UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

PROPUESTA DE UN MODELO PARA LA SELECCIÓN DE UN PROCESO DE SOFTWARE

Especialización en Proyectos Informáticos
Edgar Leonardo Orejuela Morales
Tommy Jolian Restrepo Rios

Docente: Ing. Sandro Bolaños S.





UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

TRABAJO FINAL
ESPECIALIZACIÓN EN PROYECTOS INFORMÁTICOS

PROPUESTA DE UN MODELO PARA LA SELECCIÓN DE UN PROCESO DE SOFTWARE

Autores
EDGAR LEONARDO OREJUELA M.
TOMMY JOLIAN RESTREPO RÍOS

Director
ING. SANDRO BOLAÑOS S.

Bogotá 2018

Dedicado a mis padres, quienes con su amor infinito, guía y esfuerzo me dieron todo para formarme en la vida como persona, padre y profesional. A mi esposa y mis hijos por darme todos los días grandes cantidades de amor, ternura y bondad. A toda mi familia y amigos que me han apoyado incondicionalmente a lo largo de la vida.

- Leonardo Orejuela -

A Dios por permitirme vivir esta nueva etapa de mi vida. A mi mamá, mis hermanos y mi familia les agradezco el estar siempre junto a mi en todas las facetas de mi vida. Mis amigos y allegados que siempre han creído en mí sin dudar nunca.

- Tommy Restrepo -

Agradecimientos

A Dios y a nuestras familias por el tiempo otrogado para nuestra preparación profesional.

A la Universidad Distrital Francisco José de Caldas, por ser nuesta alma máter en nuestros estudios de Pregrado y Posgrado y porque en su labor misional de democratización del conocimiento nos ha brindado los espacios necesarios para nuestro proceso de formación. A la Especialización en Proyectos Informáticos, por brindarnos los medios para crecer personalmente y laboralmente.

A nuestro Director de Trabajo de Grado, el Ingeniero Sandro Bolaños, por su guia y acompañamiento en el desarrollo de este proyecto. A nuestra revisora la Ingeniera Alexandra Abuchar por sus consejos y guía durante el desarrollo de este trabajo de grado.

A todos los docentes de la Especialización en Proyectos Informáticos, y administrativos de la Universidad Distrital Francisco José de Caldas, quienes participaron en nuestra formación a lo largo de nuestra carrera.

Índice general

I Contextualización de la Investigación

1 Descripción de la investigación 21

1.1 Planteamiento del problema 21

1.2 Pregunta de investigación 22

1.3 Sistematización del problema 22

1.4 Objetivos 22

1.4.1 Objetivo general 22

1.4.2 Objetivos específicos 22

1.5 Justificación 23

1.6 Hipótesis 23

1.7 Alcances y limitaciones 23

1.7.1 Limitaciones 23

1.7.2 Alcance 24

1.8 Metodología 24

1.8.1 Levantamiento de información 25

II Fundamentación de la Investigación

2 Fundamentación de la investigación 29

2.1 Marco teórico 29

2.1.1 Proyectos de software 29
2.1.2 ¿Qué es un proceso de software? 30
2.1.3 ¿Qué es un modelo de procesos del software? 31
2.1.4 Modelo big bang 32
2.1.5 Modelo Cascada 32
2.1.6 Modelo Espiral 34
2.1.7 Modelo en V 35
2.1.8 RUP 36
2.1.9 Métrica V3 37

3 Estado del arte 39

3.1 Proyectos de Software 39

3.1.1 Factores de éxito en los proyectos de software 43

3.2 Mejoramiento del proceso de software 43

3.3 Calidad del producto de software 44

3.4 Modelos de selección de procesos de software 45

3.4.1 Criterio para selección de procesos de software de Alexander y Davis ... 46
3.4.2 Otros modelos para la selección de procesos de software 47

4 Desarrollo de la investigación 49

4.1 Situación actual 50

4.1.1 Pregunta 1: Profesión de los encuestados 50
4.1.2 Pregunta 2: Proyectos de software exitosos 51
4.1.3 Pregunta 3: Procesos de desarrollo de software en la empresa 52
4.1.4 Pregunta 4: Uso de procesos de desarrollo de software en la empresa ... 53
4.1.5 Pregunta 5: Documentación de los proyectos de desarrollo de software . 54
4.1.6 Pregunta 6: Retrasos en tiempos de entrega de proyectos 55
4.1.7 Pregunta 7: Problemas de calidad en el software 56
4.1.8 Pregunta 8: Sobrecostos en los proyectos de software 57
4.1.9 Pregunta 9: Definición de requerimientos en los proyectos de software ... 58
4.1.10 Pregunta 10: Importancia de los procesos de software 59

III Resultado de la investigación

5 Modelo para la selección de procesos de software 63

5.1 Variables del modelo de selección de procesos de software 63

5.1.1 Criterios de evaluación 63

5.2 Criterios de evaluación 64

5.2.1 Complejidad del proyecto 64
5.2.2 Tamaño del proyecto 64
5.2.3 Presupuesto del proyecto 64
5.2.4 Tiempo de entrega del proyecto 65
5.2.5 Riesgo asociado al proyecto 65
5.2.6 Proyecto Exploratorio 65
5.2.7 Claridad en los requerimientos iniciales 66
5.2.8 Frecuencia de cambios en los requerimientos 66
5.2.9 Tamaño del equipo de desarrollo 66
5.2.10 Experiencia del equipo de desarrollo 67
5.2.11 Comunicación entre stakeholders 67
5.2.12 Entrega de funcionalidades parciales 68
5.2.13 Reusabilidad del producto de software 68
5.2.14 Documentación del proyecto 68
5.2.15 Matriz de criterios de evaluación 68

5.3 Evaluación de criterios en el modelo cascada 69

5.3.1 Complejidad del proyecto en modelo cascada 69
5.3.2 Tamaño del proyecto en modelo cascada 69
5.3.3 Presupuesto del proyecto en modelo cascada 70
5.3.4 Tiempo de entrega del proyecto en modelo cascada 70
5.3.5 Riesgo asociado al proyecto en modelo cascada 70
5.3.6 Proyecto exploratorio en modelo cascada 71
5.3.7 Claridad en los requerimientos iniciales en modelo cascada 71
5.3.8 Frecuencia de cambios en los requerimientos en modelo cascada 71
5.3.9 Tamaño del equipo de desarrollo en el modelo cascada 72
5.3.10 Experiencia del equipo de desarrollo en el modelo cascada 72
5.3.11 Comunicación entre stakeholders en el modelo cascada 72
5.3.12 Entrega de funcionalidades parciales en el modelo cascada 72
5.3.13 Reusabilidad del producto de software en el modelo cascada 73
5.3.14 Documentación del proyecto en el modelo cascada 73

5.4 Evaluación de criterios en el modelo en V 73

5.4.1 Complejidad del proyecto en el modelo en V 75
5.4.2 Tamaño del proyecto en el modelo en V 75
5.4.3 Presupuesto del proyecto en el modelo en V 75
5.4.4 Tiempo de entrega del proyecto en el modelo en V 75

5.4.5 Riesgo asociado al proyecto en el modelo en V 76
5.4.6 Proyecto exploratorio en el modelo en V 76
5.4.7 Claridad en los requerimientos iniciales en el modelo en V 76
5.4.8 Frecuencia de cambios en los requerimientos en el modelo en V 76
5.4.9 Tamaño del equipo de desarrollo en el modelo en V 77
5.4.10 Experiencia del equipo de desarrollo en el modelo en V 77
5.4.11 Comunicación entre stakeholders en el modelo en V 77
5.4.12 Entrega de funcionalidades parciales en el modelo en V 78
5.4.13 Reusabilidad del producto de software en el modelo en V 78
5.4.14 Documentación del proyecto en el modelo en V 78
5.5 Evaluación de criterios en el modelo espiral 79
5.5.1 Complejidad del proyecto en modelo espiral 80
5.5.2 Tamaño del proyecto en modelo espiral 80
5.5.3 Presupuesto del proyecto en modelo espiral 80
5.5.4 Tiempo de entrega del proyecto en modelo espiral 80
5.5.5 Riesgo asociado al proyecto en modelo espiral 81
5.5.6 Proyecto exploratorio en modelo espiral 81
5.5.7 Claridad en los requerimientos iniciales en modelo espiral 82
5.5.8 Frecuencia de cambios en los requerimientos en modelo espiral 82
5.5.9 Tamaño del equipo de desarrollo en el modelo espiral 82
5.5.10 Experiencia del equipo de desarrollo en el modelo espiral 82
5.5.11 Comunicación entre stakeholders en el modelo espiral 83
5.5.12 Entrega de funcionalidades parciales en el modelo espiral 83
5.5.13 Reusabilidad del producto de software en el modelo espiral 83
5.5.14 Documentación del proyecto en el modelo espiral 84
5.6 Evaluación de criterios en el modelo RUP 84
5.6.1 Complejidad del proyecto en el modelo RUP 84
5.6.2 Tamaño del proyecto en el modelo RUP 86
5.6.3 Presupuesto del proyecto en el modelo RUP 86
5.6.4 Tiempo de entrega del proyecto en el modelo RUP 86
5.6.5 Riesgo asociado al proyecto en el modelo RUP 86
5.6.6 Proyecto exploratorio en el modelo RUP 87
5.6.7 Claridad en los requerimientos iniciales en el modelo RUP 87
5.6.8 Frecuencia de cambios en los requerimientos en el modelo RUP 87
5.6.9 Tamaño del equipo de desarrollo en el modelo RUP 88
5.6.10 Experiencia del equipo de desarrollo en el modelo RUP 88
5.6.11 Comunicación entre stakeholders en el modelo RUP 88
5.6.12 Entrega de funcionalidades parciales en el modelo RUP 89
5.6.13 Reusabilidad del producto de software en el modelo RUP 89
5.6.14 Documentación del proyecto en el modelo RUP 89

5.7 Evaluación de criterios en el modelo en Métrica V3 89
5.7.1 Complejidad del proyecto en el modelo en Métrica V3 91
5.7.2 Tamaño del proyecto en el modelo en Métrica V3 91
5.7.3 Presupuesto del proyecto en Métrica V3 91
5.7.4 Tiempo de entrega del proyecto en Métrica V3 91
5.7.5 Riesgo asociado al proyecto en Métrica V3 92
5.7.6 Proyecto exploratorio en Métrica V3 92
5.7.7 Claridad en los requerimientos iniciales en Métrica V3 92
5.7.8 Frecuencia de cambios en los requerimientos en Métrica V3 93
5.7.9 Tamaño del equipo de desarrollo en Métrica V3 93
5.7.10 Experiencia del equipo de desarrollo en Métrica V3 93
5.7.11 Comunicación entre stakeholders en Métrica V3 94
5.7.12 Entrega de funcionalidades parciales en Métrica V3 94
5.7.13 Reusabilidad del producto de software en Métrica V3 94
5.7.14 Documentación del proyecto en Métrica V3 94
5.8 Ejemplo de aplicación del modelo de selección de procesos de software 96
5.9 Calculo proyecto ejemplo en modelos de proceso 98
5.9.1 Calculo proyecto ejemplo en modelos cascada 98
5.9.2 Calculo proyecto ejemplo modelo en V 100
5.9.3 Calculo proyecto ejemplo en espiral 101
5.9.4 Calculo proyecto ejemplo en RUP 102
5.9.5 Calculo proyecto ejemplo en Métrica v3 103
5.9.6 Resultados de la aplicación del modelo de selección procesos de software 104

IV Conclusiones

Conclusiones 107

Índice de figuras

1.1 Metodología 25

2.1 Modelo Cascada 33
2.2 Modelo Espiral 35
2.3 Modelo V 36
2.4 RUP 37
2.5 Metrica 3 38

3.1 Resolución de proyectos de software del año 2015 40
3.2 Resolución de proyectos de software entre 1994 y 2015 41
3.3 Resolución de proyectos por tamaño en 2015 42
3.4 Calidad de software 45

4.1 Pregunta 1 50
4.2 Pregunta 2 51
4.3 Pregunta 3 52
4.4 Pregunta 4 53
4.5 Pregunta 5 54
4.6 Pregunta 6 55
4.7 Pregunta 7 56
4.8 Pregunta 8 57
4.9 Pregunta 9 58
4.10 Pregunta 10 59

Índice de cuadros

5.1 Criterios de selección 69
5.2 Evaluación de criterios en el modelo cascada 74
5.3 Evaluación de criterios en el modelo en V 79
5.4 Evaluación de criterios en el modelo espiral 85
5.5 Evaluación de criterios en el modelo RUP 90
5.6 Evaluación de criterios en Métrica V3 95
5.7 Ejemplo de proyecto de software 97
5.8 Ejemplo de evaluación en modelo cascada 99
5.9 Ejemplo de evaluación en modelo en V 100
5.10 Ejemplo de evaluación en modelo espiral 101
5.11 Ejemplo de evaluación en modelo en RUP 102
5.12 Ejemplo de evaluación en modelo en Métrica v3 103
5.13 Resultados de la evaluación de los procesos de software en el ejemplo 104

Introducción

En Ingeniería de Software, el proceso es un conjunto de actividades, información consistente que producen software [31]. Además, el proceso es problema y solución en un proyecto de software, ya que, al dar dirección al proyecto, puede convertirse en apoyo u obstáculo de éste. Por otra parte, los proyectos tienen particularidades que los hacen únicos, por lo que el proceso con el cual se direccionará el proyecto debe ser particular y específico.

Actualmente, los proyectos de desarrollo de software carecen de un modelo que permita la selección del proceso adecuado al proyecto (con sus singularidades) y que cubra las necesidades del desarrollo; el proceso en muchos casos se selecciona por moda y por los resultados satisfactorios en otros proyectos.

Además, las organizaciones carecen de procesos de software definidos formalmente que apoyen el éxito de los proyectos informáticos, lo cual afecta los tiempos de entrega, la calidad de los productos y los sobrecostos en el proyecto. Por lo tanto, ésta propuesta está orientada a crear un modelo que permitirá seleccionar a empresas el proceso adecuado para un proyecto de software a partir del análisis de las características de los más importantes procesos de software.

Este proyecto contiene las diferentes etapas que se realizaron para la creación del modelo de selección de procesos de software.

Contextualización de la Investigación

1 Descripción de la investigación 21

1.1 Planteamiento del problema
1.2 Pregunta de investigación
1.3 Sistematización del problema
1.4 Objetivos
1.5 Justificación
1.6 Hipótesis
1.7 Alcances y limitaciones
1.8 Metodología

1. Descripción de la investigación

A continuación, se presenta la descripción de la investigación organizado por: planteamiento o identificación del problema, la pregunta de investigación, la sistematización del problema, los objetivos, la justificación, la hipótesis, los alcances, las limitaciones y la metodología.

1.1 Planteamiento del problema

A pesar de que a lo largo de la Ingeniería de Software se han desarrollado procesos y metodologías de desarrollo de software que prometen ser las soluciones a los problemas existentes en los proyectos de software, se encuentran en estudios e investigaciones que éstos proyectos en un importante porcentaje son no exitosos o exitosos con retrasos y sobrecostos. Dentro de los problemas existentes en los proyectos de software se encuentran: errores de la definición de requerimientos, sobrecostos, reprocesos, demoras en tiempos de entrega, reducción de la calidad y utilidad del producto, entre otros.

Una de las razones para que los proyectos de software fracasen es la ausencia de un proceso o metodología de desarrollo de software formal en el que se definan el qué y el cómo se desarrollará el software. Otra razón por la cual los proyectos de software fracasan se debe a que existen oganizaciones que adoptan procesos o metodologías de software, como la panacea que solucionará todos sus problemas en el desarrollo de software, sin tener en cuenta que los procesos o metodologías de desarrollo de software deben selecionarse y aplicarse de acuerdo a las particularidades del proyecto, las personas, la organización, los clientes, el mercado, etc.

Para dar solución a los problemas mencionados, se requiere un modelo que recomiende

a las organizaciones el proceso de desarrollo de software óptimo para el proyecto de acuerdo las particularidades del proyecto: las personas, los clientes, la complejidad, el tamaño del equipo, el nivel de documentación, los riesgos, las habilidades, entre otras tantas variables.

1.2 Pregunta de investigación

¿Cómo crear un modelo que permita seleccionar el proceso de desarrollo de software óptimo para que el proyecto de software sea exitoso, mejore la calidad y utilidad de los productos de software, disminuya los tiempos de desarrollo y los costos asociados?

1.3 Sistematización del problema

- ¿Cuáles son los problemas, necesidades, limitaciones y alcances existentes en los proyectos y productos de software en las organizaciones?
- ¿Cómo modelar los proyectos de software de manera que se tengan en cuenta todas las variables involucradas en el desarrollo de software?
- ¿Cuál es la mejor manera para analizar las alternativas de procesos de software para las organizaciones que desarrollan software?

1.4 Objetivos

A continuación, se especifican el objetivo general y los objetivos específicos del presente proyecto.

1.4.1 Objetivo general

Crear un modelo para la selección del proceso de software para las organizaciones que mejore la eficacia y eficiencia de los proyectos, así como la calidad y utilidad de los productos de software.

1.4.2 Objetivos específicos

- Identificar el estado de los proyecto de software a nivel global, nacional y local con respecto a las tasas de éxito/fracaso, los factores involucrados y los procesos de software utlizados.
- Analizar los procesos de desarrollo de software más relevantes en la industria del software, sus caraterísticas, ventajas, desventajas e impacto en los proyectos de software.
- Verificar los trabajos e investigaciones realizadas en la historia de la ingeniería de software para seleccionar los procesos de desarrollo de software.
- Crear un modelo para la selección de procesos de software a partir de las características de los procesos de desarrollo de software y su impacto en los criterios más relevantes de los proyectos de software.

1.5 Justificación

La justificación del presente proyecto se enmarca como investigación ya que estudia los proyectos de software y los problemas que llevan a que éstos fracasen: reducción de la calidad o utilidad de los productos, errores en la definición de requerimientos, sobrecostos, reprocesos, demora en toma de decisiones, entre otros. Como intento de solución a dichos problemas, en muchas organizaciones se tienen procedimientos de soluciones, los cuales podrían definirse informalmente como procesos de software que no son óptimos para satisfacer las necesidades por las empresas, ni tienen en cuenta las singularidades de cada uno de los proyectos de software.

Asimismo, éste proyecto tiene una justificación metodológica ya que para brindar una alternativa de solución a los problemas con los proyectos de software, primero se pretende realizar un estudio del estado actual de los proyectos de software, analizando a fondo las necesidades y problemas actuales en sus proyectos de software, para seguidamente diseñar un modelo que permita seleccionar el proceso acorde a los proyectos de software, aumentando la calidad y utilidad de los productos, mejorando los tiempos de entrega, disminuyendo los costos asociados y permitiendo el seguimiento de los proyectos de un modo más eficaz y eficiente.

La definición del proceso adecuado de software por medio del modelo propuesto, beneficiará además de las organizaciones, a las personas involucradas en el desarrollo de software, a los clientes internos, así como a los productos de software.

1.6 Hipótesis

El proceso de software es un conjunto de actividades que busca el desarrollo exitoso de un proyecto con productos informáticos de alta calidad. Debido a las particularidades de los proyectos de software en recursos, tiempos, contexto organizacional, entre otros, cada proyecto es diferente y por tanto requiere de un proceso particular, por lo que se requiere de un modelo que de acuerdo al proyecto determine el proceso más adecuado.

Por lo tanto, se plantea la siguiente hipótesis:

"Un modelo para la selección de procesos de software determina un proceso de software adecuado para los proyectos y productos de software de las organizaciones".

1.7 Alcances y limitaciones

Se describen a continuación las limitaciones y alcance del presente proyecto:

1.7.1 Limitaciones

Las limitaciones que podrían llegar afectar el normal desarrollo del proyecto son:

- La información que sea suministrada por las investigaciones, organizaciones o de las personas involucradas no sea 100 por ciento real.
- No se contempla la implementación del modelo de selección de un proceso de software en organizaciones.
- En el modelo a desarrollar se tendrán en cuenta los 5 procesos de software más reconocidos.
- El equipo de trabajo para el proyecto es de 2 personas.

1.7.2 Alcance

El presente proyecto es un aporte al equipo de tiene las siguientes pautas como alcance:

Estructuración y diseño de un modelo de selección de un proceso de software de acuerdo a las características de los proyectos de software, dirigiendo de manera eficaz y eficiente el proceso de software.

Documentación del modelo de selección de procesos de desarrollo de software.

El presente trabajo pretende ser un apoyo empresarial que le permita a la empresa competir con un producto de mejor calidad, mejorando los tiempos de entrega y disminuyendo los costos en los proyectos de software.

1.8 Metodología

Para crear un modelo de selección de procesos de software adecuado a los proyectos de las organizaciones, lo primero que se realizó fue el análisis del estado de los proyectos de software a nivel global, nacional y local, identificando las tasas de éxito/fracaso y los factores que impactan en los proyectos de software.

Asimismo, se estudió el marco teórico existente sobre ingeniería de software, los procesos de desarrollo de software y sus características, las cuales son aplicables al diseño del modelo propuesto impactando las entradas, el proceso y las salidas del mismo.

Seguidamente se realizó un estudio de las investigaciones existentes con respecto a la selección de procesos y metodologías de desarrollo de software, los análisis y criterios utilizados en modelos y recoemndaciones.

Por último, a partir de la investigación realizada se diseñó el modelo de selección de procesos de desarrollo de software, en el cual se definieron los criterios más relevantes en los proyectos de software, el impacto de cada uno de éstos criterios en los modelos de software y se formalizó la estrategia para evaluar un proyecto de software con respecto a los criterios y los procesos de software.

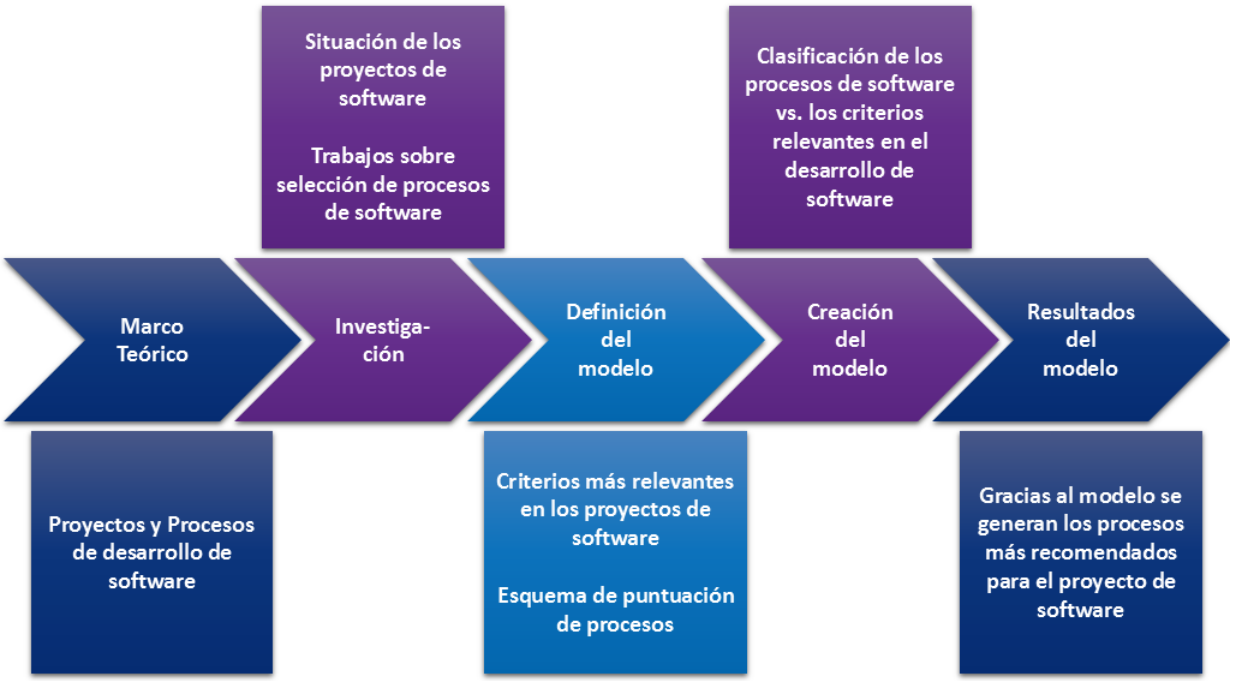


Figura 1.1: Metodología

1.8.1 Levantamiento de información

Las técnicas de investigación que se utilizarán para recolectar la información que permitirá el diseño del modelo para selección de un proceso de software son:

- Encuestas a funcionarios de las empresas involucradas con los proyectos y productos de software: clientes internos, equipo de desarrollo, líderes de equipos de desarrollo, analistas de requerimientos, analistas de pruebas, administradores funcionales, administradores técnicos, etc.
- Observación de campo para obtener información de las organizaciones y del procedimiento de desarrollo de software, registrarla y posteriormente realizar un análisis. Se tendrán en cuenta la observación científica, en la cual el investigador sabe lo que quiere observar y para que se quiere hacerlo.
- Investigación: Esta técnica consiste en la indagación del tema propuesto, analizar cada uno de los factores encontrados y dar fundamentos al tema a desarrollar, el tipo de investigación que se llevará a cabo para este modelo será sistemática, hará énfasis en la recolección y análisis de la información.

II

Fundamentación de la Investigación

2 Fundamentación de la investigación 29

2.1 Marco teórico

3 Estado del arte 39

3.1 Proyectos de Software

3.2 Mejoramiento del proceso de software

3.3 Calidad del producto de software

3.4 Modelos de selección de procesos de software

4 Desarrollo de la investigación .. 49

4.1 Situación actual

2. Fundamentación de la investigación

2.1 Marco teórico

Actualmente las organizaciones se encuentran en una búsqueda constante de estrategias, metodologías y herramientas que apoyen la automatización de sus procesos y sistema de información. Para los procesos de software, se estima tener claros el siguiente marco teórico el cual permitirá ahondar en el modelo que se plantea en este proyecto.

2.1.1 Proyectos de software

Para administrar un proyecto de software exitoso, se debe comprender qué puede salir mal, de modo que los problemas puedan evitarse. En un excelente ensayo acerca de los proyectos de software, John Reel define 10 señales que indican que un proyecto de sistemas de información está en peligro:

1. El personal del software no entiende las necesidades del cliente.
2. El ámbito del producto está pobremente definido.
3. Los cambios se gestionan pobremente.
4. Cambia la tecnología elegida.
5. Las necesidades empresariales cambian [o están mal definidas].
6. Las fechas límite son irreales.
7. Los usuarios son resistentes.
8. Pérdida de patrocinio [o nunca obtenido adecuadamente].
9. El equipo del proyecto carece de personal con habilidades adecuadas.
10. Los gerentes [y profesionales] evitan mejores prácticas y lecciones aprendidas.

Los profesionales de la industria, hastiados, con frecuencia se refieren a la regla 90-90

cuando estudian proyectos de software particularmente difíciles: el primer 90 por ciento de un sistema absorbe el 90 por ciento del esfuerzo y tiempo asignados. El último 10 por ciento toma otro 90 por ciento del esfuerzo y tiempo asignados. Las semillas que conducen a la regla 90-90 están contenidas en las señales anotadas en la lista anterior.

¿Cómo actúa un gerente para evitar los problemas recién anotados? Se sugiere un enfoque de sentido común de cinco partes en los proyectos de software:

1. Comenzar con el pie derecho. Esto se logra al trabajar duro (muy duro) para entender el problema que debe resolverse y luego establecer objetivos y expectativas realistas para todos aquellos que estarán involucrados en el proyecto. Lo anterior se refuerza al construir el equipo correcto y darle autonomía, autoridad y tecnología necesarias para realizar el trabajo.
2. Mantener la cantidad de movimiento. Muchos proyectos parten hacia un buen comienzo y luego lentamente se desintegran. A fin de mantener la cantidad de movimiento, el gerente de proyecto debe proporcionar incentivos para mantener la rotación de personal en un mínimo absoluto, el equipo debe enfatizar la calidad en cada tarea que realice y el administrador ejecutivo debe hacer todo lo posible para permanecer fuera del camino del equipo.
3. Siga la pista al progreso. Para un proyecto de software, el progreso se rastrea conforme los productos operativos (por ejemplo, modelos, código fuente, conjuntos de casos de prueba) se producen y aprueban (usando revisiones técnicas) como parte de una actividad que asegure la calidad. Además, pueden recopilarse medidas de proceso de software y proyecto y usarse para valorar el progreso contra promedios desarrollados para la organización de desarrollo del software.
4. Tome decisiones inteligentes. En esencia, las decisiones del gerente del proyecto y del equipo de software deben "mantenerse simples". Siempre que sea posible, decida usar software comercial de anaquel, o componentes o patrones de software existentes, así como evitar interfaces a la medida cuando estén disponibles enfoques estándar; decida también identificar y luego evitar los riesgos obvios, y asignar más tiempo del que se considere necesario para tareas complejas y riesgosas (necesitará cada minuto).
5. Realice un análisis postmortem. Establezca un mecanismo consistente para extraer lecciones aprendidas por cada proyecto. Evalúe los calendarios planeado y real, recopile y analice métricas de proyecto de software, consiga retroalimentación de los miembros del equipo y de los clientes, y registre los hallazgos en forma escrita [27] .

2.1.2 ¿Qué es un proceso de software?

Un proceso del software es un conjunto de actividades y resultados asociados que producen un producto de software. Estas actividades son llevadas a cabo por los ingenieros de software. Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software. Estas actividades son:

1. Especificación del software donde los clientes e ingenieros definen el software a producir y las restricciones sobre su operación.
2. Desarrollo del software donde el software se diseña y programa.

3. Validación del software donde el software se válida para asegurar que es lo que el cliente requiere.
4. Evolución del software donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

Diferentes tipos de sistemas necesitan diferentes procesos de desarrollo. Por ejemplo, el software de tiempo real en un avión tiene que ser completamente especificado antes de que empiece el desarrollo, mientras que, en un sistema de comercio electrónico, la especificación y el programa normalmente son desarrollados juntos. Por lo tanto, estas actividades genéricas pueden organizarse de diferentes formas y describirse en diferentes niveles de detalle para diferentes tipos de software. Sin embargo, el uso de un proceso inadecuado del software puede reducir la calidad o la utilidad del producto de software que se va a desarrollar y/o incrementar los costes de desarrollo [31].

2.1.3 ¿Qué es un modelo de procesos del software?

Un modelo de procesos del software es una descripción simplificada de un proceso del software que presenta una visión de ese proceso. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería del software. Algunos ejemplos de estos tipos de modelos que se pueden producir son:

1. Un modelo de flujo de trabajo. Muestra la secuencia de actividades en el proceso junto con sus entradas, salidas y dependencias. Las actividades en este modelo representan acciones humanas.
2. Un modelo de flujo de datos o de actividad. Representa el proceso como un conjunto de actividades, cada una de las cuales realiza alguna transformación en los datos. Muestra cómo la entrada en el proceso, tal como una especificación, se transforma en una salida, tal como un diseño. Pueden representar transformaciones llevadas a cabo por las personas o por las computadoras.
3. Un modelo de rol/acción. Representa los roles de las personas involucrada en el proceso del software y las actividades de las que son responsables.

La mayor parte de los modelos de procesos del software se basan en uno de los tres modelos generales o paradigmas de desarrollo de software:

1. El enfoque en cascada. Considera las actividades anteriores y las representa como fases de procesos separados, tales como la especificación de requerimientos, el diseño del software, la implementación, las pruebas, etcétera. Después de que cada etapa queda definida «se firma» y el desarrollo continúa con la siguiente etapa.
2. Desarrollo iterativo. Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones muy abstractas. Éste se refina basándose en las peticiones del cliente para producir un sistema que satisfaga las necesidades de dicho cliente. El sistema puede entonces ser entregado. De forma alternativa, se puede reimplementar utilizando un enfoque más estructurado para producir un sistema más sólido y mantenible.
3. Ingeniería del software basada en componentes (CBSE). Esta técnica supone que las partes del sistema existen. El proceso de desarrollo del sistema se enfoca en la integración

de estas partes más que desarrollarlas desde el principio [31] .

2.1.4 Modelo big bang

Este modelo es el modelo con la forma más simple. Requiere poca planificación, mucha programación y también muchos fondos. Este modelo se conceptualiza alrededor de la teoría de creación del universo 'Big Bang'. Tal como cuentan los científicos, después del big bang muchas galaxias, planetas y estrellas evolucionaron. De la misma manera, si reunimos muchos fondos y programación, quizá podemos conseguir el mejor producto de software.

Para este modelo, se requiere poca planificación. No sigue ningún proceso concreto, y a veces el cliente no está seguro de las futuras necesidades y requisitos. Por tanto, la entrada o input respecto a los requisitos es arbitraria. Este modelo no es recomendable para grandes proyectos de software, pero es bueno para aprender y experimentar.

Es un proceso para usar en situaciones en donde los requerimientos y su periodo de lanzamiento son completamente flexibles, es importante también contar con clientes flexibles; las pruebas no son formales y si ocurren estas se dan antes de lanzar el producto [6].

Ventajas del modelo Big bang

- Modelo muy simple
- Poca o ninguna planificación requiere
- Fácil de manejar
- Da flexibilidad para los desarrolladores
- Buena ayuda al aprendizaje para recién graduados y estudiantes.

Desventajas del modelo big bang

- Muy alto riesgo e incertidumbre
- Mal modelo para los proyectos largos y continuos
- Es caro si se entiende mal los requisitos
- Mal modelo para los proyectos complejos y orientado a objetos

2.1.5 Modelo Cascada

El modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada.

El modelo de desarrollo en cascada se originó en la industria y la construcción, donde los cambios a posteriori son caros y difíciles de implementar. Cuando estás creando un producto material, realizar cambios en lo ya construido es mucho más difícil que en un programa informático. En el mundo del software, todavía no se habían implantado otras metodologías de desarrollo por lo que se adaptó el modelo en cascada que se utilizaba en otros sectores [6].

Fases del modelo cascada

El modelo de desarrollo en cascada sigue una serie de etapas de forma sucesiva, la etapa siguiente empieza cuando termina la etapa anterior.

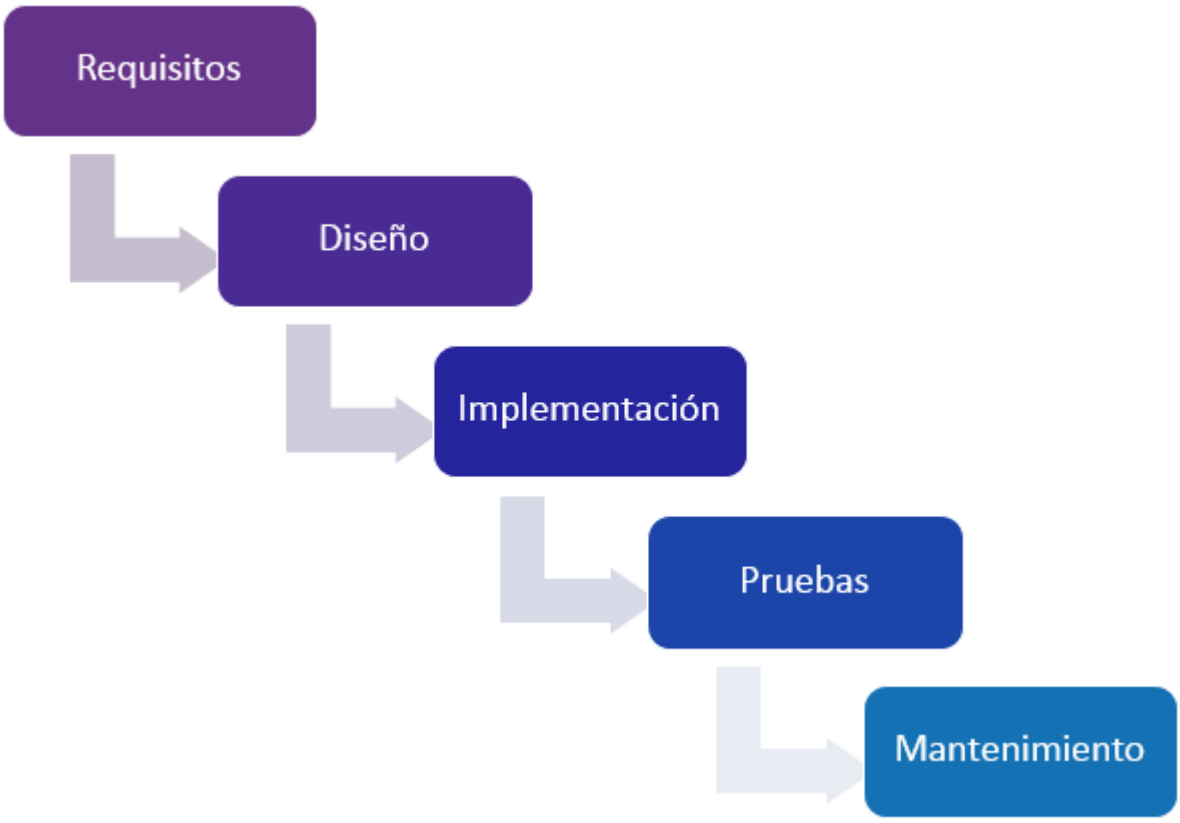


Figura 2.1: Modelo Cascada

Ventajas del modelo cascada

El modelo de cascada es el modelo más antiguo y más ampliamente utilizado en el campo de desarrollo de software. Hay ciertas ventajas del modelo de cascada, que hace que sea el modelo más ampliamente utilizado hasta el momento. Algunos de ellos se pueden enumerar como bajo.

- Documentación completa
- El plan completo se prepara al comienzo del proyecto
- Modelo bien conocido
- Fácil de medir el estado del proyecto

Desventajas del modelo cascada

- Requiere requisitos predefinidos, que son
- Muy difícil de obtener al comienzo del ciclo de desarrollo
- Resiste el cambio de requisitos, y estos los cambios son costosos en las últimas fases.

- Toma mucho tiempo entregar el software sistema
- Menos participación del usuario
- No es bueno para grandes proyectos
- Tiene una gran cantidad de riesgo

2.1.6 Modelo Espiral

El modelo en espiral (Boehm, 1986) es un modelo evolutivo del proceso de software que combina naturaleza iterativa de construcción de prototipos con los aspectos controlados y sistemáticos del modelo lineal secuencial representado en el modelo cascada [6]).

Boehm describe al modelo espiral como un generador de modelo de proceso impulsado por el riesgo, que se usa para guiar la ingeniería concurrente con participantes múltiples de sistemas intensivos en software [5]

El modelo espiral se basa en circuitos que contienen las siguientes fases:

1. Definición de objetivos: los objetivos específicos, alternativas y restricciones para la fase del proyecto son identificados.
2. Evaluación y reducción de riesgos: Los riesgos claves son identificados, analizados y la información es obtenida para reducir riesgos.
3. Desarrollo y validación: Un modelo apropiado es seleccionado para la siguiente fase de desarrollo.
4. Planeación: el proyecto es revisado y los planes son diseñados para la siguiente fase del modelo en espiral.

Como afirma [27] el primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del producto; las vueltas sucesivas se usan para desarrollar un prototipo y, luego, versiones cada vez más sofisticadas del software. Cada paso por la región de planeación da como resultado ajustes en el plan del proyecto. El costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega.

Ventajas del modelo espiral

- Los requisitos pueden ser evaluados y cambiados durante el desarrollo.
- Adecuado para proyectos complejos y grandes.
- Trata diferentes tipos de riesgos en cada ciclo, por lo tanto, los riesgos serían gradualmente reducidos.
- Un producto de software se entrega en breve tiempo.

Desventajas del modelo espiral

- Es un modelo costoso
- Necesita mayor experiencia para lidiar con el riesgo análisis
- No es bueno para pequeños proyectos

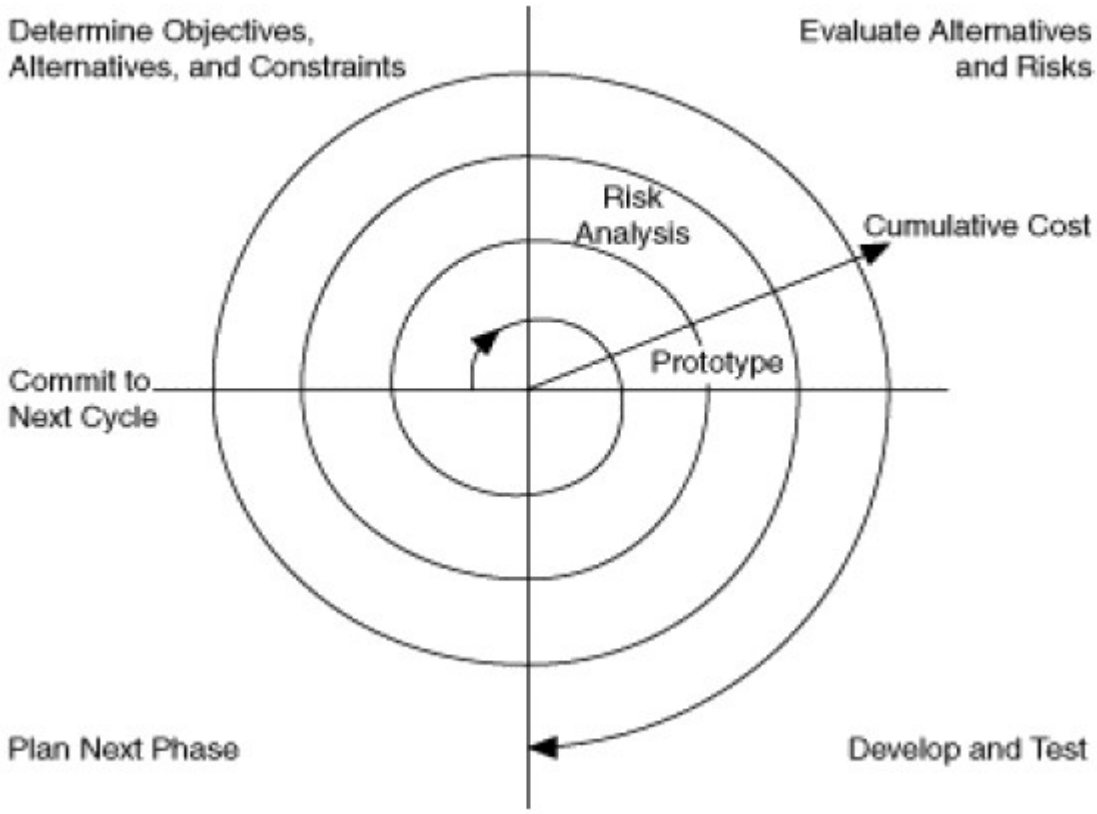


Figura 2.2: Modelo Espiral

2.1.7 Modelo en V

Enfatiza en el nivel abstracción que se maneja a lo largo del tiempo (Bolaños, 2016). Es un modelo que trata de esquematizar el desarrollo conforme se llevan a cabo las fases, contrastándolas con las pruebas. El gran aporte de este modelo de proceso está en el carácter paralelo de proceso que se establece entre el desarrollo y la prueba, ello implica romper también mitos como el de la realización de las pruebas, que bajo ese modelo se hacen desde el inicio del proyecto hasta el final.

Ventajas del modelo en V

- Específica bien los roles de los distintos tipos de pruebas a realizar.
- Hace explícito parte de la iteración y trabajo que hay que realizar.
- Este método involucra chequeos de cada una de las etapas del método Cascada.
- Es un método más robusto y completo que el método cascada y produce software de mayor calidad que con el modelo cascada.
- Es un modelo sencillo de y de fácil aprendizaje.
- Involucra al usuario en las pruebas.

Desventajas del modelo en V

- Es difícil que el cliente exponga explícitamente todos los requisitos.
- El cliente debe tener paciencia, ya que obtendrá el producto al final del ciclo de vida.
- El modelo no contempla la posibilidad de retornar a etapas inmediatamente anteriores, cosa que en la realidad puede ocurrir.
- Se pierde dinero, ya que, si algún proceso fue mal desarrollado, este debe ser revisado de nuevo, lo que puede traer como consecuencia un RollBack"de todo un proceso.
- Las pruebas pueden ser caras y a veces no lo suficientemente efectivas.

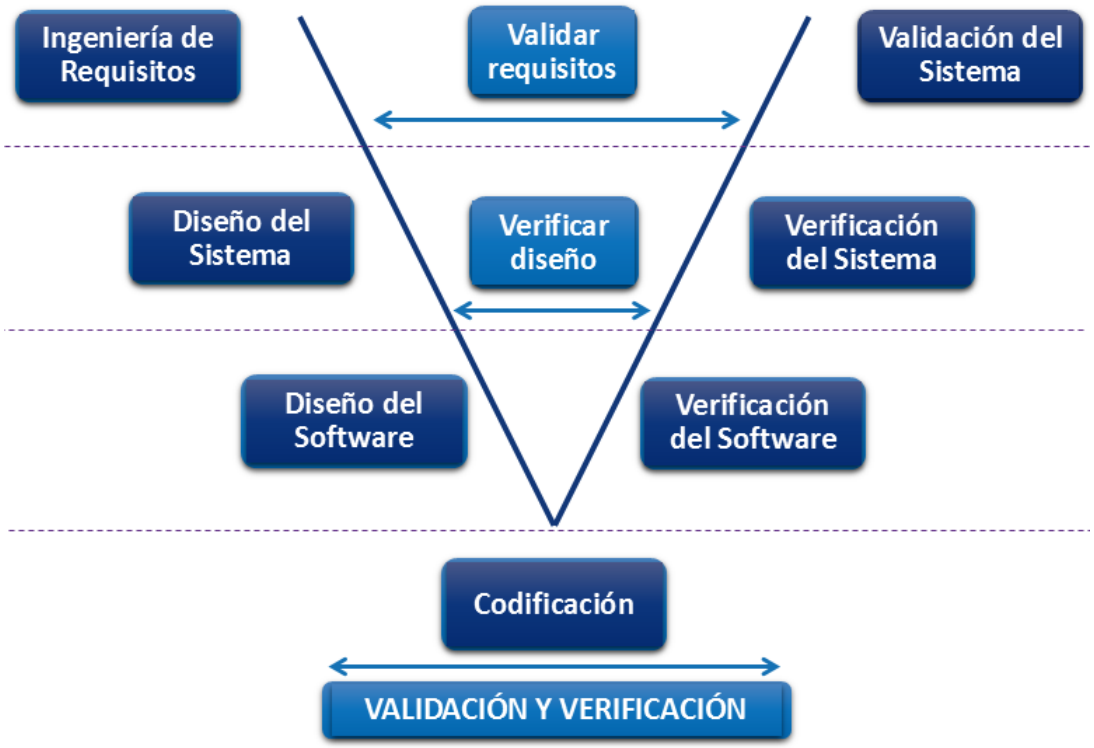


Figura 2.3: Modelo V

2.1.8 RUP

Propone una lista bidimensional de fases contra flujos de trabajo y propone un conjunto amplio de entregables, una de sus fortalezas radica en la estrecha relación que propone con el lenguaje de modelamiento UML, facilitando la realización de una documentación acompañada de artefactos de diseño trazables luego con facilidad a modelos ejecutables. El proceso de ciclo de vida de RUP se divide en cuatro fases bien conocidas llamadas Concepción, elaboración, construcción y transición. Esas fases se dividen en iteraciones, cada una de las cuales produce una pieza de software demostrable [6] .

Ventajas de RUP

- La ventaja principal de RUP es que se basa todo en las mejores prácticas que se han intentado y se han probado en el campo.
- Es el proceso de desarrollo más general de los existentes actualmente.
- Es una forma disciplinada de asignar tareas y responsabilidades en una empresa de desarrollo (quién hace qué, cuándo y cómo).
- Permite tener claro y accesible el proceso de desarrollo que se sigue.

Ventajas de RUP

- Método pesado.
- Por el grado de complejidad puede ser no muy adecuado.
- En proyectos pequeños, es posible que no se puedan cubrir los costos de dedicación del equipo de profesionales necesarios.
- Es mal aplicado en el estilo cascada.

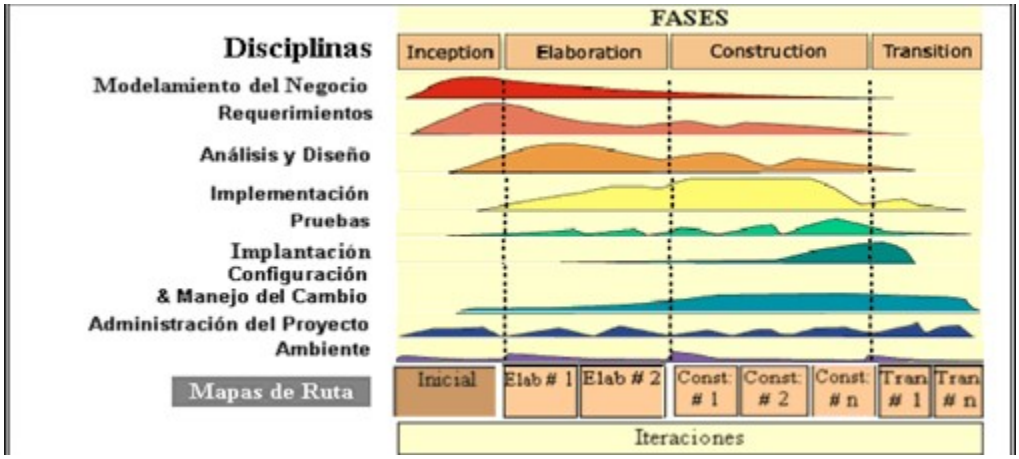


Figura 2.4: RUP

2.1.9 Métrica V3

Apoya el desarrollo de sistemas de información para asegurar sus objetivos de calidad, costo y plazos. Métrica 3 detalla participantes, productos de entrada y de salida, así como las técnicas y prácticas para su obtención. Métrica 3 tiene en cuenta estándares de ingeniería de software, calidad, seguridad y gestión de proyectos. Esta metodología facilita a través de interfaces la realización de los procesos de apoyo y organizativos.

Se enfoca en los siguientes objetivos:

- Proporcionar o definir sistemas de información con un marco estratégico para el desarrollo de los mismos.
- Dotar a la organización de productos de software que satisfagan las necesidades de sus usuarios.

- Mejorar la productividad de los departamentos TI, permitiendo una mayor capacidad de adaptación a los cambios.
- Facilita la comunicación y entendimiento entre los participantes en la producción de software a lo largo del ciclo de vida del proyecto.
- Facilitar la operación, mantenimiento y uso de los productos de software obtenidos.

¿Cuándo usar Métrica V3?

- Proyectos encargados por clientes, que no tienen suficientemente claro lo que desean, ni lo que esperan.
- Proyectos con requisitos iniciales inestables, cuyos cambios puedan suponer un alto impacto, y grandes desviaciones en los plazos de un proyecto.
- Proyectos en los que Métrica v3 es requisito no funcional del cliente (normalmente Administraciones Públicas).

Ventajas de Métrica V3

- Cuatro interfaces que definen actividades orientadas a la mejora y perfeccionamiento de los procesos principales para garantizar la consecución del objetivo del desarrollo.
- Cubre distintos tipos de desarrollo, mejorar la productividad de los departamentos de sistemas y tecnologías de la Información y las comunicaciones, permitiendo una mayor capacidad de adaptación a los cambios y las comunicaciones
- Proporcionar o definir sistemas de Información que ayuden a conseguir los fines de la organización mediante la definición de un marco estratégico para el desarrollo de los mismos.

Desventajas de Métrica V3

- Es demasiado pesada tanto en su implementación
- Se mantiene algunos factores de las anteriores versiones

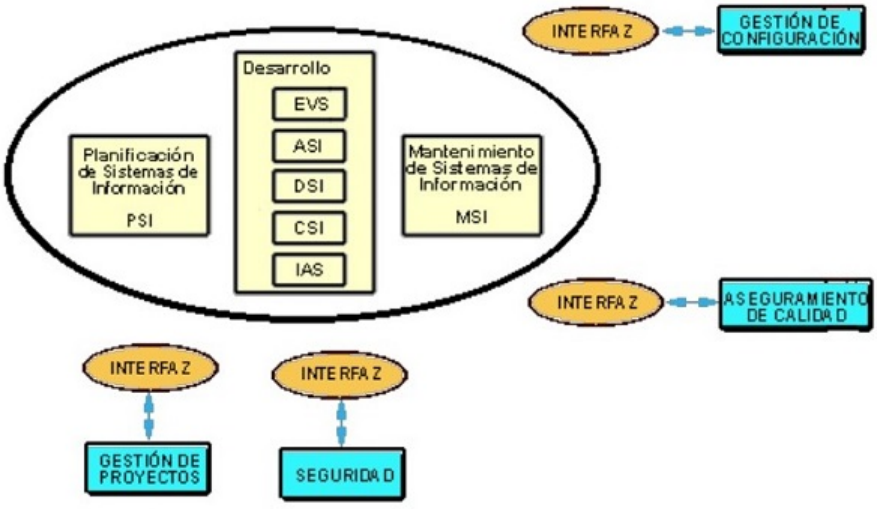


Figura 2.5: Metrica 3

3. Estado del arte

Para el desarrollo de este proyecto se abordan trabajos, libros y tesis que hayan realizado investigaciones acerca de los procesos de software y cómo pueden llegar a implementarse en las organizaciones, que factores, recursos o datos de entrada se tuvieron en cuenta para obtener el resultado esperado.

3.1 Proyectos de Software

A pesar de que la ingeniería del software fue definida en el año 1968, de que los primeros procesos formales de desarrollo de software como cascada datan de 1970, de que en los últimos 40 años se han desarrollado procesos de desarrollo de software (como el espiral, modelo en V, RUP, Métrica, entre otros), metodologías ágiles (como XP, Scrum, ASD, Crystal, entre otras) y el cuerpo de conocimiento de la Ingeniería de Softwrae (SWEBOK) y un gran etcétera, los proyectos de desarrollo de software tienen una tasa inferior al 30 % de éxitos.

Es así que de acuerdo al Standish Group [18] la tasa de éxito para proyectos de desarrollo de software en eñ 2015 fue del 29 % (entregados a tiempo, de acuerdo al presupuesto y con resultados satisfactorios), 52 % fueron discutidos (hay dudas sobre si tuvieron éxito o fracasaron) y 19 % fueron fallidos (ver Figura 3.1).

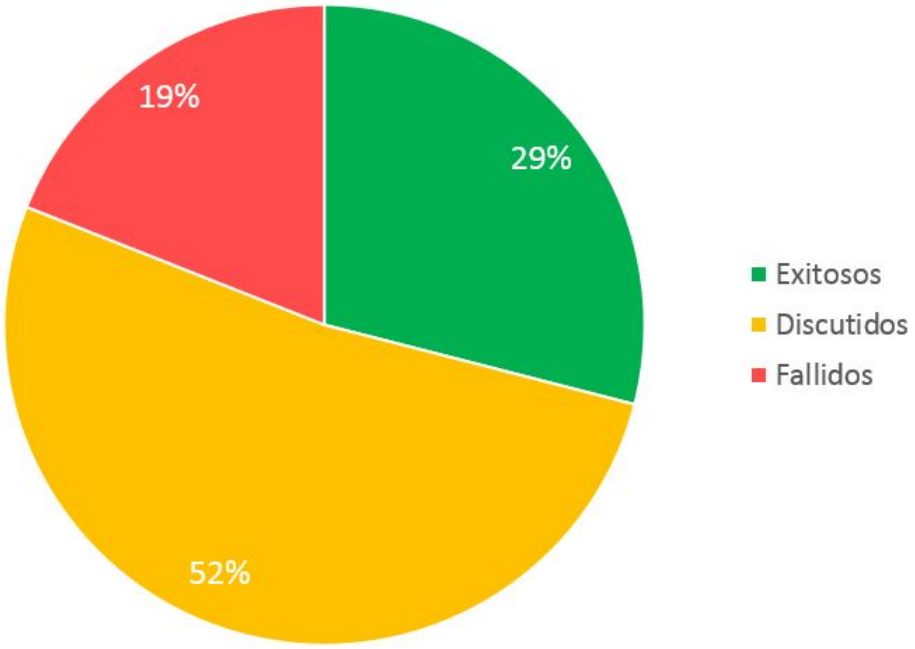


Figura 3.1: Resolución de proyectos de software del año 2015

Sin embargo, las estadísiticas recopiladas por el Standish Group en su Chaos Report desde 1994 hasta el 2015 (ver Figura 3.2) muestran que estos resultados han mantenido las tendencias y de que aún se requieren muchos esfuerzos en la Ingeniería del Software para mejorar las tasas de proyectos de software exitosos [25].

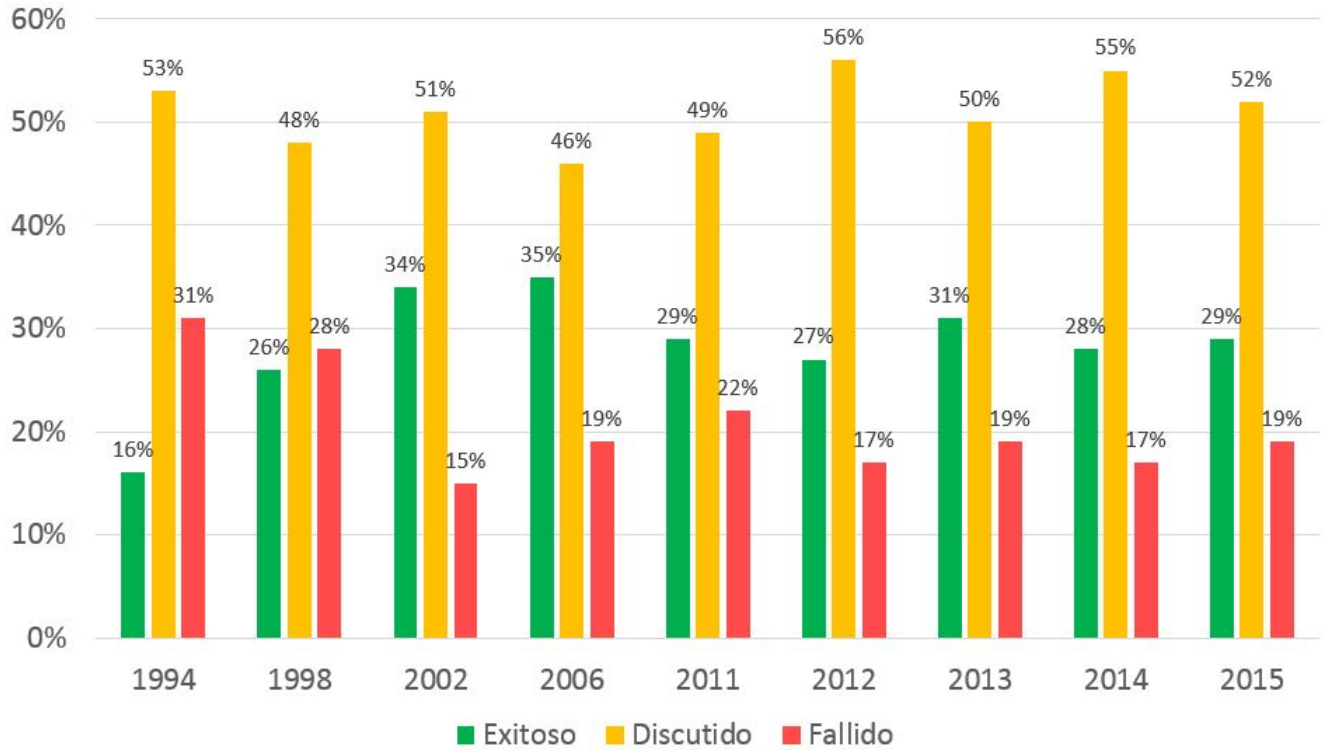


Figura 3.2: Resolución de proyectos de software entre 1994 y 2015

Otro aspecto importante a considerar y como lo menciona Jennifer Lynch [25], es que mientras más pequeño es el proyecto de software tienen una probabilidad de éxito mayor que los grandes (ver Figura 3.3).

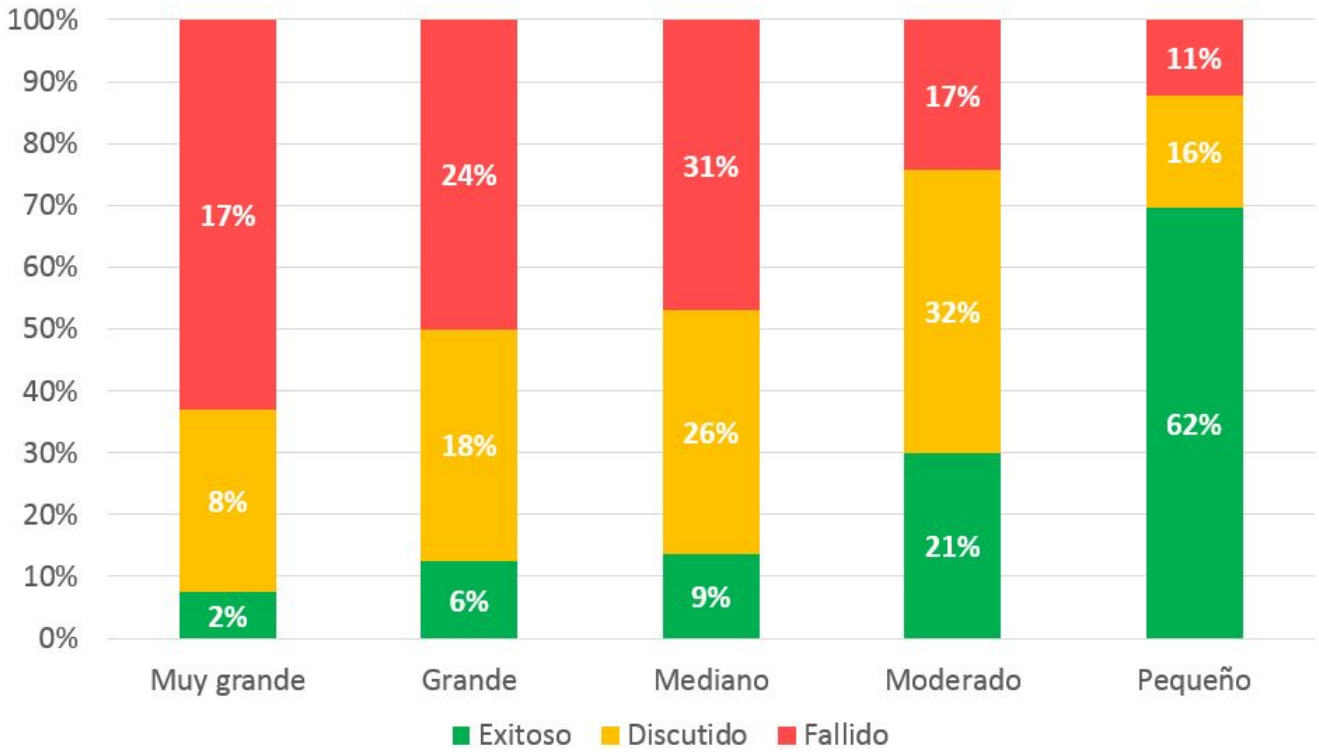


Figura 3.3: Resolución de proyectos por tamaño en 2015

Por otra parte, Galorath [15] menciona que Tata Consulting Services reportó en 2007 lo siguiente con respecto a sus proyectos de software:

- 62 % no cumplieron el cronograma
- 49 % Sobrepasaron el presupuesto
- 41 % Fallaron en la entrega del ROI (retorno de la inversión)

Estos resultados muestran el alto porcentaje de proyectos que fallaron debido a que sobrepasaron los tiempos de entrega y el presupuesto asignado.

De acuerdo a [15] una investigación de la ESSU (European Services Strategy Unit) [34] identifica 105 conratos de outsourcing en las TIC en el goierno central británico, encontrando la siguiente estadística:

- Valor total de los contratos: 29.5 mil millones de libras esterlinas
- Sobrecostos totales: 9.0 mil millones de libras esterlinas
- 57 % de los contratos experimentaron sobrecostos
- El porcentaje aproximado de sobrecostos fue de 30.5 %
- 33 % de los contratos sufrieron grandes retrasos
- 30 % de los contratos fueron terminados

3.1.1 Factores de éxito en los proyectos de software

Jennifer Lynch [25] lista los factores de éxito en los proyectos de software de acuerdo al Reporte Chaos del Standish Group, definiéndolos así:

- **Soporte Ejecutivo:** cuando un ejecutivo o un grupo de ejecutivos acuerdan brindar respaldo financiero y emocional. El ejecutivo o los ejecutivos alentarán y ayudarán en la finalización exitosa del proyecto.
- **Madurez emocional** es el conjunto de comportamientos básicos de cómo las personas trabajan juntas. En cualquier grupo, organización o empresa, es la suma de sus habilidades y el eslabón más débil que determina el nivel de madurez emocional.
- **Participación del usuario:** se lleva a cabo cuando los usuarios participan en el proceso de toma de decisiones y recopilación de información del proyecto. Esto también incluye comentarios de los usuarios, revisión de requisitos, investigación básica, creación de prototipos y otras herramientas de creación de consenso.
- **Optimización** es un medio estructurado para mejorar la efectividad del negocio y optimizar una colección de muchos proyectos pequeños o requisitos importantes. La optimización comienza con la administración del alcance en función del valor comercial relativo.
- **Personal calificado** son personas que entienden tanto el negocio como la tecnología. Un personal capacitado es altamente competente en la ejecución de los requisitos del proyecto y la entrega del proyecto o producto.
- **SAME** es el entorno de gestión arquitectónica estándar. El Standish Group define SAME como un grupo consistente de prácticas integradas, servicios y productos para desarrollar, implementar y operar aplicaciones de software.
- **Competencia ágil** significa que el equipo ágil y el propietario del producto son expertos en el proceso ágil. La competencia ágil es la diferencia entre buenos resultados ágiles y malos resultados ágiles.
- **Modesta ejecución** está teniendo un proceso con pocas partes móviles, y esas partes están automatizadas y simplificadas. La ejecución modesta también significa usar herramientas de administración de proyectos con moderación y solo unas pocas características.
- **Experiencia en Gestión de Proyectos** es la aplicación de conocimientos, habilidades y técnicas para proyectar actividades con el fin de cumplir o superar las expectativas de las partes interesadas y generar valor para la organización.
- **Objetivos comerciales claros** es la comprensión de todos los interesados y participantes en el propósito comercial para ejecutar el proyecto. También podría significar que el proyecto se está alineando con los objetivos y la estrategia de la organización.

3.2 Mejoramiento del proceso de software

- ¿Qué es? El mejoramiento del proceso de software abarca un conjunto de actividades que conducirán a un mejor proceso de software y, en consecuencia, a software de mayor calidad y a su entrega en forma más oportuna.

- ¿Quién lo hace? El personal que impulsa el MPS (mejoramiento de proceso de software) proviene de tres grupos: gerentes técnicos, ingenieros de software e individuos que tienen responsabilidad en el aseguramiento de la calidad.
- ¿Por qué es importante? Algunas organizaciones de software tienen poco más que un proceso de software ad hoc. Conforme trabajan para mejorar sus prácticas de ingeniería del software, deben abordar las debilidades en sus procesos existentes e intentar mejorar su enfoque para el trabajo de software.
- ¿Cuáles son los pasos? El enfoque del MPS es iterativo y continuo, pero puede verse en cinco pasos: valoración del proceso de software actual, educación y capacitación de profesionales y gerentes, selección y justificación de elementos de proceso, métodos de ingeniería del software y herramientas, implementación del plan MPS y evaluación y afinación con base en los resultados del plan.
- ¿Cuál es el producto final? Aunque existen muchos productos operativos MPS intermedios, el resultado final es un proceso de software mejorado que conduce a software de mayor calidad.
- ¿Cómo me aseguro de que lo hice bien? El software que produce su organización se entregará con menos defectos, se reducirá la repetición del trabajo en cada etapa del proceso de software y la entrega oportuna será mucho más probable [27].

3.3 Calidad del producto de software

Como lo plantea [31] “La calidad del producto se ve afectada si un proyecto, independientemente de su tamaño, está mal presupuestado o planificado con un tiempo de entrega irreal. Un buen proceso requiere recursos para su implementación efectiva. Si estos recursos son insuficientes, el proceso no puede desarrollarse de forma efectiva. Si los recursos no son adecuados, sólo las personas excelentes pueden salvar el proyecto. Más aún, si el déficit es demasiado grande, la calidad del producto se degradará. A menudo, la causa real de los problemas en la calidad del software no es la mala gestión, los procesos inadecuados o la poca calidad de la capacitación. Más bien, es el hecho de que las organizaciones deben competir para sobrevivir.

Muchos proyectos de software infravaloran el esfuerzo o prometen una entrega rápida con el fin de conseguir el contrato de desarrollo. En un intento de mantener estos compromisos, la compañía podría sacrificar la calidad del software.”

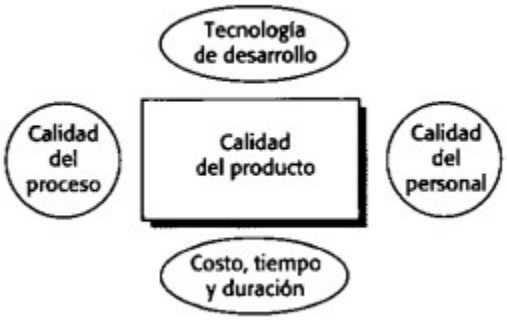


Figura 3.4: Calidad de software

Los modelos formales sirven como un punto de inicio útil para el análisis de procesos. Sin embargo, raramente incluyen suficiente detalle o reflejan las actividades reales del desarrollo de software. Los modelos de procesos formales son más abstractos y sólo definen las actividades y productos a entregar del proceso principal. Por lo general, es necesario mirar «hacia adentro» del modelo para descubrir los procesos reales. Más aún, los procesos reales que se siguen a menudo difieren significativamente de los modelos formales, aunque por lo general se tengan que desarrollar los productos a entregar. Las técnicas de análisis de procesos comprenden:

1. Cuestionarios y entrevistas. A los ingenieros que trabajan en un proyecto se les pregunta sobre lo que sucede realmente. Las respuestas a un cuestionario formal se refinan durante las entrevistas personales con los involucrados en el proceso.
2. Estudios etnográficos. Los estudios etnográficos se utilizan para comprender la naturaleza del desarrollo de software como una actividad humana. Tal análisis revela la sutileza y complejidades no descubiertas por otras técnicas.

Después de revisar algunas literaturas, se ha observado que sufren de varias debilidades. Primero que nada, la generalización de los resultados, donde concluyen que es ágil el método más eficiente para el proceso de desarrollo de software. Sin embargo, esto depende de muchos factores, como el tamaño del equipo, interacción, colaboración y el dominio del proyecto también.

En segundo lugar, la mayoría de las revisiones de la literatura comparan ágil con revisiones tradicionales en general sin considerar escenarios, encuestas o entrevistas para los encuestados que tienen experiencia en metodologías y software tradicionales basados en planes procesos de desarrollo.

3.4 Modelos de selección de procesos de software

A continueación se presentan las investigaciones, trabajos, propuestas y modelos más relevantes que se han realizado a lo largo de la historia de la Ingeniería del Software para la selección de procesos de desarrollo de software.

3.4.1 Criterio para selección de procesos de software de Alexander y Davis

Em 1991 Linda Alexander y Alan Davis [2] diseñaron un modelo para la selección de proceso de software, basado en 20 criterios, 3 clases de procesos y 3 procesos de software. Dentro de los criterios se encuentran:

1. Experiencia del usuario
2. Expresión del usuario
3. Experiencia de los desarrolladores en el dominio de la aplicación
4. Experiencia de los desarrolladores en la ingerniería de software
5. Madurez de la aplicación
6. Complejidad del problema
7. Requerimiento de funcionalidades parciales
8. Frecuencia de los cambios
9. Magnitud de los cambios
10. Tamaño del producto
11. complejidad del producto
12. Requerimientos de garantía
13. Requerimiento de interfaz de usuario
14. Perfil de los fondos
15. Disponibilidad de fondos
16. Perfil del equipo de desarrollo
17. Disponibilidad del equipo de desarrollo
18. Accesibilidad de los usuarios
19. Compatibilidad de la organización y lo procesos de software
20. Compatibilidad de la organización, la gestión de la configuración y el aseguramiento de la calidad

Las clases de procesos de software disponibles en el modelo son:

- Convencional
- Incremental
- Evolución

Los procesos de software disponibles en el modelo de Alexander y Davis son:

- Cascada
- Prototipado Híbrido
- Especificación Operacional

Este modelo cuenta con un amplio número de criterios de evaluación, sin embargo, tiene como desventajas que es muy antiguo, por lo que carece de caracterísiticas muy relevantes en el desarrollo de software actual: gestión de riesgos, documentación del proyecto, reusabilidad de software y tamaño del equipo de desarrollo.

Por otra parte, es muy limitado el alcance del modelo ya que incluye procesos de software muy utilizados en la época: cascada, prototipado híbrido y especificación operacional, dejando de lado procesos de software como el modelo RUP, Métrica V3 y Espiral, de amplio uso en la industria del desarrollo de software.

3.4.2 Otros modelos para la selección de procesos de software

Y. Leau, W. Loo, W. Tham, y S. Tan [23] explicaron las ventajass y desventajas de los tradicionales y las metodologías ágiles. Ellos entregaron una forma de selección del proceso de desarrollo adecuado de acuerdo al tamaño del proyecto, equipo y otros factores. Sin embargo, sus conclusiones carececen de evidencias y detalles [20].

M. Stoica, M. Mircea, y B. Ghilic-Micu [32] explicaron las ventajas y limitaciones de algunos métodos tradicionales como el cascada y el método incremental. Discutieron algunas tecnologías para apoyar a las organizaciones a convertirse en ágiles. Luego, diseñaron un criteeio para la selección del proceso adecuado de acuerdo a equipos, requerimeintos, escala del proyecto, costos y objetivos del proyecto [20].

N. Russo [29] conluyó basado en una encuesta en más de 100 organizaciones que los tres factores más importantes para la selección y uso de las las metodologías de software eran: técnicas para el desarrollo estructurado, políticas/procedimientos corporativos bien definidos y compartir información entre desarrolladores [16].

A. Cockburn [11] identifica dos factores que impactan sobre la metodología apropiada de desarrollo de software: las prioridades del proyecto y las particularidades de la metodologías de los diseñadores. Sin embargo, esas investigaciones no analizan procesos específicos de desarrollo de software con respecto al impacto en los criterios mencionados para seleccionar el proceso o metodologías más acorde a los proyectos de software [16].

4. Desarrollo de la investigación

Uno de los objetivos de este capítulo de investigación, es detectar que tanta relevancia tiene los procesos de desarrollo de software para las empresas colombianas, bien sean pymes o grandes organizaciones que se dedican o dentro de sus procedimientos se atiende la elaboración, creación y mantenimiento de desarrollo de software.

Las pymes en Colombia y Bogotá representan la gran mayoría de proyectos en el área de TI, no solo eso, también representan la mayor generación de empleo en el país. Gracias a esto el crecimiento en innovación en Colombia trae beneficios a la sociedad, de manera que estos nuevos ítems y sistemas adquiridos traen desarrollo no solo tecnológico, traen desarrollo, económico, industrial, expansión de mercados.

Como observamos “En Colombia la industria de software tiene el potencial de convertirse en un sector económico de gran impacto, que pueda incluso suplir la demanda interna y llegar con éxito a los mercados del exterior. También, por ser este un sector que involucra el manejo transversal a nivel empresarial, puede apoyar y brindar soporte, agilizando los procesos, facilitando la comunicación y reduciendo los costes de operación.” [1].

Para validar directamente conceptos anunciados anteriormente, se aplica una encuesta a diferentes profesionales que trabajan en el sector tecnológico, donde cada uno de ellos pertenecen a distintas empresas, dándonos a conocer sus conceptos con respecto al desarrollo de software y que tan familiarizados están estos conceptos con los procesos de desarrollo de software, directa o indirectamente en cada una de las organizaciones a las cuales pertenece cada profesional.

4.1 Situación actual de los proyectos de software en Colombia

A continuación, se presenta una encuesta realizada a diferentes profesionales del área de TI. La cual, consta de 10 sencillas preguntas y busca conocer el estado actual de los procesos de software en las empresas colombianas. Seguidamente se validarán y se analizarán las respuestas de cada interrogante, junto con otras opiniones que se traerán a este modelo.

4.1.1 Pregunta 1: Profesión de los encuestados

Observamos la variedad en los distintos cargos que pueden verse relacionados con los modelos o procesos de desarrollo de software.

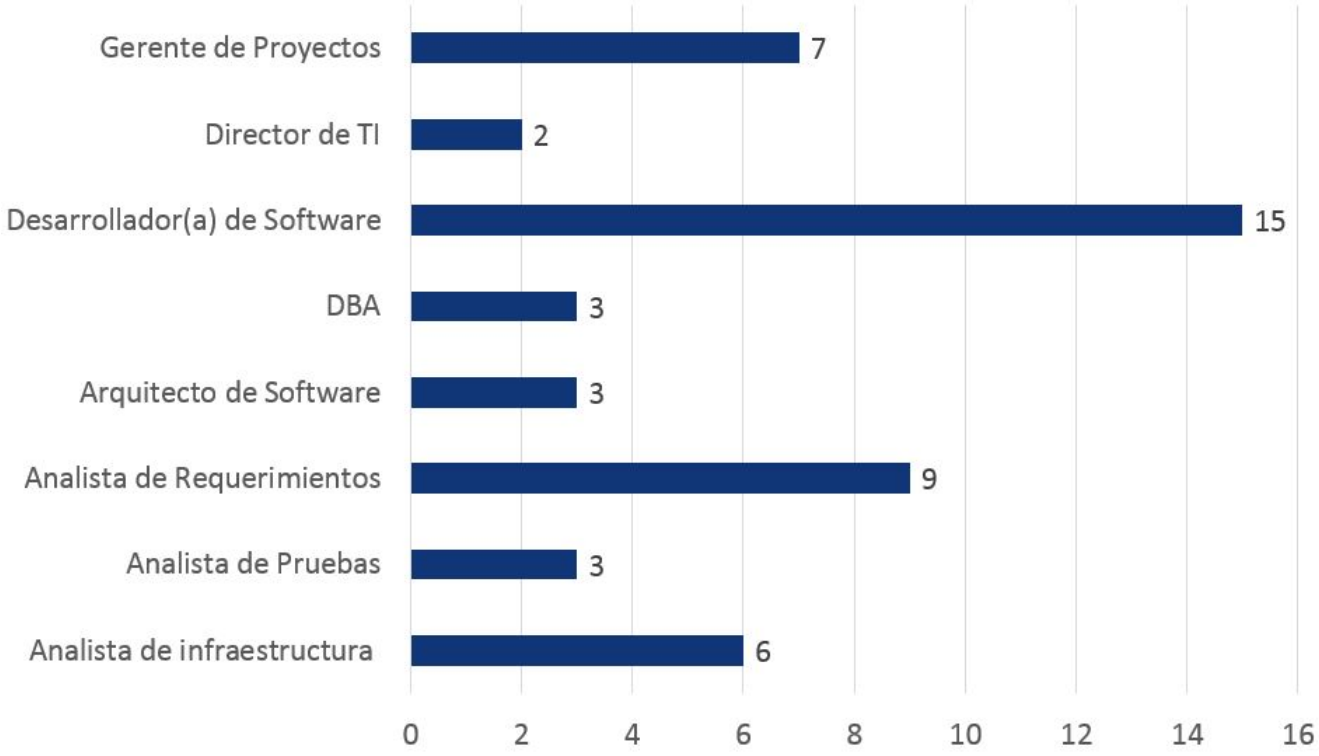


Figura 4.1: Pregunta 1

4.1.2 Pregunta 2: Proyectos de software exitosos

A la pregunta ¿Cuál es el porcentaje de proyectos de software exitosos en su empresa?, El 41 % de los encuestados esta entre los 76 y 100 % de éxito, 33 % de los encuestados pronuncia el 51 a 75 % el 12 % piensa que se está entre 26 a 50 %, el otro 12 % restante esta con el 0 y 25

De la información anterior, analizamos que a pesar que la mayoría de los desarrollos de software en las empresas son exitosos, la sumatoria del total que no son exitosos son mayoría a los que sí. Lo anterior nos muestra que se requiere tener un mejor control y planeación para que sean exitosos los proyectos.

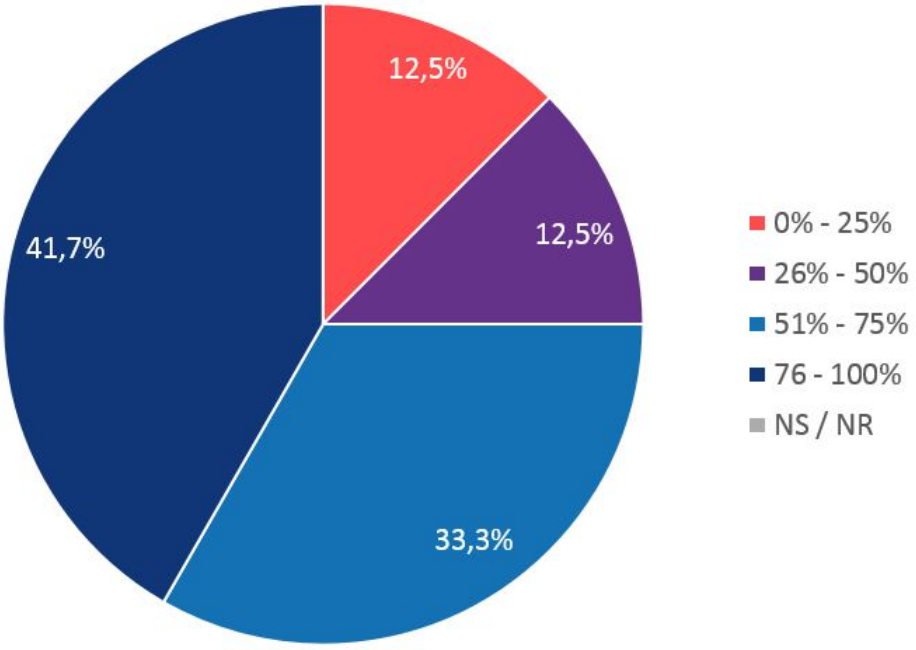


Figura 4.2: Pregunta 2

4.1.3 Pregunta 3: Procesos de desarrollo de software en la empresa

A la pregunta ¿En su empresa se siguen procesos de desarrollo de software?, El 83 % de los encuestados responde “si”, 12 % No sabe o no responde.

De lo anterior inferimos que aun vemos desconocimiento en los temas en el uso de procesos de desarrollo de software.

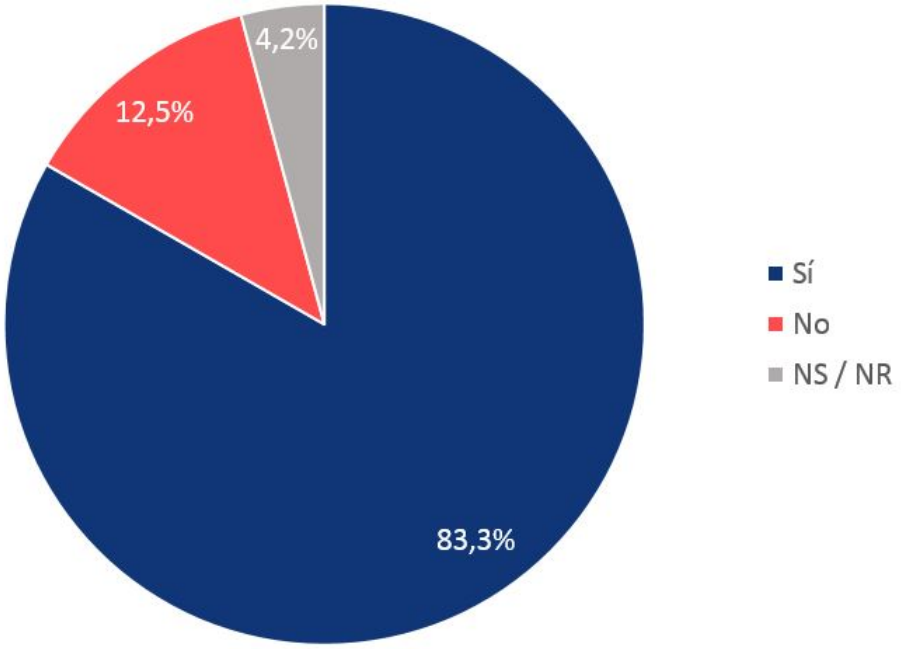


Figura 4.3: Pregunta 3

4.1.4 Pregunta 4: Uso de procesos de desarrollo de software en la empresa

A la opción, Seleccione los procesos de desarrollo de software aplicados en su empresa. Observamos que las metodologías agiles tienen gran auge en los proyectos, pues aún falta más conocimientos en el uso de procesos de software.

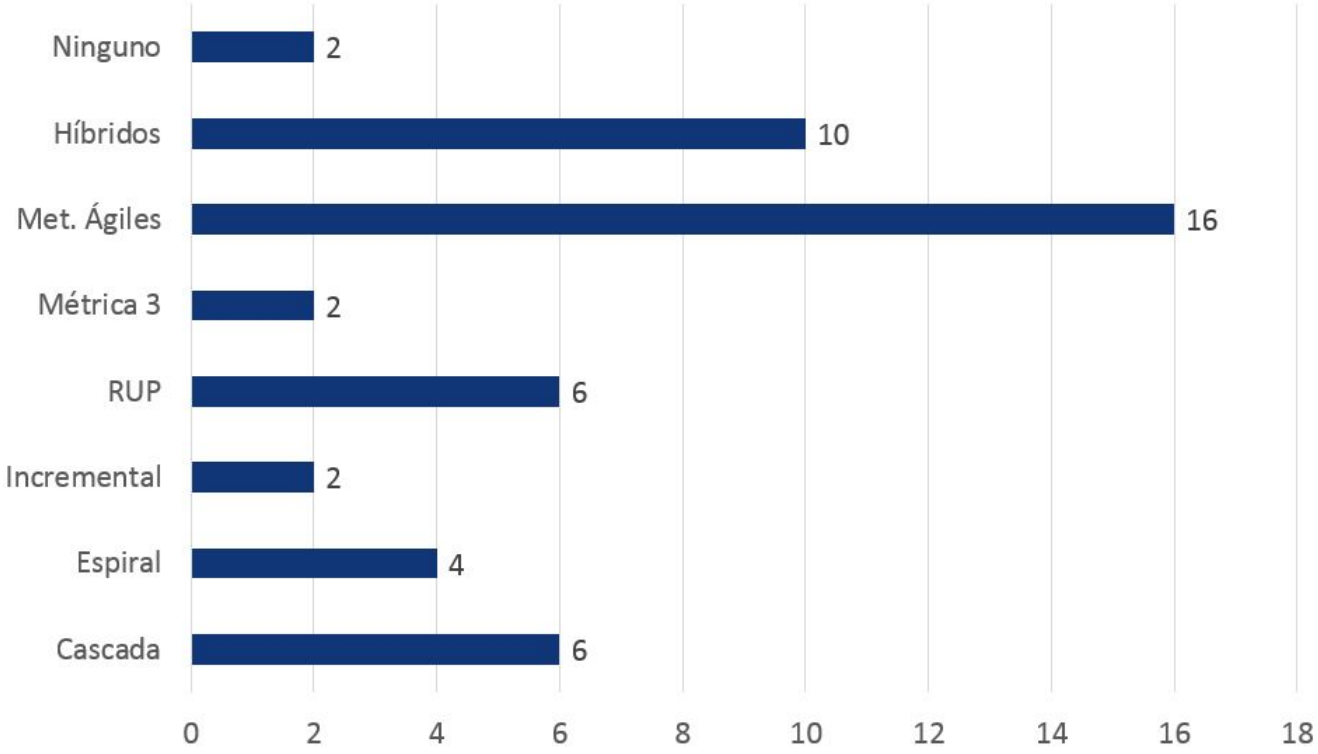


Figura 4.4: Pregunta 4

4.1.5 Pregunta 5: Documentación de los proyectos de desarrollo de software

A las opciones de la siguiente selección, Considera que la documentación de los proyectos de software es, miramos que, aunque la documentación en todo proyecto de importante, también es cierto que dependiendo de las necesidades de la empresa puede darse más importancia o no a este ítem.

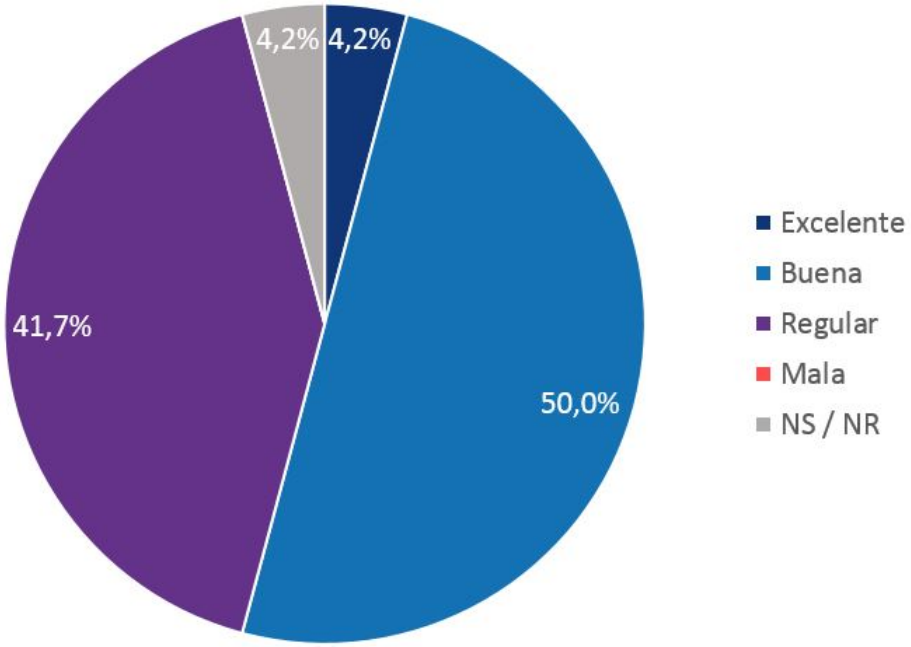


Figura 4.5: Pregunta 5

4.1.6 Pregunta 6: Retrasos en tiempos de entrega de proyectos

A la pregunta ¿Cuántos proyectos de software de la empresa han sufrido retrasos en los tiempos de entrega?, El 20 % de los encuestados esta entre los 76 y 100 % de éxito, 33 % de los encuestados pronuncia el 51 a 75 % el 16 % piensa que se está entre 26 a 50 %, el otro 25 % restante esta con el 0 y 25 %.

Se ven opiniones divididas, aunque partimos del hecho de que el éxito de los proyectos de software se basa en la buena planeación inicial, y esto incluye los tiempos, vemos gran cantidad en las opiniones acerca del retraso en los proyectos.

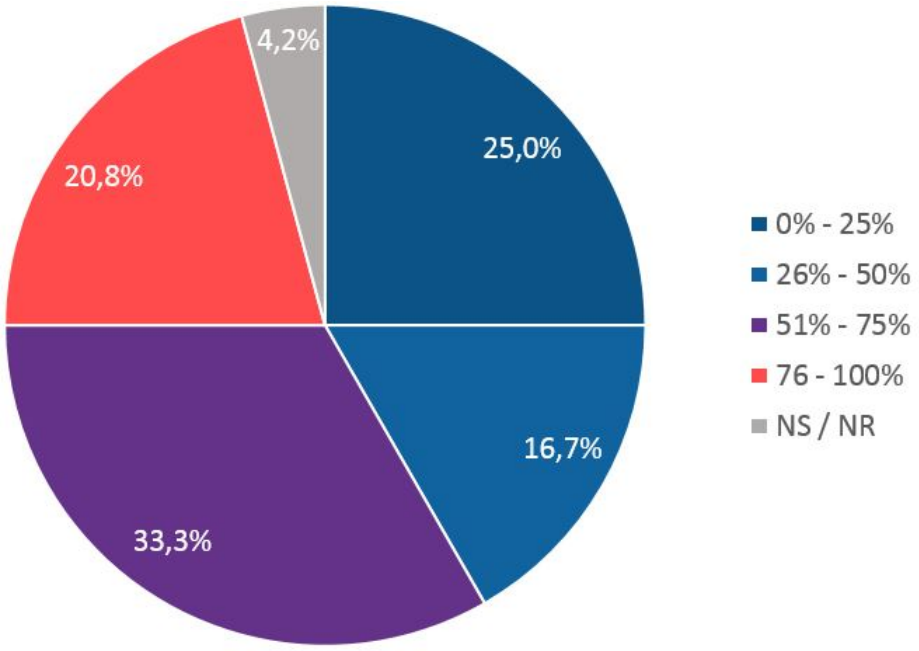


Figura 4.6: Pregunta 6

4.1.7 Pregunta 7: Problemas de calidad en el software

A la pregunta ¿Cuál es el porcentaje de los productos de software han presentado problemas de calidad?, la minoría de los encuestados esta entre los 76 y 100 % de éxito, 33 % de los encuestados pronuncia el 51 a 75 % el 25 % piensa que se está entre 26 a 50 %, el otro 33 % restante esta con el 0 y 25 %.

La calidad en cualquier producto es la presentación de la empresa.

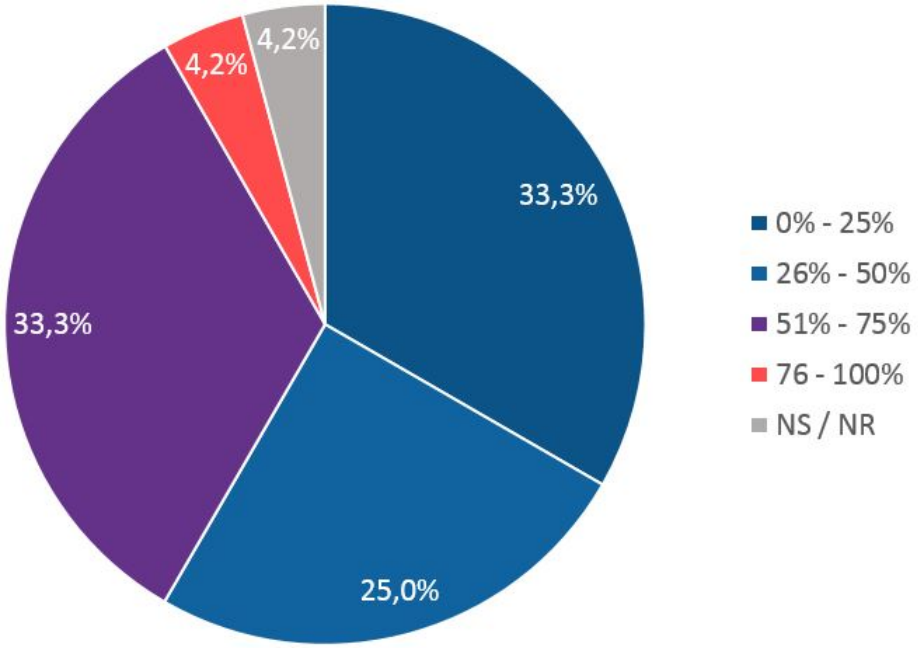


Figura 4.7: Pregunta 7

4.1.8 Pregunta 8: Sobrecostos en los proyectos de software

A la pregunta ¿Qué porcentaje de proyectos de software han sufrido sobrecostos?, la minoría de los encuestados esta entre los 76 y 100 % de éxito, 25 % de los encuestados pronuncia el 51 a 75 % el 29 % piensa que se está entre 26 a 50 %, el otro 33 % restante esta con el 0 y 25 %.

La calidad, tiempo y costos son directamente proporcionales unos de los otros, por ende, cuando falle uno de estos, los otros se verán afectados directamente, lo anterior puede ser observado por los funcionarios internos de la compañía y en ocasiones también por clientes.

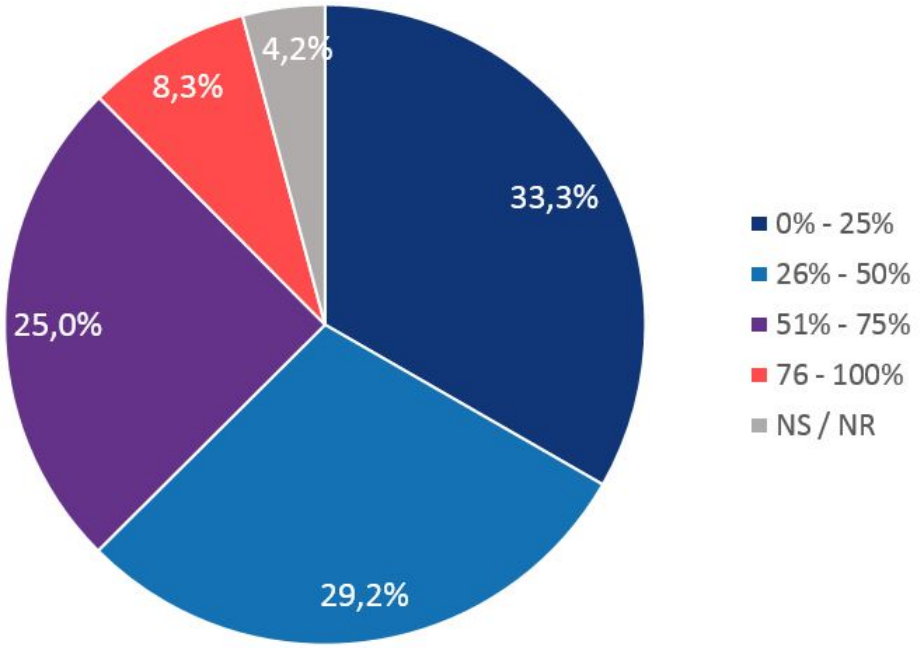


Figura 4.8: Pregunta 8

4.1.9 Pregunta 9: Definición de requerimientos en los proyectos de software

A la pregunta ¿Qué porcentaje de proyectos de software han tenido errores en la definición de requerimientos?, El 12 % de los encuestados esta entre los 76 y 100 % de éxito, 16 % de los encuestados pronuncia el 51 a 75 % el 41 % piensa que se está entre 26 a 50 %, el otro 25 % restante esta con el 0 y 25 %.

Los requerimientos iniciales para muchos modelos es la espina a seguir en los proyectos, depende del tipo de empresa y negocio que sigan las diferentes compañías para poder aplicar bien o no este ítem.

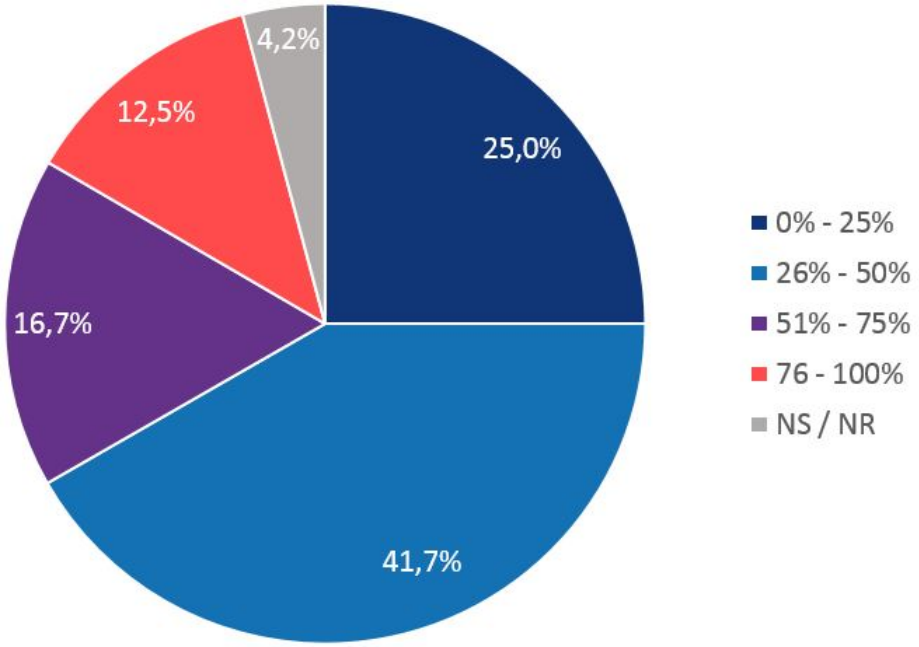


Figura 4.9: Pregunta 9

4.1.10 Pregunta 10: Importancia de los procesos de software

A la pregunta ¿Considera que un proceso de software contribuye a mejorar la calidad, tiempos de entrega y reducir los costos del software?, El 87 % de los encuestados responde "si", el otro 12 % restante responde "tal vez".

Esta pregunta a pesar de que es básica e intuitiva nos demuestra que aún falta credibilidad en el uso de los procesos de software.

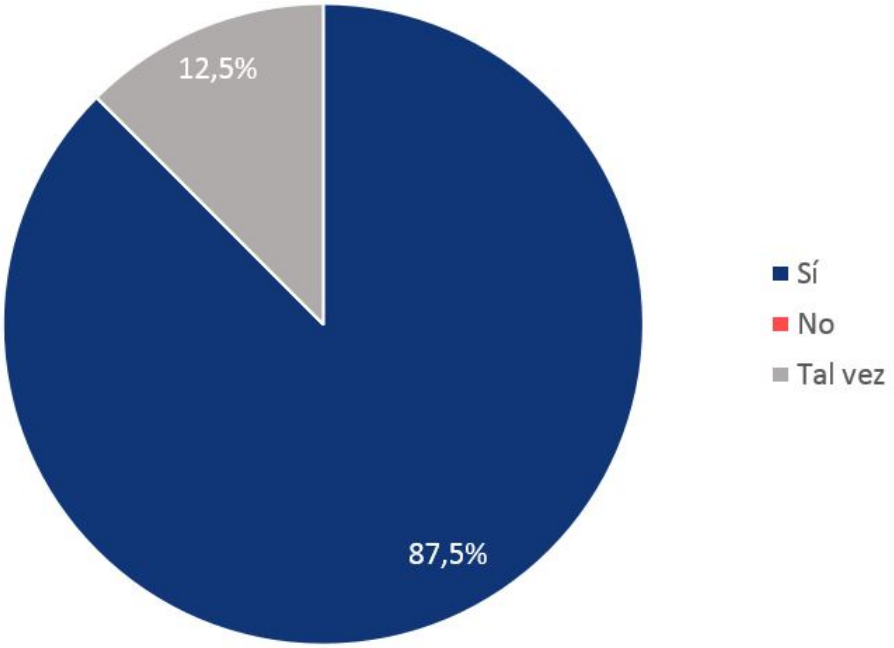


Figura 4.10: Pregunta 10

III Resultado de la investigación

5 Modelo para la selección de procesos de software 63

5.1 Variables del modelo de selección de procesos de software
5.2 Criterios de evaluación
5.3 Evaluación de criterios en el modelo cascada
5.4 Evaluación de criterios en el modelo en V
5.5 Evaluación de criterios en el modelo espiral
5.6 Evaluación de criterios en el modelo RUP
5.7 Evaluación de criterios en el modelo en Métrica V3
5.8 Ejemplo de aplicación del modelo de selección de procesos de software
5.9 Calculo proyecto ejemplo en modelos de proceso

5. Modelo para la selección de procesos

El modelo de selección de procesos de desarrollo de software se basa en la definición de 14 criterios, a través de los cuales el equipo de expertos en desarrollo de software (gerentes de proyecto, equipo de desarrollo de software, etc.) analizan el proyecto, el producto, las personas y el proceso involucrados en el desarrollo de software para que el modelo genere la mejor recomendación del proceso de desarrollo de software para lograr los objetivos del proyecto.

5.1 Variables del modelo de selección de procesos de software

5.1.1 Criterios de evaluación

C_i = Los criterios de evaluación relacionados con el desarrollo de software.
C_1 = Complejidad del proyecto
C_2 = Tamaño del proyecto
C_3 = Presupuesto del proyecto
C_4 = Tiempo de entrega del proyecto
C_5 = Riesgo asociado al proyecto
C_6 = Proyecto exploratorio
C_7 = Claridad en los requerimientos
C_8 = Frecuencia de cambios en los requerimientos
C_9 = Tamaño del equipo de desarrollo
C_{10} = Experiencia del equipo de desarrollo
C_{11} = Comunicación entre stakeholders
C_{12} = Entrega de funcionalidades parciales

C13 = Resusabilidad del producto de software
C14 = Documentación del proyecto

5.2 Criterios de evaluación

5.2.1 Complejidad del proyecto

Faridani [14] menciona que por defecto el software es intrínsecamente complejo, ya que por lo general, a los usuarios les resulta muy difícil expresar con precisión sus necesidades en una forma que los desarrolladores puedan entender.

Asimismo, Faridani [14] recalca que los tamaños de los sistemas están creciendo, lo que exige un gran equipo de desarrolladores y una comunicación y coordinación más complejas

Por otra parte, en sistemas grandes, hay una explosión combinatoria que hace que el número de rutas a través del código sea muy grande. Esto hace que las pruebas exhaustivas sean imposibles.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es la complejidad del proyecto de software?
Respestas: Baja, Media, Alta

5.2.2 Tamaño del proyecto

El tamaño del proyecto está relacionado con el presupuesto, el tamaño del equipo, la critcidad del proyecto y la cantidad de recursos físicos relacionados.

Este criterio mide el tamaño del proyecto. El gerente de proyecto debe estimar de acuerdo a los proyetos desarrollados en la organización.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el tamaño del proyecto de software?
Pequeño, Mediano, Grande

5.2.3 Presupuesto del proyecto

Este criterio mide el presupuesto disponible para el proyecto. El gerente de proyecto debe estimar de acuerdo al presupuesto disponible en su organización, si el asignado al proyecto es bajo, medio o alto.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cómo es el presupuesto disponible para el proyecto?
Respuestos: Bajo, Medio, Alto

5.2.4 Tiempo de entrega del proyecto

El tiempo de entrega de los proyectos es una de las variables más importantes que determinan la finalización exitosa o fallida de un proyecto de software. Diversas investigaciones en campo de la ingeniería de software demuestran que un amplio porcentaje de proyectos de software sufren retrasos en los tiempos de entrega.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cómo son los tiempos de entrega del proyecto de software?
Corto plazo, Mediano plazo, Largo plazo

5.2.5 Riesgo asociado al proyecto

Los riesgos son inherentes a todos los proyectos incluyendo los proyectos de software, por lo que la mayoría de procesos, metodologías, estándares y buenas prácticas realcionadas con el desarrollo de software tienen un apartado especial para la gestión de riesgos.

Por otra parte del universo de procesos de desarrollo de software existentes es evdente que unos son más enfocados a la gestión de riesgos que otros. Como mencionad [14], para los proyectos que son críticos y exigen una gran seguridad, un proceso 'pesado' como Casacada, Espiral o RUP parece ser una mejor opción, ya que requieren un conjunto documentado de planes y especificaciones y el cumplimiento de las normas.

Sommerville [31] añade que los riesgos originan problemas en el proyecto, como los de confección de agendas y excesos en los costos; por lo tanto, la disminución de riesgos es una actividad muy importante en la gestión del proyecto.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el nivel de riesgo asociado al proyecto de software?
Respuestas: Bajo, Medio, Alto

5.2.6 Proyecto Exploratorio

Los proyectos exploratorios son proyectos que se encuentran en una frontera, son críticos para la misión, dependen del tiempo o cambian constantemente. Highsmith [17], proporcionó una breve definición de los términos anteriores. Según Highsmith, los métodos tradicionales (cascada, espiral, modelo en v, entre otros) no se aplican a este tipo de proyectos.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el nivel de exploración en el proyecto de software?
Reespuestas: Bajo, Medio, Alto

5.2.7 Claridad en los requerimientos iniciales

La claridad en los requerimientos iniciales es un factor muy importante antes de iniciar con el desarrollo del software pues de acuerdo al nivel que ésta tenga impactará fuertemente en la decisión para la selección del proceso de desarrollo de software. Este criterio se define por el nivel de expresión que tenga el usuario para paltear sus requerimientos al equipo de desarrollo, así como por la definición que realizan los analistas de requerimientos.

Autores como Boehm [5], mencionan que procesos de desarrollo considerados "pesadosçomo cascada o el modelo en V, primero identifican los requerimientos, los definen y luego lo entregan al equipo apropiado.

Por otra pate, procesos eolutivos como Espiral tienen una interacción más cercana entre clientes y desarrolladores para determinar el conjunto de requerimientos con prioridades más altas para ser incluídos en cada iteración [14].

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cómo es la claridad de los requerimientos del proyecto?
Respuestas: Baja, Media, Alta

5.2.8 Frecuencia de cambios en los requerimientos

De acuerdo con Sommerville [31] después de que los sistemas hayan sido desarrollados, inevitablemente han de sufrir cambios si tienen que seguir siendo útiles. Una vez que el software comienza a utilizarse, surgen nuevos requerimientos y los requerimientos existentes cambian. Los cambios en los negocios a menudo generan nuevos requerimientos para el software existente. Algunas partes del software tienen que modificarse para corregir errores encontrados en su funcionamiento, adaptarlo a una nueva plataforma y mejorar su rendimiento u otras características no funcionales.

Asimismo indica Sommerville que en grandes compañías, la mayor parte del presupuesto de software se dedica a mantener los sistemas existentes, y no debería sorprender que algunas estudios sugieran que el 90 % de los costes del software sean costes de evolución.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es la frecuencia de cambios sobre los requerimientos durante el desarrollo?
Respuestas: Baja, Media, Alta

5.2.9 Tamaño del equipo de desarrollo

El tamaño del equipo de desarrollo es una característica muy importante en el desarrollo de software, ya que mientras más grande sea el tamaño del equipo más inconvenientes se podrían encontrar con las comunicaciones de los stakeholders del proyecto.

Como menciona Lohnes [24] las metodologías ágiles están limitadas a equipos de desarrollo pequeños con menos de 15 personas y cercanas al gerente de proyectos. Asimismo indica que los procesos de software son exitosos en ambientes con equipos grandes de 25a 75 personas yya sea en prroyectos en sitio, o en el que el equipo esté distribuido remotamente.

Awad [3] discutió la relación entre tamaño del proyecto y tamaño de los equipos. Él argumentó que, como el tamaño del proyecto aumenta, más presupuesto del proyecto, planificación, documentación, duración, y se necesitan miembros del equipo. Por estas razones, Las organizaciones eligen usar las metodologías tradicionales con proyectos grandes y grandes tamaños de equipo, porque los métodos tradicionales apoyar más planificación.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el tamaño del equipo de desarrollo?
Respuestas:Pequeño, Mediano, Grande

5.2.10 Experiencia del equipo de desarrollo

Este criterio evalúa el conocimiento de los desarrolladores sobre el dominio del problema. El conocimiento de los desarrolladores puede ser el resultado de ser un usuario en el dominio de la aplicación o desarrollar otras aplicaciones en el dominio [2].

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es la experiencia del equipo de desarrollo?
Baja, Media, Alta

5.2.11 Comunicación entre stakeholders

La comunicación entre stakeholders es un atributo que depende del proceso de software a utilizar en el proyecto así como de la cultura organizacional y del entorno.

Como mencionan varias investigaciones la comunicación entre los stakeholders es más complicada conforme el equipo de trabajo crece o si la distribución del equipo es remota.

Este criterio mide el nivel de comunicación entre los stakeholders del proyecto. Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el nivel de comunicación entre los stakeholders?
Respuesta: Bajo, Medio, Alto

5.2.12 Entrega de funcionalidades parciales

La entrega de funcionalidades parciales en el proyecto permite a los clientes conocer el avance del proyecto en fases tempranas y emitir comentarios que sirvan de retroalimentación al equipo de desarrollo.

Este criterio mide la entrega de funcionalidades parciales en el proyecto. Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cuál es el nivel de entregas funcionales parciales que requerirá el proyecto?
Respuestas: Bajo, Medio, Alto

5.2.13 Reusabilidad del producto de software

De acuerdo con Turk [33], los artefactos reutilizables necesitan soluciones más generalizadas que puedan utilizarse para obtener beneficios a largo plazo. Los proyectos de artefactos reutilizables requieren metodologías que puedan respaldar un marco de diseño que pueda reutilizarse repetidamente. La aplicabilidad de artefactos reutilizables requiere un control de alta calidad porque el efecto de baja calidad se extenderá a todas las aplicaciones que reutilizan el artefacto. Por lo tanto, los métodos tradicionales son los más adecuados para tales proyectos debido a la alta calidad controlar.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cómo es la reusabilidad de software estimado para el proyecto?
Respuestas: Baja, Media, Alta

5.2.14 Documentación del proyecto

La documentación del proyecto ha sido muy importante para los proyectos de software que siguen procesos de software. Sin embargo en los últimos años con la aparición del manifiesto ágil, se busca enfatizar el software funcional sobre la documentación extensiva. Sin embargo, los procesos de software denominados tradicionales siguen siendo los recomendados para proyectos grandes que requieren gran control a través de la docuemntación.

Para la evaluación del proyecto de software específico se responderá a la pregunta:

¿Cómo es la documentación requerido para el proyecto de software?
Respuestas: Baja, Media, Alta

5.2.15 Matriz de criterios de evaluación

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja	Media	Alta
2	Tamaño del proyecto	Pequeño	Mediano	Grande
3	Presupuesto del proyecto	Bajo	Medio	Alto
4	Tiempo de entrega del proyecto	Corto plazo	Mediano plazo	Largo plazo
5	Riesgo asociado al proyecto	Bajo	Medio	Alto
6	Proyecto Exploratorio	Bajo	Medio	Alto
7	Claridad en los requerimientos	Baja	Media	Alta
8	Frecuencia de cambios en los requerimientos	Baja	Media	Alta
9	Tamaño del equipo de desarrollo	Pequeño	Mediano	Grande
10	Experiencia del equipo de desarrollo	Baja	Media	Alta
11	Comunicación entre stakeholders	Baja	Media	Alta
12	Entrega de funcionalidades parciales	Baja	Media	Alta
13	Reusabilidad del producto de software	Baja	Media	Alta
14	Documentación del proyecto	Baja	Media	Alta

Cuadro 5.1: Criterios de selección

5.3 Evaluación de criterios en el modelo cascada

En la tabla 5.2 se evalúa el proceso de desarrollo de software vs la aplicabilidad de los criterios. Se realizó un análisis de las investigaciones, experiencias y artículos para asignar el puntaje (Alexander & Davis, 1991), (Ben-Zahia & Jaluta, 2014), (Nesma & Morgan), (Geambasu, Jianu, Jianu, & Gavrila, 2011), (Pressman, 2010), (Sommerville, 2011).

5.3.1 Complejidad del proyecto en modelo cascada

El modelo cascada puede ser aplicado totalmente a proyectos de software con un nivel de complejidad bajo y medio, debido a que por su naturaleza, la transición entre sus fases requiere que la fase previa esté completada para dar comienzo a la siguiente. Asimismo, para un proyecto de complejidad alta el modelo cascada no es recomendable.

De esta manera se asignan los siguientes valores de acuerdo al criterio Complejidad del proyecto:

C_{11} (Baja): 2 (Aplica)
C_{12} (Media): 1 (Aplica parcialmente)
C_{13} (Alta): 0 (No aplica)

5.3.2 Tamaño del proyecto en modelo cascada

El modelo cascada es aplicable para proyectos pequeños, medianos y grandes. Sin embargo, es más viable para proyectos grandes. Como manifiestan (Cervantes & Goméz, 2012) en un estudio realizado a desarrolladores de software, éstos prefieren el modelo cascada para sistemas grandes, mientras que para sistemas más pequeños los procesos evolutivos son más

aceptados.

De esta manera se asignan los siguientes valores de acuerdo al criterio Tamaño del proyecto:

C_{21} (Pequeño): 1 (Aplica parcialmente)
C_{22} (Mediano): 1 (Aplica parcialmente)
C_{23} (Grande): 2 (Aplica)

5.3.3 Presupuesto del proyecto en modelo cascada

El modelo cascada se basa en requerimientos muy bien definidos, por lo que se resiste a que exista un cambio en los requerimientos ya que puede ser muy costoso en fases finales del proyecto. La disponibilidad de presupuesto en los proyectos de software con el modelo cascada debe ser de alto o medio nivel para solventar los reprocesos que aparezcan en caso de cambios en los requerimientos.

Así, la calificación del modelo cascada con respecto al criterio Costos del proyecto es:

C_{31} (Bajo): 0 (No Aplica)
C_{32} (Medio): 1 (Aplica parcialmente)
C_{33} (Alto): 2 (Aplica)

5.3.4 Tiempo de entrega del proyecto en modelo cascada

Si en un proyecto de software el tiempo de entrega es muy importante entonces no se recomienda el uso de modelos tipo cascada. Solamente cuando la calidad del producto sea prioritaria sobre el tiempo de entrega es bueno adoptar estos modelos (Pfleeger & Atlee, 2006).

Así, la calificación del modelo cascada con respecto al criterio Tiempo de entrega del proyecto es:

C_{41} (Corto plazo): 0 (No Aplica)
C_{42} (Mediano plazo): 2 (Aplica)
C_{43} (Largo plazo): 2 (Aplica)

5.3.5 Riesgo asociado al proyecto en modelo cascada

Como indica (Faridani, 2011) los procesos de desarrollo de software considerados como tradicionales (cascada, espiral, RUP) son considerados mejor opción que las metodologías ágiles para los proyectos de desarrollo de software con altos riesgos, debido al alto nivel de documentación y planificación que tienen los llamados procesos tradicionales.

La calificación del modelo cascada con respecto al criterio Riesgo asociado al proyecto es:

C_{51} (Bajo): 1 (Aplica)
C_{52} (Medio): 2 (Aplica)
C_{53} (Alto): 1 (Aplica)

5.3.6 Proyecto exploratorio en modelo cascada

El modelo cascada es recomendado para proyectos de software con requerimientos entendibles, claros y definidos. Por lo que un para un proyecto con altos índices de incertidumbre, como lo son los proyectos exploratorios no se recomienda el uso de cascada.

La calificación del modelo cascada con respecto al nivel de exploración en el proyecto de software es:

C_{61} (Bajo): 0 (No aplica)
C_{62} (Medio): 0 (No aplica)
C_{63} (Alto): 0 (No aplica)

5.3.7 Claridad en los requerimientos iniciales en modelo cascada

Como indica (Sommerville, 2011), el modelo cascada solo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema. Por consecuente, si los requerimientos iniciales son poco claros el modelo cascada llevaría al proyecto de software al fracaso ya que ajustes a los requerimientos iniciales son costosos e implicarían volver a la fase inicial del modelo y replantear el proyecto.

La calificación del modelo cascada con respecto al nivel de claridad de los requerimientos iniciales es:

C_{71} (Bajo): 0 (No Aplica)
C_{72} (Medio): 0 (No Aplica)
C_{73} (Alto): 2 (Aplica)

5.3.8 Frecuencia de cambios en los requerimientos en modelo cascada

En el modelo cascada es difícil responder a los cambios en los requerimientos del cliente debido ya que debido a los costos de producción y aprobación de documentos, las iteraciones son costosas e implican rehacer el trabajo (Sommerville, 2011).

La frecuencia de cambios en el modelo cascada se clasifica en:

C_{81} (Baja): 1 (Aplica parcialmente)
C_{82} (Media): 0 (No Aplica)
C_{83} (Alta): 0 (No Aplica)

5.3.9 Tamaño del equipo de desarrollo en el modelo cascada

El modelo cascada es aplicable a todos los tamaños de proyectos. Sin embargo, (Pfleeger & Atlee, 2006) enfatiza en que el modelo cascada tienen mayor aplicabilidad cuando muchas personas participan en el proyecto, ya sea porque el proyecto es grande o porque se requiere la colaboración de especialistas. Afirma también que debido al tamaño del equipo es importante tener procedimientos de control estricto y una comunicación formal y disciplinada, la cual efectivamente la ofrece el modelo cascada.

El tamaño del equipo de desarrollo para el modelo cascada se clasifica en:

C91 (Pequeño): 1 (Aplica parcialmente)
C92 (Mediano): 2 (Aplica)
C93 (Grande): 2 (Aplica)

5.3.10 Experiencia del equipo de desarrollo en el modelo cascada

El modelo cascada es sencillo para gestionar los proyectos de desarrollo por lo que es bueno para equipos de desarrollo con limitaciones en las habilidades [30] así como en equipos de expertos.

La experiencia delequipo de desarrollo para el modelo cascada se clasifica en:

C101 (Pequeño): 1 (Aplica parcialmente)
C102 (Mediano): 2 (Aplica)
C103 (Grande): 2 (Aplica)

5.3.11 Comunicación entre stakeholders en el modelo cascada

El modelo cascada es uno de los modelos de proceso que menos participación del usuario requiere. Como afirma (Faridani, 2011), procesos de software como cascada, generalmente de formularios o contratos entre desarrolladores y clientes como base para el relacionamiento. Por otra parte (Ben-Zahia & Jaluta, 2014) en relación con los demás procesos afirman que cascada tiene la menor necesidad de relacionamiento con el usuario.

El nivel de comunicación con los stakeholders para el modelo cascada es:

C111 (Bajo): 2 (Aplica)
C112 (Medio): 1 (Aplica parcialmente)
C113 (Alto): 0 (Aplica)

5.3.12 Entrega de funcionalidades parciales en el modelo cascada

El modelo cascada por defecto no admite la entrega de funcionalidades parciales; el producto de software resultante solamente es entregado al final de la última fase. Por lo que la entrega de funcionalidades parciales no se aplica a este modelo.

Por lo anterior, el modelo cascada tiene la siguiente calificicaión con respecto a los niveles de entregas parciales requeridas en el proyecto:

C121 (Bajo): 0 (No aplica)
C122 (Medio): 0 (No aplica)
C123 (Alto): 0 (No aplica)

5.3.13 Reusabilidad del producto de software en el modelo cascada

De acuerdo a (Turk, France, & Rumpe, 2002) los proyectos que incluyen artefactos reutilizables requieren de metodologías que soporten un marco de trabajo que pueda usarse repetidamente. Para esto los artefactos reutilizables requieren controles de alta calidad que pueden ser alcanzados por procesos de software 'tradicionales' como lo es el modelo cascada.

Por lo anterior, el modelo cascada tiene la siguiente calificación con respecto al nivel de reusabilidad de productos:

C131 (Bajo): 2 (Aplica)
C132 (Medio): 2 (Aplica)
C133 (Alto): 2 (Aplica)

5.3.14 Documentación del proyecto en el modelo cascada

El modelo cascada se caracteriza por el alto nivel de documentación del proceso de desarrollo de software, en el que la documentación se produce en cada fase y ésta cuada con otros modelos del proceso de ingeniería (Sommerville, 2011). Para proyectos con niveles de documentación bajos no es recomendable el uso del modelo cascada debido a que este modelo requiere completa documentación del proyecto.

Por lo anterior, el modelo cascada tiene la siguiente calificación con respecto al nivel de documentación requerida en el proyecto:

C141 (Bajo): 0 (No aplica)
C142 (Medio): 1 (Aplica parcialmente)
C143 (Alto): 2 (Aplica)

5.4 Evaluación de criterios en el modelo en V

En la tabla 5.3 se evalúa el modelo V vs la aplicabilidad de los criterios. Se realizó un análisis de las investigaciones, experiencias y artículos para asignar el puntaje (Alexander & Davis, 1991), (Ben-Zahia & Jaluta, 2014), (Nesma & Morgan), (Geambasu, Jianu, Jianu, & Gavrila, 2011), (Pressman, 2010), (Sommerville, 2011).

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja 2	Media 1	Alta 0
2	Tamaño del proyecto	Pequeño 1	Mediano 1	Grande 2
3	Presupuesto del proyecto	Bajo 0	Medio 1	Alto 2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 2	Largo plazo 2
5	Riesgo asociado al proyecto	Bajo 1	Medio 2	Alto 1
6	Proyecto Exploratorio	Bajo 0	Medio 0	Alto 0
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 2
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0
9	Tamaño del equipo de desarrollo	Pequeño 1	Mediano 2	Grande 2
10	Experiencia del equipo de desarrollo	Baja 1	Media 2	Alta 2
11	Comunicación entre stakeholders	Baja 2	Media 1	Alta 0
12	Entrega de funcionalidades parciales	Baja 0	Media 0	Alta 0
13	Reusabilidad del producto de software	Baja 2	Media 2	Alta 2
14	Documentación del proyecto	Baja 0	Media 1	Alta 2

Cuadro 5.2: Evaluación de criterios en el modelo cascada

5.4.1 Complejidad del proyecto en el modelo en V

Como indica Chavhan [10] el modelo en V no es el adecuado para proyectos muy grandes ni para proyectos complejos.

De esta manera se asignan los siguientes valores de acuerdo al criterio Complejidad del proyecto para el modelo en V:

C_{11} (Baja): 2 (Aplica)
C_{12} (Media): 1 (Aplica parcialmente)
C_{13} (Alta): 0 (No aplica)

5.4.2 Tamaño del proyecto en el modelo en V

El modelo en V suele trabajar bastante bien proyectos de software pequeños donde los requisitos son entendidos fácilmente [19]. El modelo en V también es adecuado para proyectos medianos [10].

C_{21} (Pequeño): 2 (Aplica)
C_{22} (Mediano): 1 (Aplica parcialmente)
C_{23} (Grande): 0 (No Aplica)

5.4.3 Presupuesto del proyecto en el modelo en V

En los proyectos con presupuesto limitado el modelo en V no es recomendable [30], ya que tiene poca flexibilidad y ajustar el alcance es difícil y caro [19].

Así, la calificación del modelo en V con respecto al presupuesto del proyecto es:

C_{31} (Bajo): 0 (No Aplica)
C_{32} (Medio): 0 (No Aplica)
C_{33} (Alto): 2 (Aplica)

5.4.4 Tiempo de entrega del proyecto en el modelo en V

El modelo en V no es recomendado para proyectos de software con plazos cortos de entrega, ya que en caso de un detectar un error en etapas posteriores del proceso implica volver a etapas tempranas y por ende más tiempo [30].
Así, la calificación del modelo en V con respecto al criterio Tiempo de entrega del proyecto es:

C_{41} (Corto plazo): 0 (No Aplica)
C_{42} (Mediano plazo): 1 (Aplica parcialmente)
C_{43} (Largo plazo): 2 (Aplica)

5.4.5 Riesgo asociado al proyecto en el modelo en V

Una de las más conocidas desventajas del modelo en V es que carce de gestión y mitigación de riesgos [10]. Asimismo, debido a que el modelo en V no produce entregas parciales o prototipos tiene un alto riesgo de no alcanzar las necesidades del cliente.

La calificación del modelo RUP con respecto al criterio Riesgo asociado al proyecto es:

C_{51} (Bajo): 1 (Aplica parcialmente)
C_{52} (Medio): 0 (No aplica)
C_{53} (Alto): 0 (No aplica)

5.4.6 Proyecto exploratorio en el modelo en V

El modelo en V es adecuado para proyectos de sotware pequeños y fáciles de entender, por lo que no es el adecuado para proyectos exploratorios de alto nivel [8].

La calificación del modelo en V con respecto al nivel de exploración en el proyecto de software es:

C_{61} (Bajo): 2 (Aplica)
C_{62} (Medio): 0 (No Aplica)
C_{63} (Alto): 0 (No Aplica)

5.4.7 Claridad en los requerimientos iniciales en el modelo en V

De manera similar al modelo cascada, el modelo en V solo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema [31]. Asimismo el modelo en V debe se usado para proyectos de software con requerimientos claros, definidos y fijados [8].

Por consecuente, si los requerimientos iniciales son poco claros el modelo en V llevaría al proyecto de software al fracaso ya que ajustes a los requerimientos iniciales son costosos e implicarían volver a la fase inicial del modelo hasta el punto de replantearlo.

La calificación del modelo en V con respecto al nivel de claridad de los requerimientos iniciales es:

C_{71} (Bajo): 0 (No aplica)
C_{72} (Medio): 0 (No Aplica)
C_{73} (Alto): 2 (Aplica)

5.4.8 Frecuencia de cambios en los requerimientos en el modelo en V

El modelo en V es un modelo muy rígido y poco flexible [8] el cual requiere de una clara definición de los requerimientos iniciales debido a que un cambio en medio del proceso de

desarrollo es costoso. Por lo tanto y como lo afirma Chavhan [10], no es una buena opción si los requerimientos cambian frecuentemente.

La frecuencia de cambios en los requerimientos para el modelo en V se clasifica en:

C_{81} (Baja): 1 (Aplica parcialmente)
C_{82} (Media): 0 (No aplica)
C_{83} (Alta): 0 (No aplica)

5.4.9 Tamaño del equipo de desarrollo en el modelo en V

El modelo en V es recomendado para proyectos de software pequeños con equipos de trabajo pequeños [19]. De acuerdo a la clasificación de Lohnes [24] el modelo en V es éxitoso en proyectos con equipos de hasta 25 personas.

El tamaño del equipo de desarrollo para el modelo RUP se clasifica en:

C_{91} (Pequeño): 2 (Aplica)
C_{92} (Mediano): 1 (Aplica parcialmente)
C_{93} (Grande): 0 (No Aplica)

5.4.10 Experiencia del equipo de desarrollo en el modelo en V

El modelo en V requiere de poca experiencia del equipo de desarrollo para la gestión del proceso de software y como menciona Sami [30], es bueno en equipos con limitaciones en las habilidades.

El modelo en V de acuerdo a la experiencia del equipo de desarrollo se clasifica en:

C_{101} (Baja): 2 (Aplica)
C_{102} (Media): 1 (Aplica parcialmente)
C_{103} (Alta): 1 (Aplica parcialmente)

5.4.11 Comunicación entre stakeholders en el modelo en V

El modelo en V es uno de los modelos de proceso que menos participación del usuario requiere. Como afirma Faridani [14] procesos de software como modelo en V, generalmente usa formularios o contratos entre desarrolladores y clientes como base para el relacionamiento.

El nivel de comunicación con los stakeholders para el modelo en V es:

C_{111} (Bajo): 2 (Aplica)
C_{112} (Medio): 1 (Aplica parcialmente)
C_{113} (Alto): 0 (Aplica)

5.4.12 Entrega de funcionalidades parciales en el modelo en V

El modelo en V por defecto no admite la entrega de funcionalidades parciales; el producto de software es desarrollado durante la fase de implementación, así qe ni se ceran prototipos del software en etapas tempranas [8].

Esta característica del modelo en V implica alta confianza por parte del cliente ya qe sin prototipos hay un alto riesgo de alcanzar las expectativas del cliente [8].

Por lo anterior, el modelo cascada en V tiene la siguiente calificación con respecto a los niveles de entregas parciales requeridas en el proyecto:

C_{121} (Bajo): 1 (Aplica parcialmente)
C_{122} (Medio): 0 (No aplica)
C_{123} (Alto): 0 (No aplica)

5.4.13 Reusabilidad del producto de software en el modelo en V

De acuerdo a [33] los proyectos que incluyen artefactos reutilizables requieren de metodologías que soporten un marco de trabajo que pueda usarse repetidamente. Para esto los artefactos reutilizables requieren controles de alta calidad que pueden ser alcanzados por procesos de software 'tradicionales' como lo es el modelo en V.

Sami [30] añade que el modelo en V es excelente en la reusabilidad de componentes de software.

Por lo anterior, el en V tiene la siguiente calificación con respecto al nivel de reusabilidad de productos:

C_{131} (Bajo): 2 (Aplica)
C_{132} (Medio): 2 (Aplica)
C_{133} (Alto): 2 (Aplica)

5.4.14 Documentación del proyecto en el modelo en V

El modelo en V se caracteriza por el alto nivel de documentación del proceso de desarrollo de software, en el que la documentación se produce en cada fase junto con la documentación de las pruebas [31]. Para proyectos con niveles de documentación bajos no es recomendable el uso del modelo en V ya que este modelo requiere completa documentación del proyecto.

Por lo anterior, el modelo en V tiene la siguiente calificación con respecto al nivel de documentación requerida en el proyecto:

C_{141} (Bajo): 0 (No aplica)
C_{142} (Medio): 1 (Aplica parcialmente)

C143 (Alto): 2 (Aplica)

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja 2	Media 1	Alta 0
2	Tamaño del proyecto	Pequeño 2	Mediano 1	Grande 0
3	Presupuesto del proyecto	Bajo 0	Medio 0	Alto 2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 2
5	Riesgo asociado al proyecto	Bajo 1	Medio 0	Alto 0
6	Proyecto Exploratorio	Bajo 2	Medio 0	Alto 0
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 2
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0
9	Tamaño del equipo de desarrollo	Pequeño 2	Mediano 1	Grande 0
10	Experiencia del equipo de desarrollo	Baja 2	Media 1	Alta 1
11	Comunicación entre stakeholders	Baja 2	Media 1	Alta 0
12	Entrega de funcionalidades parciales	Baja 1	Media 0	Alta 0
13	Reusabilidad del producto de software	Baja 2	Media 2	Alta 2
14	Documentación del proyecto	Baja 0	Media 1	Alta 2

Cuadro 5.3: Evaluación de criterios en el modelo en V

5.5 Evaluación de criterios en el modelo espiral

En la tabla 5.4 se evalúa el proceso de desarrollo de software vs la aplicabilidad de los criterios. Se realizó un análisis de las investigaciones, experiencias y artículos para asignar el puntaje (Alexander & Davis, 1991), (Ben-Zahia & Jaluta, 2014), (Nesma & Morgan), (Geambasu, Jianu, Jianu, & Gavrila, 2011), (Pressman, 2010), (Sommerville, 2011).

5.5.1 Complejidad del proyecto en modelo espiral

El modelo espiral es útil para proyectos de software grandes, costosos y complicados (Faridani, 2011) y (Ben-Zahia & Jaluta, 2014) gracias a sus iteraciones y fases.

La clasificación del modelo espiral con respecto a la complejidad del proyecto es:

C11 (Baja): 0 (No aplica)
C12 (Media): 2 (Aplica)
C13 (Alta): 2 (Aplica)

5.5.2 Tamaño del proyecto en modelo espiral

El modelo espiral es útil para proyectos de software grandes, costosos y complicados (Faridani, 2011) y (Ben-Zahia & Jaluta, 2014), gracias a las fases, iteraciones, la entregas parciales y el análisis de riesgos los proyectos de software de gran tamaño pueden finalizar satisfactoriamente.

(Pressman, 2010) Añade: "El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que el proceso avanza, el desarrollador y cliente comprenden y reaccionan mejor ante los riesgos en cada nivel de evolución".

La clasificación del modelo espiral con respecto a tamaño del proyecto es:

C21 (Pequeño): 0 (No aplica)
C22 (Mediano): 2 (Aplica)
C23 (Grande): 2 (Aplica)

5.5.3 Presupuesto del proyecto en modelo espiral

El modelo espiral es costoso debido a las iteraciones que se deben realizar para desarrollar el producto de software, lo cual implica mayor capital humano, tiempo y recursos físicos que otros modelos tradicionales como el modelo cascada.

La clasificación del modelo espiral con respecto al presupuesto disponible del proyecto es:

C31 (Bajo): 0 (No aplica)
C32 (Medio): 0 (No Aplica)
C33 (Alto): 2 (Aplica)

5.5.4 Tiempo de entrega del proyecto en modelo espiral

El modelo espiral hace parte de los modelos evolutivo, por lo que omparte una característica: la completa documentación en sus diversas fases. Ésta documentación implica que

en proyectos de software que se desarrollan rápidamente, no es rentable la producción de documentos que reflejen cada una de las versiones del producto de software (Sommerville, 2011). P or lo tanto el modelo espiral es recomendable para proyectos con plazos largos y medianos pero no para proyectos con plazos cortos.

La clasificación del modelo espiral con respecto al tiempo de entrega del producto final es:

C41 (Corto plazo): 0 (No aplica)
C42 (Mediano plazo): 1 (Aplica parcialmente)
C43 (Largo plazo): 2 (Aplica)

5.5.5 Riesgo asociado al proyecto en modelo espiral

Una de las 4 fases del modelo espiral es la evaluación y reducción de riesgos lo cual evidencia la orientación a riesgos del modelo espiral. Es por esto que autores como (Faridani, 2011) mencionan que en sus orígenes el objetivo del modelo espiral fue cambiar el énfasis del desarrollo de software a la evaluación y reducción de riesgos.

Para (Sommerville, 2011) la diferencia principal entre el modelo en espiral y los otros modelos del proceso del software es la consideración explícita del riesgo. Añade (Pressman, 2010) que el modelo espiral usa los prototipos como mecanismo de reducción de riesgos.

La clasificación del modelo espiral con respecto al riesgo asociado al proyecto es:

C51 (Bajo): 1 (Aplica parcialmente)
C52 (Medio): 2 (Aplica)
C53 (Alto): 2 (Aplica)

5.5.6 Proyecto exploratorio en modelo espiral

Las fases e iteraciones del modelo espiral permiten su uso en proyectos exploratorios, los cuales tienen altos índices de incertidumbre. A través de las iteraciones se van haciendo entregas parciales al cliente, con lo cual se podrían redefinir los requermientos y las estrategias para alcanzar los objetivos del proyecto. Sin embargo, como menciona Cervantes [9] no es recomendable para proyectos con nivel de exploración alto porque cada nueva adición con redefinición de requerimientos puede corromper el software.

La calificación del modelo espiral con respecto al nivel de exploración en el proyecto de software es:

C61 (Bajo): 2 (Aplica)
C62 (Medio): 2 (Aaplica)
C63 (Alto): 1 (No aplica)

5.5.7 Claridad en los requerimientos iniciales en modelo espiral

El modelo espiral permite el inicio del desarrollo cuando los requerimientos no están muy claros o son muy complejos [21].

La calificación del modelo cascada con respecto al nivel de claridad de los requerimientos iniciales es:

C_{71} (Bajo): 2 (No Aplica)
C_{72} (Medio): 2 (No Aplica)
C_{73} (Alto): 1 (Aplica)

5.5.8 Frecuencia de cambios en los requerimientos en modelo espiral

En cada iteración del modelo espiral se realiza la fase de Definición de objetivos, en la cual se definen los objetivos específicos inherentes a la fase que dará inicio [5]. Por lo tanto, el modelo espiral soporta los cambios en los requerimientos durante el desarrollo, ya sean poco o muy frecuentes.

La frecuencia de cambios en el modelo espiral se clasifica en:

C_{81} (Baja): 2 (Aplica)
C_{82} (Media): 2 (Aplica)
C_{83} (Alta): 2 (Aplica)

5.5.9 Tamaño del equipo de desarrollo en el modelo espiral

El modelo espiral requiere un tamaño de equipo mediano y grande para ocupar los roles requeridos en las fases del modelo espiral: definición de objetivos, evaluación y reducción de riesgos, desarrollo-validación y planeación. Como indica [4] espiral no es recomendable para proyectos con equipos pequeños.

El tamaño del equipo de desarrollo para el modelo espiral se clasifica en:

C_{91} (Pequeño): 0 (No Aplica)
C_{92} (Mediano): 2 (Aplica)
C_{93} (Grande): 2 (Aplica)

5.5.10 Experiencia del equipo de desarrollo en el modelo espiral

El modelo espiral se caracteriza porque en cada iteración realiza la fase de evaluación y reducción de riesgos, la cual requiere de personal altamente experimentado en análisis de riesgos [4].

La aplicabilidad del modelo espiral con respecto a la experiencia del equipo de desarrollo se clasifica en:

C_{101} (Baja): 0 (No Aplica)
C_{102} (Media): 1 (Aplica parcialmente)
C_{103} (Alta): 2 (Aplica)

5.5.11 Comunicación entre stakeholders en el modelo espiral

En el modelo espiral, a diferencia del modelo cascada se requiere una mayor interacción con los usuarios [4] ya que en cada interación se espera conocer los comentarios del cliente para evaluar y retroalimentar las versiones parciales entregadas [9].

El nivel de comunicación con los stakeholders para el modelo espiral es:

C_{111} (Baja): 0 (No Aplica)
C_{112} (Media): 1 (Aplica)
C_{113} (Alta): 2 (Aplica)

5.5.12 Entrega de funcionalidades parciales en el modelo espiral

El modelo espiral está disñado para realizar entregas parciales a través de sus diferentes iteraciones. Como menciona Ben-Zahia [4] una de las ventajas del modelo espiral radica en que un producto de software se entrega brevemente.

Espiral permite mostrar al cliente versiones parciales que permiten obtener al equipo del proyecto retroalimentación y evitar problemas con la integración e un código muy grande [9].

Por lo tanto, a clasificación del modelo espiral con respecto a la entrega de funcionalidades parciales es:

C_{121} (Baja): 1 (Aplica parcialmente)
C_{122} (Media): 1 (Aplica parcialmente)
C_{123} (Alta): 2 (Aplica)

5.5.13 Reusabilidad del producto de software en el modelo espiral

De acuerdo a Turk, France, & Rumpe [33] los proyectos que incluyen artefactos reutilizables requieren de metodologías que soporten un marco de trabajo que pueda usarse repetidamente. Para esto los artefactos reutilizables requieren controles de alta calidad que pueden ser alcanzados por procesos de software 'tradicionales' como lo es el modelo espiral.

Adicionalmente, el modelo espiral permite gestionar la resubilidad de los productos de software en cada una de las entregas parciales. De acuerdo a Sommerville [31], en espiral la reutilización es indispensable para el desarrollo rápido de sistemas.

Por lo anterior, con respecto al nivel de reusabilidad de producctos se califica el modelo

espiral como:

C_{131} (Baja): 2 (Aplica)
C_{132} (Media): 2 (Aplica)
C_{133} (Alta): 2 (Aplica)

5.5.14 Documentación del proyecto en el modelo espiral

El modelo espiral se caracteriza por la documentación en sus fases y cada una de sus iteraciones. Sin embargo, Braude [7] señala que con el propósito de optimizar la productividad en equipo, con frecuencia es necesario comenzar una nueva iteración antes de que la anterior haya terminado, lo cual dificulta la coordinación de la documentación. Al respecto [9] menciona en espiral se complica mantener actualizada y correcta la documentación.

Por lo anterior, el modelocespiral tiene la siguiente calificación con respecto al nivel de documentación requerida en el proyecto:

C_{141} (Baja): 0 (No aplica)
C_{142} (Media): 2 (Aplica)
C_{143} (Alta): 1 (Aplica parcialmente)

5.6 Evaluación de criterios en el modelo RUP

En la tabla 5.5 se evalúa el modelo RUP vs la aplicabilidad de los criterios. Se realizó un análisis de las investigaciones, experiencias y artículos para asignar el puntaje (Alexander & Davis, 1991), (Ben-Zahia & Jaluta, 2014), (Nesma & Morgan), (Geambasu, Jianu, Jianu, & Gavrila, 2011), (Pressman, 2010), (Sommerville, 2011).

5.6.1 Complejidad del proyecto en el modelo RUP

Como indica Geambasu [16] el desarrollo de software usando RUP incluye el manejo de una gran cantidad de actividades para alcanzar los objetivos del proyecto, destacando la extensión de los tiempos de entrega del sistema final. RUP pude se adaptadao a los requerimientos específicos de un proyecto particular, pero el proceso de adaptación es también complejo.

la decisión de usar el modelo RUP para el desarrollo de software debe tener en cuenta la complejidad de la gestión técnica y de proyectos. es recomendable el uso de RUP cuando la complejidad de ambos factores es promedio [16].

De esta manera se asignan los siguientes valores de acuerdo al criterio Complejidad del proyecto:

C_{11} (Baja): 1 (Aplica parcialmente)
C_{12} (Media): 2 (Aplica)

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja 0	Media 2	Alta 2
2	Tamaño del proyecto	Pequeño 0	Mediano 2	Grande 2
3	Presupuesto del proyecto	Bajo 0	Medio 0	Alto 2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 2
5	Riesgo asociado al proyecto	Bajo 1	Medio 2	Alto 2
6	Proyecto Exploratorio	Bajo 2	Medio 2	Alto 1
7	Claridad en los requerimientos	Baja 2	Media 2	Alta 1
8	Frecuencia de cambios en los requerimientos	Baja 2	Media 2	Alta 2
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 2	Grande 2
10	Experiencia del equipo de desarrollo	Baja 0	Media 1	Alta 2
11	Comunicación entre stakeholders	Baja 0	Media 1	Alta 2
12	Entrega de funcionalidades parciales	Baja 1	Media 1	Alta 2
13	Reusabilidad del producto de software	Baja 2	Media 2	Alta 2
14	Documentación del proyecto	Baja 0	Media 2	Alta 1

Cuadro 5.4: Evaluación de criterios en el modelo espiral

C13 (Alta): 0 (No aplica)

5.6.2 Tamaño del proyecto en el modelo RUP

El modelo RUP impone una sobrecarga de procesos innecesaria para proyectos pequeños, por lo que RUP es recomendable para proyectos medianos y grandes.

De esta manera se asignan los siguientes valores de acuerdo al criterio Tamaño del proyecto:

C21 (Pequeño): 0 (No Aplica)
C22 (Mediano): 1 (Aplica parcialmente)
C23 (Grande): 2 (Aplica)

5.6.3 Presupuesto del proyecto en el modelo RUP

RUP consta de una gran cantidad de actividades para alcanzar los objetivos del proyecto, por lo tanto tiene una alta complejidad, la cual requiere de una gran cantidad de recursos incluidos financieros [16].

Así, la calificación del modelo RUP con respecto al presupuesto del proyecto es:

C31 (Bajo): 0 (No Aplica)
C32 (Medio): 0 (No Aplica)
C33 (Alto): 2 (Aplica)

5.6.4 Tiempo de entrega del proyecto en el modelo RUP

El modelo RUP tiene aplicabilidad en proyectos de software con tiempos de entrega a mediano y largo plazo, sin aplicabilidad a proyectos a corto plazo. Como lo indica Geambasu [16], el desarrollo de software usando RUP implica una gran cantidad de actividades para alcanzar os objetivos del proyecto, lo cual lleva a que el tiempo de entrega del sistema final se extienda.

Así, la calificación del modelo RUP con respecto al criterio Tiempo de entrega del proyecto es:

C41 (Corto plazo): 0 (No Aplica)
C42 (Mediano plazo): 1 (Aplica parcialmente)
C43 (Largo plazo): 2 (Aplica)

5.6.5 Riesgo asociado al proyecto en el modelo RUP

Una importante ventaja del uso de RUP es que impone la identificación de riesgos y establece estrategias de mitigación en etapas tempranas, lo cual ayuda a una estimación más realística de los costos y los tiempos de desarrollo del proyecto.

La calificación del modelo RUP con respecto al criterio Riesgo asociado al proyecto es:

C51 (Bajo): 1 (Aplica parcialmente)
C52 (Medio): 2 (Aplica)
C53 (Alto): 2 (Aplica)

5.6.6 Proyecto exploratorio en el modelo RUP

RUP puede adaptarse para ajustarse a requerimientos específicos de un proyecto particular, pero la adaptación es un proceso complejo.

La calificación del modelo RUP con respecto al nivel de exploración en el proyecto de software es:

C61 (Bajo): 2 (Aplica)
C62 (Medio): 2 (Aplica)
C63 (Alto): 1 (Aplica parcialmente)

5.6.7 Claridad en los requerimientos iniciales en el modelo RUP

Como indica Geambasu [16] el modelo RUP se debe utilizar cuando exista una completa y correcta definición de crequerimientos desde el comienzo del proyecto, la cual es el punto de partida del desarrollo de software que incorpora toda la funcionalidad requerida por el cliente.

El uso del modelo RUP en proyectos con requerimeintos poco claros en el inicio del proyecto puede llevar a demoras en las entregas, sobrecostos o fallas para alcanzar los requerimientos de los usuarios.

La calificación del modelo RUP con respecto al nivel de claridad de los requerimientos iniciales es:

C71 (Bajo): 0 (No Aplica)
C72 (Medio): 1 (Aplica parcialmente)
C73 (Alto): 2 (Aplica)

5.6.8 Frecuencia de cambios en los requerimientos en el modelo RUP

En el modelo RUP, la subdivisión de las fases en iteraciones permite a los desarrolladores hacer los cambios necesarios para adaptar los nuevos equerimientos durante el proceso completo del desarrollo, sin embargo el costo de los cambios realizados especialmente en las etapas posteriores del desarrollo es alto. Los aspectos relacionados con la gestión de ambios durante el proceso de desarrollo son especificados en la disciplina de Gestión de Cambios y Configuración de RUP [16].

La frecuencia de cambios en los requerimientos para el modelo RUP se clasifica en:

C_{81} (Baja): 2 (Aplica)
C_{82} (Media): 2 (Aplica)
C_{83} (Alta): 1 (Aplica parcialmente)

5.6.9 Tamaño del equipo de desarrollo en el modelo RUP

El modelo RUP es un modelo de alta complejidad, por lo que requiere de un gran número de recursos humanos [16], por lo que de acuerdo a la clasificación de Lohnes [24] el modelo RUP es éxitoso en proyectos con equipos de 25 a 75 personas.

El tamaño del equipo de desarrollo para el modelo RUP se clasifica en:

C_{91} (Pequeño): 0 (Aplica paercialmente)
C_{92} (Mediano): 1 (Aplica parcialmente)
C_{93} (Grande): 2 (Aplica) l

5.6.10 Experiencia del equipo de desarrollo en el modelo RUP

Como menciona Krutchen [22], RUP está orientado a las personas, procesos, herramientas y métodos. Las organizaciones gastan considerables recursos en tiempo y dinero desplegando el proceso, entendiendo las herramientas, entrenando sus equipos en el proceso y la gestión de proyectos. Equipos con poca experiencia son poco recomendados para el modelo RUP.

La calificación asignada al modelo RUP en el criterio Experiencia del equipo es:

C_{101} (Baja): 0 (Aplica)
C_{102} (Media): 2 (Aplica)
C_{103} (Alta): 2 (Aplica)

5.6.11 Comunicación entre stakeholders en el modelo RUP

En el modelo RUP los clientes proveen a los desarrolladores la información de los requerimientos que el futuro sistema tendrá y les dan cuando sea requerido retoralimentación sobre ciertos resultados del proceso de desarrollo, sin embargo, no están activamente involucrados en el proceso de desarrollo de software [16]. Por lo tanto, el modelo RUP es exitoso en proyectos de software en los que la comunicación requerida entre stakeholders es de nivel bajo o medio.

El nivel de comunicación con los stakeholders para el modelo RUP es:

C_{111} (Bajo): 2 (Aplica)
C_{112} (Medio): 1 (Aplica)
C_{113} (Alto): 0 (Aplica)

5.6.12 Entrega de funcionalidades parciales en el modelo RUP

El modelo RUP debido a su desarrollo iterativo del sistema produce múltiples versiones del sistema a través de su ciclo de vida, las versiones deben ser gestionadas cuidadosamente para evitar la integración al final del proceso de desarrollo o soluciones incompletas.

Por lo anterior, el modelo RUP tiene la siguiente calificicaión con respecto a los niveles de entregas parciales requeridas en el proyecto:

C121 (Bajo): 0 (No aplica)
C122 (Medio): 2 (No aplica)
C123 (Alto): 2 (No aplica)

5.6.13 Reusabilidad del producto de software en el modelo RUP

Rational [28] menciona que el proceso RUP se enfoca el desarrollo temprano y en la línea base de una arquitectura robusta, antes de comprometer recursos para el desarrollo a gran escala. Describe cómo diseñar una arquitectura resilente que es flexible, se ajusta a los cambios, es intuitivamente entendible y promueve una reutilización de softare más efectiva.

Por lo anterior, el modelo RUP tiene la siguiente calificación con respecto al nivel de reusabilidad de productos:

C131 (Bajo): 2 (Aplica)
C132 (Medio): 2 (Aplica)
C133 (Alto): 2 (Aplica)

5.6.14 Documentación del proyecto en el modelo RUP

El modelo RUP hace énfasis en la documentación precisa y provee una descripción detallada de las actividades a ser desarrolladas junto con los roles y los artefactos relacionados [16].

Por lo anterior, el modelo RUP tiene la siguiente calificación con respecto al nivel de documentación requerida en el proyecto:

C141 (Bajo): 0 (No aplica)
C142 (Medio): 1 (No Aplica)
C143 (Alto): 2 (Aplica)

5.7 Evaluación de criterios en el modelo en Métrica V3

En la tabla 5.3 se evalúa el modelo V vs la aplicabilidad de los criterios. Se realizó un análisis de las investigaciones, experiencias y artículos para asignar el puntaje (Alexander & Davis, 1991), (Ben-Zahia & Jaluta, 2014), (Nesma & Morgan), (Geambasu, Jianu, Jianu, & Gavrila, 2011), (Pressman, 2010), (Sommerville, 2011).

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja 1	Media 2	Alta 0
2	Tamaño del proyecto	Pequeño 0	Mediano 1	Grande 2
3	Presupuesto del proyecto	Bajo 0	Medio 0	Alto 2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 2
5	Riesgo asociado al proyecto	Bajo 1	Medio 2	Alto 2
6	Proyecto Exploratorio	Bajo 2	Medio 2	Alto 1
7	Claridad en los requerimientos	Baja 0	Media 1	Alta 2
8	Frecuencia de cambios en los requerimientos	Baja 2	Media 2	Alta 1
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 1	Grande 2
10	Experiencia del equipo de desarrollo	Baja 0	Media 2	Alta 2
11	Comunicación entre stakeholders	Baja 2	Media 1	Alta 0
12	Entrega de funcionalidades parciales	Baja 0	Media 2	Alta 2
13	Reusabilidad del producto de software	Baja 2	Media 2	Alta 2
14	Documentación del proyecto	Baja 0	Media 1	Alta 2

Cuadro 5.5: Evaluación de criterios en el modelo RUP

5.7.1 Complejidad del proyecto en el modelo en Métrica V3

Métrica v 3 ha sido concebida para abarcar el desarrollo completo de Sistemas de Información sea cual sea su complejidad y magnitud, por lo cual su estructura responde a desarrollos máximos y deberá adaptarse y dimensionarse en cada momento de acuerdo a las características particulares de cada proyecto [13].

De esta manera se asignan la siguiente clasificación a Métrica v.3 de acuerdo al criterio Complejidad del proyecto:

C_{11} (Baja): 2 (Aplica)
C_{12} (Media): 2 (Aplica)
C_{13} (Alta): 2 (Aplica)

5.7.2 Tamaño del proyecto en el modelo en Métrica V3

De acuerdo a Caraballo [12], Métrica v3 es un modelo que tiene un amplio análisis funcional, por lo que es recomendado para proyectos extremadamente grandes.

De esta manera se asignan los siguientes valores de acuerdo al criterio Tamaño del proyecto:

C_{21} (Pequeño): 0 (No Aplica)
C_{22} (Mediano): 0 (No Aplica)
C_{23} (Grande): 2 (Aplica)

5.7.3 Presupuesto del proyecto en Métrica V3

Métrica v3 al ser orientado a procesos en busca de ser aplicable a cualquier proyecto de software independientemente de su complejidad y magnitud, requiere de recursos desde personal, documentales que lo pueden hacer costoso. En contraparte, Métrica v3 con el proceso de Aseguramiento de la Calidad que busca disminuir lo costos y ofrece una trazabilidad a los impactos en costos asociados a cambios.

Así, la calificación de Métrica v3 con respecto al presupuesto del proyecto es:

C_{31} (Bajo): 0 (No aplica)
C_{32} (Medio): 2 (Aplica)
C_{33} (Alto): 2 (Aplica)

5.7.4 Tiempo de entrega del proyecto en Métrica V3

Métrica v3 está diseñado para adaptarse a proyectos con tiempos de entrega a corto, mediano y largo plazo. Sin embargo, la complejidad del modelo, la documentación requerida por los diferentes procesos hacen de Métrica v3 poco recomendable para proyectos de software con plazos cortos de entrega. A su vez, es ampliamente recomendado para proyectos de

software con plazos largos en sus tiempos de entrega.

La calificación otorgada a Métrica v3 con respecto al criterio Tiempo de entrega del proyecto es:

C_{41} (Corto plazo): 0 (No Aplica)
C_{42} (Mediano plazo): 1 (Aplica parcialmente)
C_{43} (Largo plazo): 2 (Aplica)

5.7.5 Riesgo asociado al proyecto en Métrica V3

El análisis de los riesgos en Métrica v3 constituye una pieza fundamental en el diseño y desarrollo de sistemas de información seguros. Si bien los riesgos que afectan a un sistema de información son de distinta índole: naturales (inundaciones, incendios, etc.) o lógicos (fallos propios, ataques externos, virus, etc.) son estos últimos los contemplados en la interfaz de Seguridad de Métrica v3 [13].

La calificación de Métrica v3 con respecto al Riesgo asociado al proyecto es:

C_{51} (Bajo): 1 (Aplica parcialmente)
C_{52} (Medio): 2 (Aplica)
C_{53} (Alto): 2 (Aplica)

5.7.6 Proyecto exploratorio en Métrica V3

Métrica v3 ha sido concebida para abarcar el desarrollo completo de Sistemas de Información sea cual sea su complejidad y magnitud, por lo cual su estructura responde a desarrollos máximos y deberá adaptarse y dimensionarse en cada momento de acuerdo a las características particulares de cada proyecto [13].

La calificación de Métrica v3 con respecto al nivel de exploración en el proyecto de software es:

C_{61} (Bajo): 2 (Aplica)
C_{62} (Medio): 2 (Aplica)
C_{63} (Alto): 2 (Aplica)

5.7.7 Claridad en los requerimientos iniciales en Métrica V3

Métrica v3 posee el proceso de Estudio de Viabilidad del Sistema (EVS) para definir los requisitos del sistemas [13]. Aún así, Métrica v3 es recomendado para proyectos de Software en los cuales los clientes no tienen claro los requerimientos iniciales [12].

La calificación de Métrica v3 con respecto al nivel de claridad de los requerimientos iniciales es:

C_{71} (Bajo): 2 (Aplica)
C_{72} (Medio): 2 (Aplica)
C_{73} (Alto): 1 (Aplica parcialmente)

5.7.8 Frecuencia de cambios en los requerimientos en Métrica V3

Métrica v3 cuenta con el proceso Mantenimiento de Sistemas de Información (MSI) el cual permite la gestión de cambios al software [13]. Caraballo [12] añade que Métrica v3 es recomendado para proyectos con requisitos iniciales inestables, cuyos cambios puedan suponer un alto impacto, y grandes desviaciones en los plazos de un proyecto. Por lo tanto, se evidencia la aplicación de Métrica v3 a proyectos con alta frecuencia de cambios en los requerimientos.

La frecuencia de cambios en los requerimientos para Métrica v3 se clasifica en:

C_{81} (Baja): 1 (Aplica parcialmente)
C_{82} (Media): 1 (Aplica parcialmente)
C_{83} (Alta): 2 (Aplica)

5.7.9 Tamaño del equipo de desarrollo en Métrica V3

Métrica v3 es recomendado por Caraballo [12] para su uso en proyectos en los que intervengan multitud de equipos de trabajo, donde la comunicación no siempre sea fácil. De acuerdo a la clasificación de Lohnes [24] Métrica v3 es éxitoso en proyectos con equipos de 25 a 75 personas y no tanto para proyectos con menos de 25 personas.

El tamaño del equipo de desarrollo para el modelo RUP se clasifica en:

C_{91} (Pequeño): 0 (No aplica)
C_{92} (Mediano): 0 (No aplica)
C_{93} (Grande): 2 (Aplica)

5.7.10 Experiencia del equipo de desarrollo en Métrica V3

Proyectos de software ejecutados por un equipo de software con poca experiencia o en con posibilidad de tener una alta rotación de personal son recomendados para Métrica v3 [12] debido a la facilidad y a la amplia documentación que implica usar Métrica v3.

La calificación a Métrica v3 de acuerdo a la experiencia del equipo de desarrollo es:

C_{101} (Baja): 2 (Aplica)
C_{102} (Media): 1 (Aplica parcialmente)
C_{103} (Alta): 1 (Aplica parcialmente)

5.7.11 Comunicación entre stakeholders en Métrica V3

Métrica v3 es útil en proyectos de software con bajo, medio o alto nivel de comunicación con los stakeholders. Sin embargo, Caraballo [12] recomienda usar Métrica v3 en proyectos donde la comunicación no siempre sea fácil.

El nivel de comunicación con los stakeholders para Métrica v3 es:

C_{111} (Bajo): 2 (Aplica)
C_{112} (Medio): 1 (Aplica parcialmente)
C_{113} (Alto): 1 (Aplica parcialmente)

5.7.12 Entrega de funcionalidades parciales en Métrica V3

Métrica v3 tiene un listado de prácticas para alcanzar los objetivos de los proyectos de software, dentro de las cuales se incluye el prototipado, el cual se enfoca a elaborar modelos o maquetas de las interfaces entre el sistema y los usuarios que ayuden a comprender como se producirá la interaccción con el sistema [13].

Por lo anterior, el Métrica v3 tiene la siguiente calificación con respecto a los niveles de entregas parciales requeridas en el proyecto:

C_{121} (Bajo): 1 (Aplica parcialmente)
C_{122} (Medio): 2 (Aplica)
C_{123} (Alto): 1 (Aplica parcialmente)

5.7.13 Reusabilidad del producto de software en Métrica V3

De acuerdo a [33] los proyectos que incluyen artefactos reutilizables requieren de metodologías que soporten un marco de trabajo que pueda usarse repetidamente. Para esto los artefactos reutilizables requieren controles de alta calidad que pueden ser alcanzados por Métrica v3

Métrica v3 tiene la siguiente calificación con respecto al nivel de reusabilidad de productos:

C_{131} (Bajo): 2 (Aplica)
C_{132} (Medio): 2 (Aplica)
C_{133} (Alto): 2 (Aplica)

5.7.14 Documentación del proyecto en Métrica V3

Métrica v3 se caracteriza por la gran cantidad de documentación asociada a cada uno de sus procesos, por lo que es recomendado para proyectos que requieran amplia documentación; sin embargo, no es recomendable para proyectos con un nivel bajo de documentación.

Por lo anterior, Métrica v3 tiene la siguiente calificación con respecto al nivel de docu-

mentación requerida en el proyecto:

C_{141} (Bajo): 0 (No aplica)
C_{142} (Medio): 1 (Aplica parcialmente)
C_{143} (Alto): 2 (Aplica)

Ci	Criterios	Ci1	Ci2	Ci3
1	Complejidad del proyecto	Baja 2	Media 2	Alta 2
2	Tamaño del proyecto	Pequeño 0	Mediano 0	Grande 2
3	Presupuesto del proyecto	Bajo 0	Medio 2	Alto 2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 2
5	Riesgo asociado al proyecto	Bajo 1	Medio 2	Alto 2
6	Proyecto Exploratorio	Bajo 2	Medio 2	Alto 2
7	Claridad en los requerimientos	Baja 2	Media 2	Alta 1
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 1	Alta 2
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 0	Grande 2
10	Experiencia del equipo de desarrollo	Baja 2	Media 1	Alta 1
11	Comunicación entre stakeholders	Baja 2	Media 1	Alta 1
12	Entrega de funcionalidades parciales	Baja 1	Media 2	Alta 1
13	Reusabilidad del producto de software	Baja 2	Media 2	Alta 2
14	Documentación del proyecto	Baja 0	Media 1	Alta 2

Cuadro 5.6: Evaluación de criterios en Métrica V3

5.8 Ejemplo de aplicación del modelo de selección de procesos de software

El gerente de proyecto y/o experto responden el cuestionario asociado a los criterios, clasificando los atributos del proyecto, producto, personas y proceso de software de acuerdo. En el modelo se asigna el valor 1 cuando la respuesta a la pregunta asociada al criterio es afirmativa y 0 cuando es negativa. De esta manera, se asignan los valores a la matriz Pij, donde i es el número del criterio y j la opción de respuesta. Las preguntas son de única respuesta.

Para el ejemplo siguiente el Gerente del proyecto de software SoftX asigna los valores asociados al proyecto. El valor de P12 es 1 debido a que la respuesta a la pregunta del criterio 1: ¿Cuál es la complejidad del proyecto de software? Es Media. Por consiguiente los valores para P11 y P13 es 0.

Los datos registrados para el proyecto se visualizan en la tabla 5.7.

Ci	Criterios	Pi1	Pi2	Pi3
1	Complejidad del proyecto	Baja 0	Media 1	Alta 0
2	Tamaño del proyecto	Pequeño 1	Mediano 0	Grande 0
3	Presupuesto del proyecto	Bajo 0	Medio 1	Alto 0
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 1
6	Proyecto Exploratorio	Bajo 0	Medio 0	Alto 1
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 1
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0
9	Tamaño del equipo de desarrollo	Pequeño 1	Mediano 0	Grande 0
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 1
11	Comunicación entre stakeholders	Baja 1	Media 0	Alta 0
12	Entrega de funcionalidades parciales	Baja 1	Media 0	Alta 0
13	Reusabilidad del producto de software	Baja 1	Media 0	Alta 0
14	Documentación del proyecto	Baja 0	Media 1	Alta 0

Cuadro 5.7: Ejemplo de proyecto de software

5.9 Calculo proyecto ejemplo en modelos de proceso

Luego de registrados los valores aplicables al Proyecto de Software SoftX el modelo calcula el nivel de aplicación (Aij) el cual es el producto de la aplicación del criterio (Cij) y el valor asignado por las respuestas al cuestionarios del modelo (Pij). Luego se suman los valores Aij de cada criterio, para asignarle un puntaje por criterio.

Finalmente, se suman los resultados Aij de cada criterio para dar un Puntaje de Aplicación A al proceso de desarrollo de software. Dicho puntaje se contrastará con los resultados de los demás procesos de desarrollo de software para sugerir el proceso de desarrollo de software más acorde.

5.9.1 Calculo proyecto ejemplo en modelos cascada

Ci	Criterios	Ai1 = Ci1 * Pi1	Ai2 = Ci2 * Pi2	Ai2 = Ci2 * Pi2	Suma
1	Complejidad del proyecto	Baja 0	Media 1	Alta 0	1
2	Tamaño del proyecto	Pequeño 1	Mediano 0	Grande 0	1
3	Presupuesto del proyecto	Bajo 0	Medio 1	Alto 0	1
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0	1
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 1	1
6	Proyecto Exploratorio	Bajo 0	Medio 0	Alto 0	0
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 2	2
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0	1
9	Tamaño del equipo de desarrollo	Pequeño 1	Mediano 0	Grande 0	1
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 2	2
11	Comunicación entre stakeholders	Baja 2	Media 0	Alta 0	2
12	Entrega de funcionalidades parciales	Baja 0	Media 0	Alta 0	0
13	Reusabilidad del producto de software	Baja 2	Media 0	Alta 0	2
14	Documentación del proyecto	Baja 0	Media 1	Alta 0	1
				Puntaje:	17

Cuadro 5.8: Ejemplo de evaluación en modelo cascada

5.9.2 Calculo proyecto ejemplo modelo en V

Ci	Criterios	Ai1 = Ci1 * Pi1	Ai2 = Ci2 * Pi2	Ai2 = Ci2 * Pi2	Suma
1	Complejidad del proyecto	Baja 0	Media 1	Alta 0	1
2	Tamaño del proyecto	Pequeño 1	Mediano 0	Grande 0	1
3	Presupuesto del proyecto	Bajo 0	Medio 1	Alto 0	1
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0	1
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 1	1
6	Proyecto Exploratorio	Bajo 0	Medio 0	Alto 0	0
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 2	2
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0	1
9	Tamaño del equipo de desarrollo	Pequeño 1	Mediano 0	Grande 0	1
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 2	2
11	Comunicación entre stakeholders	Baja 2	Media 0	Alta 0	2
12	Entrega de funcionalidades parciales	Baja 0	Media 0	Alta 0	0
13	Reusabilidad del producto de software	Baja 2	Media 0	Alta 0	2
14	Documentación del proyecto	Baja 0	Media 1	Alta 0	1
				Puntaje:	**17**

Cuadro 5.9: Ejemplo de evaluación en modelo en V

5.9.3 Calculo proyecto ejemplo en espiral

Ci	Criterios	Ai1 = Ci1 * Pi1	Ai2 = Ci2 * Pi2	Ai2 = Ci2 * Pi2	Suma
1	Complejidad del proyecto	Baja 0	Media 2	Alta 0	2
2	Tamaño del proyecto	Pequeño 0	Mediano 0	Grande 0	0
3	Presupuesto del proyecto	Bajo 0	Medio 0	Alto 0	0
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0	1
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 1	1
6	Proyecto Exploratorio	Bajo 2	Medio 0	Alto 0	2
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 1	1
8	Frecuencia de cambios en los requerimientos	Baja 2	Media 0	Alta 0	2
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 0	Grande 0	0
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 2	2
11	Comunicación entre stakeholders	Baja 0	Media 0	Alta 0	0
12	Entrega de funcionalidades parciales	Baja 1	Media 0	Alta 0	1
13	Reusabilidad del producto de software	Baja 2	Media 0	Alta 0	2
14	Documentación del proyecto	Baja 0	Media 2	Alta 0	2
				Puntaje:	16

Cuadro 5.10: Ejemplo de evaluación en modelo espiral

5.9.4 Calculo proyecto ejemplo en RUP

Ci	Criterios	Ai1 = Ci1 * Pi1	Ai2 = Ci2 * Pi2	Ai2 = Ci2 * Pi2	Suma
1	Complejidad del proyecto	Baja 0	Media 1	Alta 0	0
2	Tamaño del proyecto	Pequeño 0	Mediano 0	Grande 0	0
3	Presupuesto del proyecto	Bajo 0	Medio 0	Alto 0	0
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0	1
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 2	2
6	Proyecto Exploratorio	Bajo 2	Medio 0	Alto 0	2
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 2	2
8	Frecuencia de cambios en los requerimientos	Baja 2	Media 0	Alta 0	2
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 0	Grande 0	0
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 2	2
11	Comunicación entre stakeholders	Baja 2	Media 0	Alta 0	2
12	Entrega de funcionalidades parciales	Baja 0	Media 0	Alta 0	0
13	Reusabilidad del producto de software	Baja 2	Media 0	Alta 0	2
14	Documentación del proyecto	Baja 0	Media 1	Alta 0	1
				Puntaje:	16

Cuadro 5.11: Ejemplo de evaluación en modelo en RUP

5.9.5 Calculo proyecto ejemplo en Métrica v3

Ci	Criterios	Ai1 = Ci1 * Pi1	Ai2 = Ci2 * Pi2	Ai2 = Ci2 * Pi2	Suma
1	Complejidad del proyecto	Baja 0	Media 2	Alta 0	2
2	Tamaño del proyecto	Pequeño 0	Mediano 0	Grande 0	0
3	Presupuesto del proyecto	Bajo 0	Medio 2	Alto 0	2
4	Tiempo de entrega del proyecto	Corto plazo 0	Mediano plazo 1	Largo plazo 0	1
5	Riesgo asociado al proyecto	Bajo 0	Medio 0	Alto 2	2
6	Proyecto Exploratorio	Bajo 2	Medio 0	Alto 0	2
7	Claridad en los requerimientos	Baja 0	Media 0	Alta 1	1
8	Frecuencia de cambios en los requerimientos	Baja 1	Media 0	Alta 0	1
9	Tamaño del equipo de desarrollo	Pequeño 0	Mediano 0	Grande 0	0
10	Experiencia del equipo de desarrollo	Baja 0	Media 0	Alta 1	1
11	Comunicación entre stakeholders	Baja 2	Media 0	Alta 0	2
12	Entrega de funcionalidades parciales	Baja 1	Media 0	Alta 0	1
13	Reusabilidad del producto de software	Baja 2	Media 0	Alta 0	2
14	Documentación del proyecto	Baja 0	Media 1	Alta 0	1
				Puntaje:	18

Cuadro 5.12: Ejemplo de evaluación en modelo en Métrica v3

5.9.6 Resultados de la aplicación del modelo de selección procesos de software

De acuerdo a los puntajes obtenidos el proceso de software recomendado es Métrica v3 (Puntaje:18), seguido de Cascada (Puntaje:17) y el modelo en V (Puntaje:17).

Proceso de desarrollo de Software	Puntaje
Métrica V3	18
Cascada	17
Modelo en V	17
RUP	16
Espiral	16

Cuadro 5.13: Resultados de la evaluación de los procesos de software en el ejemplo

IV

Conclusiones

Conclusiones 107

Conclusiones

- El análisis del estado de los proyectos de software a nivel global, nacional y local demuestran que a pesar de los avances en más de 50 años de la ingeniería del software un alto procentaje de los proyectos ya que siguen presentando fallas en tiempos de entrega, sobrecostos, calidad del producto, etc.

- La evolución de los procesos de desarrollo de software es debido a que cada proceso que emerge busca ser la solución a las fallas y errores presentes en los procesos predecesores. Sin embargo, los procesos de software aplicables a los proyectos deben tener en cuenta una gran variedad de factores, como lo son: la complejidad del proyecto, el tamaño, los riesgos, el tamaño del equipo, la claridad en los requerimientos, el nivel de comunicación de los stakeholders, entre otros. Estos factores hacen que cada proyecto de software sea único por lo que el proceso de software que guiará el producto de software no debe ser asignado por moda, gusto o experiencias en otros proyectos, por el contrario, debe ser seleccionado de acuerdo a las características del proyecto, las personas, el producto y el entorno.

- Los trabajos e investigaciones realizadas con respecto a la selección de procesos de desarrollo de software, muestran concordancias entre los autores con respecto a los criterios que más impactan el desarrollo del software. Sin embargo, debido a la diversidad de estudios no hay conceptos o modelos concisos, formales y conocidos para que la selección del proceso de software sea considerada una actividad primordial de los proyectos. Por otra parte, algunos trabajos e investigaciones se quedan en la definición de los criterios, sin definir un modelo matemático o formal que apoye la selección de

procesos de software.

- El modelo de desarrollo de software creado en este trabajo de grado permite a los interesados en el proyecto seleccionar el proceso de desarrollo de software más acorde a las particularidades del mismo, teniendo en cuenta los criterios que más afectan el desarrollo de software y los procesos de desarrollo de software más relevantes.

- La planeación del trabajo de grado, en términos de actividades, tiempos, recursos y presupuesto permitieron desde una etapa temprana identificar todas las variables necesarias para lograr ejecutar con éxito la creación del modelo de selección de procesos de software.
  - El cronograma de actividades permitió tener seguimiento y control de las actividades e hitos de entrega dentro de los tiempos estipulados.
  - La identificación de los roles necesarios para la ejecución de las actividades del trabajo de grado permitió realizar la asignación de los recursos humanos. Ya que la fuerza de trabajo solo disponía de 2 recursos la asignación de tareas se realizó basándose en las habilidades, conocimientos y fortalezas de cada recurso.
  - El cálculo del gasto presupuestal permitió valorar cual es el capital requerido para la asignación de recursos humanos, materiales y logísticos.

- La utilización de los libros, artículos, investigaciones relevantes en el campo de la Ingeniería del Software fueron pieza fundamental para el creación del modelo de selección de procesos de desarrollo de software.

- Tras realizar la creación del modelo para la selección de procesos de desarrollo de software, se concluye:
  - El modelo permite ajustes y definición de procesos y/o metodologías de desarrollo de softwrae para hacerlo más robusto y completo.
  - En el proceso de definición de los atributos del proyecto de software se requerirá de personal experimentado que conozca al detalle el proyecto a desarrollar, los stakeholders involucrados, la organización y su cultura organizacional, los riesgos del proyecto, la complejidad del proyecto y demás tributos relevantes con el fin de que los resultados del modelo sean acordes al proyecto.
  - El modelo de selección de procesos de software debe estar bajo un software que permita a los interesados el ingreso de las características del proyecto a partir de la resolución de un cuestionario sencillo.

Bibliografía

[1] A. Abuchar, B. Cárdenas y D. López. “Observatorio de practicas de desarrollo de Software en MinPyme y pymes de Bogota”. En: *Revista Ciencia e Ingenieria* (2012) (véase página 49).

[2] L. Alexander y A. Davis. “Criteria for Selecting Software Process Models”. En: (1991) (véanse páginas 46, 67).

[3] M. Awad. “A Comparison between Agile and Traditional Software Development Methodologies”. En: *School of Computer Science and software Engineering* (2005) (véase página 67).

[4] M. Ben-Zahia e I. Jaluta. “Criteria for Selecting Software Development”. En: (2014) (véanse páginas 82, 83).

[5] B. Boehm. “A spiral model of software development and enhancement”. En: *ACM SIGSOFT Software Engineering Notes* (1986) (véanse páginas 34, 66, 82).

[6] S. Bolaños. *Metaproceso de Desarrollo de software*. Bogotá: Editorial UD, 2016 (véanse páginas 32, 34, 36).

[7] E. Braude. *Ingeniería de software, una perspectiva orientada a objetos*. México D.F.: Alfaomega, 2007 (véase página 84).

[8] ISTQB Exam Certification. *What is V-model- advantages, disadvantages and when to use it?* 2016. URL: `http://istqbexamcertification.com/what-is-v-model-advantages-disadvantages-and-when-to-use-it/` (véanse páginas 76, 78).

[9] O. Cervantes y M. Goméz. “Taxonomía de los modelos y metodologías de desarrollo de software más utilizados”. En: *Universidades* (2012) (véanse páginas 81, 83, 84).

[10] A. Chavhan. *V Model Advantages and Disadvantages*. 2015. URL: http://www.softwaretestingandistqb.com/v-model-advantages-and-disadvantages/ (véanse páginas 75-77).

[11] A. Cockburn. “Selecting a project’s methodology”. En: *IEEE Software* (2002) (véase página 47).

[12] Ministerio de Hacienda y Función Pública de España (véanse páginas 91-94).

[13] Ministerio de Hacienda y Función Pública de España. *Métrica v.3*. 2018. URL: https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html (véanse páginas 91-94).

[14] H. Faridani. *A guide to selecting software development methodologies*. 2011. URL: https://www.yumpu.com/en/document/view/33848071/a-guide-to-selecting-software-development-methodologies-gt-islig (véanse páginas 64-66, 77).

[15] Galorath. *Software Project Failure Costs Billions... Better Estimation Planning Can Help*. 2012. URL: http://galorath.com/blog/software-project-failure-costs-billions-better-estimation-planning-can-help/ (véase página 42).

[16] C. Geambasu y col. “Influence factors for the choice of a Software Develepment Methodology”. En: *Accounting and Management Information Systems* (2011) (véanse páginas 47, 84, 86-89).

[17] J. Highsmith. *Agile Software Development Ecosystems*. Addison-Wesley, 2002 (véase página 65).

[18] Standish Group International Inc. “Chaos chronicles”. En: (2015) (véase página 39).

[19] Inteco. *Curso de introducción a la ingeniería del software*. Madrid: Ministerio de Industria, turismo y comercio. Gobierno de España., 2009 (véanse páginas 75, 77).

[20] N. Keshta y M. Yasser. “Comparison between Traditional Plan-based and Agile Software Processes According to Team Size Project Domain”. En: () (véase página 47).

[21] R. Khan A. Qurashi y U. Khan. “A Comprehensive Study of Commonly Practiced Heavy and Light Weight Software Methodologies”. En: *IJCSI International Journal of Computer Science Issues* (2011) (véase página 82).

[22] P. Kruchten. “A Rational Development Process”. En: *Crosstalk* (1996) (véase página 88).

[23] Y. Leau y col. “Software Development Life Cycle AGILE vs Traditional Approaches”. En: (2015) (véase página 47).

[24] P. Lohnes. *Comparisons of Agile vs. Traditional Approaches*. 2016. URL: http://blogs.managementconcepts.com/comparisons-of-agile-vs-traditional-approaches/#.WJ-6ZIWcHVK (véanse páginas 67, 77, 88, 93).

[25] J. Lynch. *Standish Group 2015 Chaos Report - QA with Jennifer Lynch*. 2016. URL: https://www.infoq.com/articles/standish-chaos-2015 (véanse páginas 40, 41, 43).

[26] S. Pfleeger y J. Atlee. *Ingeniería de Software, teoría y práctica*. Buenos Aires: Pearson Education, 2016.

[27] R. Pressman. *Ingeniería del Software. Un enfoque práctico*. México D.F.: McGrawHill, 2010 (véanse páginas 30, 34, 44).

[28] Rational. *Rational Unified Process Best Practices for Software Development Teams*. 2001. URL: https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf (véase página 89).

[29] N. Russo. "The use and adaptation of system development methodologies". En: *International Resources Management Association International Conference* (1995) (véase página 47).

[30] M. Sami. *Software Engineering Architecture Practices*. 2012. URL: https://melsatar.blog/2012/03/21/choosing-the-right-software-development-life-cycle-model/ (véanse páginas 72, 75, 77, 78).

[31] I. Sommerville. *Ingeniería del Software*. Madrid: Pearson Education, 2011 (véanse páginas 17, 31, 32, 44, 65, 66, 76, 78, 83).

[32] M. Stoica, M. Mircea y B. Ghilic-Micu. "Software development:Agile vs. traditional". En: (2013) (véase página 47).

[33] B. Turk, D. France y R. Rumpe. "Limitations of Agile Software Processes". En: *Systems Engineering* (2002) (véanse páginas 68, 78, 83, 94).

[34] D. Whitfield. *Cost overruns, delays and terminations: 105 outsourced public sector ICT projects*. 2007. URL: https://www.european-services-strategy.org.uk/publications/essu-research-reports/essu-research-report-no-3-cost-overruns-delays (véase página 42).