

**Guía de configuración mínima de procesos software para el desarrollo de trabajos de grado en la universidad popular del cesar seccional Aguachica**

**Yohn Timy Lopez Gomez**

Ingeniero de Sistemas

Grupo de Investigación CICOM

Universidad de Pamplona

Trabajo para optar al título de Magister en Gestión de Proyectos Informáticos



Maestría en Gestión de Proyectos Informáticos

Facultad de Ingenierías y Arquitectura

Universidad de Pamplona

Pamplona, febrero de 2016

**Guía de configuración mínima de procesos software para el desarrollo de trabajos de grado en la universidad popular del cesar seccional Aguachica**

**Yohn Timy Lopez Gomez**

Ingeniero de Sistemas

Universidad Popular del Cesar

Director

**Mg. Luis Alberto Esteban Villamizar**

**Maestría en Gestión de Proyectos Informáticos**

Grupo de Investigación CICOM

Línea de Investigación:

Ingeniería del Software



Universidad de Pamplona

Pamplona, febrero de 2016

Nota de Aceptación:

---

---

---

---

---

Firma del Jurado

---

Firma del Jurado

---

Firma del Jurado

Pamplona, \_\_\_\_\_ de \_\_\_\_\_ 2016

## **AGRADECIMIENTOS**

Gracias a Dios, por colocar en mí camino a las personas que se convirtieron en mi apoyo, compañía y orientación especialmente durante esta etapa de estudio, por iluminar mi mente para facilitar mi trabajo y mi formación profesional.

Gracias a mi mamá y a mi tío Tacho, por su apoyo incondicional permanente, porque gracias a su amor y sus cuidados hoy soy una persona de bien.

Gracias a mi esposa, por su amor y apoyo constante, por hacer que el tiempo en especial las noches que dedique a mis estudios fueran más agradables.

Gracias a mi director de trabajo de grado Luis Alberto Esteban Villamizar, quien con su paciencia y guía, permitieron llevar a feliz término este proyecto gracias por ser más que un director, un amigo, gracias por el tiempo dedicado y sus aportes, que siempre fueron acertados y permitieron la culminación de este trabajo en las mejores condiciones.

Gracias a los Docentes de la Universidad de Pamplona, en especial a Luis Esteban Villamizar, Aylin Orjuela y Mauricio Rojas, quienes con sus conocimientos y orientaciones contribuyeron en mi formación profesional y alcanzar la meta deseada.

Gracias a todos mis compañeros de la Maestría en Gestión de Proyectos Informáticos por los momentos compartidos y hacer más agradable esta etapa de mi vida, con especial afecto a Katerine, Darwin, Yasser, Cesar.

A todos, eterna gratitud y sincero agradecimiento.

## **DEDICATORIA**

*A mi mamá: Cenaida*

*A mi tío Tacho*

*A mi esposa: Taty*

*A mis Hijas: Marianita y Majo*

*Mis Hermanos:*

*Chepito (mi hermanito)*

*Cesar*

*Sonia*

A todos, "porque Dios siga siendo generoso con  
nosotros"

## Tablas de Contenido

RESUMEN .....	12
ABSTRAT .....	13
1 Introducción .....	14
1.1 Área del conocimiento.....	15
1.2 Descripción del problema.....	16
1.3 Justificación.....	17
1.4 Objetivos .....	18
1.4.1 Objetivo general.....	18
1.4.2 Objetivos específicos.....	18
1.5 Pregunta de investigación .....	18
1.6 Metodología o diseño metodológico de la investigación .....	19
1.6.1 Enfoque cualitativo .....	19
1.6.2 Alcance descriptivo.....	20
1.6.3 Diseño del proceso de investigación .....	21
1.6.4 Actividades realizadas .....	22
1.6.5 Fuentes de datos consultadas.....	24
2 Marco Teórico y Estado del Arte.....	25
2.1 Marco teórico .....	25
2.1.1 Proceso software .....	27
2.1.2 Modelos de ciclo de vida .....	29
2.1.3 Metodología de desarrollo de software.....	38
2.1.4 Tipos de metodologías .....	39

2.1.5	Estándares para definición de procesos.....	74
2.1.6	Modelos de mejoramiento de procesos de software.....	78
2.1.7	Descripción del proceso software.....	83
2.2	Estado del arte.....	85
2.2.1	El proceso software y de la adaptación de procesos.....	87
2.2.2	Nacional.....	90
2.2.3	Análisis de contexto local.....	93
3	Guía para la configuración mínima de proceso software.....	135
3.1	Objetivo y alcance.....	135
3.2	Responsable.....	136
3.3	Definiciones.....	136
3.4	Limitaciones.....	138
3.5	Contenido.....	138
3.6	Anexos.....	144
3.7	Validación.....	144
3.7.1	Elaboración del instrumento de validación.....	145
3.7.2	Escala de juicios valorativos.....	145
3.7.3	Procedimiento de evaluación.....	146
3.7.4	Validación del instrumento.....	146
3.7.5	Selección de los expertos.....	146
3.7.6	Aplicación del instrumento de validación.....	147
	CONCLUSIONES.....	158
	BIBLIOGRAFIA.....	159
	ANEXO A.....	166
	ANEXO B.....	176

ANEXO C .....	180
ANEXO D .....	182
ANEXO E .....	183
1 Guía para la configuración mínima de proceso software .....	183
2 Objetivo y Alcance .....	183
3 Responsable.....	184
4 Definiciones .....	184
5 Limitaciones.....	186
6 Contenido .....	186
1 Determinar el alcance del software a desarrollar .....	187
2 Justificar el tamaño del equipo .....	188
3 Seleccionar la metodología.....	188
4 Describir el uso de la metodología en términos de: actividades, roles, artefactos y herramientas/ tecnología .....	188
7 Anexos.....	192



## LISTA DE TABLAS

Tabla 1 cuadro comparativo entre metodologías escogidas para el estudio: Fuente el autor .....	107
Tabla 2 Ventajas y desventajas de las metodologías seleccionadas para el estudio: Fuente el autor .....	113
Tabla 3 Artefactos y entregables Vs Metodología: Fuente el autor .....	122
Tabla 4 Distribución de los Proyectos de grado en Ingeniería de Sistemas Universidad Popular del Cesar – Seccional Aguachica: Fuente el autor.....	123
Tabla 5 Metodologías más usadas en proyectos de grado .....	124
Tabla 6 Procesos de Predesarrollo Vs Formato UPC: Fuente el autor .....	129
Tabla 7 Procesos de Desarrollo Vs Formato UPC .....	130
Tabla 8 Procesos de Postdesarrollo Vs Formato UPC: Fuente el autor.....	132
Tabla 9 Procesos Integrales Vs Formato UPC: Fuente el autor.....	133
Tabla 10 actividades guía de configuración mínima de proyectos de software. Fuente: el autor .....	139
Tabla 11 Formato de Selección de Configuración del Proceso Software: Fuente el autor .....	143
Tabla 12 Escala de juicios valorativos: Fuente el autor.....	146
Tabla 13 Relación de expertos, experiencia y formación: Fuente el autor .....	147
Tabla 14 Tabulación del formato del experto 1: Fuente el autor .....	148
Tabla 15 Promedio de resultados experto 1: Fuente el autor.....	148
Tabla 16 Tabulación del formato del experto 2: Fuente el autor .....	149
Tabla 17 Promedio de resultados experto 2: Fuente el autor.....	149
Tabla 18 Tabulación del formato del experto 3: Fuente el autor .....	150
Tabla 19 Promedio de resultados experto 3: Fuente el autor.....	151
Tabla 20 Tabulación del formato del experto 4: Fuente el autor .....	151
Tabla 21 Promedio de resultados experto 4: Fuente el autor.....	152
Tabla 22 Resultado final Juicio de Expertos: Fuente el autor .....	152
Tabla 23 Formato de Selección de Configuración del Proceso Software: Fuente el autor .....	191

## LISTA DE FIGURAS

Figura 1 Ciclo de vida en cascada. Fuente: José Salvador Sánchez Garreta, Ingeniería de proyectos informáticos: actividades y procedimientos.....	30
Figura 2 Modelo Incremental: Fuente Roger Pressman, Ingeniería del Software 6º edición.....	31
Figura 3 Modelo en espiral fuente: José Salvador Sánchez Garreta, Ingeniería de proyectos informáticos: actividades y procedimientos.....	33
Figura 4 Modelo en V. Fuente: (Universidad Nacional de Jujuy Campus Virtual - Facultad de Ingeniería / ANALISIS Y DISEÑO DE SISTEMAS I.....	34
Figura 5 Proceso y roles MDD: Fuente Desarrollo de Software Dirigido por Modelos. Conceptos Teóricos y su Aplicación Práctica.....	36
Figura 6 Forma Genérica de las Metodologías de Desarrollo .....	38
Figura 7 Proceso RUP: Fuente Roger Pressman 6º edición .....	40
Figura 8 Proceso RAD: Fuente Roger Presuman, Ingeniería del Software 6º edición .....	42
Figura 9 Proceso XP Fuente J. Donovan Wells .....	50
Figura 10 Proceso SCRUM. Fuente: <a href="http://proyectosagiles.org/como-funciona-scrum/">http://proyectosagiles.org/como-funciona-scrum/</a> .....	53
Figura 11 Proceso de AUP Fuente.....	58
Figura 12 Proceso MSF, Fuente: Microsoft Solutions Framework Essentials .....	62
Figura 13 Proceso FDD: Fuente: A practical guide to feature-driven development.....	64
Figura 14 Framework SOHDM. Fuente: A scenario-based object-oriented hypermedia design methodology.....	69
Figura 15 Proceso WSDM. Fuente: WSDM: A User Centered Design Method for Web Sites .....	70
<i>Figura 16 Modelos de la UWE. Fuente: UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio .....</i>	<i>73</i>

Figura 17 Grupos de procesos de la norma ISO-12207: Fuente Mg. Luis Omar Tangarife Téllez.....	75
Figura 18 IEEE-1074. Fuente:.....	78
Figura 19 Esquema escalonado de CMMI. Fuente: Mauricio Morales, Manual de Administración de Proyectos .....	80
Figura 20 Análisis de Proyectos de grado en Ingeniería de Sistemas Universidad Popular del Cesar – Seccional Aguachica .....	124
Figura 21 Porcentaje de proyectos por metodología: Fuente el autor .....	125
Figura 22 IEEE-1074 Grupos de Procesos Orientados al desarrollo .....	128
Figura 23 Procedimiento de la guía de configuración de procesos software .....	135
Figura 24 Grafica del resumen de la evaluación de expertos: Fuente el autor ...	153
Figura 25 Coeficiente de Kendall: Fuente .....	154
Figura 26 Definición de las variables a evaluar en SPSS: Fuente el autor .....	154
Figura 27 Datos para las variables a evaluar: Fuente el autor .....	155
Figura 28 Búsqueda del método de correlación para k muestras relacionadas: Fuente el autor .....	155
Figura 29 Selección de las variables y del método w de Kendall: Fuente el autor .....	156
Figura 30 Resultados generados por el SPSS: Fuente el autor .....	156

## **RESUMEN**

Este documento presenta como producto de investigación una guía para la configuración de procesos de software mínimos, en trabajos de grado que involucren el desarrollo de software como producto principal, en el programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica.

Dentro del proceso de investigación se realizó un análisis de los trabajos de grado ya ejecutados desde la perspectiva del estándar IEEE 1074, y un análisis de diversos modelos y metodologías de desarrollo de software que son susceptibles de utilizar en el desarrollo de dichos trabajos.

La guía se constituye entonces, en una forma de adaptar procesos de software a contextos particulares como es el caso de los trabajos de grado.

## **ABSTRAT**

This document presented as product research a guidance for setting minimum software processes in grade work involving software development as the main product in the Systems Engineering program of the Popular University of Cesar Aguachica Sectional.

Within the research process an analysis of the degree works already executed from the perspective of the standard IEEE 1074, and an analysis of various models and software development methodologies that are likely to use in the development of this work was performed.

The guide is then, in a way to tailor software processes to particular contexts such as degree of work

## **1 Introducción**

En el presente el documento se propone una guía de configuración de procesos mínimos de Software para los trabajos de grado de Universidad Popular del Cesar Seccional Aguachica, que involucren desarrollo de software como producto principal.

Este proyecto nace de la necesidad de promover un marco común de evaluación para jurados, directores y estudiantes, además de una ruta de desarrollo, adaptada a las necesidades de cada proyecto en particular, que permita establecer los elementos mínimos del proceso software sin que implique perder la formalidad de un proyecto de ingeniería, pero que permita que el estudiante pueda acelerar su proceso de grado.

El desarrollo de la guía se inició con la revisión bibliográfica de la literatura disponible, en algunas bases de datos en línea, así como también el análisis de los proyectos desarrollados en la Universidad Popular del Cesar Seccional Aguachica, la entrevista a evaluadores y directores de proyectos de grado que incluyen software como producto, además de la correlación y comparación entre el proceso de la Universidad y el estándar IEEE 1074.

Finalmente se redactó la guía y se validó su utilidad a través de la técnica del juicio de expertos, para lo cual se seleccionaron a profesores del programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica, que fueran o hubieran sido directores o evaluadores de proyectos de grado con producto software, por sus conocimientos en el proceso, además de su formación y calidades académicas.

## 1.1 Área del conocimiento

El proyecto liderado por la IEEE, (Muñoz, 2006) SWEBOK dice Juan Garbajosa “Nace como una descripción del conocimiento asociado a la disciplina de ingeniería de software, aunque no se descarta que en un futuro una entidad profesional lo utilice como certificación”<sup>1</sup>. Esto permite referenciar un tema dentro del área de conocimiento de la Ingeniería del Software con exactitud.

Este proyecto, se enmarca en la gestión del proceso de ingeniería del software y más puntualmente en la definición de procesos y como tema de estudio la adaptación de procesos software<sup>2</sup>.

Tiene como finalidad la creación de una guía de selección, a través de la cual se podrá definir la menor cantidad de procesos de software, que garanticen el cumplimiento de los objetivos de los proyectos de la Universidad Popular del Cesar Seccional Aguachica que incluyan desarrollo de Software como opción de grado. Teniendo en cuenta las limitaciones de tiempo, alcance, equipo de desarrollo de los estudiantes, además servirá para tener una base común de evaluación, porque se determinara los entregables a tener en cuenta por parte de directores, estudiantes y jurados.

La validación de la guía de selección y configuración mínima de procesos software se hizo mediante la opinión de expertos, por profesores encargados de revisar y calificar proyectos de grado en la Universidad Popular del Cesar Seccional Aguachica a través la aplicación de un instrumento de evaluación diseñado con este

---

<sup>1</sup> “Swebok (Software Engineering Body of Knowledge) surge como una necesidad dentro de la industria”. Así explicó Juan Garbajosa, catedrático en la Universidad Politécnica de Madrid (UPM), la llegada de este nuevo concepto de validación internacional de la disciplina de ingeniería de software a nuestro país, durante su intervención en la II Semana del CMMI organizada por Caelum en Madrid.

<sup>2</sup> En inglés, software process tailoring.

propósito y un análisis de consenso realizado mediante el coeficiente  $w$  de Kendall (Badii, Guillen, Lugo Serrato, & Aguilar Garnica, 2014).

## 1.2 Descripción del problema

Aunque el Desarrollo de Software es un campo disciplinar y en constante evolución, en este proceso se han dado pasos importantes en la solución de problemas inherentes a la complejidad del proceso, la aparición de la ingeniería del Software y con ella las metodologías de desarrollo, recomendaciones, buenas prácticas, modelos de mejora y otras herramientas que ayudan a lograr un producto que responda a las necesidades del usuario, dentro de los límites de tiempo, presupuesto planeados y con la calidad esperada, sin embargo se tienen problemas muchas veces con la implementación de estas herramientas.

En la Universidad Popular del Cesar Seccional Aguachica una de las opciones más solicitadas para proyecto de grado en estudiantes de la facultad de Ingeniería de Sistemas es el Desarrollo de Software, donde el futuro profesional puede profundizar conceptos y además de poner en práctica sus conocimientos en el desarrollo de un producto tangible. Esto a través de la implementación de un modelo de ciclo de vida y de la aplicación de alguna metodología, que se crea se ajusta al proyecto.

Sin embargo actualmente existen muchas metodologías de desarrollo de software, y se debe escoger una, *“la que se ajuste más al proyecto”*. Pero en la realidad estas metodologías se aplican de manera incompleta por parte los directores y estudiantes próximos a concretar su trabajo de grado, esto se debe a que los proyectos no se ajustan a los requerimientos de las metodologías, porque normalmente es un solo estudiante o a lo mucho dos o en casos extremos tres, son los encargados de ejecutar los roles y generar los artefactos, propios de la metodología que se escoge para el desarrollo del proyecto, además hay que tener



en cuenta que la implementación de ciertas metodologías demoraría el tiempo del proyecto, perjudicando de esta manera a los futuros profesionales.

Esto también produce ambigüedades en la evaluación de los proyectos, porque los diferentes evaluadores, pueden tener diferentes percepciones acerca de cuáles elementos y entregables de las metodologías deberían de tenerse en cuenta (no hay un consenso en cómo adaptar la metodología al proyecto), para un desarrollo equilibrado del proyecto (documento & Software), produciendo demoras y muchas veces inconsistencias en los solicitudes de los jurados. Todo esto por no tener claro que se debería tener en cuenta en la evaluación del proyecto software.

### **1.3 Justificación**

El SWEBOK, (IEEE, 2004) en el área de conocimiento, proceso de ingeniería del software aclara “es importante señalar que los procesos predefinidos incluso los estandarizados deben adaptarse a las necesidades locales, por ejemplo, el contexto organizacional, el tamaño del proyecto, los requisitos reguladores, las prácticas industriales y las culturas corporativas.”

El proceso software de los proyectos de desarrollo de Software de los estudiantes de la Universidad Popular del Cesar, se debe contextualizar al alcance, tiempo y equipo de desarrollo que deben disponer los estudiantes para la culminación de sus proyectos; para lograr agilidad en el proceso de grado y contar con un marco común de trabajo en el que directores, evaluadores y estudiantes sepan con exactitud cuáles son los entregable que deberían o se podrían tener en cuenta durante estos desarrollos.

El contar con una guía de selección de procesos, que minimice el número de elementos de una metodología en particular, sin que se vea afectado el resultado final, acelera los tiempos de desarrollo, y el contar con entregables bien definidos reduce las ambigüedades en la evaluación, por lo que se podrá realizar de manera

más rápida y efectiva, esto impacta de manera positiva los tiempos de ejecución de los proyectos de grado y procesos de graduación, favoreciendo al programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica y especialmente a los estudiantes, futuros profesionales.

## **1.4 Objetivos**

### 1.4.1 Objetivo general.

Definir una guía de Configuración mínima de procesos software para el desarrollo de proyectos software que sirven como opción de trabajo de grado en la Universidad Popular del Cesar - Seccional Aguachica

### 1.4.2 Objetivos específicos.

- ✓ Elaborar un estado del arte acerca del proceso de Software y de su adaptación.
- ✓ Definir los elementos de la guía de configuración mínima de procesos de software de acuerdo a las metodologías de desarrollo escogidas
- ✓ Validar la guía de configuración mínima de procesos software por medio del juicio de expertos en evaluación de proyectos de grado y desarrollo de software

## **1.5 Pregunta de investigación**

¿Cómo adaptar el proceso de software a las necesidades y limitaciones de los proyectos de desarrollo de software de la Universidad Popular del Cesar Seccional Aguachica?

## 1.6 Metodología o diseño metodológico de la investigación

En este apartado se mostraran el enfoque dado a la investigación, así como también los métodos, técnicas empleadas y las actividades necesarias, para la consecución de los objetivos.

### 1.6.1 Enfoque cualitativo

**Investigación Cualitativa.** “La investigación cualitativa se enfoca a comprender y profundizar los fenómenos, explorándolos desde la perspectiva de los participantes en un ambiente natural y en relación con el contexto.” (Hernández, Fernández, & Baptista, 2010).

Los procesos cualitativos ayudan a enfocar el estudio directamente a los individuos implicados en el<sup>3</sup>, analizar y comprender un fenómeno social, dentro de un entorno académico para profundizar sobre el problema por medio de técnicas o métodos como las encuestas, entrevistas, análisis documental, entre otros, y exponer las conclusiones obtenidas a la comunidad interesada.

La investigación se enmarco en un enfoque de investigación cualitativo porque se utilizaron instrumentos de recolección de información conversacionales y una recopilación de experiencias de expertos en donde se pretendió generar un avance en la situación actual de la problemática planteada.

Las entrevistas realizadas son de carácter abierto, donde se toman los criterios y experiencias obtenidas con la práctica de un número específico de expertos en el tema de desarrollo de software y quienes han venido haciendo la labor de guía y evaluación de trabajos de grado en la universidad.

---

<sup>3</sup> En este caso los estudiantes, directores y jurados de los trabajos de grado en el programa de ingeniería de sistemas de la universidad popular del cesar Sede Aguachica.

Además se realizó un análisis bibliográfico de la literatura disponible, un análisis de los trabajos de grado de la universidad en los que el producto es software y un análisis del contexto en que se desarrollan los productos software y se comparó frente al estándar IEEE 1074, con el objetivo de conocer y ambientar el entorno en que se construyen estos sistemas.

#### 1.6.2 Alcance descriptivo

**Investigación Descriptiva.** (Sabino, 2007) “La investigación descriptiva busca especificar propiedades, características y rasgos importantes de cualquier fenómeno que se analice<sup>4</sup>. Describe tendencias de un grupo de población”.

La investigación del presente trabajo es de tipo descriptiva, porque inspecciona los eventos de un fenómeno específico como es la adaptación de procesos de software en los trabajos de grado presentados en la Universidad Popular del Cesar Seccional Aguachica en el área de desarrollo de software, con el fin de comprender y postular unos instrumentos que colaboren con la evaluación de los futuros trabajos, así como generar la orientación a los estudiantes hacia los entregables requeridos para dar cumplimiento a satisfacción de su investigación para optar al título de pregrado en Ingeniería de Sistemas.

Además permite organizar y medir el resultado de las diferentes observaciones, características y hechos que se obtuvieron del estudio, como es el nivel de aceptación en la comunidad docente con respecto a la utilización de la **Guía de configuración mínima de procesos software para el desarrollo trabajos de grado en la Universidad Popular del Cesar Seccional Aguachica**

---

<sup>4</sup> La adaptación de procesos de software en particular en los trabajos de grado de los estudiantes de ingeniería de sistemas de la Universidad Popular del Cesar, sede Aguachica.

### 1.6.3 Diseño del proceso de investigación

Para el desarrollo del proyecto se debieron realizar una serie de actividades, estas actividades obedecen a un diseño previo, un plan de trabajo o a una forma de trabajo, para este proyecto se pensó que el orden lógico sería el siguiente.

Como primer paso se consultó la literatura usando la técnica de **revisión documental**<sup>5</sup> referente al proceso del software, acerca de los modelos de ciclo de vida, los estándares para la definición de procesos así como también de las diferentes metodologías, modelos de mejora y adaptación de procesos, con el fin de establecer un estado del arte y un marco de referencia acerca de la adaptación y simplificación de procesos en el desarrollo de software.

Luego de conocer el estado actual del proceso de software y las metodologías relevantes y sobre aquellas que se pueda conseguir mayor documentación se realizó un **análisis comparativo**<sup>6</sup> acerca de los procesos, artefactos y entregables que usa cada metodología.

Además se usó el **método de abstracción científica**<sup>7</sup> para analizar los proyectos de grado para determinar cuáles son las metodologías preferidas por docentes y profesores y de esta manera contextualizar y acotar la investigación a las necesidades de la Universidad y su comunidad.

---

<sup>5</sup> Es una técnica de revisión y de registro de documentos que fundamenta el propósito de la investigación y permite el desarrollo del marco teórico y/o conceptual.

<sup>6</sup> Método Comparativo/ análisis comparativo, procedimiento de la comparación sistemática de casos de análisis que en su mayoría se aplica con fines de generalización empírica y de la verificación de hipótesis.

<sup>7</sup> El método científico implica una abstracción, necesaria para entender el funcionamiento de la realidad. En el proceso de abstracción se omiten detalles y se establecen hipótesis y esquemas lógicos que permiten relacionar los hechos.

Luego con la ayuda de **método sistémico**<sup>8</sup> se determinaron los criterios de adaptación que se tuvieron en cuenta para proponer la guía de configuración mínima de procesos para cada metodología escogida.

Para terminar se validó la guía a través del **juicio de expertos**<sup>9</sup> en evaluación de proyectos de grado, que tengan desarrollo de software como producto principal, para lograr esto se diseñó un instrumento el cual fue aplicado por cada experto y las conclusiones de cada uno se tabularon y promediaron para obtener un resultado final. Este resultado se validó a través del método de concordancia w de Kendall (Escobar Pérez & Cuervo Martínez, 2006), que mide el coeficiente de correlación de los juicios emitidos por los expertos.

Para el logro de estas actividades se usara un enfoque cualitativo y el método descriptivo, además se usaran la técnica de conversación asociada a la entrevista, la revisión bibliográfica, la observación directa, la técnica de observación de registros, el análisis comparativo, además se usaran algunas técnicas estadísticas simples.

#### 1.6.4 Actividades realizadas

1. Consultar la literatura referente al proceso software, acerca de los modelos de ciclo de vida, los estándares para la definición de procesos así como

---

<sup>8</sup> Está dirigido a modelar el objeto mediante la determinación de sus componentes, así como las relaciones entre ellos. Esas relaciones determinan por un lado la estructura del objeto y por otro su dinámica.

<sup>9</sup> (Robles & Rojas, 2015) La evaluación mediante el juicio de expertos, método de validación cada vez más utilizado en la investigación, dicen Cabero y Llorente “consiste, básicamente, en solicitar a una serie de personas la demanda de un juicio hacia un objeto, un instrumento, un material de enseñanza, o su opinión respecto a un aspecto concreto”

también de las diferentes metodologías, modelos de mejora y adaptación de procesos.

2. Definir un estado del arte acerca de la adaptación de procesos en el marco del desarrollo de software.
3. Presentar el informe de avance del proyecto.
4. Realizar un análisis comparativo acerca de los procesos, artefactos y entregables que usa cada metodología escogida para el estudio.
5. Recopilar información acerca de los proyectos de grado del programa de Ingeniería de Sistemas de la Universidad Popular del Cesar – Seccional Aguachica con el fin de saber que tan frecuente son los proyectos de grado que involucran desarrollo de Software.
6. Analizar la metodología usada en los proyectos que involucran desarrollo de software como producto principal para determinar las metodologías más usadas en los proyectos de grado.
7. Mapear y analizar el estándar IEEE 1074 Standard for Developing Software Life Cycle Processes de forma comparativa frente a los procesos empleados en el desarrollo de software en los proyectos de grado del programa de Ingeniería de Sistemas de la Universidad Popular del Cesar - Seccional Aguachica, para entender que procesos son relevantes en el desarrollo del proyecto.
8. Proponer el formato guía para la selección de la configuración mínima de procesos software de acuerdo a la información recolectada.
9. Diseñar un instrumento de verificación, para la validación de la guía de selección de configuración mínima de procesos software.
10. Validar la guía a través del juicio de expertos en evaluación de proyectos de grado, que tengan desarrollo de software como producto principal, aplicando el instrumento diseñado para este fin.
11. Tabular y promediar las conclusiones de cada uno de los expertos para obtener un resultado final.

12. Realizar análisis del consenso de los expertos mediante el método de coeficiente  $w$  de Kendall
13. Presentar para evaluación un artículo acerca del estado del arte de la adaptación de procesos de software.

#### 1.6.5 Fuentes de datos consultadas

Entre las fuentes de datos consultadas en la elaboración del presente estudio están, las opiniones recogidas de los directores y evaluadores de proyectos de grado de la universidad (ver anexo D), Bases de Datos en internet (SCOPUS, SCIENCE DIRECT, GOOGLE BOOKS, GOOGLE ACADEMICO), así como también, informes en formato digital acerca de los proyectos de grado de la Universidad Popular del Cesar Seccional Aguachica, y los proyectos de grado algunos digitales y otros en físico consultados directamente en la biblioteca de la Universidad.



## 2 Marco Teórico y Estado del Arte

En esta sección se hace un recorrido a través de la evolución de la ingeniería del software, desde su definición, pasando por los modelos de ciclos de vida, metodologías, modelos de mejora, estándares y buenas prácticas hasta el tema central del proyecto **la adaptación del proceso software**

### 2.1 Marco teórico

Como define Benet Campderrich en su libro Ingeniería del software (CAMPDERRICH, 2003) “La ingeniería de software es un conjunto integrado de programas que en su forma definitiva se pueden ejecutar, pero comprende también las definiciones de estructuras de datos que utilizan, de igual manera la documentación referente a todo ello, tanto para el usuario como los estándares de desarrollo”<sup>10</sup>.

Por lo tanto, la ingeniería de software es una disciplina que comprende todos los aspectos de la producción de software desde etapas iniciales de la especificación hasta el mantenimiento de este después de que se utiliza, como lo orienta (Somerville, 2005) en su libro Ingeniería de software<sup>11</sup>.

---

<sup>10</sup> Es así que el proceso ingenieril de desarrollo de sistemas comprende una gran gama de elementos que hace la conceptualización exacta del manejo y realización de sistemas con el objetivo de generar soluciones informáticas a requerimientos propios de los usuarios aminorando costos en producción y evitando errores predecibles durante su programación e implementación.

<sup>11</sup> De esta manera deja al ingeniero como el que da las soluciones y aplica teorías al descubrimiento de las mismas, así mismo esta disciplina va desde la creación hasta su uso en el campo propio de aplicación de cada sistema software.

Teniendo presente lo anterior se puede considerar el software como un producto industrial y darle el tratamiento propio que este genera como se menciona en el libro Ingeniería del software (Campderrich, 2003)<sup>12</sup> .

El término ingeniería del software se usó por primera vez en una conferencia de la OTAN en 1968, para llamar la atención acerca de los problemas de esta y de buscar soluciones que se pudieran aplicar y ayudar a reducir la complejidad de construir software que funcione, en los tiempos, presupuesto y con la calidad deseada. Algunas de las definiciones de autores reconocidos son:

(Pressman, Ingeniería del Software, Un Enfoque Práctico, 2002) cita una definición de Fritz Bauer (1972) "El establecimiento y uso de principios de ingeniería sólidos (métodos) a fin de obtener software económico que sea fiable y funcione en máquinas reales".

(Software Engineering Institute, 1990) "La ingeniería de software es la forma de la ingeniería que aplica los principios de la computación y las matemáticas para lograr soluciones rentables a los problemas de software".

(IEEE, 1990) "La aplicación de un enfoque disciplinado, cuantificable y sistemático, en el desarrollo, operación y mantenimiento de software".

(Pressman, 2010) "La ingeniería del software es una disciplina que integra procesos, métodos y Herramientas para el desarrollo de sistemas software"

---

<sup>12</sup> "Un software no es una obra de arte sino un producto de consumo utilitario y masivo; para una empresa o trabajador autónomo, el software es un medio auxiliar que interviene de manera más o menos indirecta, pero a menudo imprescindible, en su gestión y cada vez más en su proceso productivo; también existe, como todos sabemos, un consumo privado del software. Por lo tanto, se puede considerar plenamente como un producto industrial."

### 2.1.1 Proceso software

Un proceso software es un conjunto estructurado de actividades usado por una organización para desarrollar, operar o mantener un producto software y que proporcionan un marco de trabajo desde el cual se puede establecer un plan para obtener un producto deseado. Tiene como característica proponer las actividades sin pretender como deben hacerse o en qué momento hacerlo (INTECO, 2009).

(Pressman, 2005) Un proceso de software se define como un marco de trabajo para las tareas que se requieren en la construcción de software de alta calidad. El proceso de software define el enfoque que se adopta mientras el software está en desarrollo.

(Pressman, 2010) La estructura del proceso establece el fundamento para el proceso completo de Ingeniería del Software, por medio de la identificación de un número pequeño de actividades estructurales que sean aplicables a todos los proyectos software, sin importar su tamaño o complejidad.

Según (Pressman, 2010) El siguiente marco de trabajo genérico del proceso describe actividades que se pueden aplicar en la inmensa mayoría de los proyectos software:

- ✓ Comunicación: Esta actividad del marco de trabajo implica una intensa colaboración y comunicación con los clientes; además, abarca La investigación de requisitos y otras actividades relacionadas.
- ✓ Planeación: Esta actividad establece un plan para el trabajo de la Ingeniería del Software. Describe las tareas técnicas que deben realizarse, los riesgos probables, los recursos que serán requeridos, Los productos del trabajo que han de producirse y un programa de trabajo.

- ✓ Modelado: Esta actividad abarca la creación de modelos que permiten al desarrollador y al cliente entender mejor los requisitos del software y el diseño que logrará satisfacerlos.
- ✓ Construcción: Esta actividad combina la generación del código (ya sea manual o automatizado) y la realización de pruebas necesarias para descubrir errores en el código.
- ✓ Despliegue: El software (como una entidad completa o un incremento completado de manera parcial) se entrega al cliente, quien evalúa el producto recibido y proporciona información basada en su evaluación.

Estas cinco actividades genéricas del marco de trabajo son útiles durante el desarrollo de programas pequeños, la creación de grandes aplicaciones en la red, y en la Ingeniería de sistemas basados en computadoras grandes y complejas. Los detalles del proceso del software serán muy diferentes en cada caso, pero las actividades dentro del marco permanecerán iguales.

Por otro lado (Somerville, 2005) en su libro ingeniería del software enmarca los procesos en cuatro actividades, que se mencionan a continuación:

- ✓ Especificación del software, en donde se define la funcionalidad y las restricciones en su operación, es decir, todos los requerimientos que tiene o que va a tener el sistema en su etapa de funcionamiento.
- ✓ Diseño e implementación, dando respuesta a los requerimientos anteriormente descritos, se construye el software cumpliendo las especificaciones.
- ✓ Validación del software, en donde se deben hacer las pruebas respectivas para verificar que el software hace lo que se necesita que realice.
- ✓ Evaluación del software, este debe evolucionar de acuerdo a las necesidades cambiantes que se puedan presentar en el entorno, sean por cambio de tecnología o adaptación de nuevos requerimientos.

Las dos descomposiciones anteriores del proceso de software nos llevan al desarrollo del ciclo de vida del software y hacen notar que los procesos en el desarrollo e implementación de software son de gran utilidad, debido a la facilidad de fragmentar las tareas tanto en el momento de desarrollarlo, como en su ejecución y futuras mejoras.

### 2.1.2 Modelos de ciclo de vida

El término modelo de ciclo de vida se refiere a un conjunto de fases por las que pasa un producto software desde que nace la idea hasta que es reemplazado o retirado, determina el orden de las fases del proceso de software, establece los criterios de transición entre fases y define las entradas y salidas de cada una de las fases.

Un proyecto de desarrollo de software, como cualquier proyecto de ingeniería se estructura en una serie de fases que configuran su ciclo de vida. En el caso de un proyecto informático estas fases se basan en el modo en que se organizan y se definen diferentes modelos o paradigmas de ciclo de vida.

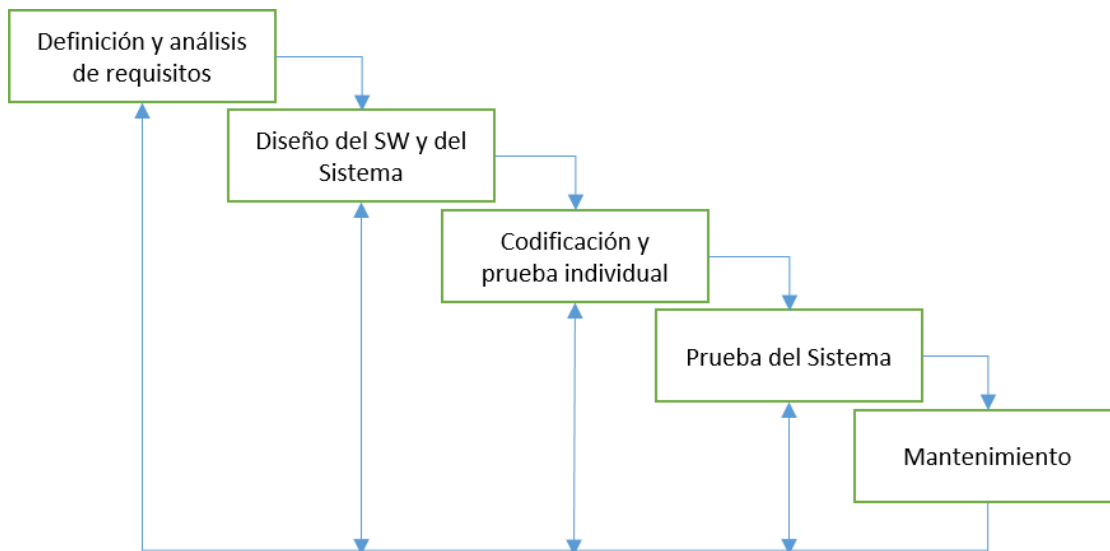
#### 2.1.2.1 *Modelo secuencial o en cascada*

Se basa en desarrollar cierto número de pasos que se realizan uno a continuación de otro, el modelo secuencial o en cascada fue uno de los más utilizados hace algunas décadas, una de las razones es la secuencialidad de las actividades que se proponen, ya que tienen un modo sencillo de seguir y comprender. Los pasos principales en que se divide son análisis, diseño, implementación y prueba, cada uno de estos pasos o fases cumple con una función específica dentro del proceso<sup>13</sup> (Sánchez, 2003).

---

<sup>13</sup> Estas fases son: Definición y análisis de requisitos; consiste en saber los requisitos del sistema, analizarlos y modelarlo, es decir establecer que necesitan los usuarios y que debe hacer el sistema

En algunos casos estas fases son más exploradas para mejorar los resultados, sin embargo las más utilizadas son las que se muestran a continuación



*Figura 1 Ciclo de vida en cascada. Fuente: José Salvador Sánchez Garreta, Ingeniería de proyectos informáticos: actividades y procedimientos.*

---

para cumplir estas necesidades teniendo en cuenta posibles restricciones que se presenten en el entorno, el ámbito, el alcance del sistema y de la información, las funciones e interfaces que se requieran, **Diseño del SW y del Sistema**; se enfoca al desarrollo de la estructura física de los datos, la arquitectura, como se van a conectar los diferentes módulos e interfaces con los usuarios, **Codificación y prueba individual**; es la traducción del diseño al lenguaje de hardware, es decir la inserción en el lenguaje de programación de la solución a las necesidades observadas, **Pruebas del sistema**; consiste en verificar todos los módulos de forma conjunta, su interconexión y funcionamiento en el sitio en donde se va a desenvolver el sistema, realizando las adaptaciones que sean necesarias durante el proceso, **El mantenimiento**; son todas las tareas o modificaciones que se realicen una vez que el software se pone en funcionamiento, en esta fase también se incluyen las posibles actualizaciones que se requieran en el tiempo.

### 2.1.2.2 Modelo Incremental

(Pressman, 2005) El modelo incremental combina elementos del modelo en cascada aplicados en forma iterativa. El modelo incremental aplica secuencias lineales de manera escalonada conforme transcurre el tiempo del proyecto.

En el modelo incremental, el primer incremento a menudo es un *producto esencial* (núcleo). Es decir, se afrontan requisitos básicos, pero muchas funciones suplementarias (algunas conocidas, otras no) quedan sin extraer. El cliente utiliza el producto central (o se realiza la revisión detallada). Como un resultado de utilización y/o de evaluación, se desarrolla un plan para el incremento siguiente. El plan afronta la modificación del producto central a fin de cumplir mejor las necesidades del cliente y la entrega de funciones, y características adicionales. Este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo (Pressman, 2005).

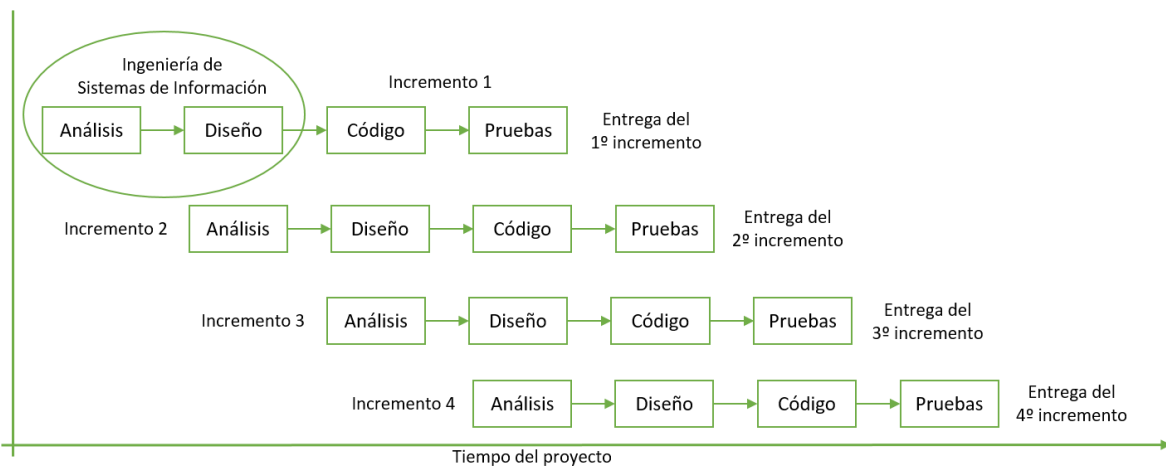


Figura 2 Modelo Incremental: Fuente Roger Pressman, Ingeniería del Software 6ª edición

### 2.1.2.3 Modelo de prototipo

Permite que todo el software o parte de él se construya rápidamente para ser comprendido fácilmente y aclarar aspectos que aseguren que el desarrollador y el cliente están de acuerdo en la solución que se está presentando para dar respuesta

a las necesidades planteadas. Este modelo se adopta cuando el usuario tiene objetivos diversos sin definir de manera clara los requerimientos del producto<sup>14</sup>. Los prototipos pueden ser utilizados posteriormente o desechados cuando se va implementando sus partes funcionales (LAWRENCE, 2002) .

Se pueden definir cuatro etapas que son: identificar requerimientos básicos del usuario, desarrollo del prototipo inicial, uso del prototipo, revisión y mejora del mismo. La construcción de prototipos se realiza en tres pasos:

- ✓ Escuchar al cliente. Se recolectan los requisitos. Se encuentran y definen los objetivos globales, se identifican los requisitos conocidos y las áreas donde es obligatorio más profundidad.
- ✓ Construir y revisar el prototipo.
- ✓ El cliente prueba prototipo y lo utiliza para refinar los requisitos del software.

#### *2.1.2.4 Modelo en espiral*

Se desarrolló para unificar en un paradigma las mejores características del ciclo de vida clásico y el modelo de prototipos añadiendo ciertas mejoras como el análisis de riesgos, en él se definen cuatro fases que se repiten, y a medida que se realizan se aumenta la complejidad estas fases son:

- ✓ Planificación, en donde se analizan los requerimientos, las restricciones y las posibles alternativas.
- ✓ Análisis de riesgo que se encarga de verificar las alternativas de costos, beneficios y probabilidades de éxito, en esta etapa se decide si se continúa o no con el desarrollo del sistema.

---

<sup>14</sup> Es decir cuando no se tienen claro el alcance, la parametrización y los resultados esperados del producto



- ✓ Ingeniería que es donde se desarrollan y mejoran los productos de las etapas anteriores
- ✓ Evaluación del usuario que es donde el cliente realiza pruebas y verifica su funcionalidad para establecer nuevos requisitos del sistema y comenzar de nuevo el ciclo

En la siguiente figura se muestra la adaptación que presenta (Sánchez, 2003) en su libro Ingeniería de proyectos informáticos: actividades y procedimientos, que orienta las fases del modelo en escala.

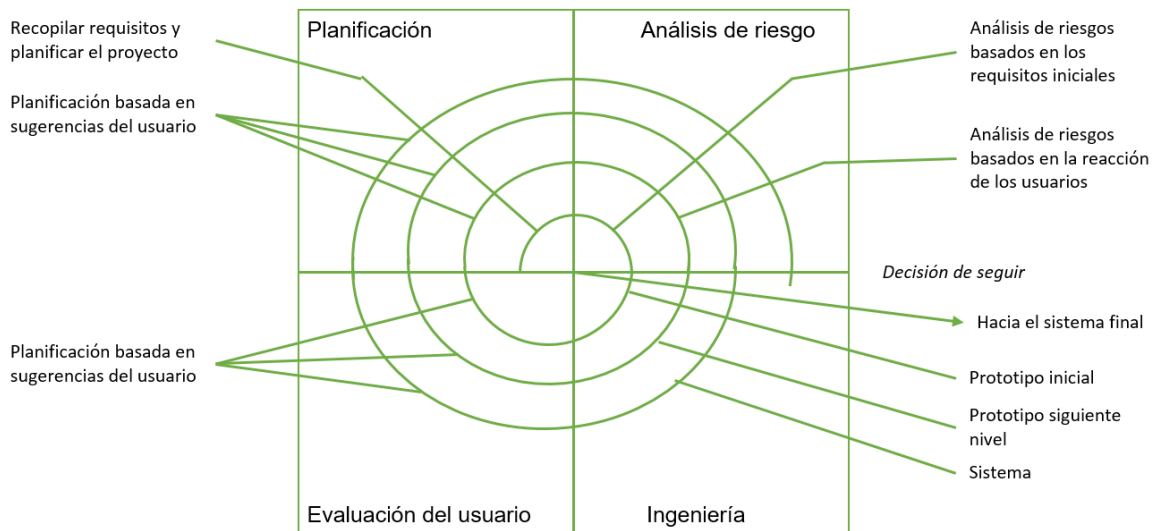


Figura 3 Modelo en espiral fuente: José Salvador Sánchez Garreta, Ingeniería de proyectos informáticos: actividades y procedimientos

En este modelo con cada iteración se obtienen nuevas versiones del software cada vez más completas, este es el enfoque más realista que se usa, debido a que permite tanto la utilización de prototipos para evaluar riesgos como el ciclo de vida del sistema para ser realizado de una forma secuencial, probando y evolucionando en cada fase su calidad y servicios

### 2.1.2.5 Modelo en V

El modelo en v<sup>15</sup> se desarrolló para terminar con algunos de los problemas que se vieron utilizando el enfoque de cascada tradicional. Los defectos estaban siendo encontrados demasiado tarde en el ciclo de vida, ya que las pruebas no se introducían hasta el final del proyecto<sup>16</sup>.

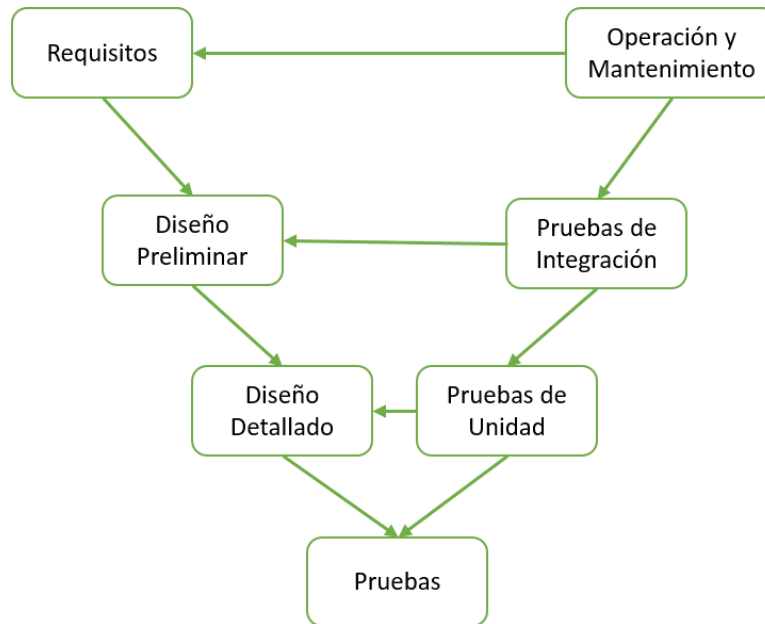


Figura 4 Modelo en V. Fuente: (Universidad Nacional de Jujuy Campus Virtual - Facultad de Ingeniería / ANALISIS Y DISEÑO DE SISTEMAS I

También muestra que las pruebas no son sólo una actividad basada en la ejecución. Hay una variedad de actividades que se han de realizar antes del fin de la fase de codificación<sup>17</sup>. Los productos de trabajo generados por los desarrolladores y

<sup>15</sup> la v representa la integración de partes y su verificación. V significa "Validación y Verificación".

<sup>16</sup> El modelo en v dice que las pruebas necesitan empezarse lo más pronto posible en el ciclo de vida.

<sup>17</sup> Estas actividades deberían ser llevadas a cabo en paralelo con las actividades de desarrollo, y los técnicos de pruebas necesitan trabajar con los desarrolladores y analistas de negocio de tal forma que puedan realizar estas actividades y tareas y producir una serie de entregables de pruebas

analistas de negocio durante el desarrollo son las bases de las pruebas en uno o más niveles. El modelo en v es un modelo que ilustra cómo las actividades de prueba (verificación y validación) se pueden integrar en cada fase del ciclo de vida. Dentro del modelo en v, las pruebas de validación tienen lugar especialmente durante las etapas tempranas<sup>18</sup>. El modelo en v es un proceso que representa la secuencia de pasos en el desarrollo del ciclo de vida de un proyecto (Universidad Nacional de Jujuy Campus Virtual - Facultad de Ingeniería / ANALISIS Y DISEÑO DE SISTEMAS I, s.f.).

Describe las actividades y resultados que han de ser producidos durante el desarrollo del producto. La parte izquierda de la v representa la descomposición de los requisitos y la creación de las especificaciones del sistema.

Realmente las etapas individuales del proceso pueden ser casi las mismas que las del modelo en cascada. Sin embargo hay una gran diferencia. En vez de ir para abajo de una forma lineal las fases del proceso vuelven hacia arriba tras la fase de codificación, formando una v. La razón de esto es que para cada una de las fases de diseño se ha encontrado que hay un homólogo en las fases de pruebas que se correlacionan.

#### *2.1.2.6 Desarrollo Dirigido por Modelos MDD*

En MDD<sup>19</sup>, este paradigma asigna a los modelos un rol central y activo; es decir los modelos son al menos tan importantes como el código fuente. Los modelos se van generando desde los más abstractos a los más concretos a través de transformaciones entre modelos y/o refinamientos, hasta llegar al código aplicando

---

<sup>18</sup> Por ejemplo, revisando los requisitos de usuario y después por ejemplo, durante las pruebas de aceptación de usuario

<sup>19</sup> MDD del inglés Model Driven Development.

una última transformación. La transformación entre modelos constituye el motor de MDD (PONS, GIANDIN, & PÉREZ, 2010). Los puntos claves de la iniciativa MDD fueron identificados en (Selic, 2008)<sup>20</sup>.

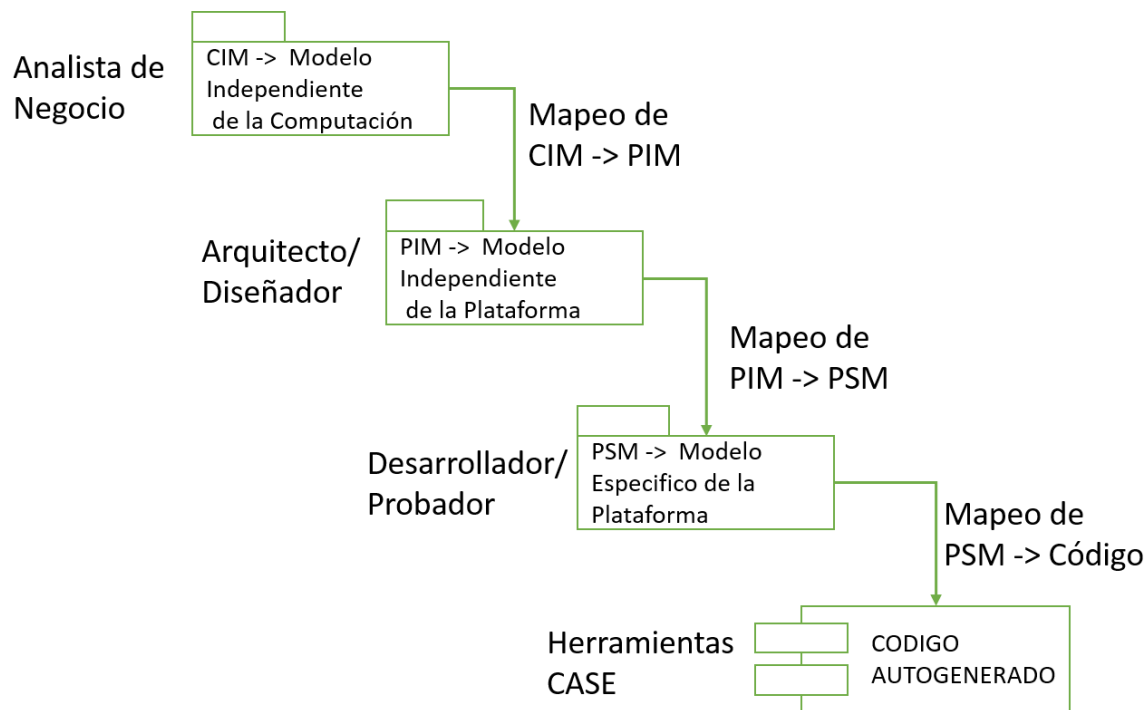


Figura 5 Proceso y roles MDD: Fuente Desarrollo de Software Dirigido por Modelos. Conceptos Teóricos y su Aplicación Práctica

La idea de crear modelos, no es nueva y en casi todas las metodologías hay alguna referencia a usar modelos para abstraer los detalles de alto nivel y separarlos de la

<sup>20</sup> a) El uso de un mayor nivel de **abstracción** en la especificación tanto del problema a resolver como de la solución correspondiente, en relación con los métodos tradicionales de desarrollo de software. b) El aumento de confianza en la **automatización** asistida por computadora para soportar el análisis, el diseño y la ejecución. c) El uso de **estándares** industriales como medio para facilitar las comunicaciones, la interacción entre diferentes aplicaciones y productos, y la especialización tecnológica.

implementación, sin embargo en la práctica los modelos quedan en la etapa de análisis y diseño y cuando llega la hora de programar, las modificaciones hechas al diseño, no se reflejan en el código, por falta de tiempo, presupuesto o simplemente por considerar la documentación poco importante (PONS, GIANDIN, & PÉREZ, 2010).

La propuesta de MDD, incorpora los modelos como artefactos del desarrollo, los modelos y la escritura de código a framework especializados y generadores de código automáticos, esto garantiza la coherencia de los modelos frente al código, si hay cambios, se actualiza el modelo y a su vez este genera una nueva versión del código.

De una u otra forma cualquier modelo de ciclo de vida comprende las siguientes fases:

- ✓ Análisis
- ✓ Diseño
- ✓ Desarrollo
- ✓ Pruebas
- ✓ Implementación
- ✓ Mantenimiento

Sin embargo, los modelos no pretenden decir cómo ni que actividades se deben llevar, para cumplir con cada una de las fases, los detalles de la implementación del modelo se especifican en las metodologías.

### 2.1.3 Metodología de desarrollo de software

Una metodología es un conjunto integrado de técnicas y métodos que permite tomar de forma homogénea y abierta cada una de las actividades del ciclo de vida<sup>21</sup> de un proyecto de desarrollo. Es un proceso software detallado y completo

Una metodología para el desarrollo de software en un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito. Comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado.

Cada una de las metodologías disponibles es más adecuada para tipos específicos de proyectos, basados en consideraciones técnicas, organizacionales, de proyecto y de equipo, es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo software.

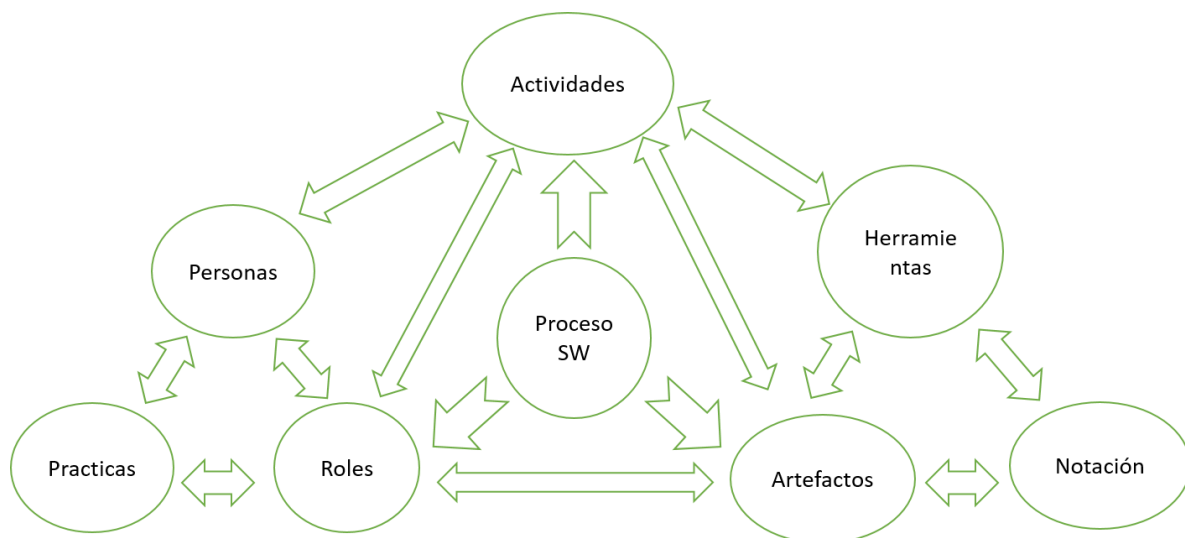


Figura 6 Forma Genérica de las Metodologías de Desarrollo

<sup>21</sup> Las metodologías detallan y hacen funcionales los ciclos de vida. Una metodología puede incorporar aspectos de más de un ciclo de vida

Una gran parte de las metodologías de desarrollo de software no solamente describen detalles del proceso al descomponerlos en actividades, sino que definen roles, artefactos y herramientas que se utilizan en cada una de las actividades además de buenas prácticas.

#### 2.1.4 Tipos de metodologías

Según la filosofía de desarrollo se pueden clasificar las metodologías en dos grupos. Las metodologías **tradicionales**, que se basan en una fuerte planificación durante todo el desarrollo, y las metodologías **ágiles**, en las que el desarrollo de software es incremental, cooperativo, sencillo y adaptado (INTECO, 2009).

##### 2.1.4.1 Metodologías tradicionales

Las metodologías tradicionales Centran su atención en llevar una documentación completa de todo el proyecto y en cumplir con un plan, definido todo esto, en la fase inicial del desarrollo. Incurren en altos costos cuando se necesita implementar un cambio y son poco flexibles<sup>22</sup>, sin embargo son útiles para grandes proyectos donde el control de las actividades es una necesidad.

#### **Proceso Racional Unificado RUP**

RUP<sup>23</sup> es un marco de trabajo de proceso de desarrollo de software iterativo creado por Rational Software Corporation, fue originalmente desarrollado por Rational Software. Resultó de la combinación de varias metodologías y se vio influenciado por métodos previos como el modelo en espiral. Surgió del deseo de elevar el modelado de sistemas a la práctica del desarrollo y de resaltar los principios de calidad que se aplican a las manufacturas en general del software. Las características principales de RUP son: es Guiado/Manejado por casos de uso,

---

<sup>22</sup> Las metodologías tradicionales (formales) se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)

<sup>23</sup> Del ingles Rational Unified Process

Centrado en arquitectura, Desarrollo basado en componentes, Utilización de un único lenguaje de modelado y Proceso Integrado (Martínez & Martínez, 2002).

En la figura se puede observar el proceso que sigue RUP.

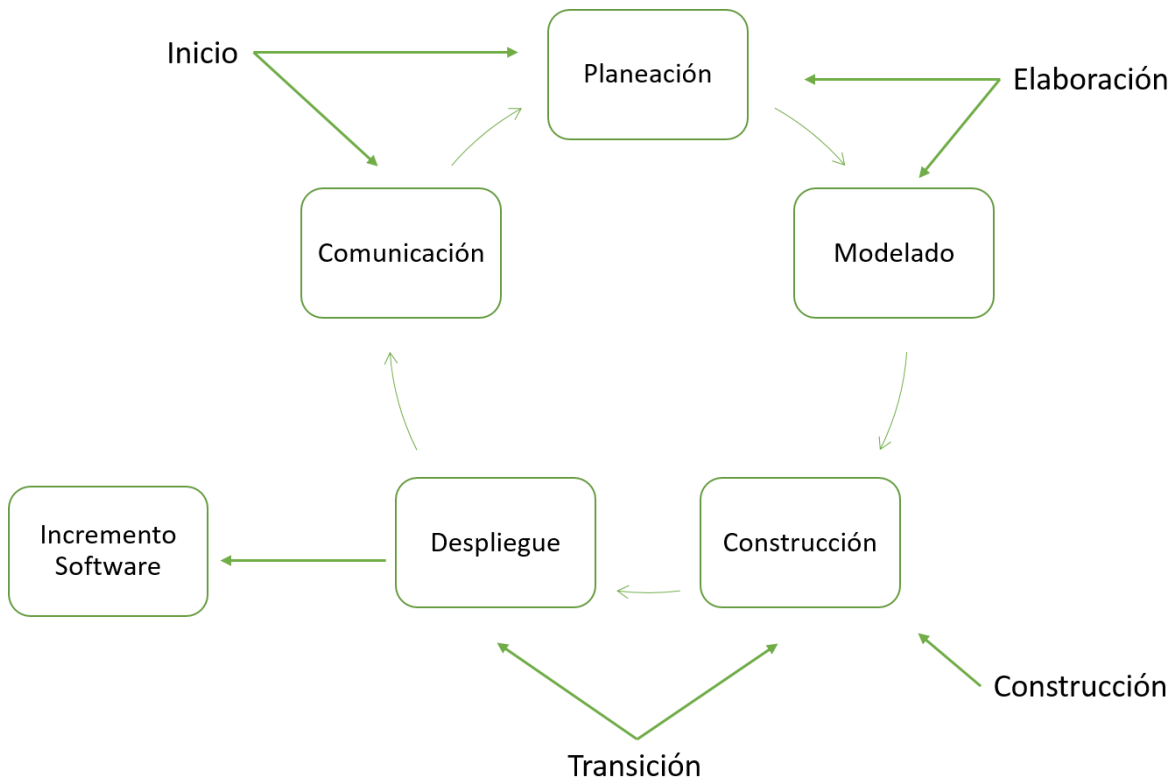


Figura 7 Proceso RUP: Fuente Roger Pressman 6º edición

El proceso fue diseñado con las mismas técnicas con las que el equipo solía diseñar software; tenía un modelo orientado a objetos subyacente, usando UML<sup>24</sup>. RUP se basa en un conjunto de módulos o elementos, que describen qué se va a producir, las habilidades necesarias requeridas y la explicación paso a paso describiendo

<sup>24</sup> Lenguaje de Modelado Unificado en inglés Unified Modeling Language.



cómo se consiguen los objetivos de desarrollo. Los módulos principales, o elementos de contenido, son: Roles, Productos de trabajo y Tareas.

RUP tiene un ciclo de vida de cuatro fases<sup>25</sup> y dentro de cada fase se definen unas iteraciones, dentro de cada iteración, las tareas se categorizan en nueve disciplinas o flujos de trabajo: Seis disciplinas de ingeniería que son el Modelaje de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas, Despliegue y Tres disciplinas de soporte<sup>26</sup>.

### **Desarrollo Rápido de Aplicaciones RAD**

La metodología de desarrollo rápido de aplicaciones (RAD)<sup>27</sup> se creó para responder a la necesidad de entregar sistemas muy rápido. No es apropiado para todos los proyectos (INTECO, 2009).

El alcance, el tamaño y las circunstancias, determinan el éxito de un enfoque RAD. El proceso<sup>28</sup> RAD se realiza utilizando las siguientes etapas: comunicación, planeación, modelado (negocios, datos, procesos), construcción rápida y transición, los cuales abarcan todo el flujo de vida de este tipo de metodología (Arbeláez, Medina, & Chaves, 2011)

---

<sup>25</sup> que son: Fase de iniciación en donde se define el alcance del proyecto, Fase de elaboración en la cual se analizan las necesidades del negocio y se definen sus principios arquitectónicos, Fase de construcción que se crea el diseño de la aplicación y el código fuente y Fase de transición en donde se entrega el sistema a los usuarios

<sup>26</sup> Gestión de la configuración y del cambio, Gestión de proyectos y entorno

<sup>27</sup> Rapid Application Development. El método RAD tiene una lista de tareas y una estructura de desglose de trabajo diseñada para la rapidez. El método comprende el desarrollo iterativo, la construcción de prototipos y el uso de herramientas CASE (Computer Aided Software Engineering).

<sup>28</sup> RAD requiere el uso interactivo de técnicas estructuradas y prototipos para definir los requisitos de usuario y diseñar el sistema final. El desarrollador primero construye modelos de datos y modelos de procesos de negocio preliminares. Los prototipos ayudan al analista y los usuarios a verificar tales requisitos y a refinar formalmente los modelos de datos y procesos.

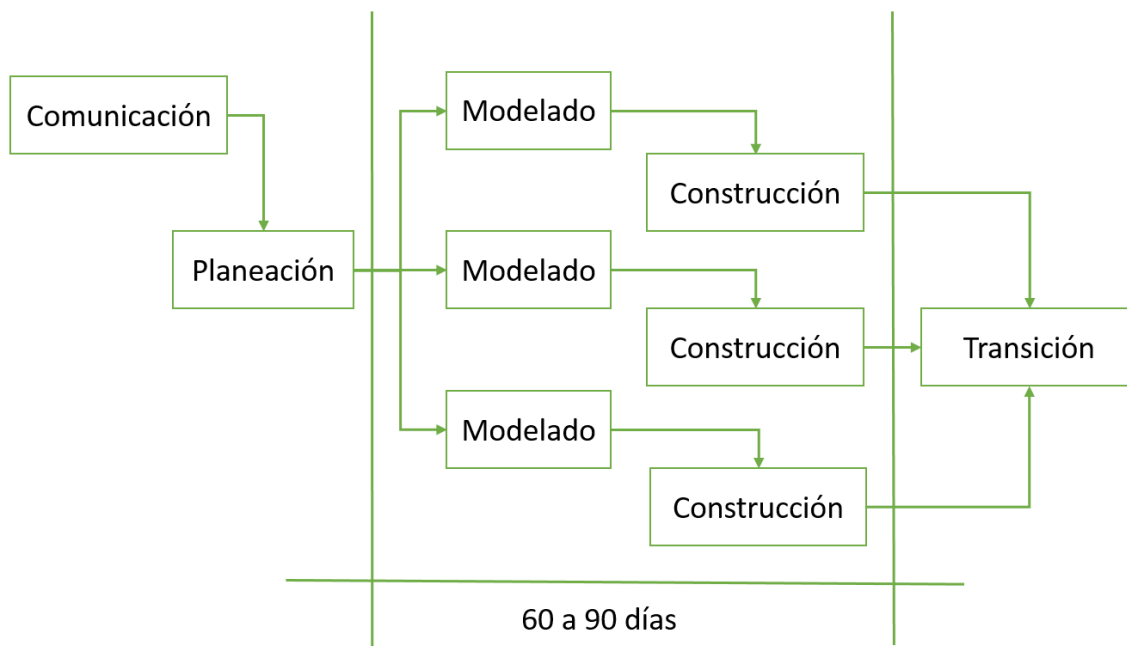


Figura 8 Proceso RAD: Fuente Roger Presuman, Ingeniería del Software 6º edición

RAD, enfatiza el uso de herramientas CASE<sup>29</sup>, para la generación automática de código, se pueden usar diversas herramientas como Visual Studio.Net, NetBeam y otras...

#### 2.1.4.2 Metodologías ágiles

Este enfoque nace como respuesta a los problemas que puedan ocasionar las metodologías tradicionales. El esquema tradicional para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos donde el entorno del sistema es muy

<sup>29</sup> Computer Aided Software Engineering.

cambiante y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad (Letelier & Penadés, 2006).

En este escenario, las metodologías ágiles<sup>30</sup> emergen como una posible respuesta para llenar ese vacío metodológico<sup>31</sup>. Por estar especialmente orientadas para proyectos pequeños, las metodologías ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

### **El manifiesto ágil**

En una reunión celebrada en febrero de 2001 en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software<sup>32</sup>, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto<sup>33</sup> (Letelier & Penadés, 2006).

---

<sup>30</sup> El desarrollo ágil de software es un grupo de metodologías de desarrollo de software que se basan en principios similares.

<sup>31</sup> Ante las dificultades para utilizar metodologías tradicionales con restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir del "buen hacer" de la ingeniería del software, asumiendo el riesgo que ello conlleva (Letelier & Penadés, 2006).

<sup>32</sup> Los firmantes de los valores y principios de este Manifiesto son: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas

<sup>33</sup> Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó "La Alianza Ágil"<sup>34</sup>, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida es fue el Manifiesto Ágil, un documento que resume esta filosofía (Letelier & Penadés, 2006).

El Manifiesto comienza enumerando los principales valores del desarrollo ágil. Se valora:

1. Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
2. Desarrollar software que funciona más que conseguir una buena documentación.
3. La colaboración con el cliente más que la negociación de un contrato
4. Responder a los cambios más que seguir estrictamente un plan.

Los valores anteriores inspiran los doce principios del manifiesto. Estos principios son las características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil. Son:

1. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
2. Dar la bienvenida a los cambios.
3. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
4. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
5. Construir el proyecto en torno a individuos motivados.

---

<sup>34</sup> El termino original en inglés es The Agile Alliance

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
7. El software que funciona es la medida principal de progreso.
8. Los procesos ágiles promueven un desarrollo sostenible.
9. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
12. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Las metodologías ágiles se basan en dos aspectos fundamentales, retrasar las decisiones y la planificación adaptativa<sup>35</sup>. Basan su fundamento en la adaptabilidad de los procesos de desarrollo.

El desarrollo ágil<sup>36</sup> elige hacer las cosas en incrementos pequeños con una planificación mínima, más que planificaciones a largo plazo. Las iteraciones son estructuras de tiempo pequeñas (conocidas como “TimeBoxes”) que típicamente duran de 1 a 4 semanas. De cada iteración se ocupa un equipo realizando un ciclo de desarrollo completo, incluyendo planificación, análisis de requisitos, diseño, codificación, pruebas unitarias y pruebas de aceptación. Esto ayuda a minimizar el

---

<sup>35</sup> Estas metodologías ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan; Es importante tener en cuenta que el uso de un método ágil no vale para cualquier proyecto. Sin embargo, una de las principales ventajas de los métodos ágiles es su peso inicialmente ligero.

<sup>36</sup> Las metodologías ágiles promueven generalmente un proceso de gestión de proyectos que fomenta el trabajo en equipo, la organización y responsabilidad propia, un conjunto de mejores prácticas de ingeniería que permiten la entrega rápida de software de alta calidad, y un enfoque de negocio que alinea el desarrollo con las necesidades del cliente y los objetivos de la compañía.

riesgo general, y permite al proyecto adaptarse a los cambios rápidamente. La documentación se produce a medida que es requerida por los agentes involucrados (Letelier & Penadés, 2006).

La composición del equipo en un proyecto ágil es normalmente multidisciplinar y de organización propia sin considerar cualquier jerarquía corporativa existente o los roles corporativos de los miembros de los equipos. Los esfuerzos de desarrollos largos deben ser repartidos entre múltiples equipos trabajando hacia un objetivo común o partes diferentes de un esfuerzo<sup>37</sup>.

### **Programación Extrema XP**

XP<sup>38</sup> es un enfoque de la ingeniería del software formulado por Kent Beck. Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos (Letelier & Penadés, 2006).

XP es una metodología ágil centrada en potenciar las relaciones interpersonales<sup>39</sup> como clave para el éxito en el desarrollo de software. Se basa en la realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y precisión para enfrentar los cambios (Letelier & Penadés, 2006). XP resalta una serie de valores y

---

<sup>37</sup> Esto puede requerir también una coordinación de prioridades a través de los equipos.

<sup>38</sup> Extreme Programming

<sup>39</sup> XP promueve el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo

principios que deben tenerse en cuenta y practicarlos durante el tiempo de desarrollo que dure un proyecto (Echeverry & Luz, 2007).

Más que una metodología, XP se considera una disciplina, la cual está sostenida por valores y principios propios de las metodologías ágiles (Echeverry & Luz, 2007):

- La comunicación: en la metodología XP es muy importante que exista un ambiente de colaboración y comunicación al interior del equipo de desarrollo, así como en la interacción de éste con el cliente<sup>40</sup>.
- La simplicidad: este valor se aplica en todos los aspectos de la programación extrema. Desde diseños muy sencillos donde lo más relevante es la funcionalidad necesaria<sup>41</sup> que requiere el cliente, hasta la simplificación del código mediante la refactorización del mismo.
- La retroalimentación: se presenta desde el comienzo del proyecto, ayuda a encaminarlo y darle forma<sup>42</sup>.
- El coraje: el equipo de desarrollo debe estar preparado para enfrentarse a los continuos cambios<sup>43</sup> que se presentarán en el transcurso de la actividad.

A partir de los valores se plantea una serie de prácticas que sirven de guía para los desarrolladores en esta metodología. Una de los aspectos más importantes para

---

<sup>40</sup> En XP la interacción con el cliente es tan estrecha, que es considerado parte del equipo de desarrollo

<sup>41</sup> La programación XP no utiliza sus recursos para la realización de actividades complejas, sólo se desarrolla lo que el cliente demanda, de la forma más sencilla.

<sup>42</sup> Ésta se presenta en los dos sentidos, por parte del equipo de trabajo hacia el cliente, con el fin de brindarle información sobre la evolución del sistema, y desde el cliente hacia el equipo en los aportes a la construcción del proyecto.

<sup>43</sup> Cada integrante debe tener el valor de exponer los problemas o dudas que halle en la realización del proyecto, asumiendo las consecuencias que esto implique. Aún con estas variaciones, las jornadas de trabajo deben proporcionar el máximo rendimiento.

XP son las doce reglas que se plantean, las cuales se caracterizan por su grado de simplicidad y por su enfoque en la practicidad, además de que cada regla se complementa con las demás. A continuación se realizará una breve descripción de cada una de ellas (INTECO, 2009).

- El desarrollo está dirigido por pruebas: antes de realizar una unidad de código, es necesario contar con su respectiva unidad de pruebas<sup>44</sup>. El cliente con ayuda del probador<sup>45</sup> se encarga de diseñar las pruebas de aceptación<sup>46</sup>.
- El juego de la planificación: desde el comienzo del desarrollo se requiere que el grupo y el cliente tengan una visión general y clara del proyecto, es decir, deben entender y estar de acuerdo con lo que el "otro" plantee<sup>47</sup>.
- Cliente in-situ: el cliente, o un representante del mismo, deben estar en el sitio de desarrollo para solucionar las preguntas o dudas que se puedan presentar a medida que se realice el proyecto.
- Programación en parejas: XP propone que exista una pareja de programadores por monitor y teclado, como medida para aumentar la calidad del código<sup>48</sup>.

---

<sup>44</sup> El programador realiza pruebas dirigidas al funcionamiento de nuevas adiciones o módulos al sistema.

<sup>45</sup> El Tester o quien realiza las pruebas, colabora en la realización de las pruebas de aceptación y es quien muestra los resultados de las mismas. En este proceso, ayuda al cliente a diseñar las pruebas y a verificar que las pruebas sean aprobadas.

<sup>46</sup> El propósito es verificar que las historias de usuario se hayan implementado correctamente.

<sup>47</sup> En el transcurso del proyecto se realizan diferentes reuniones, con el fin de organizar las tareas e ideas que surgen tanto por parte del cliente como por el equipo.

<sup>48</sup> Esta práctica busca reducir los errores de codificación, mientras uno de los programadores busca una forma de dar funcionalidad a un módulo, el otro programador aprueba dicho código y busca la forma de simplificarlo.



- Convenciones de código. La aplicación de estándares<sup>49</sup> de programación al código fuente de la aplicación.
- Entregas pequeñas: en la programación extrema se realizan entregas constantes de módulos funcionales completos, de tal forma que en todo momento el cliente tiene una parte de aplicación funcionando<sup>50</sup>.
- Refactorización sin piedad: el código se revisa de forma permanente para depurarlo y simplificarlo, buscando la forma de mejorarlo<sup>51</sup>.
- Integración continua del código: el código de los módulos debe ser integrado a cortos plazos de tiempo<sup>52</sup>, preferiblemente no mayores a un día.
- Diseño simple: sólo se realiza lo necesario para que la aplicación cumpla con la funcionalidad requerida<sup>53</sup> por el cliente.
- Utilización de metáforas del sistema: para el mejor entendimiento de los elementos del sistema por parte del equipo de desarrollo se acude a la utilización de metáforas, como una forma de universalizar el lenguaje del sistema.
- Propiedad colectiva del código: el código es conocido por todo el grupo de trabajo, esto facilita implementar cambios al programa por parte de otros integrantes del equipo.

---

<sup>49</sup> La aplicación de estándares permite que todas las personas que conforman el grupo de trabajo puedan entender y realizar modificaciones al código del sistema.

<sup>50</sup> En XP no existe el desarrollo incompleto de una tarea, ésta se ejecuta en su totalidad o no se hace.

<sup>51</sup> La refactorización se realiza durante todo el proceso de desarrollo.

<sup>52</sup> . Esto facilita la búsqueda y la corrección de errores de codificación e integración que se presenten en el proceso.

<sup>53</sup> No es conveniente realizar diseños complejos que posiblemente no aporten soluciones claras al proyecto, y que a la hora de cambiar los requerimientos se conviertan en una gran barrera de tiempo.

- No trabajar horas extras. Es preferible volver a estimar los tiempos de entrega. Con esta práctica se busca utilizar al máximo el rendimiento y energía del programador.

El ciclo de desarrollo consiste en los siguientes pasos:

1. El cliente define el valor de negocio<sup>54</sup> a implementar
2. El programador estima el esfuerzo necesario para su implementación
3. El programador construye ese valor
4. Se realizan pruebas de aceptación
5. Vuelve al paso 1

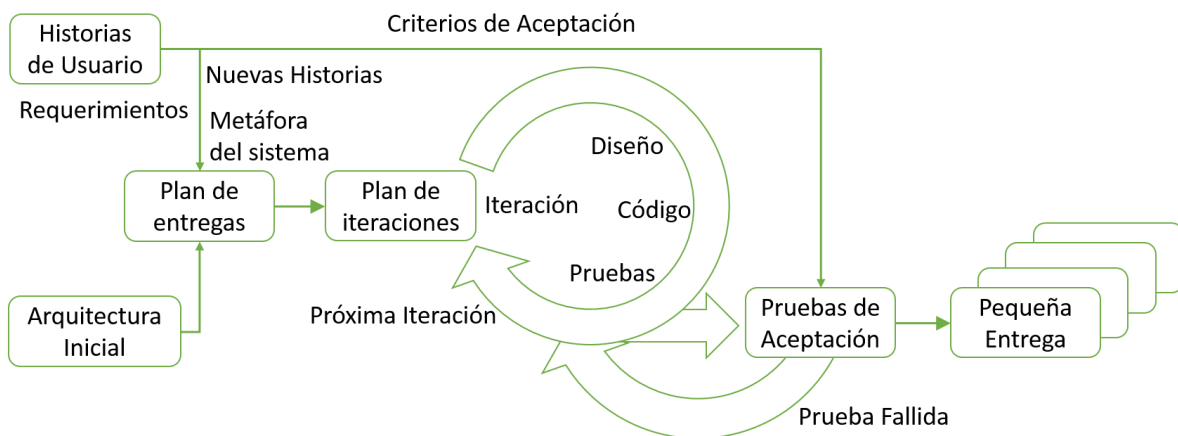


Figura 9 Proceso XP Fuente J. Donovan Wells

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del

<sup>54</sup> En XP el cliente o su representante, se reúne con el equipo de desarrollo y decide que funcionalidad debe programarse primero de acuerdo al valor o importancia que esta tenga en la organización.

producto, para asegurarse de que el sistema tenga el mayor valor de negocio posible.

El ciclo de vida ideal de XP consiste en 6 fases: exploración, planificación de la entrega, iteraciones, producción, mantenimiento y finalización del proyecto (INTECO, 2009).

- Exploración: Los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto.
- Planificación de la Entrega: El cliente establece la prioridad de cada historia de usuario, y los programadores realizan una estimación<sup>55</sup> del esfuerzo necesario de cada una de ellas.
- Iteraciones: Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas<sup>56</sup>. Al final de la última iteración el sistema estará listo para entrar en producción.
- Producción: La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente<sup>57</sup>.
- Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones<sup>58</sup>.

---

<sup>55</sup> Release Plan: Plan de lanzamiento, Al planificar se obtiene el número de iteraciones necesarias para su implementación.

<sup>56</sup> En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto.

<sup>57</sup> Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

<sup>58</sup> Para realizar esto se requiere de tareas de soporte para el cliente.

- Finalización del Proyecto: Es cuando el cliente no tiene más historias para ser incluidas en el sistema<sup>59</sup>.

XP hay 5 grupos de personas que intervienen en un proyecto y están representadas en roles, los roles en XP son:

- Programador: el programador escribe las pruebas unitarias y produce el código del sistema.
- Cliente: escribe las historias de usuario<sup>60</sup> y las pruebas funcionales para validar su implementación.
- Encargado de pruebas<sup>61</sup>: ayuda al cliente a escribir las pruebas funcionales.
- Encargado de seguimiento: proporciona realimentación<sup>62</sup> al equipo.
- Entrenador: es el responsable del proceso global<sup>63</sup>.
- Consultor: es un miembro externo del equipo con un conocimiento específico Gestor: es el vínculo entre clientes y programadores<sup>64</sup>. Su labor esencial es de coordinación.

---

<sup>59</sup> Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura.

<sup>60</sup> Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en apoyar mayor valor al negocio.

<sup>61</sup> Tester: Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para las pruebas.

<sup>62</sup> Tracker: Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones. Realiza el seguimiento del progreso de cada iteración.

<sup>63</sup> Coach: Debe proveer guías al equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.

<sup>64</sup> Big Boss: ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas.

## SCRUM

Es un proceso ágil, más que una metodología, que se puede usar para gestionar y controlar desarrollos complejos de software y productos usando prácticas iterativas e incrementales. Es un proceso incremental iterativo para desarrollar cualquier producto o gestionar cualquier trabajo (INTECO, 2009).

En Scrum un proyecto se ejecuta en bloques temporales cortos y fijos<sup>65</sup>. 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<sup>66</sup> cuando lo solicite (Albaladejo, s.f.).

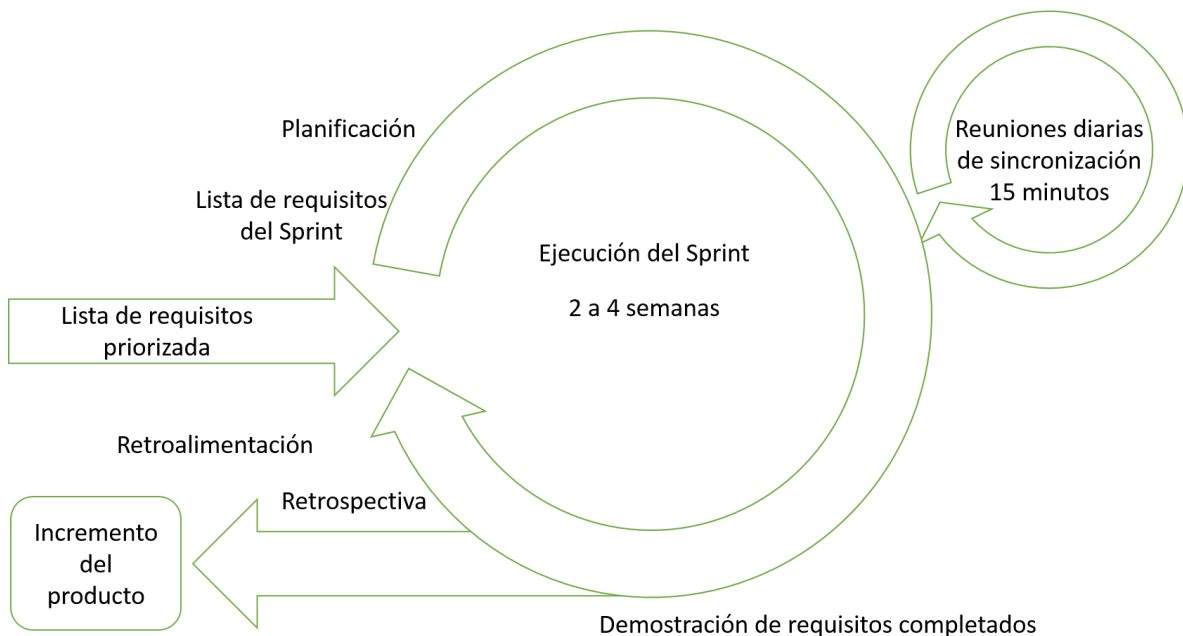


Figura 10 Proceso SCRUM. Fuente: <http://proyectosagiles.org/como-funciona-scrum/>

<sup>65</sup> Iteraciones naturales de un mes y hasta de dos semanas, si así se necesita.

<sup>66</sup> Product Owner: el representante de todas las personas interesadas en los resultados del proyecto (internas o externas a la organización, promotores del proyecto y usuarios finales, idealmente también debería ser un usuario clave o consumidores finales del producto) y actuar como interlocutor único ante el equipo, con autoridad para tomar decisiones.

Los planes de iteraciones y entregas, la estimación del esfuerzo requerido<sup>67</sup> la realiza el equipo<sup>68</sup> en conjunto (Albaladejo, s.f.).

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 y tiene dos partes a) Selección de requisitos<sup>69</sup>. b) Planificación de la iteración<sup>70</sup>
- Ejecución de la iteración: cada día el equipo realiza una reunión de sincronización Cada miembro del equipo inspecciona el trabajo que el resto

---

<sup>67</sup> la estimación se realiza usando el Planning Poker es un proceso iterativo de planificación. Funciona de la siguiente manera: a) El cliente lee un objetivo (historia de usuario escrita en una tarjeta). b) El equipo le hace preguntas para entender su alcance. c) Cada miembro del equipo piensa en el esfuerzo necesario para completar el objetivo y todos muestran sus tarjetas simultáneamente, de manera que no están condicionados por las estimaciones de los otros. d) Las personas que están más alejadas del consenso explican por qué su votación es más alta (hay algún problema en el que nadie más ha pensado o el resto no ha tenido en suficiente consideración) o más baja (conocen una manera sencilla de resolver el problema, resolvieron algo muy parecido en un proyecto anterior, etc.). e) El equipo vuelve a votar, hasta que alcanza un acuerdo. No hay democracia, dado que todos deberán comprometerse a que ese objetivo se va a acabar con el esfuerzo acordado.

<sup>68</sup> Team: grupo de personas que de manera conjunta desarrollan el producto del proyecto. Tienen un objetivo común, comparten la responsabilidad del trabajo que realizan (así como de su calidad) en cada iteración y en el proyecto.

<sup>69</sup> 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, 4 horas máximo.

<sup>70</sup> 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 autoasignan las tareas, 4 horas máximo.

está realizando<sup>71</sup> para poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso adquirido, actualiza el estado de la lista de tareas de la iteración<sup>72</sup> y los gráficos de trabajo pendiente<sup>73</sup>. En la reunión cada miembro del equipo responde a tres preguntas<sup>74</sup>. Durante la iteración el Facilitador<sup>75</sup> se encarga de que el equipo pueda cumplir con su compromiso y de que no se merme su productividad. Durante la iteración, el cliente junto con el equipo refinan la lista de requisitos y, si es necesario, cambian o replanifican los objetivos del proyecto para maximizar la utilidad de lo que se desarrolla y el retorno de inversión.

- Inspección y adaptación: el último día de la iteración se realiza la reunión de revisión de la iteración. Tiene dos partes: a) Demostración<sup>76</sup>. b) Retrospectiva<sup>77</sup>

---

<sup>71</sup> Cada miembro del equipo busca dependencias entre tareas, progreso hacia el objetivo de la iteración, obstáculos que pueden impedir este objetivo. Dura máximo 15 minutos

<sup>72</sup> Sprint Backlog: lista de tareas de la iteración.

<sup>73</sup> Burndown charts: Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los objetivos/requisitos. Permite extrapolar si el Equipo podrá completar el trabajo en el tiempo estimado.

<sup>74</sup> ¿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?

<sup>75</sup> Scrum Master: Lidera; a) Elimina los obstáculos que el equipo no puede resolver por sí mismo. b) Protege al equipo de interrupciones externas que puedan afectar su compromiso o su productividad.

<sup>76</sup> 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, replanificando el proyecto. La demostración tiene una duración máxima de 4 horas.

<sup>77</sup> 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. Tiene una duración máxima de 4 horas.

Sólo en situaciones muy excepcionales el cliente o el equipo pueden solicitar una terminación anormal de la iteración. Esto puede suceder si, por ejemplo, el contexto del proyecto ha cambiado enormemente y no es posible esperar al final de la iteración para aplicar cambios, o si el equipo encuentra que es imposible cumplir con el compromiso adquirido. En ese caso, se dará por finalizada la iteración y se dará inicio a otra mediante una reunión de planificación de la iteración (Albaladejo, s.f.).

Scrum adopta un enfoque empírico, aceptando que el problema no se puede entender o definir completamente, centrándose en cambio en maximizar las habilidades del equipo para entregar rápidamente y responder a los requisitos emergentes (INTECO, 2009).

### **Método de Desarrollo de Sistemas Dinámico DSDM**

El método de desarrollo de sistemas dinámico DSDM<sup>78</sup> es una metodología de desarrollo de software originalmente basada en la metodología RAD, es un enfoque iterativo e incremental que enfatiza la participación continua del usuario.

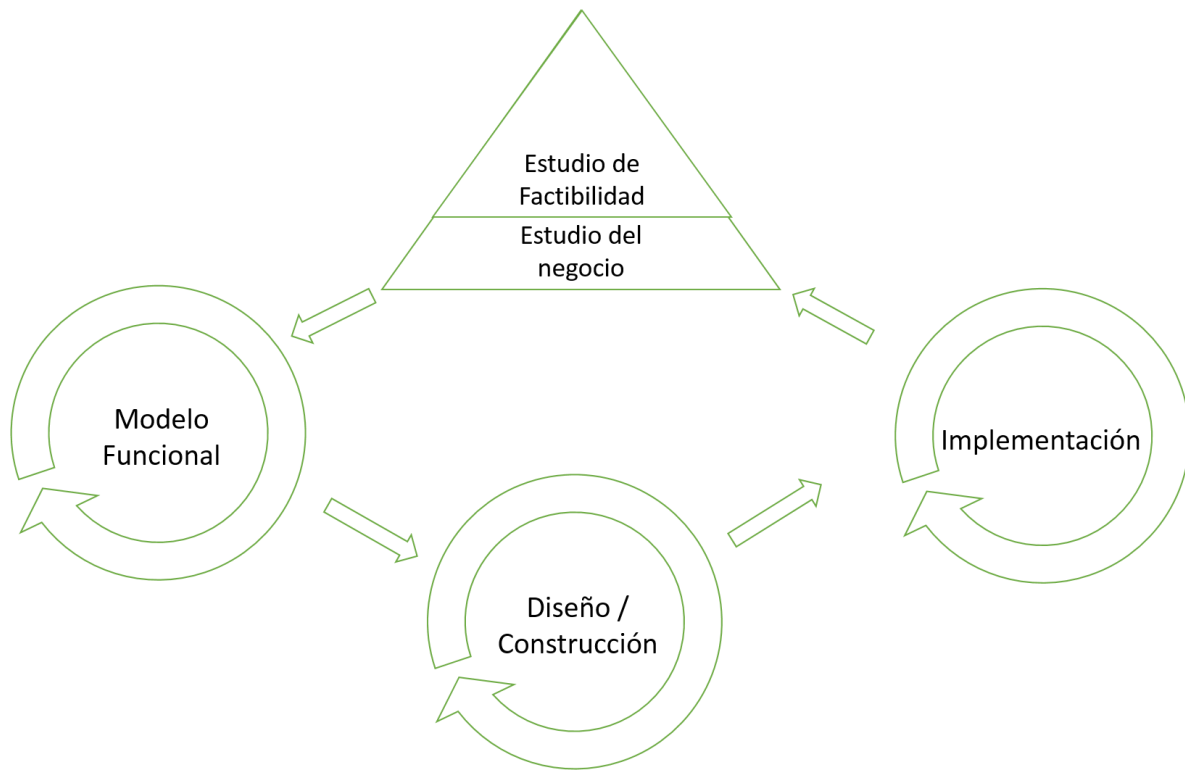
Como extensión del desarrollo rápido de aplicaciones, DSDM se centra en proyectos que se caracterizan por planificaciones y presupuestos estrictos. DSDM consiste en tres fases: fase pre-proyecto, fase de ciclo de vida del proyecto y fase post-proyecto. La fase de ciclo de vida del proyecto está subdividida en 5 etapas: estudio de viabilidad, estudio de negocio, iteración de modelo funcional, iteración de diseño y construcción e implementación.

En DSDM hay nueve procesos subyacentes que consisten en cuatro fundamentos y cinco puntos de partida.

---

<sup>78</sup> En inglés Dynamic Systems Development Method





1. La participación del usuario es la clave principal para llevar a cabo un proyecto eficiente y efectivo, de tal forma que las decisiones se puedan hacer de la forma más exacta.
2. Se le deben otorgar poderes al equipo del proyecto para tomar decisiones, sin tener que esperar a aprobaciones de más alto nivel.
3. Poner atención en la entrega frecuente de productos, entregando el producto con frecuencia desde una etapa temprana del proyecto, el producto se puede probar y revisar y el registro de pruebas y el documento de revisión se pueden tener en cuenta en la siguiente iteración o fase.
4. El principal criterio de aceptación de un entregable es que entregue un sistema que trate las necesidades del negocio actuales.
5. El desarrollo es iterativo e incremental y conducido por la retroalimentación del usuario para converger en una solución de negocio efectiva.
6. Todos los cambios durante el desarrollo son reversibles

7. Se debería hacer una línea base del alcance y los requisitos a alto nivel antes de que el proyecto empiece.
8. Las pruebas se llevan a cabo a lo largo de todo el ciclo de vida del proyecto.
9. La comunicación y la cooperación entre todas las personas involucradas en el negocio son necesarias para ser eficientes y efectivos.

### Proceso Unificado Agil AUP

AUP<sup>79</sup> es un acercamiento aerodinámico a desarrollo del software basado en el Proceso Unificado Rational de IBM RUP, basado en disciplinas y entregables incrementales en el tiempo. Describe un enfoque simple, fácil de entender, del desarrollo de software de Aplicación de negocios usando técnicas y conceptos ágiles. Aplica técnicas ágiles incluyendo desarrollo orientado a pruebas, modelado ágil, gestión de cambios ágil y refactorización de bases de datos para mejorar la productividad. La naturaleza en serie de AUP se presenta en cuatro fases al igual que RUP:

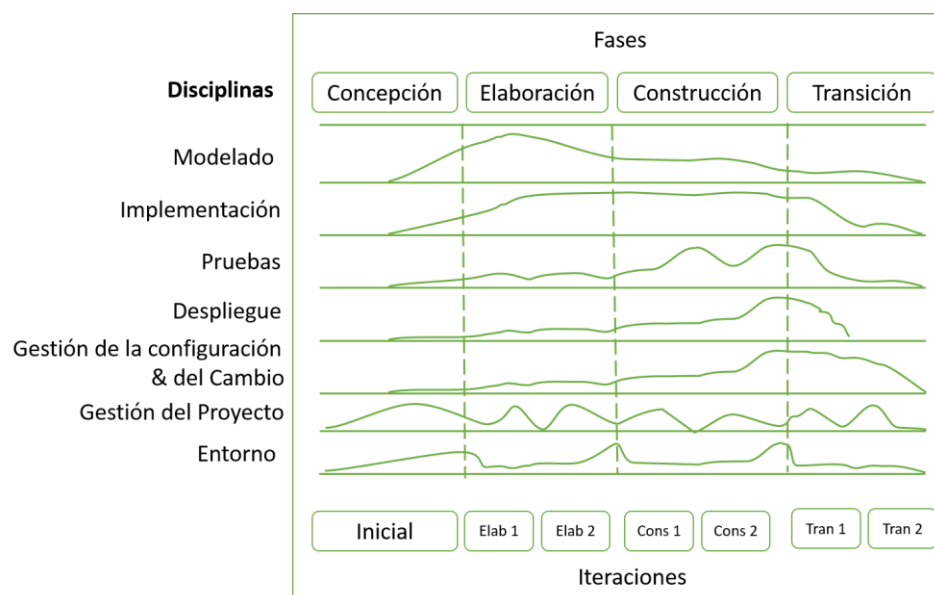


Figura 11 Proceso de AUP Fuente

<sup>79</sup> En inglés Agile Unified Process es una versión simplificada de RUP desarrollada por Scott Ambler.

1. Concepción: el objetivo es identificar el alcance inicial del proyecto, una arquitectura potencial para el sistema y obtener fondos y aceptación por parte de las personas involucradas en el negocio. Definir la razón de ser y el alcance del proyecto<sup>80</sup>, las actividades en esta etapa son especificación de los criterios del proyecto, definir los requisitos, estimar los recursos necesarios y establecer un cronograma inicial de fases, el resultado de esta fase se compila en el documento de definición del proyecto en donde están los acuerdos entre el equipo desarrollador y el cliente o usuario final.
2. Elaboración: el objetivo es probar la arquitectura del sistema. A través del análisis del dominio del problema se establece un plan de proyecto y se define la arquitectura básica correcta del sistema, es en donde se definen los tiempos y necesidades propias además de realizar un análisis de riesgos del desarrollo del proyecto, los artefactos resultantes de esta etapa son Modelo del dominio, Modelo de procesos, Modelo funcional de alto nivel, Arquitectura básica.
3. Construcción: el objetivo es construir software operativo de forma incremental que cumpla con las necesidades de prioridad más altas de las personas involucradas en el negocio. En esta fase se hace la programación de la solución propiamente dicha, va ligada con el ciclo de vida clásico del sistema, se realizan el análisis, diseño, implementación / codificación, pruebas (individuales, de integración), Los entregables propiamente dichos son los códigos fuentes y los resultados de las pruebas individuales de cada parte
4. Transición: el objetivo es validar y desplegar el sistema en el entorno de producción. Se realizan las pruebas en los entornos de producción, es decir en donde va a ser desarrollado, para poder ser implementado en el sitio de

---

<sup>80</sup> Es decir el estudio de la oportunidad presentado para el desarrollo. De igual manera en esta etapa se debe describir claramente Que se va a hacer, para que se va a realizar y como o que estrategias se va a usar para lograrlo.

producción previa aprobación del usuario final, los manuales de usuario y de sistema deben estar listos se entrega a traes del documento de entrega a satisfacción

Además AUP a diferencia de RUP cuenta con siete disciplinas; modelado<sup>81</sup>, implementación, pruebas, despliegue, gestión de la configuración y el cambio, gestión del proyecto y entorno.

### **Marco de Soluciones de Microsoft MSF**

El Marco de Soluciones de Microsoft MSF<sup>82</sup> es una metodología que define un marco de trabajo de referencia para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft para cualquier plataforma (Linux, Citrix, Microsoft, Unix) (Jurado, 2005).

MSF provee un conjunto de principios, modelos, disciplinas, conceptos y lineamientos para la entrega de tecnología de la información utilizando soluciones Microsoft.

Los siguientes principios y conceptos de MSF sirven de guía al equipo de proyecto para entregar una solución de calidad. Cada miembro del equipo deberá comprender y aplicar estos principios en sus interacciones con otros miembros del equipo, con la organización y con las partes interesadas. MSF se basa en nueve principios fundamentales (Microsoft, 2013):

1. Fomentar una comunicación abierta.

---

<sup>81</sup> La disciplina de modelado agrupa las disciplinas de RUP, modelado de negocios, requisitos y análisis y diseño

<sup>82</sup> Microsoft Solutions Framework fue desarrollada por Microsoft Consulting Services

2. Intentar lograr una visión compartida.
3. Empoderar a los miembros del equipo.
4. Establecer responsabilidades claras y compartidas.
5. Ofrecer valor incremental<sup>83</sup>.
6. Mantenerse ágil, esperar cambios y adaptarse a ellos.
7. Invertir en la calidad.
8. Aprender de todas las experiencias<sup>84</sup>.
9. Colaborar con clientes internos y externos.

MSF utiliza un proceso iterativo e incremental, compuesto de 5 grupos de actividades (pistas de ejecución) y una actividad a todo lo largo del proyecto llamada pista de gobernanza (Microsoft, 2002).

El modelo de equipo de MSF segmenta las actividades y responsabilidades típicas de entrega de una solución en siete roles<sup>85</sup>. Estos roles son interdependientes y multidisciplinarios, para conseguir un enfoque equilibrado, cada uno aporta una perspectiva exclusiva sobre lo que se necesita y cuáles deben ser los objetivos asociados con la entrega de una solución. Estos roles se pueden combinar para situaciones de equipos pequeños y ampliar para situaciones de equipos grandes (Microsoft, 2013).

---

<sup>83</sup> Este principio tiene dos facetas: a) Asegurarse de que lo que se entrega tiene un valor óptimo para las partes interesadas. b) Determinar los incrementos óptimos en los que se entregará valor (o la "frecuencia de entrega").

<sup>84</sup> En tres niveles: a) A nivel de proyecto, como por ejemplo, al perfeccionar un proceso válido para todo el proyecto b) A nivel individual, como por ejemplo, al buscar la manera de interactuar mejor con otros miembros del equipo c) A nivel de la organización, como por ejemplo, al ajustar las métricas de calidad que se recopilan para cada proyecto

<sup>85</sup> Roles MSF: Administración de productos, Administración del programa, Arquitectura, Desarrollo, Experiencia del usuario, Prueba, Lanzamiento/Operaciones

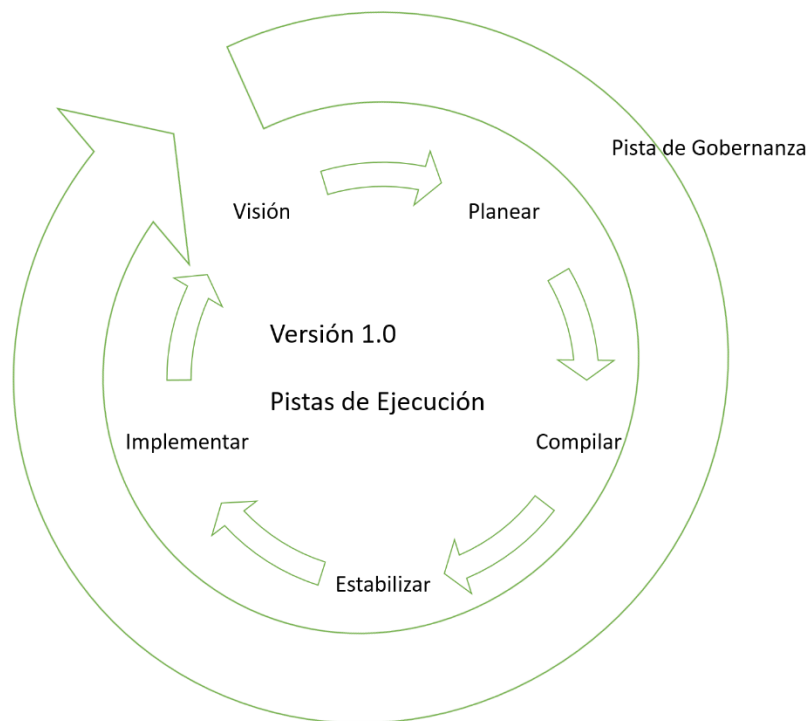


Figura 12 Proceso MSF, Fuente: Microsoft Solutions Framework Essentials

Estos roles no dan a entender ni sugieren ningún tipo de organigrama ni conjunto de cargos, porque varían enormemente según la organización y el equipo. En la mayoría de casos, los roles se distribuirán entre distintos grupos de una organización de TI y a veces con la comunidad de usuarios empresariales, así como con consultores y asociados externos (Microsoft, 2013).

### **Desarrollo Dirigido por Características FDD**

FDD por sus siglas en inglés<sup>86</sup> tiene como rasgo característico la planeación y el diseño por adelantado. En consecuencia, el modelo de objetos, la lista de características y la planeación se hacen al inicio del proyecto. Las iteraciones son incrementos con características identificadas (Navarro, Fernández, & Morales, 2013).

---

<sup>86</sup> Feature Driven Development

Las prácticas que FDD pregona son: el modelado de objetos de dominio<sup>87</sup>, el desarrollo por características<sup>88</sup>, los equipos de características<sup>89</sup>, las inspecciones, la construcción regular de planificación<sup>90</sup>, la gestión de configuración y los reportes y visibilidad de los resultados.

Su ciclo de vida está compuesto por cinco etapas: el desarrollo de un modelo general, la construcción de la lista de características, la planeación por característica, el diseño por característica y la construcción por característica. FDD se enfoca en las fases de diseño, diseño por característica y desarrollo/construcción por característica siendo estas las etapas del proceso que se iteran.

FDD es un enfoque ágil para el desarrollo de software. Dicho enfoque no hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción. Sin embargo, fue diseñado para trabajar con otras actividades de desarrollo de software y no requiere la utilización de ningún modelo de proceso específico. Además, hace énfasis en aspectos de calidad durante todo el proceso e incluye un monitoreo permanente del avance del proyecto. Al contrario de otras metodologías ágiles, FDD afirma ser conveniente para el desarrollo de sistemas críticos (Calabria, 2003).

Esta metodología propone tener etapas de cierre cada dos semanas, lo cual implica que los desarrolladores tendrán nuevas actividades que realizar en dicho período de tiempo. Esto hace que la motivación del equipo se mantenga durante todo el proyecto debido a que se ven los resultados periódicamente.

---

<sup>87</sup> domain object modeling, modelado de objetos de dominio

<sup>88</sup> Class (code) ownership, construcción por características

<sup>89</sup> Feature Teams, equipo de características

<sup>90</sup> Regular Build Schedule, construcción regular de la planificación

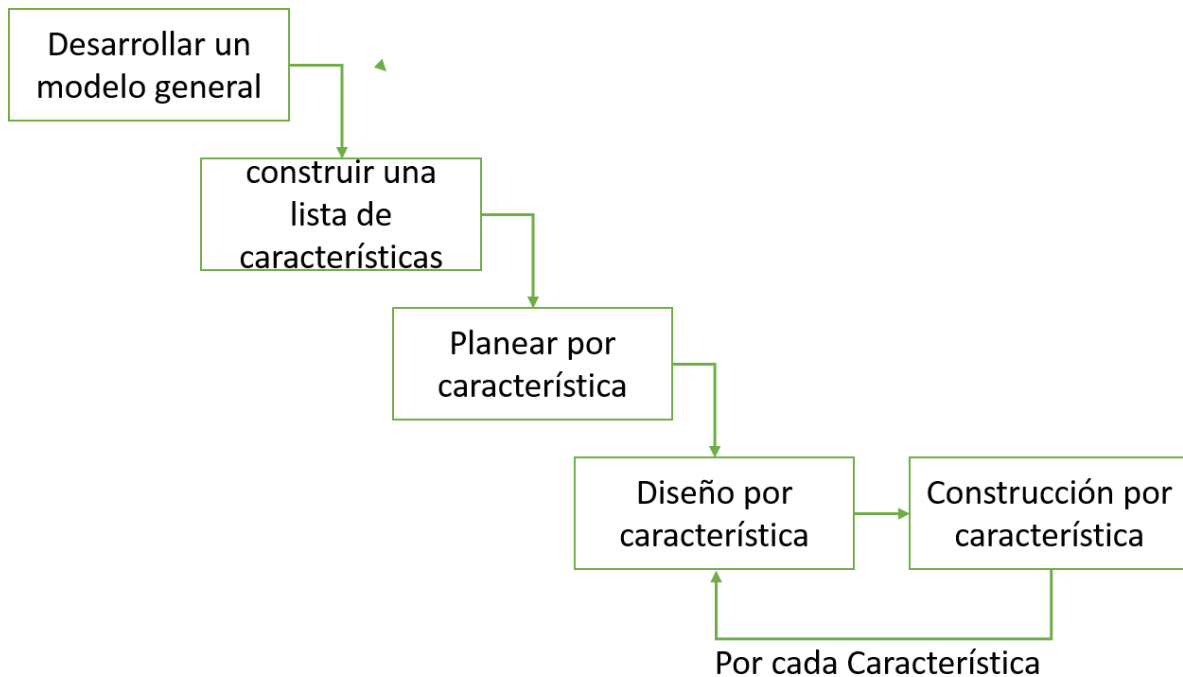


Figura 13 Proceso FDD: Fuente: *A practical guide to feature-driven development*

#### 2.1.4.3 Metodologías web

Son metodologías orientadas al desarrollo y mantenimiento de aplicaciones y portales web, que los usuarios usaran a través de un navegador para acceder a recursos en internet o dentro de una intranet

#### **Método de diseño de Hipermedia HDM**

El método de diseño de Hipermedia HDM<sup>91</sup>. Es basado en la diagrama E-R, pero extiende el concepto de entidad e introduce nuevas primitivas como unidades<sup>92</sup> y enlaces. Las entidades HDM tienen una estructura interior y asocian una

<sup>91</sup> En ingles Hypermedia Design Method, es uno de los primeros métodos que han sido desarrollados para definir la estructura y la interacción en aplicaciones de hipermedia

<sup>92</sup> Correspondiente a "nodos"



navegación semántica a ellos<sup>93</sup>. Una entidad es una jerarquía de componentes y los componentes están hechos de unidades.

Se definen tres tipos de enlaces: estructural, la perspectiva y vínculos de aplicación. Los vínculos estructurales conectan componentes; las estructuras de perspectiva unen unidades<sup>94</sup>. Los enlaces de aplicación los define el autor de la conexión de las entidades de componentes del mismo o diferente tipo (Koch, 1999).

En HDM hay dos grupos diferentes de entidades de salida: las entidades de aplicación descritas anteriormente, y los llamados "esquemas"<sup>95</sup> que permiten el acceso a las entidades de aplicación que ofrece puntos de entrada para empezar la navegación y la posibilidad de localizar y seleccionar entidades.

En el análisis de aplicaciones hipermedia se pueden distinguir las siguientes dimensiones: contenido, estructura, presentación, dinámica y la interacción. El contenido trata las piezas de información mientras que la estructura es la organización del contenido. La presentación define cómo el contenido de la aplicación y las funciones se muestran a los usuarios. La interacción de HDM es la funcionalidad dinámica presente en los elementos de presentación. En otros métodos la interacción se considera como parte de la dinámica y la presentación ya que es una combinación de ambos factores. Los contornos definidos por HDM son enlaces indexados, visitas guiadas y colecciones de enlaces. Enlaces indexados conectan un nodo de la colección a cada miembro de la colección. Un enlace de la visita guiada conecta los nodos de la colección en una secuencia lineal con cada

---

<sup>93</sup> es decir, una especificación de cómo se puede realizar la navegación y cómo la información se visualiza

<sup>94</sup> Estos enlaces se pueden derivar automáticamente de la estructura de las entidades.

<sup>95</sup> Estos esquemas o estructuras de acceso están clasificadas en árboles de componentes. Las entidades de aplicación constituyen la hiperbase.

miembro conectado a la siguiente y a la anterior. En colecciones circulares el último miembro se conecta a la primera. Las Colecciones de enlaces son índices o enlaces de visita guiada que permite recorrer los nodos de una colección (Koch, 1999).

Para apoyar el diseño de presentación HDM define dos conceptos: ranura y marco. Una ranura es un dato atómico<sup>96</sup>. Los marcos están compuestos de ranuras. Un marco es una unidad de presentación, es decir, lo que se muestra al usuario.

Esta metodología distingue entre autoría en grande<sup>97</sup> y a la autoría en pequeño<sup>98</sup>. El primero de ellos identifica las entidades, componentes y unidades mientras que el segundo llena estas unidades con contenido. HDM especifica la estructura de la hiperbase, como de autoría en el pequeño no está dentro de su ámbito de aplicación (Koch, 1999).

### **Metodología de Relación entre Objetos EORM**

La Metodología de Relación entre Objetos EORM<sup>99</sup>, es definida por un proceso iterativo que se concentra en el modelado orientado a objetos por la representación de relaciones entre los objetos (acoplamientos) como objetos, es por ello que fue una de las primeras propuestas para Web centrada en el paradigma de la orientación a objetos<sup>100</sup>.

---

<sup>96</sup> Pueden ser simples o complejas, como un video sincronizado con el sonido

<sup>97</sup> Se refiere a la especificación y diseño de los aspectos globales y estructurales de la aplicación

<sup>98</sup> Se refiere al desarrollo del contenido de los nodos.

<sup>99</sup> En ingles Enhanced Object Relationship Methodology

<sup>100</sup> Se basa en muchas de las ideas que se definen en HDM, pero las traslada a la orientación a objetos. La adopción del enfoque orientado a objetos garantiza todas las ventajas reconocidas para esta técnica de modelado, como la flexibilidad (posible existencia de múltiples formas de relaciones entre nodos) y la reutilización, por la existencia de una librería de clases de enlaces que pueden ser reutilizados en diferentes proyectos de desarrollo hipermedia.

### **Método de Diseño de Hipermedia Orientado a Objetos OOHDM**

EL Método de Diseño de Hipermedia Orientado a Objetos OOHDM<sup>101</sup> abarca cuatro actividades: El modelado conceptual, diseño navegacional, diseño abstracto de interfaz e implementación. Estas actividades se realizan en una mezcla de estilo incremental, iterativo y basado en prototipos de desarrollo (Del Valle Rodríguez, 2009).

Modelado conceptual: se construye un esquema conceptual<sup>102</sup> representado por los objetos de dominio o clases y las relaciones entre dichos objetos.

Diseño navegacional: el diseñador define clases navegacionales tales como nodos, enlaces y estructuras de acceso (índices y visitas guiadas) inducidas del esquema conceptual. Los enlaces derivan de las relaciones y los nodos representan ventanas lógicas sobre las clases conceptuales. A continuación, el diseñador describe la estructura navegacional en términos de “contextos navegacionales. Los nodos se enriquecen con un conjunto de clases especiales que permiten presentar atributos así como métodos o comportamientos cuando se navega en un contexto particular. Durante esta etapa, es posible adaptar los objetos navegacionales para cada contexto, de forma similar a las perspectivas de HDM (Lamarca, 2013).

OOHDM no propone un modelo enriquecido para el dominio de la aplicación, por lo que deja libre al diseñador para elegir el modelo de especificación del dominio. Sin embargo, el modelo hipermedia está definido en dos niveles de abstracción: las clases navegacionales<sup>103</sup> y los contextos navegacionales.

---

<sup>101</sup> Object-Oriented Hypermedia Design Method es una extensión de HDM con orientación a objetos

<sup>102</sup> Se puede usar un modelo de datos semántico estructural (como el modelo de entidades y relaciones). El modelo OOHDM propone como esquema conceptual basado en clases, relaciones y subsistemas.

<sup>103</sup> En el momento de la especificación de las clases navegacionales es cuando el diseñador define las correspondencias clases.

Los nodos inducidos de las clases del modelo del dominio y los enlaces inducidos de las relaciones del modelo del dominio se pueden precisar. Como el segundo nivel está consagrado a la especificación de la navegación, expresada exclusivamente sobre los objetos navegacionales (no sobre los elementos del modelo del dominio), constituye un mecanismo que permite enriquecer el modelo hipermedia (Lamarca, 2013).

Diseño abstracto de interfaz: está dedicada a la especificación de la interfaz abstracta. Así, se define la forma en la cual deben aparecer los contextos navegacionales. También se incluye aquí el modo en que dichos objetos de interfaz activarán la navegación y el resto de funcionalidades de la aplicación, esto es, se describirán los objetos de interfaz y se los asociará con objetos de navegación. La separación entre el diseño navegacional y el diseño de interfaz abstracta permitirá construir diferentes interfaces para el mismo modelo navegacional.

Implementación: esta fase está dedicada a la puesta en práctica, es donde se hacen corresponder los objetos de interfaz con los objetos de implementación (Lamarca, 2013).

### **Método de Diseño de Hipermedia Orientado a Objetos Basado en Escenarios SOHDM**

SOHDM<sup>104</sup> Es un Método que Desarrolla Diseño en escenarios, Orientado a Objetos en Hipermedia. Presenta la necesidad de disponer de un proceso que permita capturar las necesidades del sistema. Para ello, propone el uso de escenarios como medio de extender la capacidad expresiva del modelado en situaciones particulares (Lee, lee, & Yoo, 1998).

---

<sup>104</sup> Scenario-Based Object-Oriented Hypermedia Design Methodology

Es una de las primeras propuestas para la web y brinda más importancia a la tarea de tratamiento de requisitos. Se caracteriza principalmente porque su ciclo de vida comienza con la aplicación de los escenarios como técnica de elicitación y definición de requisitos.

El proceso de definición de requisitos parte de la realización de un diagrama de contexto. En este diagrama de contexto se identifican las entidades externas que se comunican con el sistema, así como los eventos que provocan esa comunicación.

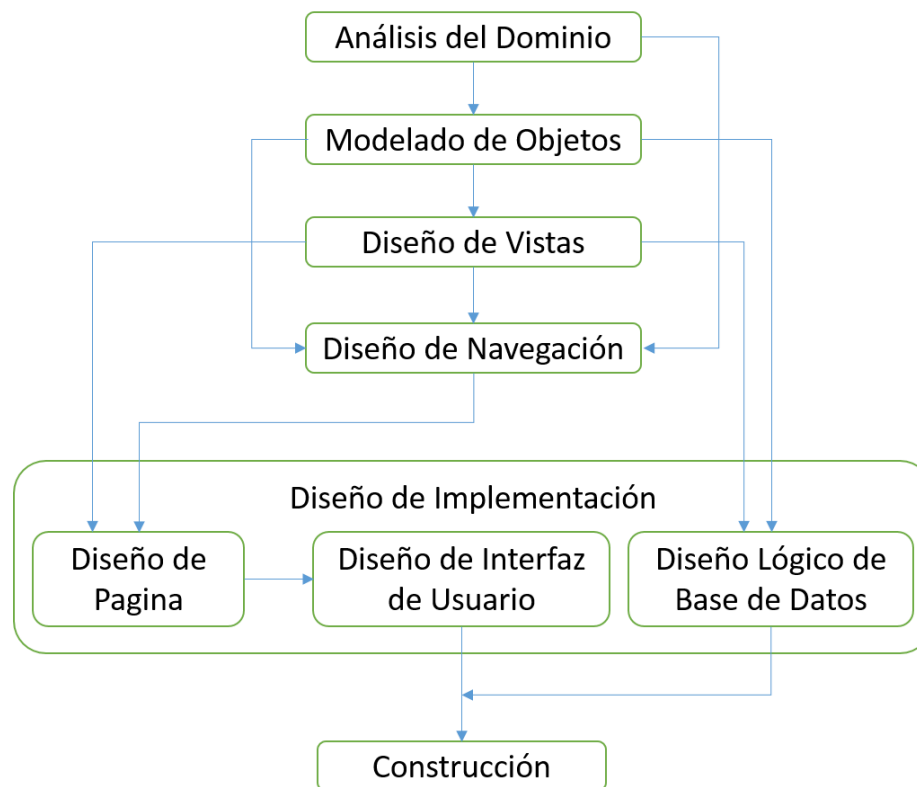


Figura 14 Framework SOHDM. Fuente: A scenario-based object-oriented hypermedia design methodology

SOHDM, tiene seis fases: análisis del dominio, modelado de objetos, diseño de vistas, diseño de navegación, diseño de implementación<sup>105</sup> y construcción.

<sup>105</sup> Que a su vez se descompone en: diseño de página, diseño de interface de usuario y diseño lógico de bases de datos

## Método de Diseño para Sitios Web WSDM

WSDM es un enfoque centrado en el usuario, donde el punto de partida en el análisis es el conjunto de los posibles usuarios de aplicaciones Web. Se compone de cuatro fases: 1) Modelado de usuario, 2) Diseño conceptual, 3) Diseño de la Implementación e 4) Implementación (De Troyer & Leune, 1998).

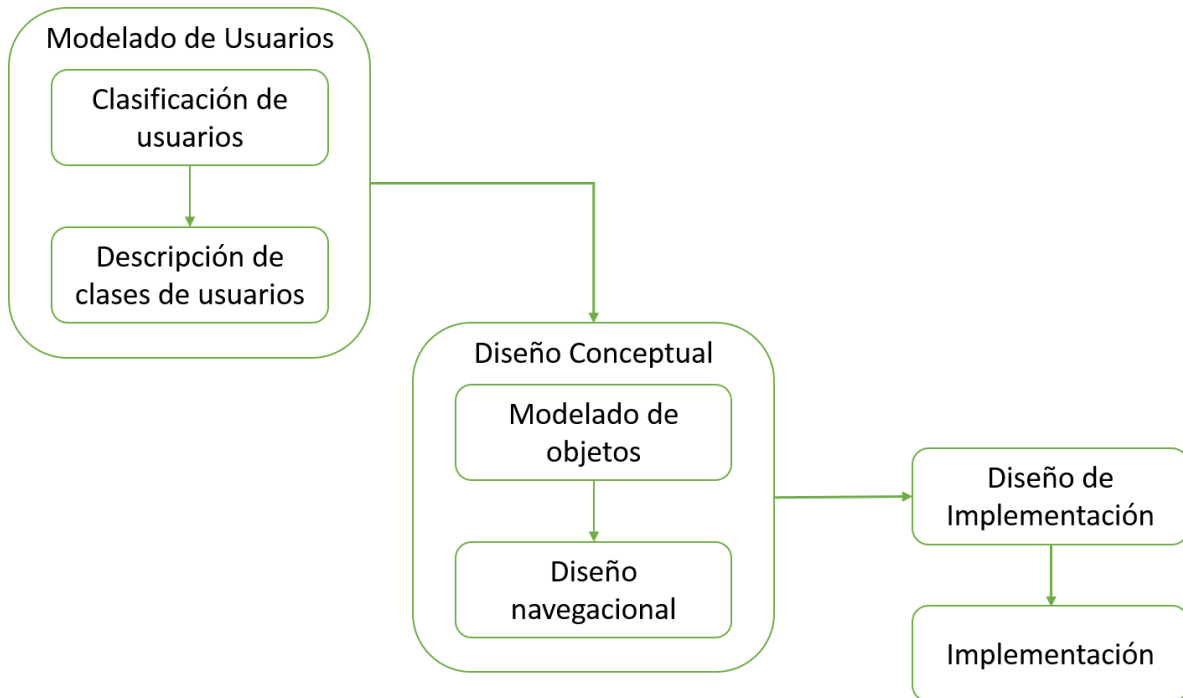


Figura 15 Proceso WSDM. Fuente: WSDM: A User Centered Design Method for Web Sites

Modelado de usuario: en esta fase se clasifica a los usuarios en función de la audiencia<sup>106</sup> Web mirando a la organización o el proceso por el cual se construirá el sitio Web. A continuación, se analizan las clases de usuario, se centra en los

---

<sup>106</sup> Las clases de usuario se definen basándose en subconjuntos de todos los usuarios potenciales que son similares en cuanto a sus necesidades de información

requisitos de información y las características del público objetivo<sup>107</sup> (De Troyer & Leune, 1998).

Diseño conceptual: esta fase consiste en dos sub-fases, el modelado de objetos y el diseño de navegación. Durante modelado de objetos con los requisitos de información de las diferentes clases de usuarios y sus perspectivas se describe formalmente. En la fase de navegación Diseño se describe cómo los diferentes usuarios pueden navegar a través del sitio web. Cada perspectiva tendrá su propia pista de navegación (De Troyer & Leune, 1998).

Diseño de Implementación: En este paso, esencialmente diseñar como quedara el sitio web. El objetivo es crear una apariencia consistente, agradable y eficiente y coherente con el diseño conceptual realizado en la fase anterior. El resultado del diseño de implementación es un modelo de implantación (De Troyer & Leune, 1998).

Implementación: La última fase es la realización real del sitio web utilizando el entorno de aplicación elegido. Por ejemplo, para una aplicación HTML esto significa que el modelo de implementación se debe convertir en un conjunto de archivos que contienen código fuente HTML (De Troyer & Leune, 1998).

### **Método de Análisis de Navegación Relacional RNA**

RNA<sup>108</sup> es un método que define una secuencia de pasos que se utilizarán para el desarrollo de sitios Web<sup>109</sup>. Es especialmente útil para uso sitios Web creados con base en sistema de herencia.

---

<sup>107</sup> Si los usuarios tienen características diferentes dentro de una clase de usuario, este se divide en perspectivas que representan diferentes requisitos de usabilidad.

<sup>108</sup> Del inglés Relationship Navigational Analysis

<sup>109</sup> En este método encontramos cinco fases las cuales son: Análisis del entorno, donde el propósito de esta fase es el de estudiar las características de la audiencia, luego se encuentran las definiciones

La propuesta de RNA es quizás una de las que más ha resaltado la necesidad de trabajar con la especificación de requisitos, incluyendo tareas como el análisis del entorno y de los elementos de interés. Además, resulta interesante que plantee la necesidad de analizar los requisitos conceptuales de manera independiente a los navegacionales (Cárdenas, 2011).

### **Metodología Ingeniería Web IWeb**

La inmediatez, evolución y crecimiento continuos, son características de las aplicaciones Web, IWeb<sup>110</sup> es un proceso incremental y evolutivo, que permite que el usuario se involucre activamente, facilitando el desarrollo de productos que se ajustan a sus requerimientos. IWeb incorpora siete actividades que forman parte del proceso de la IWeb y que son aplicables a cualquier WebApp independientemente de su tamaño y complejidad<sup>111</sup> (Del Valle Rodríguez, 2009).

### **Metodología Ingeniería Web UML UWE**

UWE<sup>112</sup> es una metodología que permite especificar el proceso de creación una aplicación Web, UWE mantiene una notación estándar basada en el uso de UML<sup>113</sup> para sus modelos y sus métodos, lo que facilita la transición. La metodología define claramente la construcción de cada uno de los elementos del modelo.

---

de elementos de interés, el análisis del conocimiento y navegación y finalmente la implementación de los análisis realizados.

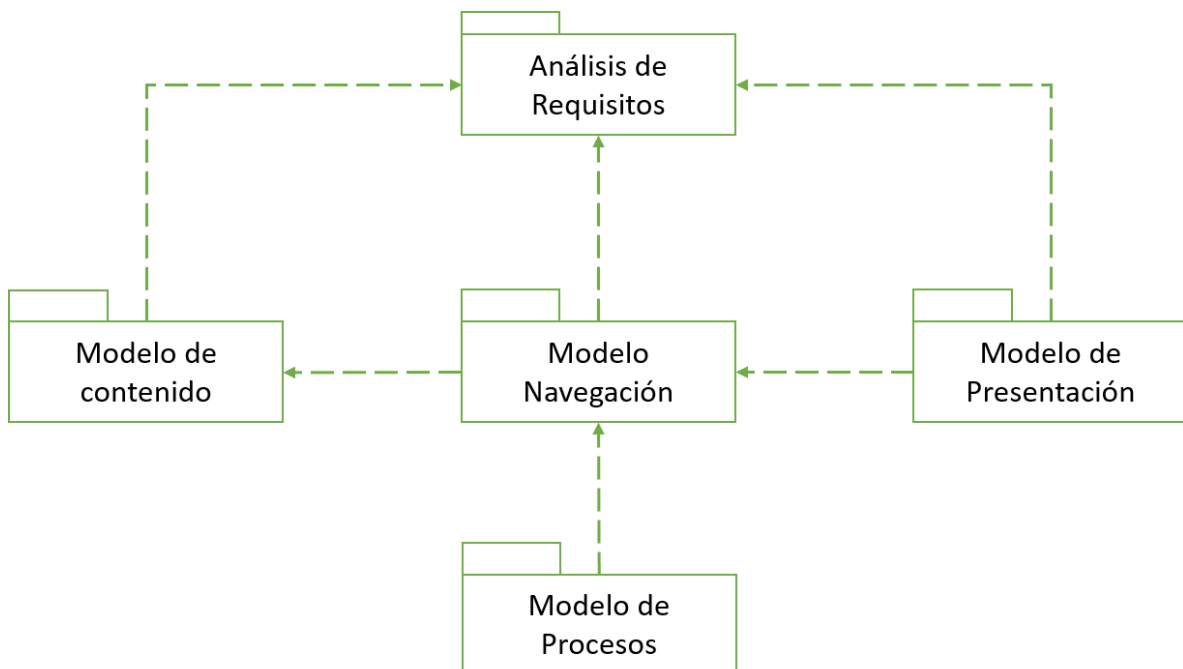
<sup>110</sup> Engineer Web

<sup>111</sup>Las fases de la IWeb son: 1) La *Formulación* identifica objetivos y establece el alcance de la primera entrega. 2) La *Planificación* genera la estimación del coste general del proyecto, la evaluación de riesgos y el calendario del desarrollo y fechas de entrega. 3) El *Análisis* especifica los requerimientos e identifica el contenido. 4) La *Modelización* se compone de dos secuencias paralelas de tareas. Una consiste en el diseño y producción del contenido que forma parte de la aplicación. La otra, en el diseño de la arquitectura, navegación e interfaz de usuario.

<sup>112</sup> UWE del ingles UML Web Engineering.

<sup>113</sup> UML del ingles Unified Modeling Language





*Figura 16 Modelos de la UWE. Fuente: UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio*

En su implementación se deben contemplar las siguientes etapas y modelos (Nieves, Ucán, & Pech, 2014):

- **Análisis de requisitos:** Plasma los requisitos funcionales de la aplicación Web mediante un modelo de casos de uso.
- **Modelo de contenido:** Define, mediante un diagrama de clases, los conceptos a detalle involucrados en la aplicación.
- **Modelo de navegación:** Representa la navegación de los objetos dentro de la aplicación y un conjunto de estructuras como son índices, menús y consultas.
- **Modelo de presentación:** Representa las interfaces de usuario por medio de vistas abstractas.

- Modelo de proceso: Representa el aspecto que tienen las actividades que se conectan con cada clase de proceso.

Como se hace notar, UWE provee diferentes modelos que permite describir una aplicación Web desde varios puntos de vista abstractos. Cada uno de estos modelos se representa como paquetes UML, estos paquetes son procesos relacionados que pueden ser refinados en iteraciones sucesivas durante el desarrollo del UWE.

#### 2.1.5 Estándares para definición de procesos

Para la Organización internacional para la estandarización ISO<sup>114</sup> y para el IEEE<sup>115</sup>. Un “modelo de proceso” se puede definir como el conjunto de tareas a realizar para conseguir el desarrollo de un determinado elemento, así como los elementos que se producen en cada una de las tareas (salidas) y los elementos que son necesarios para realizar una determinada tarea (entradas) (Pressman, 2005)

##### 2.1.5.1 Norma ISO 12207 de procesos estándar del ciclo de vida

La norma ISO<sup>116</sup> 12207 fue creada con el propósito de establecer un marco común para el ciclo de vida del software para:

- ✓ Adquirir, suministrar, desarrollar, operar y mantener software.
- ✓ Gestionar, controlar y mejorar el marco de trabajo del software.
- ✓ Como base para el comercio internacional de software

El estándar se basa en dos principios fundamentales: Modularidad y responsabilidad: Con la modularidad se pretende conseguir procesos con un

---

<sup>114</sup> ISO del inglés International Organization for Standardization

<sup>115</sup> IEEE del inglés Institute of Electrical and Electronics Engineers, Inc.

<sup>116</sup> se toman apartes de la traducción de la norma ISO/IEC 12207 Publicada por la IEEE e ISO en el 2008 debido a que se trata de una normatividad internacional.

mínimo acoplamiento y una máxima cohesión. En cuanto a la responsabilidad, se busca establecer un responsable para cada proceso, facilitando la aplicación del estándar en proyectos en los que pueden existir distintas personas u organizaciones involucradas.

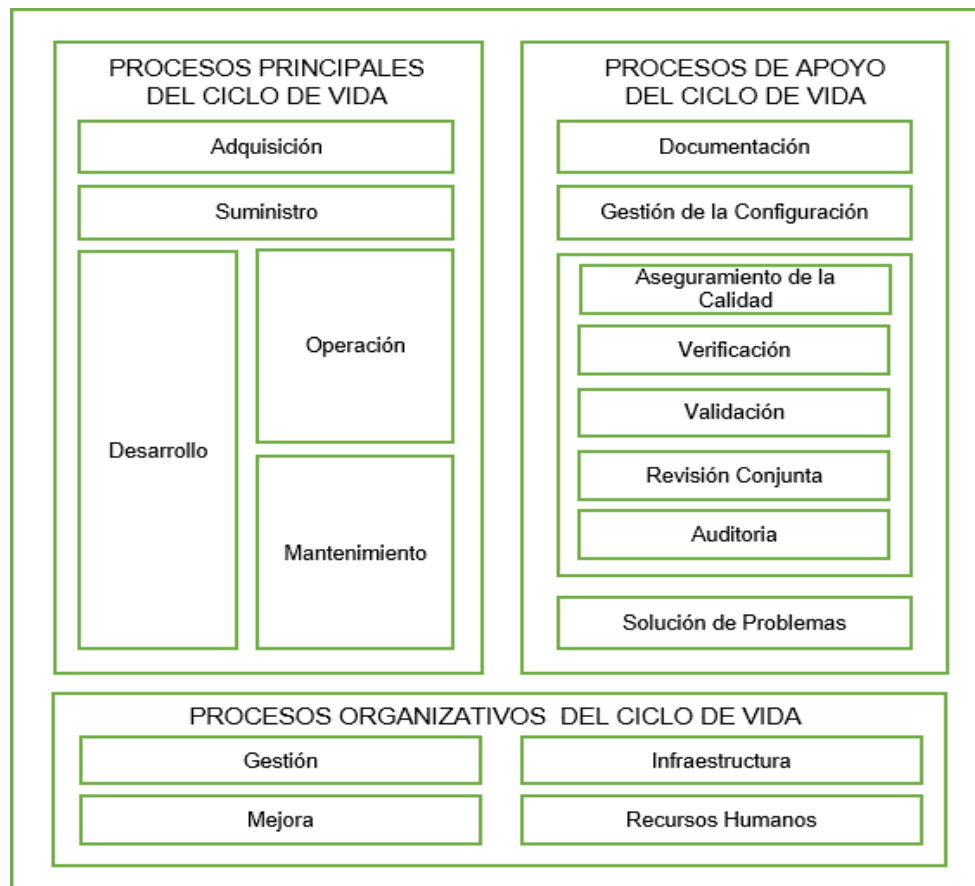


Figura 17 Grupos de procesos de la norma ISO-12207: Fuente Mg. Luis Omar Tangarife Téllez

Los procesos se clasifican en tres tipos: Principales, de soporte y de la organización. Los procesos de soporte y de organización deben existir independientemente de la organización y del proyecto ejecutado.

- Procesos principales del ciclo de vida: Son los procesos para iniciar o llevar a cabo el desarrollo, operación o mantenimiento del software

- Procesos de apoyo del ciclo de vida: Un proceso de apoyo es el que apoya a otro proceso como parte esencial del mismo. Un proceso de apoyo se emplea y ejecuta por otro proceso, según sus necesidades.
- Procesos organizativos del ciclo de vida: Se emplean por una organización para establecer e implementar una infraestructura constituida por procesos y personal asociado al ciclo de vida y para mejorar continuamente esta infraestructura. Se usan habitualmente fuera del ámbito de proyectos y contratos, contribuye a la mejora de la organización.

#### *2.1.5.2 ISO 9000*

la norma ISO 9000 da parámetros de calidad a las empresas, y gestiona la manera de comportarse frente a diversas situaciones presentadas, siempre teniendo como premisa la satisfacción del usuario y la oferta de valor que se le da, entendiéndose valor como ese fin que busca el cliente de la organización y que apropie su gestión hacia la misma.

Para este caso se toman apartes de la norma para orientar sus propósitos fundamentales orientados hacia la calidad.

La familia de Normas ISO 9000 citadas a continuación se ha elaborado para asistir a las organizaciones, de todo tipo y tamaño, en la implementación y la operación de sistemas de gestión de la calidad eficaces.

La norma ISO 9000 describe los fundamentos de los sistemas de gestión de la calidad y especifica la terminología para los sistemas de gestión de la calidad.

La Norma ISO 9001 especifica los requisitos para los sistemas de gestión de la calidad aplicables a toda organización que necesite demostrar su capacidad para proporcionar productos que cumplan los requisitos de sus clientes y los

reglamentarios que le sean de aplicación, y su objetivo es aumentar la satisfacción del cliente.

La Norma ISO 9004 proporciona directrices que consideran tanto la eficacia como la eficiencia del sistema de gestión de la calidad. El objetivo de esta norma es la mejora del desempeño de la organización y la satisfacción de los clientes y de otras partes interesadas.

La Norma ISO 19011 proporciona orientación relativa a las auditorías de sistemas de gestión de la calidad y de gestión ambiental.

Todas estas normas juntas forman un conjunto coherente de normas de sistemas de gestión de la calidad que facilitan la mutua comprensión en el comercio nacional e internacional.

#### *2.1.5.3 Estándar IEEE 1074-1998*

Es un estándar para la generación de procesos de desarrollo de software y el mantenimiento de procesos, esta norma exige la realización de los proyectos que tienen como modelo el ciclo de vida de sistemas basados en la misión, visión, metas y recursos de la organización.

El producto de esta norma es el proceso de ciclo de vida del proyecto de software, es compatible con la norma ISO 9000 y la IEEE 12207 y se enmarca en el desarrollo de procesos de la organización.

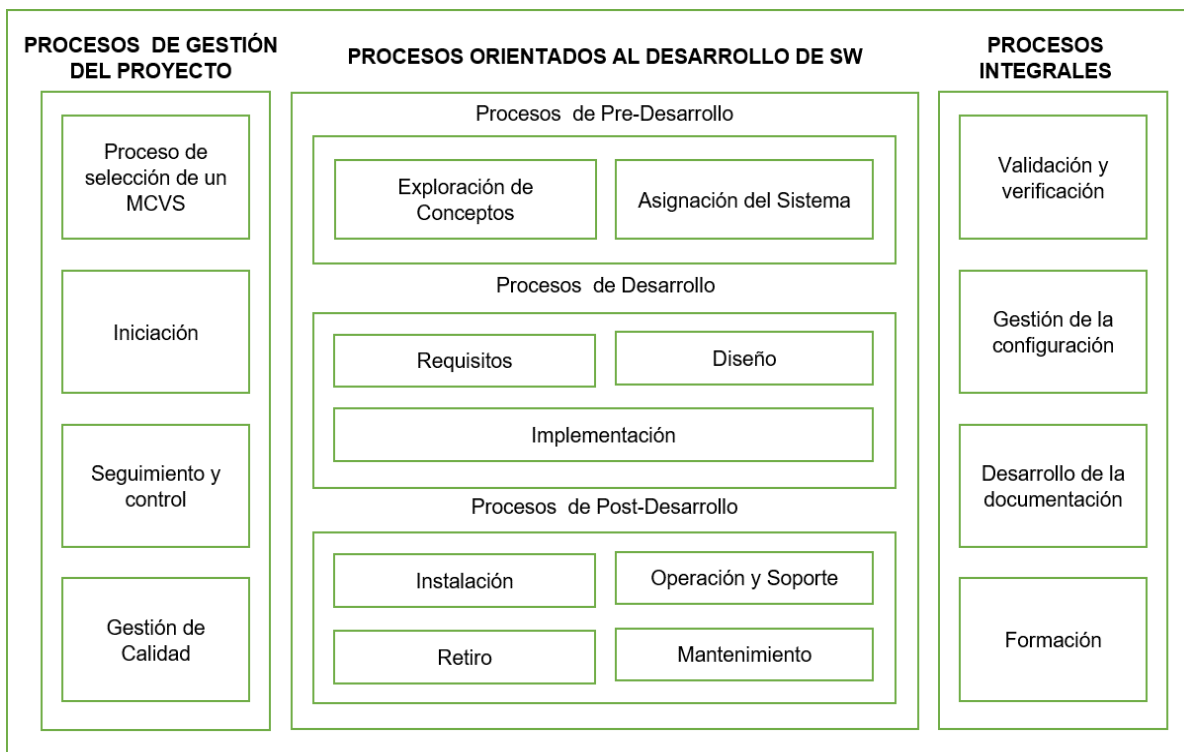


Figura 18 IEEE-1074. Fuente:

El conjunto de estándares<sup>117</sup> de la IEEE abarcan todos los aspectos técnicos relacionados con la Ingeniería de Software, pero deben ser interpretados y adaptados a las necesidades particulares de cada organización para sacarles el máximo provecho.

#### 2.1.6 Modelos de mejoramiento de procesos de software

Para las compañías un producto o servicio es de calidad cuando satisface las necesidades y expectativas del cliente otorgando a éste seguridad sobre su uso, fiabilidad de sus funciones esperadas y confianza en un producto o servicio sin fallos y duradero según tiempos establecidos y acordados (García Fernández, 2010).

<sup>117</sup> En el (Anexo A) se mencionan los estándares aprobados por esta organización que tienen algún nexo con la ingeniería de software.

El SEI ha tomado la premisa de la gestión de proceso, “la calidad de un sistema o de un producto está muy influenciada por la calidad del proceso empleado para desarrollarlo y para mantenerlo” (KONRAD & SHRUM , 2009).

#### 2.1.6.1 CMMI (*Capability Maturity Model Integration*)

Es una aproximación a la mejora de los procesos usados para guiar el desarrollo de los proyectos. Requiere preparación previa, a menudo la ayuda de un consultor para determinar los procesos y de un asesor externo para evaluar un grupo o proyecto y ver en qué nivel se encuentra (valorando del 1 al 5). Originalmente CMMI era estrictamente para la ingeniería del software y la gestión de los proyectos<sup>118</sup> (KONRAD & SHRUM , 2009).

CMMI se refiere a los modelos que contienen las mejores prácticas que ayudan a las organizaciones a mejorar sus procesos. Han sido desarrollados por equipos de trabajo formados por especialistas de la industria, el gobierno y el SEI<sup>119</sup>.

CMMI Es una guía que ayuda en la mejora de procesos. El enfoque del modelo permite evolucionar desde un proceso en crisis a un proceso controlado, estandarizado, medido y optimizado que sienta las bases de la mejora continua y permite a la organización adoptar nuevas prácticas sobre un proceso estable y controlado<sup>120</sup>.

---

<sup>118</sup> En su última versión, CMMI, aspira a adaptar aspectos como la implementación de paquetes e infraestructuras.

<sup>119</sup> Software Engineering Institute. Quien transfirió los derechos al CMMI Institute para su operación y comercialización.

<sup>120</sup> Es utilizado por las organizaciones para entender las mejores prácticas de la industria, para priorizar y adoptar las mejoras a los procesos existentes, para compararse con su competencia dentro del mercado o para que los clientes puedan identificar las prácticas que necesitan demostrar sus proveedores.

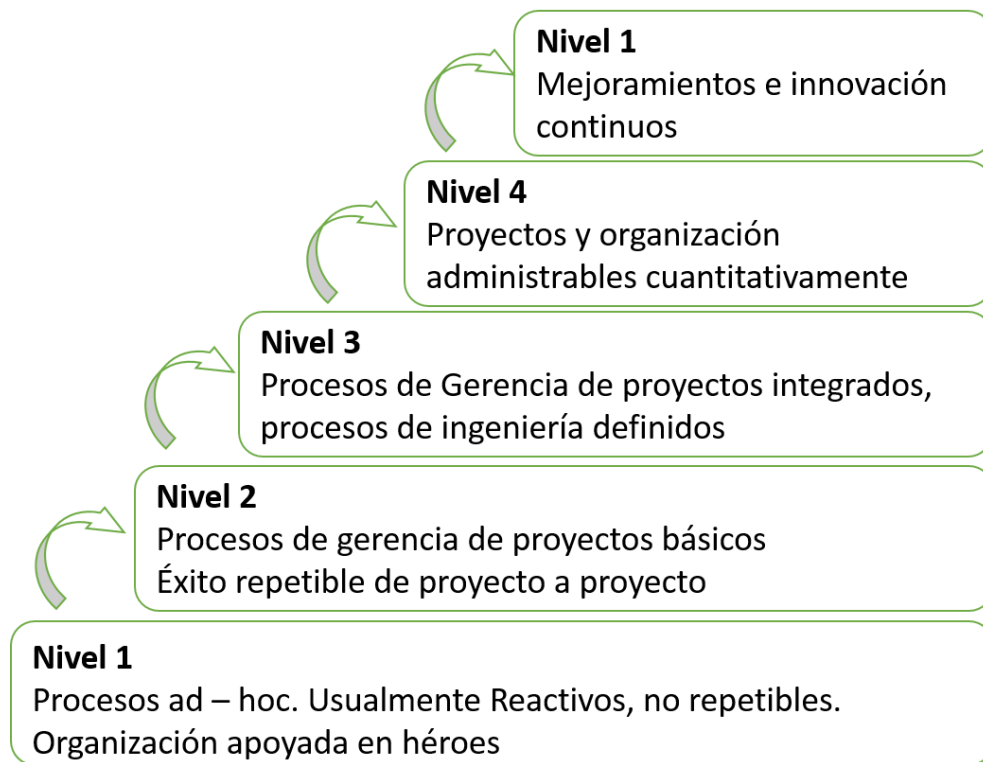


Figura 19 Esquema escalonado de CMMI. Fuente: Mauricio Morales, Manual de Administración de Proyectos

Según el modelo que se utilice se puede obtener el documento con un conjunto de guías que ayudan en:

- ✓ Desarrollo y mantenimiento de productos y servicios (CMMI DEV).
- ✓ Adquisición de productos y servicios (CMMI ACQ).
- ✓ Establecimiento, entrega y gestión de los servicios (CMMI SVC).

Contiene elementos esenciales de un proceso efectivo y propone una forma de adopción para las organizaciones que permite incrementar la calidad y productividad, al tiempo que controla el presupuesto y los compromisos establecidos. Cada una debe interpretar, adoptar y aplicar aquellas prácticas que le apoyan en el logro de sus objetivos y cumplimiento de sus necesidades de manera eficiente.



Está estructurado para facilitar su uso en elementos que definen la forma y modo de aplicarlo, considerando los elementos que son obligatorios, sugeridos o el material informativo en las áreas de proceso. En general el documento se puede revisar en función de metas, prácticas y subprácticas con el resto del material informativo

#### *2.1.6.2 Proceso de Software en Equipo TSP*

TSP<sup>121</sup> es el conjunto de prácticas de estrategias que debe seguir un administrador para poder aprovechar el valor que le ofrece a una empresa o grupo de trabajo (Del Valle Rodríguez, 2009).

El TSP proporciona un procedimiento operativo definido para guiar a los ingenieros y administradores a través de las etapas de formación de equipos. Este proceso especifica los pasos necesarios para establecer un entorno eficaz de trabajo en equipo. Sin una guía, los ingenieros deben trabajar en los detalles de La formación de equipos y trabajo en equipo por sí mismos. Dado que la definición de estos datos implica una considerable habilidad y esfuerzo, y puesto que pocos ingenieros tienen la experiencia o el tiempo para hacer este ejercicio con todos los detalles necesarios, los equipos de ingeniería en general siguen procesos de construcción del equipo y trabajo en equipo ad-hoc. Esta representa una pérdida de tiempo y que a menudo se produzcan equipos deficientes en el funcionamiento.

Con un proceso definido y un plan que sigue ese proceso, los ingenieros pueden ser altamente eficientes. Si ellos no tienen un proceso de este tipo, tienen que parar

---

<sup>121</sup> TSP en ingles Team Software Process Comienza con un proceso de inicio del proyecto en el que se establecen las estrategias, objetivos, roles y responsables de cada uno de esos roles en el proyecto, los planes individuales, las métricas que se van a llevar y se reparte el trabajo, se ejecuta, se le da seguimiento y al final se determina el cumplimiento de los planes y que tan buena o mala fue la calidad del trabajo realizado.

en cada paso de averiguar qué hacer a continuación y cómo hacerlo. La mayoría de los procesos de ingeniería son bastante complejas e implican muchos pasos. Sin una guía específica, los ingenieros son propensos a saltarse los pasos, hacer pasos en una improductiva ordenar, o que perder tiempo en averiguar qué hacer a continuación. El TSP proporciona los procesos operativos necesarios para formar los equipos de ingeniería, para establecer un entorno eficaz de los equipos, y para orientar a los equipos en hacer el trabajo.

#### 2.1.6.3 Proceso de Software Personal PSP

El PSP es un marco de trabajo diseñado para enseñar a los programadores a hacer mejor su trabajo. Muestra cómo estimar y planificar el trabajo, como controlar el rendimiento frente a esos planes y como mejorar la calidad de los programas.

El PSP es un conjunto ordenado de procesos definidos que orientan a los ingenieros de software a medir, evaluar y monitorear la manera de hacer sus tareas. Los principales objetivos del PSP son:

1. Mejorar las estimaciones
2. Mejorar la planeación y acompañamiento de cronogramas
3. Proteger contra el exceso de compromisos
4. Crear un compromiso personal con la calidad
5. Compromiso del desarrollador en la mejora continua del proceso de desarrollo
6. Aumento de la calidad a través de la reducción de la incidencia de errores
7. Mayor precisión en las estimaciones de tamaño del software y tiempo de desarrollo

El modelo PSP está dividido en niveles, implantados de manera incremental. Los niveles superiores adicionan características a los niveles ya implantados lo que

minimiza el impacto de los cambios en los hábitos del desarrollador<sup>122</sup>. Lo más importante en el proceso de aprendizaje son los datos recogidos después de cada fase<sup>123</sup>.

#### 2.1.7 Descripción del proceso software

Un Proceso Software se puede describir como (KONRAD & SHRUM , 2009) “Un conjunto coherente de políticas, estructuras organizacionales, tecnologías, procedimientos y artefactos que son necesarios para concebir, desarrollar, instalar y mantener un producto software”.

El formato en que se maneja esta información es usualmente documentos de texto en lenguaje natural. Esto tiene la gran desventaja de que toda la manipulación y generación de documentación para dichas metodologías es un proceso puramente manual.

##### 2.1.7.1 *Lenguaje de Modelado Unificado UML*

El Lenguaje de Modelado Unificado<sup>124</sup> es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables (wikipedia, s.f.).

UML es un lenguaje gráfico para especificar, construir y documentar los artefactos que modelan un sistema. UML fue diseñado para ser un lenguaje de modelado de

---

<sup>122</sup> Este deberá tan sólo adaptar nuevas técnicas a las ya existentes y conocidas.

<sup>123</sup> Con base en los resultados obtenidos en la fase actual se propone mejorar el desempeño personal para la siguiente fase.

<sup>124</sup> UML acrónimo de Unified Modeling Language

propósito general, por lo que puede utilizarse para especificar la mayoría de los sistemas basados en objetos o en componentes, y para modelar aplicaciones de muy diversos dominios de aplicación (telecomunicaciones, comercio, sanidad, etc.) y plataformas de objetos distribuidos (como por ejemplo J2EE, .NET o CORBA) (Hilera & Palomar, 2005).

#### 2.1.7.2 SPEM

OMG<sup>125</sup> ha desarrollado y aprobado recientemente el estándar SPEM<sup>126</sup> versión 2.0. SPEM es un metamodelo para modelos de procesos de ingeniería del software y de ingeniería de sistemas (Francisco Ruiz, Javier Verdugo, 2008).

SPEM 2 sirve para definir procesos de desarrollo de software y sistemas y sus componentes. Su alcance se limita a los elementos mínimos necesarios para definir dichos procesos sin añadir características específicas de un dominio o disciplina particular; pero sirve para métodos y procesos de diferentes estilos, culturas, niveles de formalismo, o modelos de ciclos de vida.

#### 2.1.7.3 Diagramas de flujo de datos DFD

Un diagrama de flujo es una representación gráfica de un proceso. Cada paso del proceso es representado por un símbolo diferente que contiene una breve descripción de la etapa de proceso. Los símbolos gráficos del flujo del proceso están unidos entre sí con flechas que indican la dirección de flujo del proceso.

---

<sup>125</sup> El Object Management Group, la organización industrial promotora de UML

<sup>126</sup> SPEM del inglés Software & Systems Process Engineering metamodel: Sirve para representar procesos de ingeniería de software, se pretende que este sea el estándar industrial para la representación de modelos de procesos de ingeniería del software e ingeniería de sistemas.

El diagrama de flujo ofrece una descripción visual de las actividades implicadas en un proceso mostrando la relación secuencial entre ellas, facilitando la rápida comprensión de cada actividad y su relación con las demás, el flujo de la información y los materiales, las ramas en el proceso, la existencia de bucles repetitivos, el número de pasos del proceso, las operaciones de interdepartamentales... Facilita también la selección de indicadores de proceso (Talavera Pleguezuelos, 1999).

## **2.2 Estado del arte**

La evolución de la ingeniería de software ha traído propuestas diferentes para mejorar los resultados del proceso de construcción. Recabando sobre las metodologías tradicionales basadas en la planeación y las metodologías ágiles que se basan en la adaptabilidad, de igual forma, los modelos para el mejoramiento de los procesos de desarrollo tienen gran relevancia en la búsqueda de la metodología adecuada para producir software de calidad en cualquier contexto de desarrollo como se enuncia en “Modelos de Ciclo de Vida de Desarrollo de Software en el Contexto de la Industria Colombiana de Software” escrito por Hugo F. Arboleda Jiménez“. De una u otra forma, las características de los proyectos de software hacen necesario seguir prácticas específicas para optimizar los resultados de los desarrollos” (Jiménez H. F., 2002).

Algunas empresas como Motorola han adaptado su proceso software, (O’Kane, Russo, & Fitzgerald, 2003) ...”El proceso de software es adaptado para cada proyecto particular y es seguido rigurosamente en todos los proyectos”. Motorola hizo la adaptación de su proceso software para proyectos y a nivel de organización, e instruye y entrena a sus empleados en el proceso adaptado como parte de su cultura corporativa.

(IEEE, 2004) El SWEBOK, dice “importante señalar que los procesos predefinidos incluso los estandarizados deben adaptarse a las necesidades locales, por ejemplo, el contexto organizacional, el tamaño del proyecto, los requisitos reguladores, las

prácticas industriales y las culturas corporativas”. Esto es particularmente importante en el entorno al que va dirigido este proyecto (académico), ya que carece de los medios y recursos con los que cuenta la industria, además de que los roles, no siempre tienen la distribución e importancia de otros tipos de proyectos

Y (Pedreira, Piattini, Luaces, & Brisaboa, 2007) Aunque continuamente aparecen nuevos trabajos y propuestas en el área de proceso software, es difícil que encajen en su forma original en una empresa dada. De ahí la necesidad de adaptar los procesos estándar a las características particulares de la empresa o proyecto.

También es cierto que nadie discute la necesidad de adaptar, las metodologías de desarrollo y aunque la adaptación del proceso software es señalada como una actividad obligatoria por la mayoría de las metodologías, en general se lleva a cabo siguiendo un enfoque adhoc (Pedreira, Piattini, Luaces, & Brisaboa, 2007). Sin que esas experiencias se puedan replicar en otros contextos.

Oscar Pedreira y otros en su artículo “Una revisión sistemática de la adaptación del proceso software”, argumentan que adaptar un proceso software a las necesidades concretas de una organización o un proyecto dado tiene lugar en dos niveles, nivel de la organización o nivel del proyecto (Pedreira, Piattini, Luaces, & Brisaboa, 2007). Pero en la práctica es complejo por las diferencias en las organizaciones que se dedican al desarrollo de software.

Para lograr adaptar procesos en el desarrollo de software, en los últimos años se han hecho esfuerzos y trabajos orientados a definir procesos de adaptación, (Sanchez, 2012) dice “La propuesta es definir un conjunto de características compartidas entre la gran variedad de PYMES desarrolladoras de software que existen en el país, que permita unificar criterios para la adaptabilidad de estándares y procesos de software”. Esto es aplicable a cualquier tipo de empresa y redundante en una mejora en los procesos porque estos fluyen naturalmente.

### 2.2.1 El proceso software y de la adaptación de procesos

Una de las ramas más investigadas en la actualidad son los procesos de software, todas las entidades dedicadas al desarrollo desean implementar unas pautas claras para sus propósitos, sin embargo no todos los proyectos son iguales y muchas veces determinan una metodología diferente para poder llegar a un buen término. Es así que se han inspeccionado muchos trabajos de investigación en donde se plantean diferentes tipos de procesos de desarrollo.

Por su parte Javier Mogollón Afanador, y Luis Alberto Esteban Villamizar en su artículo “El desarrollo individual de proyectos de software: una realidad sin método” que *“Los proyectos de desarrollo de software de manera individual, es decir abordados por una sola persona, son frecuentes en contextos particulares, donde se requiere automatizar pequeñas tareas, y para lo cual no se cuenta con grandes presupuestos. Esta informalidad de contratación de proyectos de desarrollo de software, exige unos requisitos mínimos de calidad, por lo cual es ideal contar con una metodología de desarrollo de software”*. (Esteban & Afanador, 2010)

De lo anterior se ve que en diferentes ámbitos de estudio se ven los procesos de ingeniería de software como una meta muy investigada pero poco resuelta, por lo que se van haciendo necesarios estudios como el presente en donde se den estrategias concretas para un contexto particular, de acuerdo a la aplicación se pueden proponer procesos de desarrollo sin dejar de lado las técnicas que para ello existen. Sin embargo en el objeto de estudio de esta investigación se visualiza lo planteado por el profesor Esteban en donde los desarrollos obtenidos en la institución se llevan con un método establecido en el mejor de los casos o en algunos no se evidencia.

Existen muchos casos tanto de compañías exitosas como en proyectos de software que debido a la mejora de procesos y de mediciones en ellos han tenido buenos

resultados, sin dejar de lado que la gran preocupación plasmada en varias investigaciones, en donde se pretenden acercarse a la manera de diseñar eficientemente sistemas, plasmando inconvenientes como falta de objetivos, resultados no esperados o deserción del proyecto.

En todo este compendio de investigaciones se denota la necesidad de uso del ciclo de vida del sistema, siendo aplicado a cada una de las necesidades planteadas, así mismo como técnicas y metodologías de desarrollo que junto con estrategias organizacionales dan una bitácora de trabajo para el desarrollo de cada sistema

Por lo anterior se plantea en este trabajo de investigación unos parámetros para ser aplicados en la institución teniendo pautas claras en los artefactos e instrumentos entregables en futuros trabajos de tecnología aplicada en la Universidad Popular del Cesar Seccional Aguachica, dando esto un avance y una contribución más al desarrollo de técnicas y procesos de ingeniería de software que aunque son mundiales tienen su aplicabilidad local como estrategia de desarrollo de cada región, en donde algunas prácticas particulares de proceso resultan en mayor probabilidad de éxito. De igual forma, la adaptabilidad de los modelos de acuerdo al contexto ayude a mejorar la calidad de los productos de software desarrollados de esta forma, generando un avance tecnológico y científico.

Aportes importantes encontrados en artículos como “*el desarrollo individual de proyectos de software*” (Esteban & Afanador, 2010), en donde hablan de los desarrollos individuales desarrollados por una sola persona con poco presupuesto y pocos estándares de calidad, lo que conlleva a formalizar estos pequeños desarrollos a un criterio propio y estándares de calidad que sean competitivos con las grandes empresas.

Por su parte Jones en su libro “*Introduction and Definitions of Software Best Practices*” (Jones, 2010) muestra las mejores prácticas probadas en el desarrollo



de software de donde se puede enfatizar en las metodologías descritas en este trabajo.

*“Generación Automática de Software para Sistemas de Tiempo Real: Un Enfoque basado en Componentes, Modelos y Frameworks”* (Alonso, Pastor, Sánchez, Álvarez, & Chicote, 2012), que habla sobre las diversas tendencias de calidad aplicadas a sistemas heterogéneos de esta modalidad, que por sus características tiene unas visiones diferentes en cuanto a su desarrollo.

*En el artículo “Estado del arte de la investigación en verificación formal. Ingeniería”* (Serna-M. & Morales-V., 2014), *cabe resaltar que* “La función explícita de la verificación formal es encontrar errores y mejorar la confianza en la exactitud del diseño del sistema, lo que supone un reto para la ingeniería de software de este siglo”, viéndose esto como una necesidad del planteamiento de calidad en los diferentes desarrollos del área

Así mismo en el escrito *“Estimación y control de costos en métodos ágiles para desarrollo de software: un caso de estudio”* en donde se argumenta que las metodologías ágiles tienen una buena aplicabilidad en pequeños proyectos por su veracidad y facilidad en la implementación, por lo cual es una buena forma de aplicar técnicas de ingeniería en el desarrollo de proyectos propios. (Hernández & Ortega-Martínez, 2014),

Otros estudios como, *“Diseño de un programa de medición estratégico para organizaciones de ingeniería de Software: descubriendo dificultades y problemas.”* (Mitre-Hernández, García-Guzmán, Amescua-Seco, & Velasco-Elizondo, 2014) enuncia que con la medición de software, rama de la ingeniería dedicada a la calidad de los desarrollos, generan grandes incursiones y demuestran mejoría en los procesos de la empresa, aportando así al buen desempeño y la implementación de

sistemas de calidad, aunque tiene diversos problemas ya que en algunos casos no es posible generar esta medición.

En el artículo, *“Requirements modeling languages for software product lines: A systematic literature review”* (Sepúlveda, Cravero, & Cachero, 2016), en donde se plantean casos y formas de aplicación a los modelos de desarrollo y procesos aplicados que se han probado en diferentes ámbitos con buenos resultados y le dan base contextual a lo anteriormente descrito y a la finalidad del presente trabajo de investigación.

### 2.2.2 Nacional

“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” (Jiménez H. F., 2002).

Como lo enuncia el Profesor Jiménez en su escrito “Modelos de Ciclo de Vida de Desarrollo de Software en el Contexto de la Industria Colombiana de Software”, la inclusión del tema de planeación y seguimiento del software en Colombia, empieza con la llegada de nuevas tecnologías y la vinculación de estas a nuevos planes de estudio de las diferentes universidades nacionales, antes de ello se manejaban sistemas enteramente incrementales y de una forma empírica o como lo enuncia el auto “artesanal”.

De igual manera enuncia que el modelo cascada, de prototipos y en espiral fueron los más usados aunque no de la mejor forma en la mayoría de ocasiones, ya que

no se realiza un buen análisis de riesgos ni pruebas de manera consiente sobre los resultados sino que por el contrario se convierte en tendencia de ensayo y error.

“Uno de los temas más abordados en la industria del software Colombiano es el tema de planeación y seguimiento de proyectos. Con este tema viene adjunto el tema de la definición de modelos de ciclo de vida de desarrollo. Sin embargo, normalmente el tema se ha abordado desde una perspectiva tradicional de planeación de proyectos, sin hacer énfasis en las características propias de los proyectos de software”. (Jiménez H. F., 2002)

Los cambios dados por la actualización tecnológica o diversidad de las necesidades no se tienen totalmente en cuenta por lo que se registran usos de metodologías de gerencia de proyectos como la del PMI<sup>127</sup>, donde algunos procesos se realizan de manera instrumental y no con la debida planeación requerida para llegar a un buen final de los desarrollos de sistemas

Basándose en lo anterior se hace necesario abordar el desarrollo o creación de sistemas desde un enfoque práctico, de tal forma que se tenga un buen análisis de requerimientos en donde se tengan claro las necesidades que se requiere suplir, una claridad en la etapa de desarrollo con el fin de reducir tiempos de programación y una muy buena implementación de estrategias de pruebas para verificar el buen funcionamiento y cumplimiento de las necesidades del sistema.

Por su parte Urrego (Urrego N. J., 2007), en su artículo “Aseguramiento de Calidad en el Desarrollo del Software”, asegura que la calidad en el Ciclo de Vida del

---

<sup>127</sup> PMI acrónimo de Project Management Institute, quien desarrollo la *Guía de los Fundamentos de Gestión de Proyectos* (Guide to the Project Management Body of Knowledge o PMBOK), contiene una descripción general de los fundamentos de la Gestión de Proyectos reconocidos como buenas prácticas.

Software es un proceso transversal y no se debe considerar como una etapa más dentro de este proceso, ya que en la mayoría de proyectos informáticos la etapa de pruebas viene después del desarrollo cuando los costos en detección y resolución de defectos son bastante costosos en tiempo y dinero. En donde se enfatiza en la necesidad de tener claras las tendencias para hacer pruebas sobre todo el sistema y sus etapas con el fin de evitar contratiempos reflejados en tiempo y dinero que retrasara la producción de los sistemas. La creación de una cultura de calidad en las organizaciones hace que los avances sean precisos y de mejor manera implementados.

En la universidad Católica del Norte, Castrillón propone una metodología de desarrollo, aunque aplicada a objetos virtuales de aprendizaje, puede ser vista como desarrollo de software, ya que está basada en metodologías como XP, RUP y UP, en donde se plantean como fases del desarrollo: la concepción del objeto, diseño y desarrollo, integración y despliegue, pruebas de aprendizaje y consolidación (Castrillón, 2011). De lo anterior se puede observar que el desarrollo de sistemas en diferentes ámbitos puede ser sustentado en metodologías y teorías de desarrollo de software.

Por su parte en la universidad nacional de Colombia, sede Manizales, se encuentra en publicación del profesor Jaramillo Villegas en donde se marcan siete pasos de las metodologías ágiles para el desarrollo de software. El ciclo de vida de los sistemas es la manera como se describen los diferentes pasos que se deben seguir para el desarrollo de un software, empezando por una necesidad hasta llegar a la puesta en marcha de una solución. (Villegas, 2015) Los pasos que sigue en su escrito son: ingeniería, análisis, diseño, implementación, pruebas, documentación y mantenimiento. Todos ellos enmarcados en una metodología cascada con el fin de generar un desarrollo secuencial de la labor de creación de software

### 2.2.3 Análisis de contexto local

Dentro de los trabajos de grado del programa de ingeniería de sistemas de la Universidad popular del Cesar, sede Aguachica se encontró tan solo un trabajo directamente relacionado directamente con el estudio de los procesos de software. El proyecto (Parada Robles & León Castro, 2013) **Estructurar una propuesta metodológica para guiar el proceso de desarrollo de software en los proyectos de los estudiantes de ingeniería de sistemas de la universidad popular del cesar seccional Aguachica**, se desarrolló, con el objetivo de buscar un proceso software estándar para los proyecto que incluyan desarrollo de software como producto, en el programa Ingeniería de Sistemas.

#### 2.2.3.1 Revisión Bibliográfica

Como parte de esta investigación, se recopiló información sobre las Metodologías existentes, más representativas o más comunes y de las que más documentación se encuentra, entre Metodologías tradicionales, ágiles y web.

#### **Metodologías y modelos tradicionales**

- Modelo en Cascada
- Modelo Incremental
- Modelo en Espiral
- Modelo de Construcción de Prototipos
- Análisis de Sistemas Estructurados y Metodología Diseño SSADM
- Análisis Estructurado y Técnica de Diseño SADT
- Desarrollo de Sistemas Estructurado de Datos DSED
- Desarrollo Rápido de Aplicaciones DRA
- Proceso Unificado de Rational RUP

### **Metodologías y modelos ágiles**

- Scrum
- Programación Extrema XP
- Proceso Unificado Ágil AUP
- Modelo de Desarrollo de Sistemas Dinámicos DSDM
- Desarrollo adaptativo del software ASD
- Modelado Ágil AM
- Desarrollo Dirigido por Características FDD
- Marco de Soluciones de Microsoft MSF

### **Metodologías web**

- Metodología Ingeniería Web IWEB
- Metodología de Relación Entre Objetos EORM
- Método de Diseño de Hipermedia Orientado a Objetos OOHDM
- Método de Diseño de Hipermedia Orientado a Objetos Basado en Escenarios SOHDM
- Método de Diseño para Sitios Web WSDM
- Método de Análisis de Navegación Relacional RNA
- Método UWE

De las 24 metodologías escogidas para el estudio inicial se seleccionaron 10 por ser las más reconocidas y de las cuales se encontró suficiente documentación para el análisis.

A continuación se explica por qué no se incluyeron las demás Metodologías de Desarrollo de Software mostradas anteriormente:

- Análisis De Sistemas Estructurados y Metodología Diseño SSADM: su enfoque consiste en completar una fase antes de iniciar la próxima, causando

demoras en la programación y entrega del sistema. Además, hay una gran cantidad de normas para la preparación y presentación de la documentación. Requiere que el personal esté capacitado en el uso de la Metodología. Por lo tanto, no se incluyeron como parte del estudio.

- No se encontró información detallada sobre los artefactos y entregables usados en las Metodologías: Análisis Estructurado y Técnica de Diseño SADT y el Desarrollo de sistemas estructurado de datos DSED, por lo que no se usaron como referencia para el desarrollo de esta Investigación.
- El Desarrollo rápido de aplicaciones DRA es ideal para grandes proyectos, característica que no es útil para el entorno al que va dirigido esta Investigación. Además, esta Metodología no funciona con proyectos con requisitos cambiantes, afectando el rendimiento.
- Las Metodologías: Modelo de Construcción de Prototipos, Proceso Unificado Ágil, Modelo de Desarrollo de Sistemas Dinámicos DSDM, Desarrollo conducido por características DCC, Desarrollo adaptativo de software ASD y Modelado Ágil; poseen características similares a las de las Metodologías ágiles de Desarrollo de Software<sup>128</sup> y están cubiertos por las cuatro (4) Metodologías ágiles de Desarrollo de Software tomadas como referencia para este estudio.

Con las metodologías restantes se estudió sus artefactos, entregables, diagramas y modelos empleados en cada fase, conocer y poder seleccionar finalmente las metodologías que harán parte de este estudio en el que los estudiantes de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica podrán apoyarse para determinar los requerimientos mínimos de la metodología de desarrollo que escojan y así terminar satisfactoriamente su proyecto de grado que incluya desarrollo de software

---

<sup>128</sup> Son iterativas e incrementales, responden a requisitos cambiantes que puedan surgir durante el proyecto y los incrementos se obtienen en un corto periodo de tiempo.

Las 10 metodologías/Modelos escogidos para el estudio, se muestran en la siguiente tabla, se escogen por ser más comunes en nuestro medio y por encontrarse bibliografía suficiente.



Metodología o modelo	Fases	Artefactos	Entregables	Roles
<b>Modelo en Cascada</b>	Análisis	<ul style="list-style-type: none"> <li>- Minuta de junta con el cliente</li> <li>- Listado de Requisitos de usuario</li> <li>- Listado de Requerimientos funcionales y no funcionales</li> </ul>	-Documento de especificación de requisitos	<ul style="list-style-type: none"> <li>- Analista</li> <li>- Diseñador</li> <li>- Programador</li> <li>- Tester</li> </ul>
	Diseño	<ul style="list-style-type: none"> <li>- Definir Arquitectura del software</li> <li>-Requerimientos de hardware y software</li> <li>- Diseño Arquitectónico</li> <li>- Diseño detallado</li> </ul>	-Documento de Diseño del Software	
	Codificación	Prototipos	- Prototipo del Producto Software.	
	Pruebas	<ul style="list-style-type: none"> <li>- Pruebas de unidad</li> <li>- Pruebas de integración</li> <li>- Pruebas de sistema</li> <li>- Pruebas de aceptación</li> </ul>	- Documento de Pruebas	

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
<b>Modelo Incremental</b>	Análisis	<ul style="list-style-type: none"> <li>- Plan de Gestión del proyecto</li> <li>- Listado Requerimientos funcionales y no funcionales</li> <li>- Presupuesto</li> </ul>	<ul style="list-style-type: none"> <li>- Documento de especificación de requisitos</li> </ul>	<ul style="list-style-type: none"> <li>- Analista</li> <li>- Diseñador</li> <li>- Programador</li> <li>- Tester</li> </ul>
	Diseño	<ul style="list-style-type: none"> <li>- Modelo de diseño</li> </ul>	<ul style="list-style-type: none"> <li>- Modelo de arquitectura de Software</li> </ul>	
	Codificación	<ul style="list-style-type: none"> <li>- Plan de incremento</li> <li>- Programación y pruebas</li> </ul>	<ul style="list-style-type: none"> <li>- Producto Software</li> <li>- Manuales de usuario</li> </ul>	
	Pruebas	<ul style="list-style-type: none"> <li>- Pruebas de unidad</li> <li>- Pruebas de integración</li> <li>- Pruebas de aceptación</li> <li>- Tareas de marketing</li> <li>- Capacitaciones a usuarios</li> </ul>	<ul style="list-style-type: none"> <li>- Documento de Pruebas</li> </ul>	
<b>Modelo en Espiral</b>	Planeación	<ul style="list-style-type: none"> <li>- Minuta de junta con el cliente</li> <li>- Listado de Requerimientos funcionales y no funcionales</li> <li>- Presupuesto</li> <li>- Requisitos de interfaz</li> </ul>	<ul style="list-style-type: none"> <li>- Documento Especificación de Requisitos</li> </ul>	<ul style="list-style-type: none"> <li>- Analista</li> <li>- Diseñador</li> <li>- Programador</li> <li>- Tester</li> </ul>

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
		- Plan de gestión		
	Análisis de riesgo	- Estudios de viabilidad	- Documento análisis de riesgos - Sistema prototipo - Diseño arquitectónico	
	Ingeniería	- Diagrama de proceso - Programación y pruebas	- Plan de desarrollo - Diseño detallado - Producto Software - Diseño de Bases de Datos	
	Evaluación	- Plan de integración - Generación de mejoras, errores, defectos y ampliación	- Documento pruebas	
<b>Metodología RUP</b>	Concepción	- Listado de requerimientos funcionales y no funcionales - Casos de uso - Diagrama de Casos de uso - Modelo Entidad-Relación	- Documento de Especificación de requisitos - Documento de Visión - Documento de Misión	Analistas Desarrolladores Gestores Apoyo

Metodología o modelo	Fases	Artefactos	Entregables	Roles
			- Documento de Requerimientos de Calidad	Especialista en pruebas Stakeholders
	Elaboración	- Diagrama de clases - Diagrama de secuencia - Diagrama de estados - Diagrama de colaboración - Modelo de dominio	- Documento Arquitectura del Software	Revisor Coordinación de revisiones Revisor técnico en Pruebas
	Construcción	- Programación y pruebas	- Plan de desarrollo - Pruebas de los casos de uso desarrollados - Pruebas de Regresión	Analista de pruebas Diseñador de pruebas.
	Transición	Pruebas finales de aceptación.	- Caso de prueba - Acta de entrega y finalización del proyecto	
<b>Modelo MSF</b>	Visión y Alcance	- Minuta de reunión	- Documento de Visión y Alcance	- Gerente de Producto

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
		- Listado de requerimientos funcionales y no funcionales	- Documento Análisis de riesgo	- Gerente de Programa
	Planificación	- Borrador plan maestro del proyecto - Cronograma - Presupuesto	- Documento del Plan del Proyecto - Documento Diseño de la Arquitectura	- Desarrollador - Gerente de Logística
	Desarrollo	- Diagrama de Clases	- Producto Software - Manuales de usuario - Manuales operativos - Manuales de instalación	
	Estabilización	Pruebas piloto	- Release Readiness aprobado - Documento registro de prueba	
	Implantación	Entrenamiento de administradores y usuarios	- Documento Implantación completa	

Metodología o modelo	Fases	Artefactos	Entregables	Roles
			- Acta de entrega y finalización del proyecto	
<b>Metodología XP</b>	Planificación	<ul style="list-style-type: none"> <li>- Historias de usuario</li> <li>- Plan de Iteraciones</li> <li>- Calculo de Velocidad del proyecto</li> <li>- Plan de entregas</li> <li>- Metáfora del Sistema</li> </ul>	- Metáfora del sistema	<ul style="list-style-type: none"> <li>- Programador</li> <li>- Cliente</li> <li>- Encargado de pruebas (Tester)</li> <li>- Encargado de seguimiento (Tracker)</li> <li>- Entrenador (Coach)</li> <li>- Consultor</li> <li>- Gestor (Big Boss)</li> </ul>
	Diseño	<ul style="list-style-type: none"> <li>- Tarjetas CRC</li> <li>- Solución pico</li> <li>- Tareas de Ingeniería</li> </ul>	- Diseños simples	
	Codificación	<ul style="list-style-type: none"> <li>- Plan de pruebas</li> <li>- Historia de iteración</li> <li>- Código base</li> <li>- Refactorización</li> </ul>	<ul style="list-style-type: none"> <li>- Producto Software</li> <li>- Manuales de usuario</li> <li>- Manual de desarrollo</li> </ul>	
	Pruebas	<ul style="list-style-type: none"> <li>Pruebas unitarias</li> <li>Pruebas de integración</li> <li>Pruebas de regresión</li> </ul>	<ul style="list-style-type: none"> <li>- Casos de prueba</li> <li>- Prueba Aceptación</li> </ul>	

Metodología o modelo	Fases	Artefactos	Entregables	Roles
		Pruebas del sistema Pruebas de aceptación		
<b>Metodología SCRUM</b>	Planteamiento	- Planificación de los Sprints - Casos de uso - Modelo Entidad-Relación - Presupuesto - Burn down (gráfico de trabajo) - Cronograma de actividades	- Documento Especificación de requisitos - Formato del Backlog del Producto - Documento Sprint Backlog	-Dueño del Producto -Scrum Master -Equipo
	Montaje	- Scrum Board - Diagrama de clases	- Diseño exploratorio - Prototipos - Diseño de la infraestructura	
	Desarrollo	- Incremento	- Producto software	
	Liberación	- Marketing y promoción - Pruebas de integración - Pruebas del sistema	- Documento de Pruebas	

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
<b>Metodología FDD</b>	Desarrollar Modelo General	- Minuta junta con el cliente	- Documento Modelo General	<ul style="list-style-type: none"> <li>- Administrador del Proyecto</li> <li>- Arquitecto Jefe</li> <li>- Manager de Desarrollo</li> <li>- Programador Jefe</li> <li>- Propietarios de clases</li> <li>- Cliente, usuario o Analista de Negocios</li> </ul>
	Construir lista de rasgos	<ul style="list-style-type: none"> <li>- Priorización de las Funcionalidades</li> <li>- División de las Funcionalidades complejas</li> </ul>	- Documento Listado de Funcionalidades	
	Planear por rasgo	<ul style="list-style-type: none"> <li>- Cronograma</li> <li>- Asignación de las Clases a los programadores</li> </ul>	- Plan de entregas	
	Diseñar por rasgo	<ul style="list-style-type: none"> <li>- Diagrama de Secuencia</li> <li>- Diagrama de clases</li> </ul>	- Documento Arquitectura del Sistema	
	Construir por rasgo	<ul style="list-style-type: none"> <li>- Incrementos</li> <li>- Inspección de código</li> <li>- Pruebas unitarias</li> <li>- pruebas integrales</li> </ul>	- Producto Software	



<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
<b>Metodología Iweb</b>	Formulación	<ul style="list-style-type: none"> <li>- Identificar los requisitos de contenido</li> <li>- Listado de requerimientos funcionales</li> <li>- Definir los escenarios de interacción de usuario</li> </ul>	Modelo de Análisis	<ul style="list-style-type: none"> <li>-Desarrolladores y Proveedores de Contenido</li> <li>-Editores de Web</li> <li>-Ingeniero de Web</li> <li>-Especialista de Soporte</li> <li>-Administrador</li> </ul>
	Planificación	<ul style="list-style-type: none"> <li>- Casos de uso</li> <li>- Presupuesto</li> <li>- Listado Requerimientos funcionales y no funcionales</li> <li>- Diagrama Entidad-Relación</li> <li>- Diccionarios de datos</li> </ul>	Plan de proyecto software	
	Análisis de riesgos	<ul style="list-style-type: none"> <li>- Análisis de contenido</li> <li>- Análisis de interacción</li> <li>- Análisis funcional</li> <li>- Análisis de configuración</li> </ul>	Boceto	
	Ingeniería	<ul style="list-style-type: none"> <li>- Diseño de contenido</li> <li>- Diseño de navegación</li> <li>- Diseño de interfaz de usuario</li> </ul>	<ul style="list-style-type: none"> <li>- Prototipo del objeto</li> <li>- Diseño de las estructuras de datos</li> </ul>	

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
			- Diseño de los componentes	
	Generación de páginas	<ul style="list-style-type: none"> <li>- Especificación de tareas</li> <li>- Especificación de escenarios</li> <li>- Especificación de Casos de Uso</li> <li>- Evaluación del cliente</li> <li>- Interfaz para elaborar páginas web ejecutables (HTML, XML, ASP)</li> </ul>	Diseño arquitectónico	
	Puesta a prueba y Evaluación del cliente	Reporte de correcciones	Casos de prueba	
<b>Metodología OOHDM</b>	Determinación de Requerimientos	<ul style="list-style-type: none"> <li>- Casos de uso</li> <li>- Diagrama de casos de uso</li> <li>- Modelo Entidad - Relación</li> </ul>	-Diagramas de Interacción de Usuario (UIDs)	<ul style="list-style-type: none"> <li>-Diseñador Gráfico</li> <li>-Desarrollador</li> </ul>

<b>Metodología o modelo</b>	<b>Fases</b>	<b>Artefactos</b>	<b>Entregables</b>	<b>Roles</b>
	Diseño Conceptual	Diagrama de Clases	Diseño Modelo semántico de la aplicación	
	Diseño Navegacional	Modelo de diseño de la navegación	Estructura navegacional	
	Diseño de Interfaz Abstracto	Diagramas de configuración ADVs-Charts	Modelo de vista abstracta de datos ADV	
	Implementación	Entorno del lenguaje de programación	Aplicación ejecutable	

*Tabla 1 cuadro comparativo entre metodologías escogidas para el estudio: Fuente el autor*

Sobre las metodologías escogidas para el estudio se consulto acerca de sus características y puntos fuertes y débiles (ventajas y desventajas) y se relacionan en la siguiente tabla.

<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
<b>Modelo en Cascada</b>	La documentación se produce en cada fase, formando un modelo sencillo y organizado	Su principal problema es su inflexibilidad al dividir el proyecto en distintas etapas. Se deben hacer compromisos en las etapas iniciales, lo que hace difícil responder a los cambios en los requerimientos del cliente	<ul style="list-style-type: none"> <li>- Los requisitos se definen al inicio del proyecto.</li> <li>- No es un modelo iterativo.</li> </ul>
<b>Modelo Incremental</b>	<ul style="list-style-type: none"> <li>- Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él.</li> <li>- Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema</li> <li>- Se reducen los fallos de funcionamiento del Software</li> </ul>	<ul style="list-style-type: none"> <li>- Los incrementos deben ser relativamente pequeños y cada uno debe entregar una funcionalidad del sistema, puede ser difícil adaptar los requerimientos del Cliente a incrementos del tamaño apropiado</li> </ul>	<ul style="list-style-type: none"> <li>- Entrega el Software en partes pequeñas, pero utilizables, llamadas incrementos. En general, cada incremento se construye sobre aquel que ya ha sido entregado.</li> <li>- Es iterativo.</li> </ul>

<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
<b>Modelo en Espiral</b>	<ul style="list-style-type: none"> <li>- Es un enfoque realista del Desarrollo del sistema, como el software evoluciona, a medida que progresa el proceso el Desarrollador y el Cliente comprenden y reaccionan mejor ante riesgos.</li> <li>- Utiliza la construcción de prototipos en cualquier etapa de evolución del producto</li> </ul>	<p>Requiere una considerable habilidad para la evaluación del riesgo, y cuenta con esta habilidad para el éxito</p>	<ul style="list-style-type: none"> <li>- Enfocada a la identificación de riesgos</li> </ul>
<b>Metodología RUP</b>	<ul style="list-style-type: none"> <li>- Desarrolla el Software de forma iterativa</li> <li>- Se documenta y gestiona los requerimientos</li> <li>- Utiliza arquitecturas basadas en componentes</li> <li>- Modela el Software visualmente</li> <li>- Verifica la calidad del Software</li> </ul>	<ul style="list-style-type: none"> <li>- Requiere conocimientos del proceso y de UML</li> <li>- Capacitar al equipo requiere de tiempo y consultoría</li> </ul>	<ul style="list-style-type: none"> <li>- Proceso dirigido por Casos de Uso.</li> <li>- Orientado a objetos.</li> <li>- Proceso iterativo e incremental.</li> <li>- Proceso centrado en la Arquitectura.</li> <li>- Recomendado para proyectos a largo plazo.</li> </ul>

<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
<b>Modelo MSF</b>	<ul style="list-style-type: none"> <li>- Vinculación con el cliente como también orientado al trabajo en equipo.</li> <li>- Tiene facilidad de soporte y mantenimiento.</li> <li>- Es adaptable, se puede utilizar para proyectos de cualquier magnitud.</li> <li>- Incentiva al trabajo en equipo y a la colaboración</li> </ul>	<ul style="list-style-type: none"> <li>- Al estar basado en tecnología Microsoft, trata de obligar a usar herramientas Microsoft.</li> <li>- Solicita demasiada documentación en sus fases.</li> <li>- Los precios de licencias, capacitación y soporte de Microsoft son caros.</li> </ul>	<ul style="list-style-type: none"> <li>- Soluciones tecnológicas, con menos personas y menos riesgo, pero con resultados de más calidad.</li> </ul>
<b>Metodología XP</b>	<ul style="list-style-type: none"> <li>- Programación organizada.</li> <li>- Menor tasa de errores.</li> <li>- El cliente tiene control sobre las prioridades.</li> <li>- Para proyectos donde los requerimientos cambian rápidamente.</li> <li>- Propicia la satisfacción del programador.</li> </ul>	<ul style="list-style-type: none"> <li>- Recomendado para proyectos a corto plazo.</li> <li>- Requiere de un rígido ajuste a los principios de XP.</li> </ul>	<ul style="list-style-type: none"> <li>- Desarrollo iterativo e incremental.</li> <li>- Pruebas unitarias continuas.</li> <li>- Programación en parejas.</li> <li>- Refactorización del código.</li> <li>- Propiedad del código compartida.</li> </ul>

<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
<b>Metodología SCRUM</b>	<ul style="list-style-type: none"> <li>- Se obtiene software lo más rápido posible y este cumple con los requerimientos más importantes.</li> <li>- Se trabaja en iteraciones cortas, de alto enfoque.</li> <li>- Se acepta que el cambio es una constante universal y se adapta el desarrollo para integrar los cambios que son importantes.</li> <li>- Permite producir software de una forma consistente, sostenida y competitiva.</li> <li>- Las reuniones diarias se dedican a inconvenientes recientes, evitando el estancamiento.</li> </ul>	<ul style="list-style-type: none"> <li>- Sólo funciona bien en equipos pequeños y ágiles.</li> <li>- Se requieren miembros del equipo experimentado.</li> </ul>	<ul style="list-style-type: none"> <li>- Equipos auto-organizado</li> <li>- Es un modo de desarrollo de carácter adaptable más que predictivo.</li> <li>- Orientado a las personas más que a los procesos.</li> <li>- Incremental basada en iteraciones y revisiones.</li> </ul>
<b>Metodología FDD</b>	<ul style="list-style-type: none"> <li>- El equipo de desarrollo no malgasta el tiempo y dinero del</li> </ul>	<ul style="list-style-type: none"> <li>- Falta de documentación del diseño.</li> </ul>	<ul style="list-style-type: none"> <li>- Se preocupa por la calidad, por lo que incluye</li> </ul>

<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
	<p>cliente desarrollando soluciones innecesarias.</p> <ul style="list-style-type: none"> <li>- Cada componente del producto final ha sido probado y satisface los requerimientos.</li> <li>- Rápida respuesta a cambios de requisitos a lo largo del desarrollo.</li> <li>- Entrega continua y en plazos cortos de software funcional.</li> <li>- Trabajo conjunto entre el cliente y el equipo de desarrollo.</li> <li>- Minimiza los costos frente a cambios.</li> <li>- Atención continuada a la excelencia técnica y al buen diseño.</li> </ul>	<ul style="list-style-type: none"> <li>- Problemas derivados de la comunicación oral.</li> <li>- Fuerte dependencia de las personas. Como se evita en lo posible la documentación y los diseños convencionales, los proyectos ágiles dependen críticamente de las personas.</li> <li>- La falta de documentación hacen difícil que pueda reutilizarse el código ágil.</li> </ul>	<p>un monitoreo constante del proyecto.</p> <ul style="list-style-type: none"> <li>- Propone tener etapas de cierre cada dos semanas. Se obtienen resultados periódicos y tangibles.</li> <li>- Se basa en un proceso iterativo con iteraciones cortas que producen un software funcional que el cliente y la dirección de la empresa pueden ver y monitorear.</li> </ul> <p>No hace énfasis en la obtención de los requerimientos sino en cómo se realizan las fases de diseño y construcción.</p>



<b>Metodología o modelo</b>	<b>Ventajas</b>	<b>Desventajas</b>	<b>Características</b>
<b>Metodología Iweb</b>	<ul style="list-style-type: none"> <li>- Entornos de aplicaciones dinámicos controlados por el usuario</li> </ul>	Limitado número de plantillas, aunque se pueden crear fácilmente.	<ul style="list-style-type: none"> <li>- Es incremental y evolutivo</li> <li>- Arquitectura especializada (exigencias en el diseño)</li> </ul>
<b>Metodología OOHDM</b>	<ul style="list-style-type: none"> <li>- La recuperación de la información puede realizarse sin problemas.</li> <li>- Se pueden crear enlaces entre nodos cualesquiera.</li> <li>- La modularidad y la consistencia se potencian.</li> <li>- Soporte a diferentes modos de acceso a la información.</li> </ul>	<ul style="list-style-type: none"> <li>- Genera una cantidad considerable de documentación a través de sus distintas etapas de desarrollo</li> <li>- El diseño navegacional es un tanto tedioso, para resolverlo adecuadamente.</li> </ul>	<ul style="list-style-type: none"> <li>- Separación clara entre lo conceptual, lo navegacional y lo visual.</li> <li>- Estudio profundo de los aspectos de interfaz.</li> <li>- Uso de la orientación a objetos y del diagrama de clases, para representar el aspecto de la navegación a través de las clases navegacionales.</li> </ul>

*Tabla 2 Ventajas y desventajas de las metodologías seleccionadas para el estudio: Fuente el autor*

Con la información recopilada, se organizó mediante una tabla los artefactos y entregables de cada fase para determinar cuáles son comunes y cuales únicos, dentro de cada metodología además de saber cuáles a pesar de llamarse diferente, cumplen la misma función dentro de una fase del proceso, como pueden ser métodos de obtención de requerimientos, codificación y otros

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA																
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM							
Análisis / Planeación																			
	Minuta de junta con el cliente	Documentos de especificación de requisitos	x	x		x	x	x		x	x				x				
	Listado de requisitos de usuario	Documento de visión	x							x		x							
	Plan de gestión del proyecto	Documento de misión	x		x					x									
	Listado de definición de requerimientos funcionales y no funcionales	Documento de requerimientos de calidad	x		x			x	x	x								x	
	Presupuesto	Documento de alcance	x							x	x			x					

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA												
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM			
	Requisitos de interfaz	Documento de análisis de riesgo			x			x							
	Casos de uso	Documento plan de proyecto				x		x		x			x		x
	Diagrama de casos de uso	Metáfora del sistema	x			x			x						x
	Modelo entidad relación	Formato del backlog del producto	x						x	x	x				x
	Cronograma de Actividades	Documento sprint backlog						x	x	x	x				
	Borrador del Plan Maestro del Proyecto	Documento del modelo general						x				x			
	Metáfora del Proyecto	Modelo de análisis							x				x		

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA												
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM			
	Historia de usuario	Diagramas de interacciones de usuario (UDIs)							x						x
	Diccionario de datos								x						
	Velocidad del proyecto								x						
	Plan de lanzamiento								x						
	Planificación del sprint									x					
	Burn Down (grafico de trabajo)									x					
	requisitos de contenidos											x			

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA										
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM	
	Escenarios de interacción de usuario											x	
Diseño													
	Definir arquitectura del software	Documento de diseño del software	x	x									
	Requerimientos de hardware y software	Modelo de arquitectura del software	x		x								
	Diseño Arquitectónico	Documentos de análisis de riesgos	x			x							
	Diseño Detallado	Diseño arquitectónico	x			x							



FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA													
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM				
	Tarjetas CRC	Estructura Navegacional							x							x
	Tarjetas de Ingeniería								x							
	Scrum Board									x						
	Diseño de Contenido														x	
	Diseño de Navegación														x	
	Diseño de Interfaz de Usuario														x	
Desarrollo																
	Prototipos	Prototipo del producto	x	x												
	Plan de Incremento	Producto software		x	x	x		x		x		x		x		

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA														
			Cascada	Incremental		Espiral	RUP		MSF		XP	SCRUM	FDD	Iweb	OOHDM		
	Código Fuente	Manuales de usuarios		x	x	x		x		x	x	x	x				
	Diagrama de Procesos	Plan de desarrollo				x		x									
	Junta de revisión	Manuales operativos								x	x						
	Plan de pruebas	Manuales de instalación								x	x						
	Plan de iteración	Manuales de desarrollo								x	x						
	Incremento	Aplicativo web										x	x		x		x
	Especificación de tareas													x			



FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA											
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM		
	Especificación de Escenarios												x	
	Diagrama de configuración													x
Pruebas														
	Pruebas de unidad	Documento de pruebas	x	x	x	x					x			
	Pruebas de integración		x		x							x		
	Pruebas del sistema		x		x							x		
	Pruebas de aceptación		x		x			x			X			
	Tareas de marketing				x							x		

FASES	ARTEFACTOS	ENTREGABLE	MODELO O METODOLOGIA											
			Cascada	Incremental	Espiral	RUP	MSF	XP	SCRUM	FDD	Iweb	OOHDM		
	Capacitaciones de usuarios			x				x						
	Pruebas piloto							x						
	Reporte de correcciones											x		
Implementación														
	Casos de prueba						x		X				x	
	Acta de entrega y finalización del proyecto						x	x						
	Documento implantación completa							x						

Tabla 3 Artefactos y entregables Vs Metodología: Fuente el autor

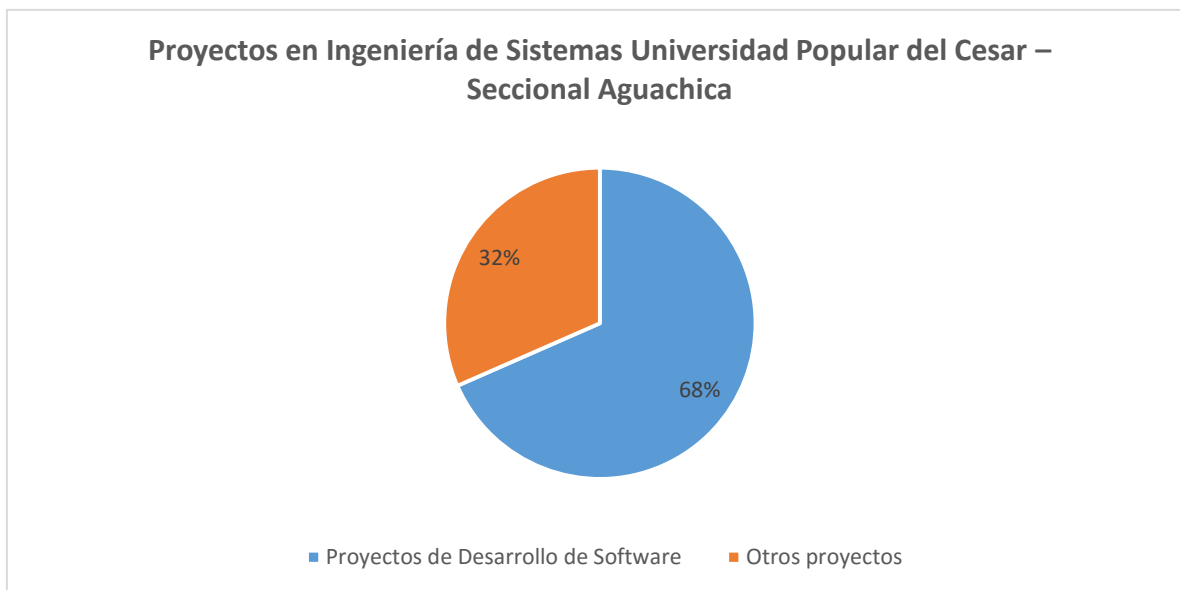
### 2.2.3.2 Análisis del Desarrollo de Software en los proyectos de la Universidad

Además de buscar en la literatura los modelos y metodologías más documentadas se analizaron las más usadas en los proyectos de grado en los que se desarrolló software en la Universidad Popular del Cesar Seccional Aguachica, el análisis se hizo en dos fases.

Una primera fase consistió en buscar la lista de proyectos de grado (ver anexo B) en la biblioteca de la seccional y analizar los títulos de los trabajos, si el título sugería desarrollo de software se marcaba para su posterior análisis, luego se procedía en una segunda fase, a revisar la metodología que se había empleado en cada uno de los proyectos, de este ejercicio se obtuvo los siguientes resultados.

<b>Proyectos en Ingeniería de Sistemas Universidad Popular del Cesar – Seccional Aguachica</b>	
Proyectos de Desarrollo de Software	39
Otros proyectos	18
Total Proyectos	57

*Tabla 4 Distribución de los Proyectos de grado en Ingeniería de Sistemas Universidad Popular del Cesar – Seccional Aguachica: Fuente el autor*



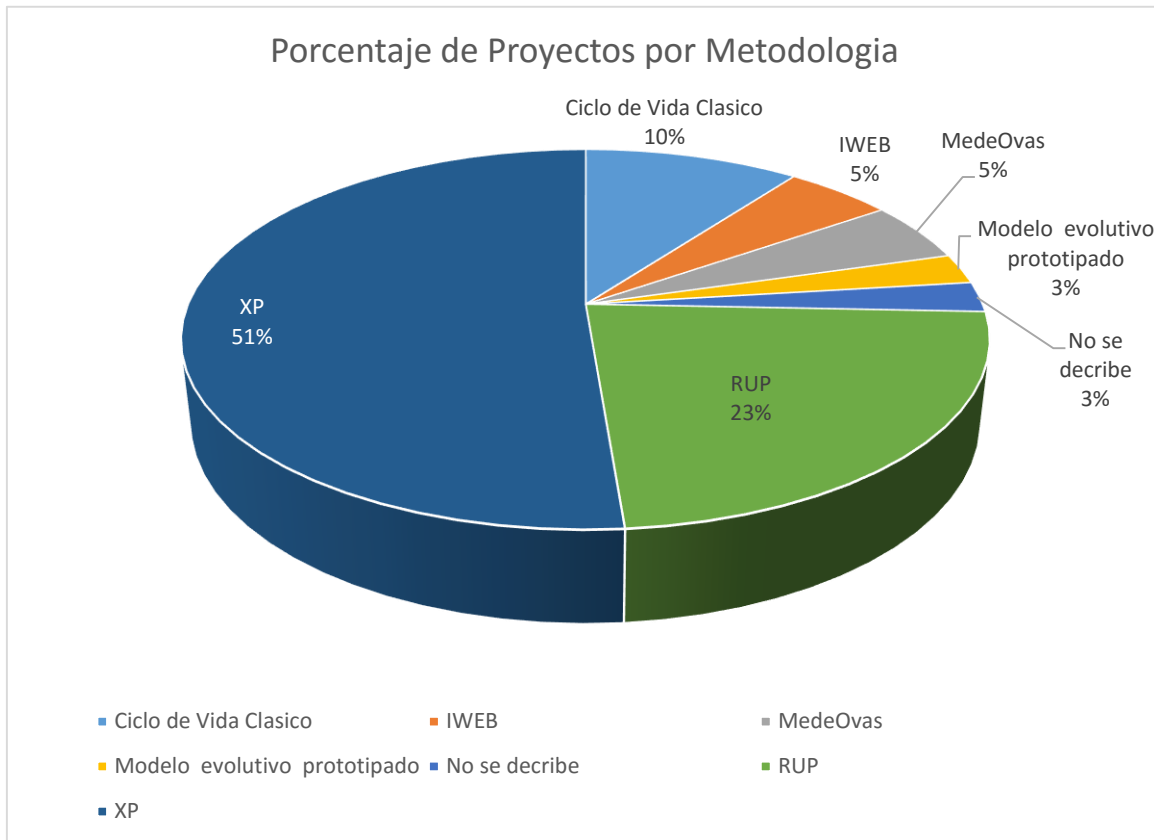
*Figura 20 Análisis de Proyectos de grado en Ingeniería de Sistemas Universidad Popular del Cesar –  
Seccional Aguachica*

De esta gráfica se puede deducir que la opción de grado preferida por los estudiantes de Ingeniería de Sistemas en la Universidad Popular del Cesar – Seccional Aguachica es el Desarrollo de Software con casi un 70%.

De la segunda fase de este análisis, la revisión de las metodologías de desarrollo, se obtuvieron los siguientes resultados.

<b>METODOLOGIA</b>	<b>CANTIDAD</b>
Ciclo de Vida Clásico	4
IWEB	2
MedeOvas	2
Modelo evolutivo prototipado	1
No se describe	1
RUP	9
XP	20
<b>Total Proyectos</b>	<b>39</b>

*Tabla 5 Metodologías más usadas en proyectos de grado*



*Figura 21 Porcentaje de proyectos por metodología: Fuente el autor*

Como se observa en la gráfica las metodologías más usadas en los proyectos de desarrollo de software en la Universidad Popular del Cesar – Seccional Aguachica son la Programación Extrema (51%) y el Proceso Racional Unificado (23%), por lo que son las metodologías escogidas para el estudio.

### *2.2.3.3 Análisis de trabajos de grado a la luz del Estándar IEEE 1074*

Los proyectos de software en un ambiente académico, son particulares de muchas formas. Se propuso analizar el estándar IEEE 1074 Estándar para el desarrollo del

modelo del ciclo de vida del software<sup>129</sup> (IEEE, 1997) y comparar con la forma en que se cumplen o se aplican (o no se aplican) las actividades de los grupos de proceso, para determinar en qué actividades se debía centrar la guía de configuración mínima de proceso de software, en un proyecto que usa XP, o RUP como metodología de desarrollo.

El estándar IEEE 1074 está dividido en grupos de procesos, de gestión del proyecto, orientados al desarrollo y los procesos integrales. Se realizó un mapeo con cada uno de estos grupos y se analizó como se realizan las actividades de cada grupo de procesos y el formato en el que se cumple esta tarea (o si no se cumple) en un proyecto de software en un trabajo de grado en el programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica.

#### 2.2.3.4 *Procesos de Gestión del Proyecto*

Los procesos de gestión del proyecto del estándar IEEE 1074 son Iniciación, seguimiento y control, gestión de la calidad, los procesos y sus actividades se presentan en esta tabla.

<b>Proceso/Actividad IEEE 1074</b>	<b>Actividad / Formato UPC</b>
<b>Iniciación</b>	
Crear Procesos de Software Ciclo de Vida	N/A
Realizar estimaciones	N/A
Asignar los recursos del proyecto	N/A
definir Métricas	N/A
<b>Seguimiento y Control</b>	

---

<sup>129</sup> Standard For Developing Software Life Cycle Processes. Las actividades que constituyen los procesos necesarios para el desarrollo y el mantenimiento de software, ya sea parte de un sistema mayor o autónomo (stand-alone) y Los procesos de gestión y de soporte a lo largo de todo el ciclo de vida

Manejo de riesgos	N/A
Gestión del proyecto	N/A
Identificar las necesidades de mejoras SLCP	N/A
Almacenar los registros	N/A
Recopilar y analizar datos de gestión	N/A
<b>Gestión de la Calidad</b>	
Plan de evaluación	Evaluación de los formatos de propuesta, formato 2 (anteproyecto) y formato 3 (documento final), definidos por la Universidad.
Plan de gestión de la configuración	N/A
Plan de transición del sistema	N/A
Plan de instalación	N/A
Plan de documentación	N/A
Plan de capacitación	N/A
Plan de gestión del proyecto	N/A
Plan de integración	N/A

#### 2.2.3.5 Procesos Orientados al Desarrollo de Software

El estándar IEEE 1074 está dividido en grupos de procesos, se realizó un mapeo con cada uno de estos grupos y se analizó como se realizan las actividades de cada grupo, en los proyectos de desarrollo de la Universidad Popular del Cesar Seccional Aguachica.

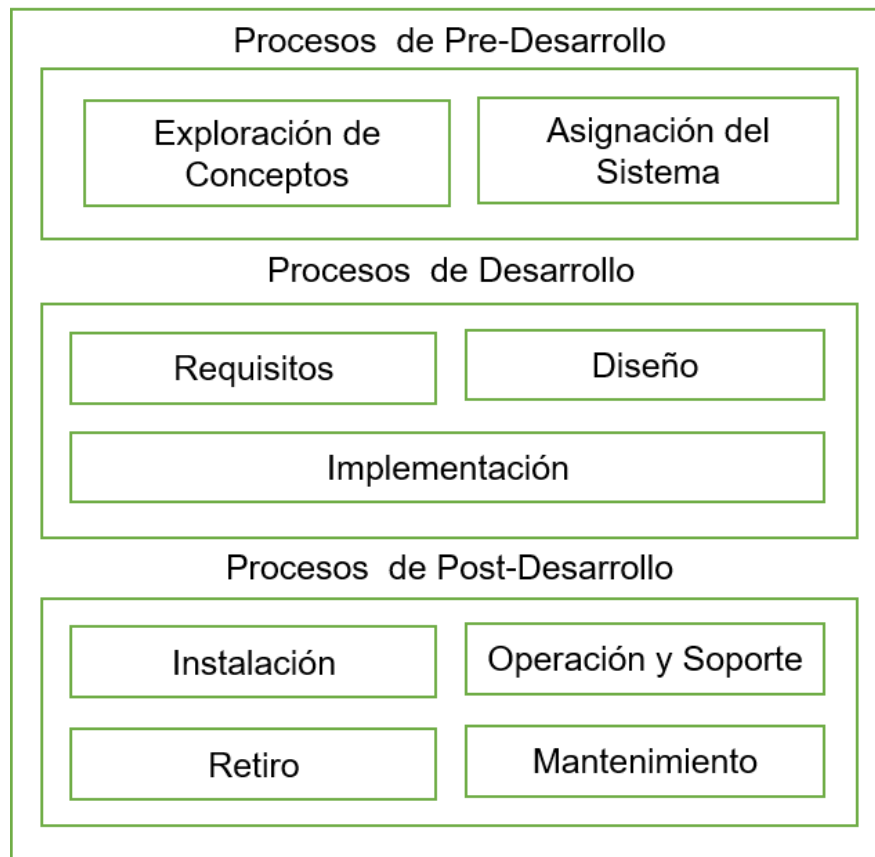


Figura 22 IEEE-1074 Grupos de Procesos Orientados al desarrollo

### Procesos Pre – Desarrollo

Son los procesos que deberían ser ejecutados antes de que el desarrollo de software inicie. El grupo de procesos de pre desarrollo considerados por el estándar IEEE 1074 son:

- Exploración de conceptos
- Asignación del Sistema
- Adquisiciones de Software

Se consideró no tomar en cuenta este grupo de procesos para la investigación, porque las actividades que las componen se cumplen en los documentos Propuesta y Formato 2 del proyecto de grado reglamentado por la Universidad.



En la tabla se observa la dinámica de estos procesos en la universidad

<b>Proceso/Actividad IEEE 1074</b>	<b>Actividad / Formato UPC</b>
<b>Exploración de conceptos</b>	
• Identificar ideas y necesidades.	Propuesta de proyecto de grado
• Formular soluciones potenciales.	Anteproyecto
• Realizar estudios de factibilidad.	
• Refinar y finalizar la idea o necesidad.	Evaluación del anteproyecto
<b>Asignación del Sistema</b>	
• Analizar las Funciones	Anteproyecto
• Desarrollar Arquitectura del Sistema	Formato 3 de acuerdo con la metodología a usar
• Descomponer Requisitos del sistema	Formato 3 de acuerdo con la metodología a usar
<b>Adquisiciones de Software</b>	
• Identificar los Requisitos del Software a Adquirir	Si aplica en Formato 3 de acuerdo con la metodología a usar
• Evaluar Fuentes de Adquisición de Software (si corresponde)	N/A
• Definir Método de Adquisición de Software (si corresponde)	N/A
• Adquisición de Software (si corresponde)	N/A

*Tabla 6 Procesos de Predesarrollo Vs Formato UPC: Fuente el autor*

### **Procesos de Desarrollo**

Este es el grupo de actividades que se realizan durante el desarrollo de un producto software. En este grupo de procesos se centrara el proyecto por es donde al estudiante define el proceso software que usara para la construcción del software

<b>Proceso/Actividad IEEE 1074</b>	<b>Actividad / Formato UPC</b>
Requisitos	
<ul style="list-style-type: none"> <li>Definir y desarrollar requerimientos de Software</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Definir los requerimientos de interfaz</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Priorizar e integrar requisitos de Software</li> </ul>	Formato 3 de acuerdo con la metodología a usar
Diseño	
<ul style="list-style-type: none"> <li>Realizar diseño arquitectónico</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Diseño de Base de datos (si corresponde)</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Diseño de Interfaces</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Realizar el diseño detallado</li> </ul>	Formato 3 de acuerdo con la metodología a usar
Implementación	
<ul style="list-style-type: none"> <li>Crear código ejecutable</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Crear documentación de operación</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>Realizar la integración</li> </ul>	Formato 3 de acuerdo con la metodología a usar

*Tabla 7 Procesos de Desarrollo Vs Formato UPC*

### **Procesos de Post – Desarrollo**

Estos son los grupos de actividades que se realizan para instalar, operar, apoyar, mantener y retirar un producto software (IEEE 1074, 1998)

Todos procesos muy importantes en la producción industrial de software, pero que sin embargo en el entorno académico en que se desenvuelven los proyectos en la Universidad, muchas veces innecesarios. Por tanto para el estudio solo se tendrá en cuenta el proceso de instalación.

Esto debido a que un proyectos software terminado, no siempre se instala por diversas razones, y de instalarse, no hay un compromiso de mantenimiento o retiro por parte de la Universidad o del futuro profesional, que de hacerlo sería de forma privada en el ejercicio de su profesión y sin ningún control de parte de la Universidad.

Proceso/Actividad IEEE 1074	Actividad / Formato 3 UPC
Instalación	
<ul style="list-style-type: none"> <li>Distribuir el Software</li> </ul>	Formato 3 de acuerdo con la metodología a usar (en los casos que aplique)
<ul style="list-style-type: none"> <li>Instalar el Software</li> </ul>	Formato 3 de acuerdo con la metodología a usar (en los casos que aplique)
<ul style="list-style-type: none"> <li>Aceptación del Software en un Entorno Operacional</li> </ul>	Formato 3 de acuerdo con la metodología a usar
Operación y soporte	
<ul style="list-style-type: none"> <li>Operar el sistema</li> </ul>	Capacitación inicial
<ul style="list-style-type: none"> <li>Proveer asistencia técnica y consultoría</li> </ul>	N/A
<ul style="list-style-type: none"> <li>Mantener registros de solicitud de apoyo</li> </ul>	N/A
Mantenimiento	
<ul style="list-style-type: none"> <li>Identificar necesidades de mejora del software</li> </ul>	N/A

<ul style="list-style-type: none"> <li>• Implementar método de reporte de problemas</li> </ul>	N/A
<ul style="list-style-type: none"> <li>• Reaplicación de SLC</li> </ul>	N/A
Retiro	
<ul style="list-style-type: none"> <li>• Notificar a los Usuarios</li> </ul>	N/A
<ul style="list-style-type: none"> <li>• Realizar operaciones paralelas (si corresponde)</li> </ul>	N/A
<ul style="list-style-type: none"> <li>• Retiro del sistema</li> </ul>	N/A

Tabla 8 Procesos de Postdesarrollo Vs Formato UPC: Fuente el autor

### 2.2.3.6 Procesos integrales

Los procesos integrales son las actividades que se necesitan para llevar a cabo las actividades del proyecto. Estas actividades se utilizan para asegurar la terminación y calidad de las funciones del proyecto.

<b>Proceso/Actividad IEEE 1074</b>	<b>Actividad / Formato UPC</b>
Validación y Verificación	
<ul style="list-style-type: none"> <li>• Llevar a cabo revisiones</li> </ul>	Por el comité de proyectos(solo propuesta), el director y evaluadores (Formatos 2 y 3)
<ul style="list-style-type: none"> <li>• Crear matriz de trazabilidad</li> </ul>	N/A
<ul style="list-style-type: none"> <li>• Realizar las auditorías</li> </ul>	N/A
<ul style="list-style-type: none"> <li>• Desarrollar procedimientos de prueba</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>• Crear datos de prueba</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>• Ejecutar pruebas</li> </ul>	Formato 3 de acuerdo con la metodología a usar
<ul style="list-style-type: none"> <li>• Resultados de la evaluación informe</li> </ul>	Por el comité de proyectos(solo propuesta), el director y evaluadores (Formatos 2 y 3)
Gestión de la Configuración	

• Desarrollar la identificación de la configuración	N/A
• Realizar Control de configuración	N/A
• Realizar la contabilidad del estado	N/A
Desarrollo de la Documentación	Desarrollar los manuales de usuario
• Implementar la documentación	Formato 3 de acuerdo con la metodología a usar
• Producir y distribuir la documentación	Formato 3 de acuerdo con la metodología a usar (si aplica)
Formación	Capacitación inicial
• Desarrollar el material de formación	Formato 3 de acuerdo con la metodología a usar (si aplica)
• Validar el programa de formación	N/A
• Implementar el programa de formación	Formato 3 de acuerdo con la metodología a usar (si aplica)

*Tabla 9 Procesos Integrales Vs Formato UPC: Fuente el autor*

Este análisis fue importante para entender el contexto de los proyectos de grado que involucran desarrollo de software en la Universidad Popular del Cesar Seccional Aguachica y para determinar que procesos eran realmente relevantes para esta investigación. En este caso como se observa en las tablas se concluyó que el grupo de proceso más importante para la investigación son los **Orientados al desarrollo**

#### *2.2.3.7 Otras consideraciones*

Teniendo en cuenta el resultado del análisis de los proyectos y el estándar IEEE 1074, además se tuvo en cuenta la información recogida a través de una entrevista no estructurada, informal a un grupo de docentes, en donde se conversó, sobre el tiempo que debería demorar un proyecto de grado, el procedimiento de la universidad para los proyectos de grado, ¿cómo determinar el alcance de un proyecto de desarrollo? y ¿cuántos integrantes debe tener el equipo o proyecto?

Las conclusiones a las que se llegó luego de la entrevista son las siguientes:

1. El programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica, define en su malla curricular dos materias, proyecto de grado I (propuesta y formato 2) y proyecto de grado II (proyecto, formato 3) (ver anexo F), para ser cursadas en 9 y 10 semestre respectivamente, o sea que un estudiante debería terminar su proyecto de grado en 1 año, tiempo estimado por la universidad para esta actividad. De esto se puede concluir que el tiempo para el desarrollo de un proyecto de software, como trabajo de grado sería de 6 meses, lo que implica una restricción al alcance que debe ajustarse a este tiempo.

Sin embargo la Universidad da un margen de dos años para que completen su proceso de grado.

2. El comité de proyectos de Ingeniería de Sistemas es quien evalúa las propuestas y determina la viabilidad técnica y el alcance del proyecto a partir de la propuesta presentada. Este comité aprueba o recomienda cambios que aporten a ser más viable el proyecto
3. Los estudiantes en compañía de su director son los encargados de proponer el tamaño de equipo que usaran en su proyecto, los estudiantes exponen ante el comité de proyectos y justifican el equipo propuesto. No obstante el comité es quien debe decidir si lo permite o no.

### 3 Guía para la configuración mínima de proceso software

En este capítulo se describe la guía de configuración mínima de procesos software (ver Anexo E), realizada para que los estudiantes del programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica, puedan adaptar el proceso software que usaran en el desarrollo de su trabajo de grado, cuando este implique un software como producto principal.

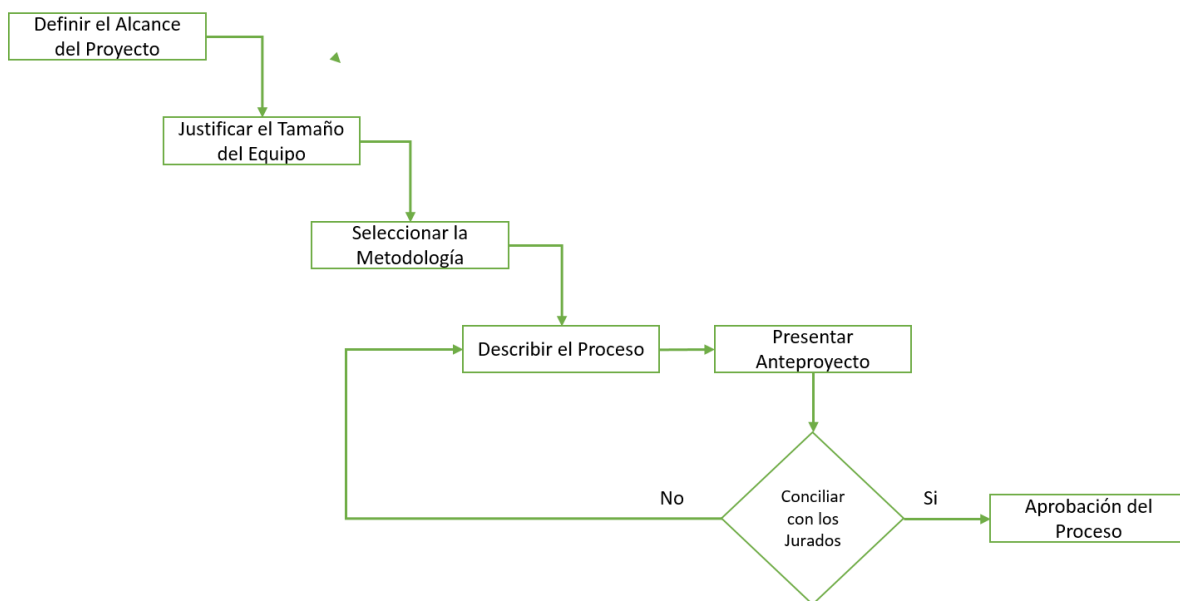


Figura 23 Procedimiento de la guía de configuración de procesos software

#### 3.1 Objetivo y alcance

Los objetivos que se pretenden alcanzar con esta guía son:

- Proporcionar a estudiantes, directores y jurados de proyectos de grado del programa ingeniería de sistemas de la Universidad popular del cesar seccional Aguachica una guía para configurar el proceso software que se usara en su proyecto con la cantidad mínima elementos que pueda contener.

- Definir un marco común de evaluación, para estudiantes, directores y jurados de proyectos de grado

La guía Inicia con la definición del alcance del proyecto software, para lo cual se usa el formato de **definición del alcance del proyecto**, que ayuda a justificar el tamaño del equipo a partir del número de funcionalidades a desarrollar (requisitos funcionales), por parte del estudiante y su director, hasta realizar el proceso de conciliación de estudiantes, directores y jurados de proyectos de grado.

### **3.2 Responsable**

El responsable de garantizar la adecuada aplicación del presente procedimiento es el Coordinador del programa de ingeniería de sistemas de la Universidad popular del cesar seccional Aguachica. Como responsable de aplicar o definir los lineamientos para la **presentación de proyectos de grado**

### **3.3 Definiciones**

Estas son las definiciones de términos encontrados de forma frecuente en la Guía de configuración mínima de proceso software.

#### **Trabajo de grado**

Una de las modalidades de grado, que ofrece la universidad a los estudiantes, consiste en una serie de documentos, donde recopilan la información y procedimientos empleados en la investigación de un tema específico o la solución de un problema.

#### **Software**

(INTECO, 2009) En su informe cita “el Estándar IEEE 610 define el software como programas, procedimientos y documentación y datos asociados, relacionados con la operación de un sistema informático”



**Modelo de ciclo de vida**

(Jiménez A. , 2013) El término modelo de ciclo de vida se refiere a un conjunto de fases por las que pasa un producto software desde que nace la idea hasta que es reemplazado o retirado.

**Proceso software**

(Pressman, 2005) El proceso software, puede definirse como una colección de patrones que definen un conjunto de acciones, tareas de trabajo o comportamientos relacionados, que requiere el desarrollo de un software.

**Artefacto**

El termino artefacto, denota cualquier producto, derivado de alguna actividad del proceso de software.

**Entregable**

(INTECO, 2009) Son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.

**Metodología**

(INTECO, 2009) Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental...). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

## **RUP**

(Martínez & Martínez, 2002) El Proceso Unificado de Rational es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles.

## **XP**

(Letelier & Penadés, 2006) Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

### **3.4 Limitaciones**

La guía de configuración mínima de proyectos de software ha sido diseñada para un entorno académico, específicamente los trabajos de grado que incorporen software como producto principal. No ha sido probada ni pretende ser probada en entornos de producción.

### **3.5 Contenido**

La guía de configuración mínima de proyectos de software contempla las actividades relacionadas en la siguiente tabla.

N° DE ACTIVIDAD	ACTIVIDAD	RESPONSABLE
1	Determinar el alcance del software a desarrollar	Estudiante/director de trabajo de grado
2	justificar el tamaño del equipo	Estudiante/director de trabajo de grado
3	seleccionar la metodología a usar en el desarrollo	Estudiante/director de trabajo de grado
4	Describir el uso de la metodología en términos de: actividades, roles, artefactos y herramientas/ tecnología	Estudiante/director de trabajo de grado
5	Presentación del formato de anteproyecto (formato 2)	Estudiante/Jurados de trabajo de grado
6	Conciliar el proceso definido con los jurados	Estudiante/Jurados de trabajo de grado
7	Aprobación de configuración mínima de proceso de software	Jurados de trabajo de grado

*Tabla 10 actividades guía de configuración mínima de proyectos de software. Fuente: el autor*

Las actividades se describen a continuación:

### **1. Determinar el alcance del software a desarrollar**

El alcance del proyecto se debe determinar teniendo en cuenta un límite de tiempo que los profesores por consenso en conversaciones informales (entrevista no estructurada) determinaron en 6 meses para el proceso de desarrollo, a partir de la aprobación del anteproyecto.

El alcance del proyecto se determinara a partir de la información recolectada con el formato de **definición del alcance del proyecto** (ver anexo D), de

acuerdo a los objetivos del proyecto a desarrollar o con base a los requerimientos, el estudiante debe proyectar de manera aproximada, consensuada con su director y proponer un tamaño de alcance entre pequeño, mediano y grande.

Esta propuesta después será aceptada o no por el comité de investigación del programa ingeniería de sistemas, quien es el encargado de evaluar las propuestas. Este deba dar el visto bueno para que el proyecto siga su curso.

## **2. Justificar el tamaño del equipo**

Una vez que se tiene definido el alcance del proyecto se puede de esta forma justificar el tamaño del equipo (personal) que planea emplear siendo un proyecto pequeño viable para un estudiante, mediano dos estudiantes, grande tres o más estudiantes, para no extender el tiempo del proyecto.

## **3. Seleccionar la metodología**

Como siguiente paso el estudiante debe seleccionar una metodología de desarrollo, en el caso particular de la Universidad y para esta guía, se ha optado por usar solo XP y RUP. De estas dos metodologías, el estudiante debe seleccionar una y se deben analizar las actividades, sus artefactos, roles y herramientas o tecnología que pueden ser usados.

Para esto el estudiante puede valerse de los anexos de la guía.

## **4. Describir el uso de la metodología en términos de: actividades, roles, artefactos y herramientas/ tecnología**

Luego de un estudio concienzudo se debe seleccionar los artefactos para definir el proceso de software, personalizado y a la medida del proyecto a desarrollar.

Se utilizó una matriz de correlación en la que el estudiante puede seleccionar las actividades genéricas del proceso, con esta lista de actividades el estudiante de común acuerdo con su director podrá definir el proceso solo con los elementos, artefactos y entregables que se consideren convenientes<sup>130</sup>,

La guía pretende que el estudiante configure su proceso software considerado como una secuencia de actividades en las que intervienen personas y herramientas/tecnología guiadas por un flujo (análisis, diseño, implementación), sin pretender decir cómo o cuando debe hacerlo.

El formato de selección de actividades que se propone consta de una lista de chequeo donde el estudiante puede escoger, los artefactos por flujo así:

<b>Selección de Actividades/Entregables</b>		
<b>Actividad/Entregable</b>	<b>Rol</b>	<b>Herramientas</b>
<b>Análisis</b>		
Historias de usuario	Cliente	Formato historias de usuario, Word
Metáfora del sistema	Cliente, equipo de desarrollo	Formato metáfora del Sistema, Word
Priorizar las actividades	Equipo de desarrollo, cliente	Historias de usuario
Calculo de Velocidad del proyecto	Equipo de desarrollo	Excel
Plan de Iteraciones	Equipo de desarrollo	Word, Excel
Plan de entregas	Equipo de desarrollo	Word, Excel

---

<sup>130</sup> También se podrían incluir elementos que sin ser de la metodología, se consideren convenientes para el proyecto

<b>Diseño</b>		
Tarjetas CRC	Director, Responsable, Equipo de desarrollo	Formato Tarjetas CRC, Excel, Word
Tareas de Ingeniería	Director, Responsable, Equipo de desarrollo	Formato Tarea de ingeniería, Excel, Word
<b>Implementación</b>		
Seguir plan de pruebas	Rastreador, probador	Formato Plan de Pruebas, Word
Realizar seguimiento al proyecto	Rastreador	Cronograma, plan de lanzamientos, plan de iteraciones
Refactorización	Programador	IDE seleccionado
Historia de iteración	Programador	Word
Escribir Código fuente	Programador	IDE
Pruebas unitarias	Programador	Módulo de pruebas automáticas
Realizar pruebas de Integración	Probador	IDE, Plan de Pruebas
Escribir manuales de usuario	Programador	Editor HTML, Word
Seguir casos de prueba	Rastreador	Listas de chequeo
Realizar Pruebas Aceptación	Probador	Plan de pruebas, listas de chequeo
Entregar Producto Software	Director	Formato de Acta de entrega, Word
<b>Descripción de las tareas de soporte del proceso</b>		

<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Estudiante</p>	<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Director</p>
<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Evaluador 1</p>	<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Evaluador 2</p>

*Tabla 11 Formato de Selección de Configuración del Proceso Software: Fuente el autor*

### **5. Presentación del formato de anteproyecto (formato 2)**

Luego de seleccionar los componentes que serán parte de su proceso de software, el estudiante podrá anexar dicho formato al formato 2 de la Universidad como anexo de la metodología de desarrollo que eligió usar en su trabajo de grado.

### **6. Conciliar el proceso definido con los jurados**

Los elementos escogidos, luego deben ser presentados por el estudiante, como directo responsable de su trabajo de grado a los evaluadores del mismo, para llegar a un consenso donde el estudiante pueda defender su propuesta de proceso software y los jurados puedan expresar que consideran ellos que no puede faltar, de esta forma los estudiantes, directores y evaluadores sabrán exactamente qué y cuales cosas deben hacer o evaluar; porque han sido previamente acordadas. Esto favorece notablemente el proceso académico y disminuye el tiempo que el estudiante tarda en terminar su proyecto porque sabe que cosas debe entregar, además evita que se le pidan más cosas después de las revisiones evitando ambigüedades en la evaluación.

Sin embargo el estudiante debe especificar como llevara a cabo las iteraciones y tareas adicionales que soporten el uso de los artefactos escogidos.

## **7. Aprobación de configuración mínima de proceso de software**

Después de llegar a un acuerdo entre estudiantes y evaluadores, las partes deben firmar el formato mencionado, para formalizar la aceptación del proceso software propuesto como válido.

Cabe mencionar que la guía podría ser aplicada a cualquier metodología sin embargo el estudio y construcción de la guía se basó en RUP y XP por ser las más comunes dentro de los proyectos de grado analizados.

### **3.6 Anexos**

A Descripción de la metodología XP

B Descripción de la metodología RUP

C Formato de definición de alcance del proyecto

### **3.7 Validación**

Como ya se ha mencionado en el documento la guía de selección de configuración mínima de proceso software se validó mediante el juicio de expertos, según el método de agregados individuales<sup>131</sup> para lo cual se diseñó un instrumento, para recolectar la información dada por los expertos y poder tabular y promediar las

---

<sup>131</sup> Se pide individualmente a cada experto que dé una estimación directa de la probabilidad de éxito o de fracaso en cada una de las tareas descritas. Después se tratan estadísticamente los datos recogidos. Lo habitual es calcular la media aritmética del conjunto de estimaciones individualmente obtenidas, para cada tarea. Esto se hace presuponiendo que el conjunto de los datos posibles tiene una distribución simétrica y, por tanto, la media aritmética es un buen índice de tendencia central.



conclusiones de cada uno de los expertos para obtener un resultado final (ver Anexo C), los expertos fueron escogidos por su experiencia en la evaluación de proyectos de grado de la Universidad Popular del Cesar Seccional Aguachica, que tengan desarrollo de software como producto principal.

### 3.7.1 Elaboración del instrumento de validación

Los pasos usados para realizar la validación fueron los siguientes, un primer paso fue la elaboración del instrumento que recogería la opinión de los expertos, para lo cual se tuvieron en cuenta los siguientes factores, **suficiencia**, **relevancia** y **claridad** y luego se pensó en la forma de evaluar cada aspecto.

Teniendo en cuenta la **suficiencia** como la capacidad de cumplir con el aspecto evaluado y en qué medida cumple, la **relevancia** como la importancia que la guía tendrá en el desarrollo de los proyectos de grado es decir si su impacto es alto o bajo o la cantidad de esfuerzo que deberá ser aplicado, para que sea alto y la **claridad** como la capacidad de ser usada sin una capacitación previa, sino a partir de la misma descripción de la guía, es decir si la guía es lo suficientemente sencilla para ser usada sin problemas.

### 3.7.2 Escala de juicios valorativos

Además se diseñó la escala valorativa que se usaría, esta escala es de tipo cualitativo sin embargo se les asignó una equivalencia numérica para facilitar los cálculos estadísticos. La escala va de 0 a 4 siendo 0 no cumple y 4 cumple completamente, en la tabla se muestran la escala valorativa completa.

<b>Escala Valorativa</b>	
<b>Valor Cualitativo</b>	<b>Numérico</b>
No cumple	0
Cumple con muchas modificaciones	1
Cumple con modificaciones	2

Cumple con modificaciones menores	3
Cumple completamente	4

*Tabla 12 Escala de juicios valorativos: Fuente el autor*

### 3.7.3 Procedimiento de evaluación

Como información complementaria, se le agrego al formato, el objetivo de la investigación, el contexto al que va dirigido el proyecto y explicación de la escala de juicios de valor.

### 3.7.4 Validación del instrumento

Para lograr una mayor objetividad y validez del instrumento, este fue revisado<sup>132</sup> por la Mg© Katherine Beleño Caselles, Ingeniera de Sistemas después de considerar su experiencia y conocimientos en métricas y calidad de software.

### 3.7.5 Selección de los expertos

Luego, se debió escoger las personas que se consideraran expertas en el tema. (Escobar Pérez & Cuervo Martínez, 2006) La identificación de las personas que Formarán parte del juicio de expertos es una parte crítica en este proceso<sup>133</sup>.

Para realizar esta labor se seleccionaron a las siguientes personas; considerándolas como expertos, por su participación activa en la dirección y evaluación de proyectos de grado en diversas universidades incluyendo a la

---

<sup>132</sup> (López & Sandoval) Una vez se ha diseñado y estructurado el cuestionario, es requisito fundamental probarlo en una pequeña muestra de la población de estudio, la muestra no necesariamente debe ser aleatoria, más bien podría ser con criterio experto o por conveniencia.

<sup>133</sup> (Skjong, Wentworht, 2000) proponen los siguientes criterios de selección: (a) Experiencia en la realización de juicios y toma de decisiones basada en evidencia o experticia (grados, investigaciones, publicaciones, posición, experiencia y premios entre otras), (b) reputación en la comunidad, (c) disponibilidad y motivación para participar, y (d) imparcialidad y cualidades inherentes como confianza en sí mismo y adaptabilidad

Universidad Popular del Cesar – Seccional Aguachica así como también sus logros académicos.

<b>Nombre</b>	<b>Ha evaluado proyectos en</b>	<b>Formación</b>
Darwin Navarro Pino	Universidad Popular del Cesar – Seccional Aguachica, Universidad de Santander	Maestría
Cesar Ramos Pineda	Universidad Popular del Cesar – Seccional Aguachica, Corporación Centrosistemas	Maestría
Edwin Barrientos	Universidad Popular del Cesar – Seccional Aguachica, Universidad Francisco de Paula Santander	Maestría
Wilson Castaño	Universidad Popular del Cesar – Seccional Aguachica, Universidad de Santander	Maestría

*Tabla 13 Relación de expertos, experiencia y formación: Fuente el autor*

### 3.7.6 Aplicación del instrumento de validación

Un tercer paso fue la explicación de cada factor y de la escala valorativa que se usaría a los expertos, y se procedió a que diligenciaran el formulario.

Después se tabulo y promedio las conclusiones de cada uno de los expertos para obtener un resultado final. Los factores evaluados y los puntajes obtenidos, así como el promedio del juicio de los expertos se pueden apreciar en las siguientes tablas.

<b>Aspectos a Evaluar</b>	<b>Valoración</b>				
	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>Suficiencia de la Guía</b>					
Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)					4
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología				3	

La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software					4
La guía abarca los elementos necesarios de cada metodología para poder realizar un proyecto de desarrollo				3	
	4				
<b>Relevancia de la Guía</b>					
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software				3	
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.					4
La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que cosas debe entregar en su trabajo de grado				3	
La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfocados a desarrollo de software				3	
	3				
<b>Claridad de la Guía</b>					
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional					4
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía		1			
Se exponen claramente las actividades, entregables y artefactos que se seleccionan en la guía				3	
	3				
<b>Puntaje Total</b>	<b>3</b>				

Tabla 14 Tabulación del formato del experto 1: Fuente el autor

Factor	Promedio	Juicio valorativo
<b>Suficiencia</b>	4	
<b>Relevancia</b>	3	Cumple con modificaciones menores
<b>Claridad</b>	3	Cumple con modificaciones menores
<b>Total</b>	3	Cumple con modificaciones menores

Tabla 15 Promedio de resultados experto 1: Fuente el autor

Cuestionario					
Aspectos a Evaluar	Valoración				
	0	1	2	3	4
<b>Suficiencia de la Guía</b>					

Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)				3
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología				4
La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software				4
La guía abarca los elementos necesarios de cada metodología para poder realizar un proyecto de desarrollo	2			
	3			
<b>Relevancia de la Guía</b>				
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software				4
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.				4
La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que cosas debe entregar en su trabajo de grado			3	
La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfocados a desarrollo de software		2		
	3			
<b>Claridad de la Guía</b>				
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional			3	
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía	0			
Se exponen claramente las actividades, entregables y artefactos que se seleccionan en la guía				4
	2			
<b>Puntaje Total</b>	<b>3</b>			

Tabla 16 Tabulación del formato del experto 2: Fuente el autor

Factor	Promedio	Juicio valorativo
Suficiencia	3	Cumple con modificaciones menores
Relevancia	3	Cumple con modificaciones menores
Claridad	3	Cumple con modificaciones menores
<b>Total</b>	3	Cumple con modificaciones menores

Tabla 17 Promedio de resultados experto 2: Fuente el autor

<b>Cuestionario</b>					
<b>Aspectos a Evaluar</b>	<b>Valoración</b>				
	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>Suficiencia de la Guía</b>					
Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)				3	
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología				3	
La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software					4
La guía abarca los elementos necesarios de cada metodología para poder realizar un proyecto de desarrollo				3	
	<b>3</b>				
<b>Relevancia de la Guía</b>					
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software					4
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.					4
La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que cosas debe entregar en su trabajo de grado				3	
La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfocados a desarrollo de software				3	
	<b>4</b>				
<b>Claridad de la Guía</b>					
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional					4
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía			2		
Se exponen claramente las actividades, entregables y artefactos que se seleccionan en la guía					4
	<b>3</b>				
<b>Puntaje Total</b>	<b>3</b>				

Tabla 18 Tabulación del formato del experto 3: Fuente el autor

<b>Factor</b>	<b>Promedio</b>	<b>Juicio valorativo</b>
<b>Suficiencia</b>	3	Cumple con modificaciones menores
<b>Relevancia</b>	4	Cumple Completamente

<b>Claridad</b>	3	Cumple con modificaciones menores
<b>Total</b>	3	Cumple con modificaciones menores

Tabla 19 Promedio de resultados experto 3: Fuente el autor

<b>Cuestionario</b>					
<b>Aspectos a Evaluar</b>	<b>Valoración</b>				
	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>Suficiencia de la Guía</b>					
Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)				3	
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología				3	
La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software				3	
La guía abarca los elementos necesarios de cada metodología para poder realizar un proyecto de desarrollo				3	
				3	
<b>Relevancia de la Guía</b>					
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software				3	
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.				3	
La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que cosas debe entregar en su trabajo de grado					4
La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfocados a desarrollo de software			2		
				3	
<b>Claridad de la Guía</b>					
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional				3	
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía		1			
Se exponen claramente las actividades, entregables y artefactos que se seleccionan en la guía				3	
				2	
<b>Puntaje Total</b>				<b>3</b>	

Tabla 20 Tabulación del formato del experto 4: Fuente el autor

<b>Factor</b>	<b>Promedio</b>	<b>Juicio valorativo</b>
<b>Suficiencia</b>	3	Cumple con modificaciones menores
<b>Relevancia</b>	3	Cumple con modificaciones menores
<b>Claridad</b>	2	Cumple con modificaciones
<b>Total</b>	3	Cumple con modificaciones menores

*Tabla 21 Promedio de resultados experto 4: Fuente el autor*

Con los resultados de la evaluación de los cuatro expertos se promedió para obtener el resultado final.

<b>Cuestionario</b>	<b>Experto 1</b>	<b>Experto 2</b>	<b>Experto 3</b>	<b>Experto 4</b>
<b>Aspectos a Evaluar</b>	<b>Valor</b>	<b>Valor</b>	<b>Valor</b>	<b>Valor</b>
Suficiencia de la Guía	4	3	3	3
Relevancia de la Guía	3	3	4	3
Claridad de la Guía	3	3	3	2
<b>Total</b>	<b>3</b>	<b>2</b>	<b>3</b>	<b>2</b>

*Tabla 22 Resultado final Juicio de Expertos: Fuente el autor*

El promedio<sup>134</sup> final del juicio de los expertos fue de 3 -> **Cumple con modificaciones menores.**

En la gráfica se puede observar el comportamiento de las respuestas de los expertos.

---

<sup>134</sup> Todos los cálculos se hicieron aproximando al entero más cercano.



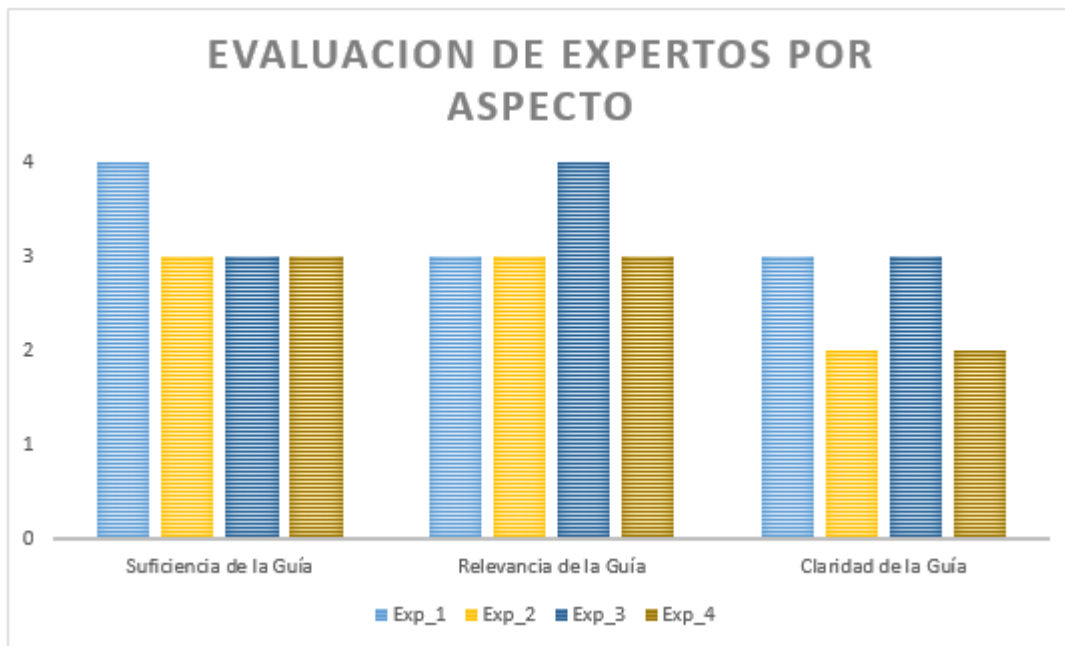


Figura 24 Grafica del resumen de la evaluación de expertos: Fuente el autor

Para validar este resultado se usó el coeficiente de concordancia ( $\omega$ ) de Kendall, que es una técnica estadística de correlación no paramétrica (Badii, Guillen, Lugo Serrato, & Aguilar Garnica, 2014) “Se considera a este coeficiente como un promedio de un grupo de coeficientes de Spearman, es decir, el  $\omega$  es una medida del grado de acuerdo (concordancia) entre  $m$  conjuntos de  $n$  rangos. Por ejemplo, para un grupo de  $n$  objetos evaluados por  $m$  jueces, la  $\omega$  provee información sobre el grado de acuerdo entre  $m$  rangos otorgados por los jueces. Una diferencia entre  $r_s$  (coeficiente de Spearman) y  $\omega$  es que el de Kendall siempre es un valor positivo entre 0 y 1, es decir, sí la evaluación de cada juez a los  $n$  objetos es similar, entonces la  $\omega$  es igual a 1, en cambio si hay un total desacuerdo, entonces el  $\omega = 0$ . Sin embargo, hay que tomar en cuenta que un  $\omega = 0$  puede indicar que los atributos a evaluar son ambiguos o están pobremente definidos, consecuentemente, no se puede discriminar y por tanto hay desconcordancia”.

El coeficiente de Kendall se define como:

$$\omega = 12 \frac{\sum \left( \sum R - \frac{\sum R}{n} \right)^2}{m^2 n (n^2 - 1)}$$

Donde  $\omega$  es el coeficiente de Kendall

$m$  es el número de jueces

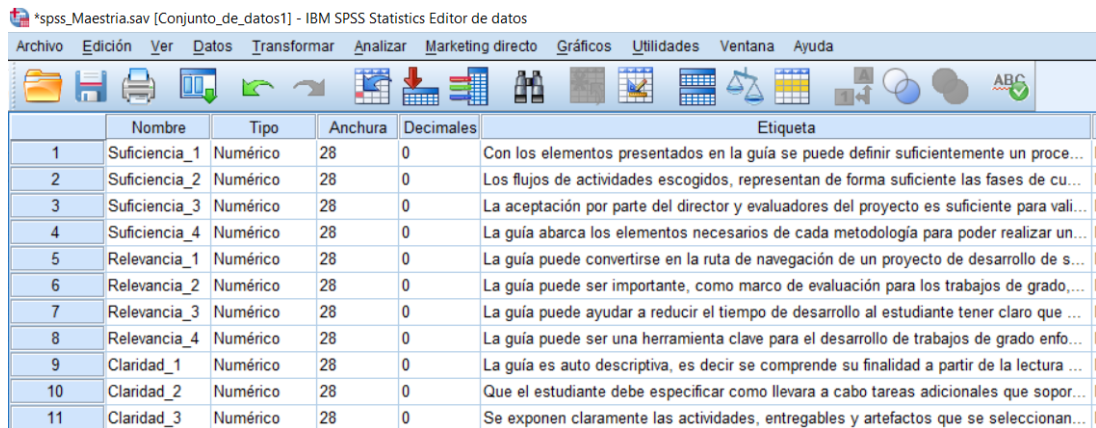
$n$  es la cantidad de aspectos a evaluar

$\sum R$  es la suma de las evaluaciones de un aspecto

$\frac{\sum R}{n}$  es el promedio de la sumatoria de las evaluaciones individuales

Figura 25 Coeficiente de Kendall: Fuente Correlación No-Paramétrica y su Aplicación en la Investigaciones Científica

En este caso tenemos a  $m = 4$ , los expertos y  $n = 11$  los ítems a evaluar. Para hallar  $\omega$  se utilizó el paquete estadístico SPSS de IBM, para lo cual se procedió a ingresar las variables (aspectos a evaluar).



\*spss\_Maestria.sav [Conjunto\_de\_datos1] - IBM SPSS Statistics Editor de datos

	Nombre	Tipo	Anchura	Decimales	Etiqueta
1	Suficiencia_1	Numérico	28	0	Con los elementos presentados en la guía se puede definir suficientemente un proce...
2	Suficiencia_2	Numérico	28	0	Los flujos de actividades escogidos, representan de forma suficiente las fases de cu...
3	Suficiencia_3	Numérico	28	0	La aceptación por parte del director y evaluadores del proyecto es suficiente para vali...
4	Suficiencia_4	Numérico	28	0	La guía abarca los elementos necesarios de cada metodología para poder realizar un...
5	Relevancia_1	Numérico	28	0	La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de s...
6	Relevancia_2	Numérico	28	0	La guía puede ser importante, como marco de evaluación para los trabajos de grado,...
7	Relevancia_3	Numérico	28	0	La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que ...
8	Relevancia_4	Numérico	28	0	La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfo...
9	Claridad_1	Numérico	28	0	La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura ...
10	Claridad_2	Numérico	28	0	Que el estudiante debe especificar como llevara a cabo tareas adicionales que sopor...
11	Claridad_3	Numérico	28	0	Se exponen claramente las actividades, entregables y artefactos que se seleccionan...

Figura 26 Definición de las variables a evaluar en SPSS: Fuente el autor

Luego se ingresaron los datos, tomados de los formatos de evaluación de expertos y se ingresaron como valores para las variables definidas.

\*spss\_Maestria.sav [Conjunto\_de\_datos1] - IBM SPSS Statistics Editor de datos

Archivo Edición Ver Datos Transformar Analizar Marketing directo Gráficos Utilidades Ventana Ayuda

22 : Relevancia\_3

	Suficiencia_1	Suficiencia_2	Suficiencia_3	Suficiencia_4	Relevancia_1	Relevancia_2	Relevancia_3	Relevancia_4	Claridad_1	Claridad_2	Claridad_3
1	4	3	4	3	3	4	3	3	4	1	3
2	3	4	4	2	4	4	3	2	3	0	4
3	3	3	4	3	4	4	3	3	4	1	4
4	3	3	3	3	3	3	4	2	3	1	3

Figura 27 Datos para las variables a evaluar: Fuente el autor

Se procedió a hacer el respectivo análisis, usando métodos no paramétricos con  $k$  muestras relacionadas,

\*spss\_Maestria.sav [Conjunto\_de\_datos1] - IBM SPSS Statistics Editor de datos

Archivo Edición Ver Datos Transformar Analizar Marketing directo Gráficos Utilidades Ventana Ayuda

22 : Relevancia\_3

	Suficiencia_1	Suficiencia_2	Suficiencia_3	Suficiencia_4	Relevancia_2	Relevancia_3	Relevancia_4	Claridad_1	Claridad_2	Claridad_3
1	4	3			4	3	3	4	1	
2	3	4			4	3	2	3	0	
3	3	3			4	3	3	4	1	
4	3	3			3	4	2	3	1	

Informes

Estadísticos descriptivos

Tablas

Comparar medias

Modelo lineal general

Modelos lineales generalizados

Modelos mixtos

Correlaciones

Regresión

Loglineal

Redes neuronales

Clasificar

Reducción de dimensiones

Escala

Pruebas no paramétricas

Predicciones

Superviv.

Respuesta múltiple

Análisis de valores perdidos...

Imputación múltiple

Muestras complejas

Control de calidad

Cuya COR...

Una muestra...

Muestras independientes...

Muestras relacionadas...

Cuadros de diálogo antiguos

Chi-cuadrado...

Binomial...

Rachas...

K-S de 1 muestra...

2 muestras independientes...

K muestras independientes...

2 muestras relacionadas...

K muestras relacionadas...

Figura 28 Búsqueda del método de correlación para  $k$  muestras relacionadas: Fuente el autor

Se escogieron todas las variables, y se escogió la prueba  $w$  de Kendall.



Figura 29 Selección de las variables y del método w de Kendall: Fuente el autor

\*Resultado6 [Documento6] - IBM SPSS Statistics Visor

Archivo Edición Ver Datos Transformar Insertar Formato Analizar Marketing directo Gráficos Utilidades

ultado  
Log  
Pruebas no paramétricas  
  - Título  
  - Notas  
  - Conjunto de datos activo  
  - Prueba W de Kendall  
    + Rangos  
    - Estadísticos de contraste  
Pruebas no paramétricas

Rangos	
	Rango promedio
Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)	6,25
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología	6,00
La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software	8,50
La guía abarca los elementos necesarios de cada metodología para poder realizar un proyecto de desarrollo	4,38
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software	7,25
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.	8,50
La guía puede ayudar a tener claro el desarrollo al estudiante tener claro que hacer en su trabajo de grado	6,13
La guía puede ser una herramienta clave para el desarrollo de trabajos de grado enfocados a desarrollo de software	3,25
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional	7,50
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía	1,00
Se exponen claramente las actividades, entregables y artefactos que se seleccionan en la guía	7,25

Haga doble clic para activar

**Estadísticos de contraste**

N	4
W de Kendall <sup>a</sup>	,620
Chi-cuadrado	24,796
gl	10
Sig. asintót.	,006

Figura 30 Resultados generados por el SPSS: Fuente el autor

Finalmente, el SPSS, nos arroja un informe donde se pueden apreciar los aspectos evaluados y el coeficiente de concordancia  $w$  de Kendall, con un valor superior a 0,5, indicando que hubo concordancia entre los jueces, por lo que se puede decir que el juicio es válido.

La adaptación del proceso software es una tema tan actual hoy día como en los comienzos de la llamada crisis del software, es necesario ajustar las metodologías a las necesidades del proyecto a desarrollar, a las restricciones de tiempo, alcance y otras que puedan surgir.

Esta investigación es un paso más en la búsqueda de soluciones al problema que implica, la necesidad de seguir un derrotero sin que se vuelva un problema más difícil que él se quiere solucionar, conservando altos estándares de calidad y un proceso ingenieril, repetible y acumulable en términos de experiencia.

## CONCLUSIONES

La adaptación del proceso software es una preocupación que recientemente cobra fuerza entre investigadores y desarrolladores de software, sin embargo y a pesar de que en la literatura se encuentre gran variedad de artículos dedicados a este tema, no existe una forma general de realizar la adaptación, normalmente se encuentran esfuerzos de organizaciones particulares y algunos criterios de adaptación como guía para su particularización.

La adaptación del proceso software a las necesidades particulares de un proyecto u organización permite que las actividades se lleven de manera más natural y por ende de forma más eficiente, permitiendo cumplir con los objetivos del proyecto de forma más armónica, el proyecto guía de configuración mínima de procesos software para el desarrollo trabajos de grado en la Universidad Popular del Cesar Seccional Aguachica, permitirá a estudiantes, directores y evaluadores tener un marco común de evaluación y una guía de desarrollo para el estudiante, lo que redundara en un mejor desarrollo de los proyectos.

Para la evaluación de la guía se buscó a expertos que cumplieran con dos características importantes, formación y experiencia en proyectos de grado con producto software; porque estas condiciones garantizan para este caso en particular la pertinencia e idoneidad de los jueces. Para verificar el grado de coherencia en las respuestas de los jueces se usó el coeficiente de correlación  $\omega$  de Kendall.

## BIBLIOGRAFIA

- Afanado, J. M., & Villamizar, L. A. (2010). EL DESARROLLO INDIVIDUAL DE PROYECTOS DE SOFTWARE: *Revista Colombiana de Tecnologías de Avanzada*, 8.
- Albaladejo, X. (s.f.). *ProyectosAgiles.org*. Recuperado el 2015, de <http://proyectosagiles.org/>
- Alonso, D., Pastor, J., Sánchez, P., Álvarez, B., & Chicote, C. V. (2012). Generación Automática de Software para Sistemas de Tiempo Real: Un Enfoque basado en Componentes, Modelos y Frameworks. *Revista Iberoamericana de Automática e Informática Industrial RIAI*.
- Arbeláez, O., Medina, F., & Chaves, J. (2011). HERRAMIENTAS PARA EL DESARROLLO RÁPIDO DE APLICACIONES WEB. (U. T. Pereira, Ed.) *Scientia et Technica*.
- Badii, M., Guillen, A., Lugo Serrato, O., & Aguilar Garnica, J. (2014). Correlación No-Paramétrica y su Aplicación en la Investigaciones Científica. *Daena Journal*.
- Calabria, L. (2003). *Metodología FDD*. Universidad ORT de Uruguay.
- Campderrich, B. (2003). *Ingeniería del Software*. Barcelona: UOC.
- CAMPDERRICH, B. (2003). *INGENIERIA DEL SOFTWARE*. UOC (UNIVERSITAT OBERTA DE CATALUNYA).
- Cárdenas, J. (2011). La Utilización de la Ingeniería del Software en Hipermedia.
- Caro, P. S., & Kahler, N. H. (20 de 10 de 2015). *Unified Modeling Language. Tutorial*. Obtenido de <http://users.dcc.uchile.cl/~psalinas/uml/>
- Castrillón, E. P. (2011). Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA-. *Revista Virtual Universidad Católica del Norte*.
- De Troyer, O., & Leune, K. (1998). WSDM: A User Centered Design Method for Web Sites.
- Del Valle Rodríguez, A. N. (2009). *METODOLOGÍAS DE DISEÑO USADAS EN INGENIERÍA WEB, SU VINCULACIÓN CON LAS NTICS*. Obtenido de

- [http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Tecnologia\\_Infomatica\\_Aplicada\\_en\\_Educacion/Trabajos\\_Finales/Rodriguez\\_Ana.pdf](http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Tecnologia_Infomatica_Aplicada_en_Educacion/Trabajos_Finales/Rodriguez_Ana.pdf)
- Echeverry, L., & Luz, D. (2007). *Caso practico de la metodologia agil XP al desarrollo de software*.
- Escobar Pérez, J., & Cuervo Martínez, Á. (2006). VALIDEZ DE CONTENIDO Y JUICIO DE EXPERTOS: UNA APROXIMACIÓN A SU UTILIZACIÓN.
- Esteban, V. L., & Afanador, M. J. (2010). EL DESARROLLO INDIVIDUAL DE PROYECTOS DE SOFTWARE:.. *Revista Colombiana de Tecnologías de Avanzada*, 8.
- Garbajosa, J. (17 de 03 de 2006). *Computerworld digital*. Obtenido de <http://www.computerworld.es/archive/swebok-profesionaliza-la-ingenieria-de-software>
- García Fernández, E. (2010). *ESTUDIO SOBRE EL MODELO PARA LA MEJORA DE PROCESOS DE SISTEMAS SOFTWARE (CMMI)*. Proyecto de Grado, UNIVERSIDAD CARLOS III DE MADRID - ESCUELA POLITÉCNICA SUPERIOR. Obtenido de [https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0ahUKEwj\\_ev0\\_a\\_KAhUH8z4KHUGMCIkQFghHMAc&url=http%3A%2F%2Farchivo.uc3m.es%2Fbitstream%2Fhandle%2F10016%2F10656%2FFPC\\_Emilio%2520Garcia.pdf%3Fsequence%3D1&usg=AFQjCNE2orEg](https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=8&cad=rja&uact=8&ved=0ahUKEwj_ev0_a_KAhUH8z4KHUGMCIkQFghHMAc&url=http%3A%2F%2Farchivo.uc3m.es%2Fbitstream%2Fhandle%2F10016%2F10656%2FFPC_Emilio%2520Garcia.pdf%3Fsequence%3D1&usg=AFQjCNE2orEg)
- Hernández, H. M., & Ortega-Martínez, E. (2014). Estimación y control de costos en métodos ágiles para desarrollo de software: un caso de estudio. *Ingeniería, Investigación y Tecnología*.
- Hernández, S. R., Fernández, C., & Baptista, L. M. (2010). *METODOLOGIA DE LA INVESTIGACION 5TA EDICION*. México D.F: McGRAW-HILL / INTERAMERICANA EDITORES, S.A. DE C.V.
- Hilera, J., & Palomar, D. (2005). Modelado de procesos de enseñanza-aprendizaje reutilizables con XML, UML e IMS-LD. *RED. Revista de Educación a Distancia*.



- Hundermark, P. (2009). *Un Mejor SCRUM*.
- IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology*. IEEE.
- IEEE. (1997). *1074-1997 - IEEE Standard for Developing Software Life Cycle Processes*. IEEE.
- IEEE. (2004). *GUÍA AL CUERPO DE CONOCIMIENTO DE LA INGENIERÍA DEL SOFTWARE*. California: IEEE Computer Society.
- INTECO, D. N. (2009). *INGENIERÍA DEL SOFTWARE: Metodologías y Ciclos de Vida*. Departamento Nacional de la Calidad del Software.
- Jiménez, A. (2013). *Definición de la metodología para la gestión y verificación de proyectos Ágiles, haciendo uso de Microsoft Team Foundation Server, para su posterior uso en una aplicación real*.
- Jiménez, H. F. (2002). Modelos de Ciclo de Vida de Desarrollo de Software en el Contexto de.
- Jiménez, H. F. (2002). Modelos de Ciclo de Vida de Desarrollo de Software en el Contexto de la Industriai Colombiana de Software. 6.
- Jones, C. (2010). *Introduction and Definitions of Software Best Practices*. McGraw Hill.
- Jurado, M. (2005). Desarrollo de portales corporativos de las instituciones del sector público del gobierno del Ecuador utilizando Microsoft Solutions Framework. Caso de estudio: Banco Nacional de Fomento. Obtenido de <http://bibdigital.epn.edu.ec/handle/15000/2397>
- Kalus, G., & Kuhrmann, M. (2013). *Criteria for software process tailoring A systematic review*.
- KENDALL, K. E., & KENDALL, J. E. (2005). *ANÁLISIS Y DISEÑO DE SISTEMAS*. Pearson education.
- Koch, N. (1999). A Comparative Study of Methods for Hypermedia Development.
- KONRAD, M., & SHRUM , S. (2009). CMMI (2ª ED.): GUIA PARA LA INTEGRACION DE PROCESOS Y LA MEJORA D E PRODUCTOS.
- Lamarca, L. M. (2013). Hipertexto, el nuevo concepto de documento en la cultura de la imagen.

- LAWRENCE, S. (2002). *Ingeniería de software: teoría y práctica*. Pearson Education.
- Lee, H., lee, C., & Yoo, C. (1998). A scenario-based object-oriented hypermedia design methodology.
- Letelier, P., & Penadés, M. (2006). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. Universidad Politécnica de Valencia.
- López, N., & Sandoval, I. (s.f.). Metodos y tecnicas de la investigacion cuantitativa y cualitativa. (U. d. Guadalajara, Ed.) Obtenido de [http://www.pics.uson.mx/wp-content/uploads/2013/10/1\\_Metodos\\_y\\_tecnicas\\_cuantitativa\\_y\\_cualitativa.pdf](http://www.pics.uson.mx/wp-content/uploads/2013/10/1_Metodos_y_tecnicas_cuantitativa_y_cualitativa.pdf)
- Mantilla, D. A., & Santos, A. C. (2007). *DESARROLLO DE UN PORTAL WEB PARA EL INGRESO Y CONSULTA DE NOTAS PARA EL COLEGIO NACIONAL MIXTO*. QUITO.
- Martínez, A., & Martínez, R. (2002). *Guía a Rational Unified Process*. Escuela Politécnica Superior de Albacete – Universidad de Castilla La Mancha.
- Microsoft. (2002). *MicrosoftÆSolutions Framework Essentials*.
- Microsoft. (2013). *MSDN*. Obtenido de <https://msdn.microsoft.com/es-es/library/jj161047%28v=vs.120%29.aspx>
- Mitre-Hernández, H. A., García-Guzmán, J., Amescua-Seco, A. D., & Velasco-Elizondo, P. (2014). Diseño de un programa de medición estratégico para organizaciones de ingeniería de Software: descubriendo dificultades y problemas. *Ingeniería, Investigación y Tecnología*.
- MOGOLLON AFANADOR, J. O. (2010). *ADAPTACION DE PROCESOS AGILES DE DESARROLLO DE SOFTWARE AL DESARROLLO INDIVIDUAL DE APLICATIVOS PEQUEÑOS Y DE BAJO PRESUPUESTO*.
- Muñoz, F. (2006). *Swebok profesionaliza la ingeniería de software*. Obtenido de COMPUTERWORLD: <http://www.computerworld.es/archive/swebok-profesionaliza-la-ingenieria-de-software>
- Navarro, A., Fernández, J., & Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software.

- Nieves, C., Ucán, J., & Pech, M. (2014). UWE en Sistema de Recomendación de Objetos de Aprendizaje. Aplicando Ingeniería Web: Un Método en Caso de Estudio. *Revista Latinoamericana de Ingeniería de Software*.
- O'Kane, T., Russo, N., & Fitzgerald, B. (2003). *SOFTWARE DEVELOPMENT METHOD TAILORING AT MOTOROLA*.
- Parada Robles, I., & León Castro, B. (2013). *Estructurar una propuesta metodológica para guiar el proceso de desarrollo de software en los proyectos de los estudiantes de ingeniería de sistemas de la universidad popular del cesar seccional Aguachica*.
- Pedreira, O., Piattini, M., Luaces, M. R., & Brisaboa, N. R. (2007). Una revisión sistemática de la adaptación del proceso. *Revista Española de Innovación, Calidad e Ingeniería del Software (REICIS)*, 22.
- Pedreira, O., Piattini, M., Luaces, M. R., & Brisaboa, N. R. (2007). *Una revisión sistemática de la adaptación del proceso software*. REICIS. Revista Española de Innovación, Calidad e Ingeniería del Software.
- PONS, C., GIANDIN, R., & PÉREZ, G. (2010). *DESARROLLO DE SOFTWARE DIRIGIDO POR MODELOS Conceptos teóricos y su aplicación práctica*. Editorial de la Universidad Nacional de La Plata.
- Pressman, R. (2002). *Ingeniería del Software, Un Enfoque Practico* (5 ed.). McGraw-Hill.
- Pressman, R. (2005). *INGENIERIA DEL SOFTWARE* (6 ed.). McGraw-Hill.
- Pressman, R. (2010). *Ingeniería del Software un enfoque Práctico*. Madrid: Mc Graw Hill.
- Robles, G. P., & Rojas, M. d. (2015). La validación por juicio de expertos: dos investigaciones cualitativas en Lingüística aplicada. *Revista Nebrija*.
- Ruiz, F., & Verdugo, J. (2008). *Guía de Uso de SPEM2 con EPF Composer*. Castilla: Universidad de Castilla-La Mancha.
- Sabino, C. (2007). *El proceso de investigación*. Caracas: Panapo.
- Sampieri, R. H. (2010). *METODOLOGÍA DE LA INVESTIGACIÓN*. MCGRAW HILL.

- Sánchez, J. S. (2003). *Ingeniería de proyectos informáticos: actividades y procedimientos*. Universitat Jaume I.
- Sanchez, S. (2012). *Criterios para la Adaptabilidad de Estándares y Modelos de Procesos de software en PYMES Ecuatorianas*.
- Selic, B. (2008). MDA Manifestations. *CEPIS UPGRADE the European Journal for the Informatics Professional*.
- Sepúlveda, S., Cravero, A., & Cachero, C. (2016). Requirements modeling languages for software product lines: A systematic literature review. *Information and Software Technology*.
- Serna-M., E., & Morales-V., D. (2014). Estado del arte de la investigación en verificación formal. *Ingeniería, Investigación y Tecnología*.
- Software Engineering Institute. (1990). 1990 SEI Report on Undergraduate Software Engineering Education. *CMU/SEI-90-TR-003*, 93.
- Somerville, I. (2005). *Ingeniería del Software*. Pearson.
- Talavera Pleguezuelos, C. (1999). *Calidad Total en la Administración Pública*. Granada: Unión Iberoamericana de Municipalistas.
- Universidad Nacional de Jujuy Campus Virtual - Facultad de Ingeniería / ANALISIS Y DISEÑO DE SISTEMAS I. (s.f.). Obtenido de <http://www.campus.fi.unju.edu.ar/main/document/document.php?curdirpath=%2FTeoria>
- Urrego, N. E. (2007). Aseguramiento de Calidad en el. *Revista de Tecnología - Journal of Technology • Volumen 6, No. 2, Julio - Diciembre 2007*.
- Urrego, N. J. (2007). Aseguramiento de Calidad en el Desarrollo de Software. *Revista de Tecnología - Journal of Technology • Volumen 6, No. 2, Julio - Diciembre 2007*.
- Villegas, E. J. (1 de 12 de 2015). *Dirección Nacional de innovación Académica*. Obtenido de <http://www.virtual.unal.edu.co/cursos/sedes/manizales/4060024/Lecciones/Capitulo%20I/problemas.htm>

Wells, j. D. (2000). *extremeprogramming.org*. Obtenido de  
<http://www.extremeprogramming.org/map/project.html>  
*wikipedia*. (s.f.). Recuperado el 12 de Agosto de 2015, de  
[https://es.wikipedia.org/wiki/Lenguaje\\_unificado\\_de\\_modelado](https://es.wikipedia.org/wiki/Lenguaje_unificado_de_modelado)

## ANEXO A

### ESTANDARES APROBADOS POR IEEE

Fuente <http://standards.ieee.org/cgi-bin/status?Computer%20Society/Software>

Designación	Título	Estado
	IEEE Software & Systems Engineering Standards Essential Collection - Systems (VuSpec)	Approved Publication of IEEE , Published Date: Jul 24, 2013
	IEEE Software & Systems Engineering Collection: VuSpec	Approved Publication of IEEE
1008-1987	IEEE Standard for Software Unit Testing	Approved Publication of IEEE
1012-2012	IEEE Standard for System and Software Verification and Validation	Approved Publication of IEEE , Published Date: May 25, 2012 **Revision of IEEE Std 1012-2004
1016-2009	IEEE Standard for Information Technology--Systems Design--Software Design Descriptions	Approved Publication of IEEE , Published Date: Jul 20, 2009
1028-2008	IEEE Standard for Software Reviews and Audits	Approved Publication of IEEE , Published Date: Aug 15, 2008

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
1044-2009	IEEE Standard Classification for Software Anomalies	Approved Publication of IEEE , Published Date: Jan 07, 2010
1061-1998	IEEE Standard for a Software Quality Metrics Methodology	Approved Publication of IEEE
1062-1993	IEEE Recommended Practice for Software Acquisition	Approved Publication of IEEE
1062a-1998	Recommended Practice for Software Acquisition	Approved Publication of IEEE **This supl will be incorporated into the base std.
1074-2006	IEEE Standard for Developing a Software Project Life Cycle Process	Approved Publication of IEEE , Published Date: Jul 28, 2006
1175.1-2002	IEEE Guide for CASE Tool Interconnections - Classification and Description	Approved Publication of IEEE , Published Date: Feb 09, 2003
1175.2-2006	IEEE Recommended Practice for CASE Tool Interconnection - Characterization of Interconnections	Approved Publication of IEEE , Published Date: Jan 15, 2007

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
1175.3-2004	IEEE Standard for CASE Tool Interconnections - Reference Model for Specifying Software Behavior	Approved Publication of IEEE , Published Date: Sep 08, 2004
1175.4-2008	IEEE Standard for CASE Tool Interconnections--Reference Model for Specifying System Behavior	Approved Publication of IEEE , Published Date: Apr 17, 2009
12207-2008	Systems and software engineering -- Software life cycle processes	Approved Publication of IEEE , Published Date: Jan 31, 2008
12207-2007-2007	Systems and Software Engineering -- Software Life Cycle Processes	Approved Publication of IEEE **International Standard jointly developed by ISO/IEC/IEEE
1228-1994	IEEE Standard for Software Safety Plans	Approved Publication of IEEE
1320.2-1998	IEEE Standard for Conceptual Modeling Language - Syntax and Semantics for IDEF1X97 (IDEFObject)	Approved Publication of IEEE **Adopted and Replaced by ISO/IEC/IEEE 31320-2-2012



<b>Designación</b>	<b>Título</b>	<b>Estado</b>
14102-2010	IEEE Standard for Adoption of ISO/IEC 14102:2008 Information Technology- -Guideline for the Evaluation and Selection of CASE Tools	Approved Publication of IEEE , Published Date: Sep 09, 2010
14471-2010	IEEE Guide for Adoption of ISO/IEC TR 14471:2007 Information Technology -- Software Engineering -- Guidelines for the Adoption of CASE Tools	Approved Publication of IEEE , Published Date: Sep 09, 2010
1462-1998	1462-1998 IEEE Adoption of ISO/IEC 14102:1995 Information Technology - Guideline for the Evaluation and Selection of CASE Tools	Approved Publication of IEEE , Published Date: Mar 19, 1998
14764-2006	ISO/IEC 14764:2006, Standard for Software Engineering - Software Life Cycle Processes - Maintenance	Approved Publication of IEEE , Published Date: Sep 01, 2006
1490-2011	IEEE Guide--Adoption of the Project Management Institute (PMI(R)) Standard A Guide to the Project Management Body of Knowledge (PMBOK(R) Guide)--Fourth Edition	Approved Publication of IEEE , Published Date: Nov 21, 2011

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
15026-2-2011	IEEE Standard--Adoption of ISO/IEC 15026-2:2011 Systems and Software Engineering--Systems and Software Assurance--Part 2: Assurance Case	Approved Publication of IEEE , Published Date: Oct 11, 2011
15026-3-2013	IEEE Standard Adoption of ISO/IEC 15026-3 -- Systems and Software Engineering -- Systems and Software Assurance -- Part 3: System Integrity Levels	Approved Publication of IEEE , Published Date: Jul 12, 2013
15026-4-2013	IEEE Standard Adoption of ISO/IEC 15026-4--Systems and Software Engineering--Systems and Software Assurance--Part 4: Assurance in the Life Cycle	Approved Publication of IEEE , Published Date: Aug 30, 2013
1517-2010	IEEE Standard for Information Technology--System and Software Life Cycle Processes--Reuse Processes	Approved Publication of IEEE , Published Date: Aug 25, 2010
15288-2008	Systems and software engineering System life cycle processes	Approved Publication of IEEE , Published Date: Jan 31, 2008
15289-2011	ISO/IEC/IEEE Systems and software engineering -- Content of life-cycle information products (documentation)	Approved Publication of IEEE , Published Date: Nov 01, 2011

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
15939-2008	IEEE Standard Adoption of ISO/IEC 15939:2007--Systems and Software Engineering--Measurement Process	Approved Publication of IEEE , Published Date: Jan 30, 2009
16085-2006	ISO/IEC 16085:2006, Standard for Software Engineering - Software Life Cycle Processes - Risk Management	Approved Publication of IEEE , Published Date: Dec 15, 2006
16326-2009	ISO/IEC/IEEE International Standard Systems and Software Engineering-- Life Cycle Processes--Project Management	Approved Publication of IEEE , Published Date: Dec 15, 2009
20000-1-2013	IEEE Standard - Adoption of ISO/IEC 20000-1:2011, Information technology -- Service management -- Part 1: Service management system requirements	Approved Publication of IEEE , Published Date: Jun 03, 2013 **Adoption of ISO/IEC 20000-1:2011
20000-2-2013	IEEE Standard -- Adoption of ISO/IEC 20000-2:2012, Information technology -- Service management -- Part 2: Guidance on the application of service management systems	Approved Publication of IEEE , Published Date: Jun 03, 2013 **Adoption of ISO/IEC 20000-2:2012
2001-2002	Recommended Practice for the Internet - Web Site Engineering, Web Site Management and Web Site Life Cycle	Approved Publication of IEEE , Published Date: Mar 03, 2003

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
24748-1-2011	IEEE Guide--Adoption of ISO/IEC TR 24748-1:2010 Systems and Software Engineering--Life Cycle Management--Part 1: Guide for Life Cycle Management	Approved Publication of IEEE , Published Date: Jun 03, 2011
24748-2-2012	IEEE Guide--Adoption of ISO/IEC TR 24748-2:2011 Systems and Software Engineering-- Life Cycle Management-- Part 2: Guide to the Application of ISO/IEC 15288 (System Life Cycle Processes)	Approved Publication of IEEE , Published Date: Apr 18, 2012
24748-3-2012	IEEE Guide:--Adoption of ISO/IEC TR 24748-3:2011, Systems and software engineering-Life cycle management-Part 3: Guide to the application of ISO/IEC 12207 (Software life cycle processes)	Approved Publication of IEEE , Published Date: Apr 20, 2012
24765-2010	Systems and software engineering -- Vocabulary	Approved Publication of IEEE , Published Date: Dec 15, 2010
24774-2012	IEEE Guide--Adoption of ISO/IEC TR 24474:2010 Systems and Software Engineering-- Life Cycle Management--Guidelines for Process Description	Approved Publication of IEEE , Published Date: Apr 27, 2012

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
26511-2012	Systems and software engineering -- Requirements for managers of user documentation	Approved Publication of IEEE , Published Date: Mar 15, 2012
26512-2011	IEEE/ISO/IEC Systems and software engineering -- Requirements for acquirers and suppliers of user documentation	Approved Publication of IEEE , Published Date: Jun 01, 2011
26513-2010	IEEE Standard for Adoption of ISO/IEC 26513:2009 Systems and Software Engineering--Requirements for Testers and Reviewers of Documentation	Approved Publication of IEEE , Published Date: Jan 27, 2011
26514-2010	IEEE Standard for Adoption of ISO/IEC 26514:2008 Systems and Software Engineering--Requirements for Designers and Developers of User Documentation	Approved Publication of IEEE , Published Date: Jan 27, 2011
26515-2012	Systems and software engineering -- Developing user documentation in an agile environment	Approved Publication of IEEE , Published Date: Mar 15, 2012
26702-2007	ISO/IEC Standard for Systems Engineering - Application and Management of the Systems Engineering Process	Approved Publication of IEEE , Published Date: Jul 15, 2007

<b>Designación</b>	<b>Título</b>	<b>Estado</b>
29119-1-2013	Software and systems engineering - Software testing - Part 1: Concepts and definitions	Approved Publication of IEEE , Published Date: Sep 01, 2013
29119-3-2013	Software and systems engineering - Software testing - Part 3: Test documentation	Approved Publication of IEEE , Published Date: Sep 01, 2013
29148-2011	Systems and software engineering -- Life cycle processes -- Requirements engineering	Approved Publication of IEEE , Published Date: Dec 01, 2011
31320-1-2012	Information technology -- Modeling Languages -- Part 1: Syntax and Semantics for IDEF0	Approved Publication of IEEE , Published Date: Oct 30, 2012 **Adoption of IEEE Std 1320-1-1998
31320-2-2012	Information technology -- Modeling Languages -- Part 2: Syntax and Semantics for IDEF1X97 (IDEFObject)	Approved Publication of IEEE, Published Date: Oct 30, 2012 **This is an adoption of IEEE Std 1320.2-1998.
42010-2011	ISO/IEC/IEEE Systems and software engineering -- Architecture description	Approved Publication of IEEE , Published Date: Dec 01, 2011

Designación	Título	Estado
730-2014	IEEE Standard for Software Quality Assurance Processes	Approved Publication of IEEE , Published Date: Jun 13, 2014 **Revision of IEEE Std 730-2002
828-2012	IEEE Standard for Configuration Management in Systems and Software Engineering	Approved Publication of IEEE , Published Date: Mar 16, 2012 **Revision of IEEE Std 828-2005
829-2008	IEEE Standard for Software and System Test Documentation	Approved Publication of IEEE , Published Date: Jul 18, 2008
90003-2008	IEEE Guide--Adoption of ISO/IEC 90003:2004 Software Engineering-- Guidelines for the Application of ISO 9001:2000 to Computer Software	Approved Publication of IEEE , Published Date: Nov 14, 2008
982.1-2005	IEEE Standard Dictionary of Measures of the Software Aspects of Dependability	Approved Publication of IEEE , Published Date: May 08, 2006

## ANEXO B

### RELACION DE PROYECTOS ANALIZADOS: Fuente el autor

TITULO	METODOLOGIA	AÑO
DISEÑO E IMPLEMENTACION DE LA ASIGNATURA CATEDRA UPECISTA I DE LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA EN EL AMBIENTE VIRTUAL DE APRENDIZAJE MOODLE	Ciclo de Vida Clásico	2010
APLICATIVO WEB PARA LA GESTION DE LOS PROCESOS DE CONTRATACION DE EGUROS EXEQUIALES Y RECAUDO EN LA FUNERARIA PREVICESAR	IWEB	2013
DESARROLLO DE UNA APLICACIÓN WEB QUE PERMITA A LA INSTITUCION TECNICA EDUCATIVA NUESTRA SEÑORA DEL CARMEN LLEVAR UN HISTORIAL DE VIDA ACADEMICO DE CADA UNO DE LOS ESTUDIANTES, GENERAR INFORMES ACADEMICOS Y REALIZAR SUGERENCIA VOCACIONALES	IWEB	2014
DISEÑO E IMPLEMENTACION DE UN CURSO E-LEARNING PARA APOYAR A LOS DOCENTES EN LOS PROCESOS DE PRODUCCION DE OVAS Y CURSOS PARA EL AULA VIRTUAL DE LA UNIVESIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA	MedeOvas	2013
CONSTRUCCION E IMPLEMENTACION DE UN OBJETO VIRTUAL DE APRENDIZAJE EN LA PLATAFORMA PARA FACILITAR EL PROCESO DE ENSEÑANZA - APRENDIZAJE DE LA ASIGNATURA DE CIENCIAS BASICAS I (CALCULO I), UNIDAD FORMATIVA: DERIVADAS PARA PRIMER SEMESTRE DE LOS ESTUDIANTES DE INGENIERIAS, DE LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA	MedeOvas	2014
DESARROLLO DE UN APLICATIVO PARA SISTEMATIZAR EL REGISTRO DE AFILIADOS Y CONTROL DE LOS RECURSOS FINANCIEROS DE LA ORGANIZACIÓN SINDICAL SINTRAELECOL SECCIONAL OCAÑA, NORTE DE SANTANDER	Modelo evolutivo prototipado	2013
ANALISIS, DISEÑO E IMPLEMENTACION DE UN SISTEMA DE BASE DE DATOS QUE ADMINISTRELA INFORMACION DE LOS AFILIADOS EN LA COOPERATIVA DE TRABAJO ASOCIADO DE RADIO Y TELEVISION " COOP - ANDALUZ", UBICADA EN EL MUNICIPIO DE AGUACHICA, CESAR	No se describe	2006
ESTUDIO DE VIABILIDAD Y DESARROLLO DE UN PROTOTIPO DE SISTEMA EXPERTO PARA DIAGNOSTICO DE ENFERMEDADES BOVINAS EN CINCO MUNICIPIO DEL SUR DEL CESAR	No se describe	2006
SISTEMA DE INFORMACION SIDEFUS PARA EL CONTROL DE LOS SEGUROS EXCEQUIALES DE LA FUNERARIA SAN PEDRO DE AGUACHICA CESAR	No se describe	2006
IMPLEMENTACION DE UN TUTORIAL INFORMATIVO MEDIANTE EL USO DE HERRAMIENTAS MULTIMEDIALES, APLICADO A LA RECOPIACION HISTORICA DEL MUNICIPIO DE AGUACHICA CESAR, PARA EL CENTRO CULTURAL "ELADIO VARGAS".	No se describe	2009




<b>TITULO</b>	<b>METODOLOGIA</b>	<b>AÑO</b>
DISEÑO E IMPLEMENTACION DE UN CURSO VIRTUAL DE PRE - CALCULO EN EL AREA DE MATEMATICAS PARA LOS ESTUDIANTES DE PRIMER SEMESTRE DE LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA	No se describe	2010
DISEÑO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION EN AMBIENTE WEB PARA LA CLINICA ODONTOLOGICA DE ESPECIALISTAS (CODES) DE AGUACHICA CESAR	RUP	2009
DISEÑO E IMPLEMENTACION DE UNA APLICACIÓN WEB PARA ACCEDER A LOS RECURSOS BIBLIOGRAFICOS EXISTENTES EN LA BIBLIOTECA DE LA UNIVERSIDAD POULAR DEL CESAR, SECCIONAL AGUACHICA	RUP	2009
DISEÑO E IMPLEMENTACION DE UN OBJETO VIRTUAL DE APRENDIZAJE (OVA) QUESIRVA DE APOYO PARA ADQUIRIR LAS COMPETENCIAS EN LOS TEMAS TIEMPOS SIMPLES Y PRONOMBRES DE LA ASIGNATURA DE INGLES I EN LA UNIVERSIDAD POPULAR DEL CESAR - SECCIONAL AGUACHICA	RUP	2010
SISTEMA DE INFORMACION BAJO PLATAFORMA WEB PARA GESTIONAR LOS PROCESOS DE SEGUIMIENTO REVISION Y CONTROL DE CORRECCIONES REALIZADAS POR EL COMITÉ DE INVESTIGACION Y DEMAS ACTORES A LAS MODALIDADES DE PROYECTOS DE GRADO EN LA UNIVERSIDAD POPULAR DEL CESAR - SECCIONAL AGUACHICA	RUP	2010
DISEÑO IMPLEMENTACION DE UN SISTEMA DE INFORMACION ACADEMICO PARA LA GESTION DE LOS PROCESOS ADMINISTRATIVOS DE LA INSTITUCION EDUCATIVA RAFAEL SALAZAR UBICADA EN EL MUNICIPIO DE GAMARRA, DESPARTAMENTO DEL CESAR	RUP	2011
IMPLEMENTACION DEL SOFTWARE PARA LA GESTION DE LA INFORMACION EN LA ESTACION DE GAS NATURAL NOREAN (CESAR)	RUP	2011
SISTEMATIZACION DE LOS PROCESOS DEL AREA DE VENTAS DE LA EMPRESA INDUSTRIAS ARDICAR S.A.S. UBICADA EN AGUACHICA CESAR	RUP	2014
DISEÑO E IMPLEMENTACION DE UN PORTAL INFORMATIVO PARA EL PERIODICO EL NUEVO SUR DE AGUACHICA CESAR	XP	2009
SISTEMA DE INFORMACION PARA EL CONTROL DE LOS CULTIVOS EN LA COOPERATIVA CAMPESINA INTEGRAL DEL CESAR COOCIC LIMITADA EN EL MUNICIPIO DE TAMALAMEQUE CESAR	XP	2010
DISEÑO E IMPLEMENTACION DE UNA APLICACIÓN QUE AUTOMATICE Y GESTIONE LOS PROCESOS DEL BANCO DE SANGRE DEL HOSPITAL REGIONAL " JOSE DAVID PADILLA VILLAFAÑE" EN AGUACHICA CESAR	XP	2012
CONSTRUCCION DE UN APLICATIVO DE ESCRITORIO WINDOWS CON ACCESO A BASE DE DATOS EN SERVIDORES WEB LINUX A TRAVES DE SERVICIOS WEB PARA PROMOVER Y APORPIAR ESTA TECNOLOGIA EN LOS PROYECTOS DE DESARROLLO DE SOFTWARE EN LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA.	XP	2012

<b>TITULO</b>	<b>METODOLOGIA</b>	<b>AÑO</b>
DISEÑO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION EN AMBIENTE WEB PARA EL PROCESO DE VENTAS DE LA EMPRESA AMBIFLORES LOCALIZADA EN EL MUNICIPIO DE AGUACHICA, CESAR	XP	2012
DISEÑO E IMPLEMENTACION DE UN SOFTWARE DE GESTION PARA EL MANEJO OPERATIVO Y COMERCIAL DEL PARQUE CEMENTERIO JARDINES DE LA ANTROPOLIS DE LA EMPRESA ORGANIZACIÓN PARANA E.U. DE AGUACHICA CESAR	XP	2012
IMPLEMENTACION DE UNA APLICACIÓN WEB PARA FACILITAR LOS SERVICIOS PRESTADOS POR EL INSTITUTO MUNICIPAL DE TRANSITO Y TRANSPORTE DE AGUACHICA CESAR	XP	2013
DISEÑO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION QUE GESTIONE LA FACTURACION Y PERMITIR CONTROLAR LOS PAGOS DE SERVICIOS DE AGUA DE LA EMPRESA DE SERVICIOS PUBLICOS DEL MUNICIPIO DE GAMARRA - CESAR	XP	2013
DISEÑO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION EN AMBIENTE PARA LA GESTION DEL PROCESO DE REFERENCIA Y CONTRAREFERENCIA DEL HOSPITAL JOSE DAVID PADILLA VILLAFANE DEL MUNICIPIO DE AGUACHICA CESAR	XP	2013
DISEÑO E IMPLEMENTACION DE UNA PLATAFORMA PARA SOPORTAR UNA COMUNIDAD VIRTUAL ACADEMICA EN ENTORNO WEB QUE GESTIONE Y APOYE LOS PROCESOS DE INTERACCION DE CONOCIMIENTOS EN LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA	XP	2013
DESARROLLO DE UN SISTEMA DE INFORMACION INTEGRAL PARA EL MANEJO Y CONTROL DE LOS PROCESOS DE LA DROGUERIA " LA MAYORIA " DE LA JAGUA DE IBIRICO	XP	2013
IMPLEMENTACION DE LA PLATAFORMA WEB PARA ORGANIZAR LOS PROCESOS DE ADMINISTRACION ACADEMICA EN LA INSTITUCION EDUCATIVA TECNICA INDUSTRIAL " LAUREANO GOMEZ CASTRO " EN EL MUNICIPIO DE AGUACHICA CESAR	XP	2014
DISEÑO E IMPLEMENTACION DE UN SISTEMA DE INFORMACION PARA LA GESTION DEL PROCESO DE TRAZABILIDAD DEL GANADO BOVINO DE CEBA EN LA FINCA EL DARIEN EN EL MUNICIPIO DE GAMARRA CESAR	XP	2014
DISEÑO E IMPLEMENTACION DE UN APLICATIVO WEB PARA EL PROCESO DE AUTOEVALUACION DE LOS PROGRAMAS DE LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA	XP	2014
DESARROLLO DE UN APLICATIVO WEB PARA EL MANEJO DEL DESPARTAMENTO DE INVENTARIOS Y EL CONTROL DE MANTENIMIENTO DE MAQUINARIA Y EQUIPO DE LA EMPRESA CONSTRUCCION GEOTECNIA & MANTENIMIENTO LIMITADA (CG&M LTDA)	XP	2014

TITULO	METODOLOGIA	AÑO
DISEÑO E IMPLEMENTACION DE LA PLATAFORMA WEB DE EDUCACION VIRTUAL EN LA UNIVERSIDAD POPULAR DEL CEAR SECCIONAL AGUACHICA (WUPC) MEDIANTE UTILIZACION DE SOFTWARE DE DISTRIBUCION LIBRE	N/A	
ANALISIS Y DISEÑO DE LA RED DE DATOS DE LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA EN INTERCONEXION CON LOS MUNICIPIOS DEL CORREDOR UNIVERSITARIO	N/A	
ESTUDIO DE FACTIBILIDAD TECNOLÓGICA PARA LA INTERCONEXION EN RED DE COMPUTADORES DESDE LA SEDE PRINCIPAL DEL INSTITUTO NACIONAL CENTRO DE SISTEMAS EN VALLEDUPAR, HACIA LAS SEDES UBICADAS EN EL DEPARTAMENTO DEL CESAR	N/A	
IMPLEMENTACION DE UN AMBIENTE DE APRENDIZAJE MULTIMEDIAL POLIVALENTE MEDIANTE LA INCORPORACION DE LAS TICS PARA EL COLEGIO PASACORRIENDO CORREGIMIENTO DE ANTEQUERA MUNICIPIO DE TGAMALAMEQUE DESPARTAMENTO DEL CESAR	N/A	
VIRTUALIZACION DE SERVIDORES PARA INCREMENTAR EL APROVECHAMIENTO DE LOS RECURSOS INFORMATICOS ACTUALES UTILIZANDO HERRAMIENTAS DE SOFTWARE LIBRE QUE SOPORTE LAS ACTIVIDADES ACADEMICAS DEL PROGRAMA INGENIERIA DE SISTEMAS DE LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA	N/A	
ESTUDIO PARA CONOCER EL ESTADO ACTUAL DEL USO Y APROPIACION DE LAS TECNOLOGIAS DE LA INFORMACION Y LAS COMUNICACIONES (TIC) EN EL MUNICIPIO DE AGUACHICA - CESAR, COMO FASE DE INICIO DEL PROYECTO "AGUACHICA DIGITAL"	N/A	
ESTRUCTURAR UNA PROPUESTA METODOLOGICA PARA GUIAR EL PROCESO DE DESARROLLO EN LOS PROYECTOS DE LOS ESTUDIANTES DE INGENIERIA DE SISTEMAS DE LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA	N/A	
DISEÑO DEL SISTEMA DE GESTION DE SEGURIDAD DE LA INFORMACION BASADO EN EL ESTANDAR ISO 27001, EN LA UNIVERSIDAD POPULAR DEL CESAR, SECCIONAL AGUACHICA REALIZANDO UNA PRUEBA PILOTO EN LA COORDINACION DEL PROGRAMA DE INGENIERIA DE SISTEMAS	N/A	
ESTUDIO TECNICO Y PRESUPUESTAL PARA LA IMPLEMENTACION DE LA RED INALAMBRICA WI-FI QUE APOYARA LA ESTRATEGIA DE TERRITORIOS DIGITALES PARA EL PROYECTO AGUACHICA DIGITAL	N/A	

## ANEXO C

Formato de validación por juicio de expertos: Fuente el autor

	<b>GUÍA DE CONFIGURACIÓN MÍNIMA DE PROCESOS SOFTWARE PARA EL DESARROLLO TRABAJOS DE GRADO EN LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA</b>	Versión 2.0
	<b>INSTRUMENTO DE VALIDACION POR JUICIO DE EXPERTOS</b>	20/02/2016

Respetado Señor usted ha sido seleccionado como Juez para evaluar el instrumento de selección de configuración del proyecto de investigación **GUÍA DE CONFIGURACIÓN MÍNIMA DE PROCESOS SOFTWARE PARA EL DESARROLLO TRABAJOS DE GRADO EN LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA.**

**Datos del Juez**

Nombre			
Formación Académica			
Areas de experiencia profesional			
Tiempo de experiencia	Años	Cargo Actual	
Institución			

**Objetivo de la Investigación**

Definir una guía de Configuración mínima de procesos software para el desarrollo de proyectos software que sirven como opción de trabajo de grado en la Universidad Popular del Cesar - Seccional Aguachica

**Objetivo de la prueba**

El objetivo de este cuestionario es recolectar la opinión de expertos acerca de la de la fiabilidad que puede tener un proceso software, definido con la guía de configuración presentada a usted, para esto se tendrán en cuenta los siguientes factores suficiencia, relevancia y claridad

**Contexto**

La guía de configuración mínima de procesos software presentada, busca que un estudiante defina el proceso software de su proyecto de grado a partir de la adaptación de una metodología en particular y que director y evaluadores validen esta configuración, para que sirva como guía de trabajo y marco de evaluación.

**De la Evaluación**

La prueba consistirá en una serie de aspectos por factor, las que usted como experto calificara en una escala de 0 a 5, siendo 0 cuando no se cumpla y 5 si se cumple completamente; en la tabla se ve la equivalencia entre el valor numérico y el cualitativo

<b>Escala Valorativa</b>	
<b>Valor Cualitativo</b>	<b>Numérico</b>
No cumple	0
Cumple con muchas modificaciones	1
Cumple con modificaciones	2
Cumple con modificaciones menores	3
Cumple completamente	4



**GUÍA DE CONFIGURACIÓN MÍNIMA DE PROCESOS SOFTWARE  
PARA EL DESARROLLO TRABAJOS DE GRADO EN LA  
UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA  
INSTRUMENTO DE VALIDACION POR JUICIO DE EXPERTOS**


Versión 2.0

20/02/2016

<b>Cuestionario</b>					
<b>Aspectos a Evaluar</b>	<b>Valoración</b>				
	<b>0</b>	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>
<b>Suficiencia de la Guía</b>					
Con los elementos presentados en la guía se puede definir suficientemente un proceso software para un proyecto de desarrollo (proyecto de grado)					
Los flujos de actividades escogidos, representan de forma suficiente las fases de cualquier metodología					
La aceptación por parte del director y evaluadores del proyecto es suficiente para validar los elementos seleccionados como proceso software					
<b>Relevancia de la Guía</b>					
La guía puede convertirse en la ruta de navegación de un proyecto de desarrollo de software					
La guía puede ser importante, como marco de evaluación para los trabajos de grado, que incluyan desarrollo de software.					
La guía puede ayudar a reducir el tiempo de desarrollo al estudiante tener claro que cosas debe entregar en su trabajo de grado					
<b>Claridad de la Guía</b>					
La guía es auto descriptiva, es decir se comprende su finalidad a partir de la lectura de la misma, sin información adicional					
Que el estudiante debe especificar como llevara a cabo tareas adicionales que soporten el uso de los artefactos escogidos, hace más fácil el uso de la guía					
<b>Puntaje Total</b>					
<b>Observaciones</b>					
<b>Convenciones</b>					
<b>S -&gt; Suficiencia, R -&gt; Relevancia, C-&gt; Claridad</b>					
_____			_____		
<b>Firma del Juez</b>			<b>Fecha</b>		

## ANEXO D

Formato de definición del alcance del proyecto: Fuente el autor

	<b>GUÍA DE CONFIGURACIÓN MÍNIMA DE PROCESOS SOFTWARE PARA EL DESARROLLO TRABAJOS DE GRADO EN LA UNIVERSIDAD POPULAR DEL CESAR SECCIONAL AGUACHICA</b>	Versión 1.0
	<b>DEFINICION DEL ALCANCE DEL PROYECTO</b>	20/01/2016

DATOS DEL PROYECTO			
Nombre			
Director			
Integrantes			
Nombre		Correo	
Nombre		Correo	
Nombre		Correo	
<b>Objetivo General del Proyecto</b>			
<b>Objetivos Especificos</b>			
<b>Descripción del Producto del Proyecto</b>			
<b>Requisitos Iniciales</b>			
<b>Estimación del Tamaño del Proyecto</b>			
Pequeño		Mediano	
			Grande

## ANEXO E

### Guía para la configuración mínima de proceso software

#### 1 Guía para la configuración mínima de proceso software

La guía de configuración mínima de procesos software, fue realizada para que los estudiantes del programa de Ingeniería de Sistemas de la Universidad Popular del Cesar Seccional Aguachica, puedan adaptar el proceso software que usaran en el desarrollo de su trabajo de grado, cuando este implique un software como producto principal.

#### 2 Objetivo y Alcance

Los objetivos que se pretenden alcanzar con esta guía son:

- Proporcionar a estudiantes y directores de proyectos de grado del programa ingeniería de sistemas una guía para configurar el proceso software que se usara en su proyecto con la cantidad mínima elementos que pueda contener.
- Definir un marco común de evaluación, para estudiantes, directores y jurados de proyectos de grado

La guía Inicia con la definición del alcance del proyecto software, para lo cual se usa el formato de **definición del alcance del proyecto**, que ayuda a justificar el tamaño del equipo a partir del número de funcionalidades a desarrollar (requisitos funcionales) o los objetivos del proyecto (si los requisitos del mismo aún no se han definido suficientemente), por parte del estudiante y su director, hasta realizar el proceso de conciliación de estudiantes, directores y jurados de proyectos de grado.

### **3 Responsable**

El responsable de garantizar la adecuada aplicación del presente procedimiento es el Coordinador del programa de ingeniería de sistemas de la Universidad popular del cesar seccional Aguachica. Como responsable de aplicar o definir los lineamientos para la **presentación de proyectos de grado**

### **4 Definiciones**

Estas son las definiciones de términos encontrados de forma frecuente en la Guía de configuración mínima de proceso software.

#### **Trabajo de grado**

Una de las modalidades de grado, que ofrece la universidad a los estudiantes, consiste en una serie de documentos, donde recopilan la información y procedimientos empleados en la investigación de un tema específico o la solución de un problema.

#### **Software**

(INTECO, 2009) En su informe cita “el Estándar IEEE 610 define el software como programas, procedimientos y documentación y datos asociados, relacionados con la operación de un sistema informático”

#### **Modelo de ciclo de vida**

(Jiménez A. , 2013) El término modelo de ciclo de vida se refiere a un conjunto de fases por las que pasa un producto software desde que nace la idea hasta que es reemplazado o retirado.



## **Proceso software**

(Pressman, 2005) El proceso software, puede definirse como una colección de patrones que definen un conjunto de acciones, tareas de trabajo o comportamientos relacionados, que requiere el desarrollo de un software.

## **Artefacto**

El termino artefacto, denota cualquier producto, derivado de alguna actividad del proceso de software.

## **Entregable**

(INTECO, 2009) Son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, software). Los entregables permiten evaluar la marcha del proyecto mediante comprobaciones de su adecuación o no a los requisitos funcionales y de condiciones de realización previamente establecidos.

## **Metodología**

(INTECO, 2009) Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo. Es un proceso de software detallado y completo.

Las metodologías se basan en una combinación de los modelos de proceso genéricos (cascada, incremental...). Definen artefactos, roles y actividades, junto con prácticas y técnicas recomendadas.

## **RUP**

(Martínez & Martínez, 2002) El Proceso Unificado de Rational es un proceso de ingeniería del software. Proporciona un acercamiento disciplinado a la asignación de tareas y responsabilidades en una organización de desarrollo. Su propósito es

asegurar la producción de software de alta calidad que se ajuste a las necesidades de sus usuarios finales con unos costos y calendario predecibles.

## **XP**

(Letelier & Penadés, 2006) Es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico.

## **5 Limitaciones**

La guía de configuración mínima de proyectos de software ha sido diseñada para un entorno académico, específicamente los trabajos de grado que incorporen software como producto principal. No ha sido probada ni pretende ser probada en entornos de producción.

## **6 Contenido**

La guía de configuración mínima de proyectos de software contempla las actividades relacionadas en la siguiente tabla.

<b>N° DE ACTIVIDAD</b>	<b>ACTIVIDAD</b>	<b>RESPONSABLE</b>
1	Determinar el alcance del software a desarrollar	Estudiante/director de trabajo de grado
2	justificar el tamaño del equipo	Estudiante/director de trabajo de grado

3	seleccionar la metodología a usar en el desarrollo	Estudiante/director de trabajo de grado
4	Describir el uso de la metodología en términos de: actividades, roles, artefactos y herramientas/ tecnología	Estudiante/director de trabajo de grado
5	Presentación del formato de anteproyecto (formato 2)	Estudiante/Jurados de trabajo de grado
6	Conciliar el proceso definido con los jurados	Estudiante/Jurados de trabajo de grado
7	Aprobación de configuración mínima de proceso de software	Jurados de trabajo de grado

Las actividades se describen a continuación:

#### 1 Determinar el alcance del software a desarrollar

El alcance del proyecto se debe determinar teniendo en cuenta un límite de tiempo que los profesores por consenso en conversaciones informales (entrevista no estructurada) determinaron en 6 meses para el proceso de desarrollo, a partir de la aprobación del anteproyecto.

El alcance del proyecto se determinara a partir de la información recolectada con el formato de **definición del alcance del proyecto** (ver anexo C), de acuerdo a los objetivos del proyecto a desarrollar, el estudiante debe proponer un tamaño de alcance entre pequeño, mediano y grande. Esta propuesta después será aceptada o no por el comité de investigación del programa ingeniería de sistemas, quien es el encargado de evaluar las propuestas. Este deba dar el visto bueno para que el proyecto siga su curso.

## 2 Justificar el tamaño del equipo

Una vez que se tiene definido el alcance del proyecto se puede de esta forma justificar el tamaño del equipo (personal) que planea emplear siendo un proyecto pequeño viable para un estudiante, mediano dos estudiantes, grande tres o más estudiantes.

## 3 Seleccionar la metodología

Como siguiente paso el estudiante debe seleccionar una metodología de desarrollo, en el caso particular de la Universidad y para esta guía, se ha optado por usar solo XP y RUP. De estas dos metodologías, el estudiante debe seleccionar una y se deben analizar las actividades, sus artefactos, roles y herramientas o tecnología que pueden ser usados.

Para esto el estudiante puede valerse de los anexos de la guía.

## 4 Describir el uso de la metodología en términos de: actividades, roles, artefactos y herramientas/ tecnología

Luego de un estudio concienzudo se debe seleccionar los artefactos para definir el proceso de software, personalizado y a la medida del proyecto a desarrollar.

La guía se vale de una matriz de correlación en la que el estudiante puede seleccionar las actividades del proceso, con esta lista de actividades el estudiante de común acuerdo con su director podrá definir el proceso solo con los elementos, artefactos y entregables que se consideren convenientes<sup>135</sup>,

La guía pretende que el estudiante configure su proceso software considerado como una secuencia de actividades en las que intervienen personas y

---

<sup>135</sup> También se podrían incluir elementos que sin ser de la metodología, se consideren convenientes para el proyecto

herramientas/tecnología guiadas por un flujo, sin pretender decir cómo o cuando debe realizar alguna actividad.

El formato para la selección de actividades que se propone consta de una lista de chequeo donde el estudiante puede escoger, los artefactos por flujo así:

<b>Selección de Actividades/Entregables</b>		
<b>Actividad/Entregable</b>	<b>Rol</b>	<b>Herramientas</b>
<b>Análisis</b>		
Historias de usuario	Cliente	Formato historias de usuario, Word
Metáfora del sistema	Cliente, equipo de desarrollo	Formato metáfora del Sistema, Word
Priorizar las actividades	Equipo de desarrollo, cliente	Historias de usuario
Calculo de Velocidad del proyecto	Equipo de desarrollo	Excel
Plan de Iteraciones	Equipo de desarrollo	Word, Excel
Plan de entregas	Equipo de desarrollo	Word, Excel
<b>Diseño</b>		
Tarjetas CRC	Director, Responsable, Equipo de desarrollo	Formato Tarjetas CRC, Excel, Word
Tareas de Ingeniería	Director, Responsable, Equipo de desarrollo	Formato Tarea de ingeniería, Excel, Word
<b>Implementación</b>		
Seguir plan de pruebas	Rastreador, probador	Formato Plan de Pruebas, Word

Realizar seguimiento al proyecto	Rastreador	Cronograma, plan de lanzamientos, plan de iteraciones
Refactorización	Programador	IDE seleccionado
Historia de iteración	Programador	Word
Escribir Código fuente	Programador	IDE
Pruebas unitarias	Programador	Módulo de pruebas automáticas
Realizar pruebas de Integración	Probador	IDE, Plan de Pruebas
Escribir manuales de usuario	Programador	Editor HTML, Word
Seguir casos de prueba	Rastreador	Listas de chequeo
Realizar Pruebas Aceptación	Probador	Plan de pruebas, listas de chequeo
Entregar Producto Software	Director	Formato de Acta de entrega, Word

**Descripción de las tareas de soporte del proceso**

<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Estudiante</p>	<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Director</p>
<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Evaluador 1</p>	<hr style="width: 80%; margin: 0 auto;"/> <p>Firma del Evaluador 2</p>

8. Presentación del formato de anteproyecto (formato 2)

Luego de seleccionar los componentes que serán parte de su proceso de software, el estudiante podrá anexar dicho formato al formato 2 de la Universidad como anexo de la metodología de desarrollo que eligió usar en su trabajo de grado.

9. Conciliar el proceso definido con los jurados

Los elementos escogidos, luego deben ser presentados por el estudiante, como directo responsable de su trabajo de grado a los evaluadores del mismo, para llegar a un consenso donde el estudiante pueda defender su propuesta de proceso software y los jurados puedan expresar que consideran ellos que no puede faltar, de esta forma los estudiantes, directores y evaluadores sabrán exactamente qué y cuales cosas deben hacer o evaluar; porque han sido previamente acordadas. Esto favorece notablemente el proceso académico y disminuye el tiempo que el estudiante tarda en terminar su proyecto porque sabe que cosas debe entregar, además evita que se le pidan más cosas después de las revisiones evitando ambigüedades en la evaluación.

Sin embargo el estudiante debe especificar como llevara a cabo las iteraciones y tareas adicionales que soporten el uso de los artefactos escogidos.

10. Aprobación de configuración mínima de proceso de software

Después de llegar a un acuerdo entre estudiantes y evaluadores, las partes deben firmar el formato mencionado, para formalizar la aceptación del proceso software propuesto como válido.

Cabe mencionar que la guía podría ser aplicada a cualquier metodología sin embargo el estudio y construcción de la guía se basó en RUP y XP por ser las más comunes dentro de los proyectos de grado analizados.

## **7 Anexos**

A Descripción de la metodología XP

B Descripción de la metodología RUP

C Formato de definición de alcance del proyecto