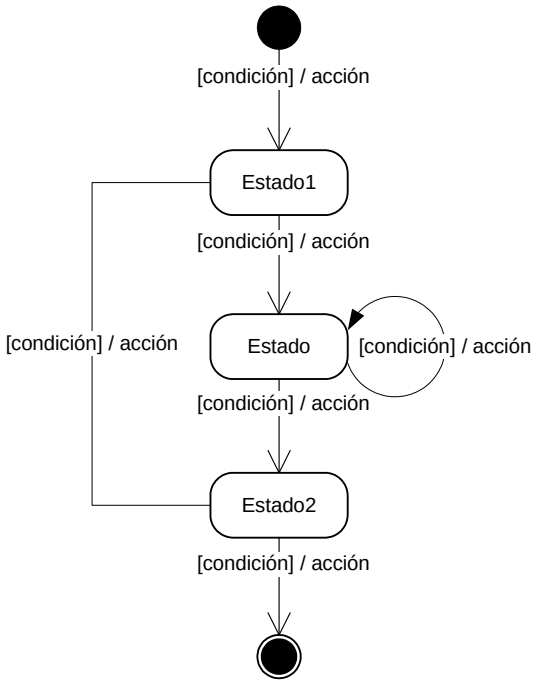
El nivel de profundidad (nivel 2, 3, 4, N) se define de acuerdo a la complejidad de los procesos a sistematizar.

E. Perfiles

Listado de los cargos que juegan un rol en el sistema de información, a partir de estos se generan los grupos de permisos.

F. Diagrama de Motivos o Diagrama de Transición de Estados

Modela los estados por los cuales un concepto puede pasar durante la existencia en el sistema. Para mayor claridad se deben especificar los estados necesarios dentro de los procesos en el sistema.



Estado Inicial

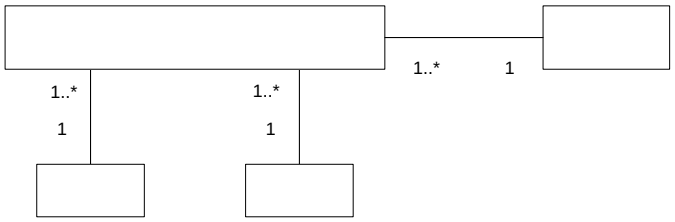
Estado: Valor que tiene e atributo de una entidad en un momento dado

Transición: Se utiliza como ciclo en el mismo estado hasta que se cumpla la condición

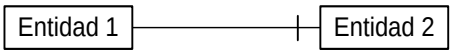
Transición: Tiene como efecto un cambio en los valore o atributos de la entidad

Estado Final

G. Modelo Entidad-Relación (MER)

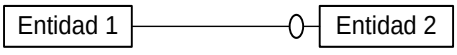


Representación estática del sistema, muestra los elementos y relaciones que existen entre estos. Para las relaciones se maneja una cardinalidad establecida de 1:1, 1:*, *:*, y a su vez las relaciones pueden ser de carácter obligatorio u opcional.



Ent1 debe tener al menos 1…* Ent2

Relación Obligatoria



Ent1 puede tener 0…* Ent2

Relación Opcional

H. Diccionario de Datos

Listado de tablas: Se define atributo, descripción del atributo, tipo, longitud, observaciones.

TBNombreTabla

Atributo	Descripción	Tipo Campo/ Longitud	PK	FK	Observaciones

Entre los posibles valores que puede recibir un atributo están:

- En el caso de Estado: Activo (A), Inactivo (I)
- En el caso de Evento de Incidencia: Nuevo (N), Modificación (M), Anterior (A)

I. Diagrama de Funciones y Navegación

Definición de funciones y desplazamiento entre las mismas con base en los siguientes parámetros:

Funciones:

CR: Consulta de Registro

TR: Trabajar con Registro

CA: Consulta Registro con parámetro

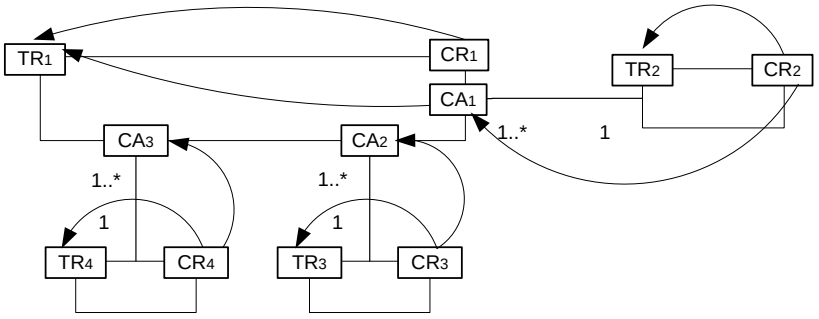
IR: Ingreso masivo de registro

OT: Otro tipo

IA: Ingreso masivo de registro con parámetro

BA: Batch

PR: Listado



De la CR_2 pasa a la TR_2, de la CR_2 se pasa a la CA_1 (si la tiene), de la CA_1 se pasa a la TR_1

El Coordinador de Desarrollo estará encargado de listar las tareas a desarrollar en la nueva herramienta:

Funciones	Tipo	Observación	Responsable

6.3 DOCUMENTO DE DESARROLLO / MANUAL TÉCNICO

El grupo de desarrolladores debe entregar por escrito la tarea de programación asignada para construir el manual técnico, manteniendo la siguiente plantilla:

A. Alcance de Desarrollo

Campo en el cual se determina la cobertura de la tarea asignada al programador.

B. Descripción de Desarrollo

C. Anexo Modelo Entidad-Relación

Caso opcional en cuanto a modificaciones generadas en el modelo original

D. Condiciones especiales

Descripción detallada de restricciones en el funcionamiento

E. Listado de clases, funciones y/o similares por programador

Nombre		Nombre dado dentro del código fuente
Tipo		Subrutina, Función, Clase, Procedimiento almacenado, Trigger y otros
Descripción		
Funciones invocadas		
Pre		Precondiciones
Post		Postcondiciones
Parámetros	IN	Parámetros de Entrada
	OUT	Parámetros de Salida

F. Glosario de términos

G. Convenciones (si aplica)

H. Cronograma de Desarrollo

7 REGISTRO DE MODIFICACIÓN PARA SISTEMAS DE INFORMACIÓN

Responsabilidad de cada **INTERVENTOR** en un Sistema de Información:
Como carácter obligatorio, se debe reunir el personal involucrado y/o relacionado con el Sistema de Información a tratar (Programadores, Analistas de Sistemas, Analistas de Proceso, Personal a Fin), con el objetivo de comunicar la decisión tomada respecto a un tipo de Sistema de Información próximo a ser modificación; esto se realiza buscando verificar el grado de factibilidad y viabilidad de los nuevos requerimientos solicitados por el usuario, y mejorar la comunicación entre el personal.

Se contempla la realización de este tipo de documentación después de liberar la concepción de cambios para el Sistema de Información a tratar.

Formato Obligatorio de Cambios:

A. Nombre del Sistema de Información

B. Responsable: Nombre de la(s) persona(s) a cargo de la(s) modificación(es).

C. Tipo de Cambio: Selección de casos

- Creación de Funciones
- Modificación de Funciones
- Eliminación de Funciones
- Creación de Módulo

D. Justificación: Especificación del por qué se realizan los cambios.

E. Vigencia a partir de: Fecha en la que los nuevos cambios empiezan a regir.

F. Anexo Registro de Modificación para Sistemas de Información.

ESTÁNDARES DE DESARROLLO

Estándares de Objetos

Consideraciones: Todos los objetos que se mencionan a continuación se ubicarán dentro de la librería respectiva del proyecto. Sólo si se trata de un módulo genérico a varias aplicaciones se ubicará en BDUTIL.

Los archivos fuente para la creación de estos objetos se almacenarán en el miembro QSQLSCR.

Objeto	**Mnemotecnia**
Base de Datos	Bd – NombreProyecto
Tablas	Tb – Pr – NombreTabla
Vistas	Vw – Pr – Nombre
Trigger	Tr – Pr – Nombre
Función	Fn – Pr – Nombre
Constraint	Ct – Pr – Nombre
Procedimiento Almacenado	Sp – Pr – Nombre
Formato de Pantalla	Fm – Pr – Nombre
Programas RPG	Pr – Pr – Nombre
Programas CL	Cl – Pr – Nombre
Reportes	Rp – Pr – Nombre
Comandos	Cm – Pr – Nombre
Índices	In – Pr – Nombre
Joins	Jn – Pr – Nombre
Script	Sc – Pr – XXX – YYY

En todos los casos anteriores Pr es Proyecto y la longitud máxima de caracteres es de 10 para consideraciones de AS/400

XXX → Tipo de Script
YYY → Tabla

Estándares de Campos

TT – CCC – _ _ _ _ _

Donde: TT: Nombre de Tabla (Caracteres 5 y 6 del Nombre)
CCC: Apellido del Nombre (tipo o clase del campo), así como los siguientes:
Cod: Código
Nom: Nombre / Descripción
Fec: Fecha
Num: Número / Cantidad
Vlr: Valor

Por Ejemplo: BE-COD-BENE
BE: Tabla de Beneficiarios
COD: Tipo código
BENE: Nombre del campo

La excepción:

Los campos de auditoría siempre inician con AU.

Wk – nombre de campo – Variables de Trabajo

Características de tablas a tener en cuenta:

- Las llaves foráneas se repite el nombre de la llave foránea
  - Ej. : Tabla de Trabajadores Código : TrCodTra
    Tabla de PersCargo/GrupoFam: TrCodTra
- Las llaves en lo posible serán de tipo numérico
- Se manejará campo de Estado Activo / Inactivo y Motivo Activo / Inactivo
- Las tablas manejarán Campos de Usuario (ingreso al sistema), Fecha (Numérico de 8 caracteres y formato AAAAMMDD) y Hora (Numérico de 6 caracteres)
- Tablas de movimiento serán históricas teniendo el período como campo
- Se manejarán tablas de incidencias para los que la requieran
- Para AS/400 la longitud máxima es de 10 Caracteres

Estándares para Visual Basic

Objeto	**Mnemotecnia**
Proyecto	Pry – NombreProyecto
Formatos	Frm – Pr – Tf – NombreTabla Pr - Proyecto Tf – Tipo Función Tr Trabajar con Registro Cr Consulta de Registro

	Ca Consulta Registro con Parámetro Ba Batch Pr Proyecto Ir Ingreso Masivo de Registro
Módulos	Mdl – Pr – Funciones (del Proyecto) Mdl – Pr – Definiciones Mdl – Pr – Configuración Mdl – Comfamiliar (estándar de la Caja)
Clases	Cls – Pr – NombreClase
Controles	Ctl – Pr – NombreControl
Recursos	Res – Pr – NombreRecurso Iconos – Imágenes – Mensajes de Texto
Listados de Cristal Report	Rpt – Pr – NombreProceso

Para desarrollo en Visual Basic, se hará uso de las siguientes carpetas: NombreProyecto

- Datos: Base de Datos y archivos de configuración
- Módulos
- Formas
- Controles
- Reportes
- Recursos
- Documentos
- Paquete: Archivos de instalación

Estándares para Java

Si se utiliza el generador del modelo para las aplicaciones en Java se hará uso de las siguientes carpetas:

Ruta del proyecto: El proyecto completo se almacenará en la ruta
C:\pryComf\proyectos\Comf_ris_hist_cli\dev\src\apps_nombreproyecto

Internamente en el src (Fuentes *.java) se genera las siguientes carpetas:

- com.comfris.App: Archivos básicos de inicio, como el menú principal, el archivo de conexión inicial y la ventana de password.
- com.comfris.Bean: Se almacenan las reglas del negocio propio de la aplicación.
- com.comfri.CasosdeUso: Casos de uso, indican las operaciones aplicadas al modelo.
- com.comfris.Dao: Se relacionan todas las consultas a la base de datos.
- com.comfris.Interface: Define las interfaces que aplican al modelo.
- com.comfris.Model: Contiene los atributos, get y set, que aplican al modelo de datos.
- com.comfris.View: Contiene las formas (CR, TR, CA… etc.)
- com.comfris.View.Panel: Contiene los paneles de datos que usan las TRs
- classes : Archivos compilados *.class
- Datos: Archivos de configuración
- distrib.: Archivos de Instalación

El Generador del Modelo (ModelGen) construye estas rutas automáticamente al construir cualquier forma (CR, CA, TR,…etc.) en el interior de la ruta del proyecto.

Nota de Exclusión: La estructura anterior aplica para los nuevos proyectos informáticos como SISFAMI, SISESI, SICOVEN, por lo tanto, se excluyen aquellos proyectos desarrollados posteriormente a la estandarización como el Sistema de Información SIIS.

Tablas Estándar

Los motivos se ingresaran un la librería BDUTIL en la tabla TBUTMOTIVO

```
TABLE DBUTIL.TBUTMOTIVO (
   MOCODMOTIV      CHAR          (4)      NOT NULL PRIMARY KEY,
   MONOMTABLA      CHAR          (10),
   MODESMOTIV      CHAR          (20)     NOT NULL,
   MOTIPMOTIV      CHAR          (1),
```

```
   MOCODCOLOR      CHAR          (10)     NOT NULL,
   MOCODDEFAU      CHAR          (1),
   MOCODESTAD      CHAR          (1)      NOT NULL,
   MOCODESTAU      CHAR          (1)      NOT NULL,
   MOCODMOTAU      CHAR          (4)      NOT NULL,
   MOCLASIF1       CHAR          (2),
   MOCLASIF2       CHAR          (2),
   MOCLASIF3       CHAR          (2),
   MOCLASIF4       CHAR          (2),
   MOCLASIF5       CHAR          (2),
   AUNOMEQUIP      CHAR          (20)     NOT NULL,
   AUDIRIP         CHAR          (20)     NOT NULL,
   AUNOMUSU        CHAR          (10)     NOT NULL,
   AUFECMOD        DECIMAL       (8, 0)   NOT NULL,
   AUHORMOD        DECIMAL       (6, 0)   NOT NULL
)
```

Los combos desplegables de tipo código descripción, como tipo de documento, sexo, parentesco, entre otros se almacenan también en BDUTIL en las tablas TBBDCODVAR y TBBDVARIOS

```
TABLE BDUTIL.TBBDVARIOS (
   VACODTABLA      DECIMAL       (5, 0)   NOT NULL PRIMARY KEY,
   VANOMTABLA      CHAR          (30),
   USCODUSUAR      CHAR          (10)     NOT NULL,
   AUCODESTAD      CHAR          (1),
   MOCODMOTIV      CHAR          (4)      NOT NULL,
   AUNOMEQUIP      CHAR          (20),
   AUDIRIP         CHAR          (20),
   AUFECMOD        DECIMAL       (8, 0),
   AUHORMOD        DECIMAL       (6, 0)
)

TABLE BDUTILPRU.TBBDCODVAR (
   VACODTABLA      DECIMAL       (5, 0)   NOT NULL,
   VACODDESCR      CHAR          (30),
   VADESCRIPC      CHAR          (150),
   USCODUSUAR      CHAR          (10)     NOT NULL,
   VACODEST        CHAR          (1),
   MOCODMOTIV      CHAR          (4)      NOT NULL,
   AUNOMEQUIP      CHAR          (20),
   AUDIRIP         CHAR          (20),
   AUFECMOD        DECIMAL       (8, 0),
   AUHORMOD        DECIMAL       (6, 0)
)
```