MARCO DE TRABAJO PARA LA ADOPCIÓN DE UN ENFOQUE EN EL DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

Jhovanny Andrés Cañas Pino

Trabajo de tesis para optar al título de Magister en Ingeniería de Software

Asesor temático: Fernán Alonso Villa Garzón PhD.

Asesor metodológico: Juan Bernardo Quintero Magister en ingeniería informática.

UNIVERSIDAD DE MEDELLIN

FACULTAD DE INGENIERIAS

MEDELLÍN - ANTIOQUIA

2017

Tabla de contenido

LISTA DE TABLAS

LISTA DE FIGURAS

RESUMEN DEL TRABAJO DE GRADO: MARCO DE TRABAJO PARA LA ADOPCIÓN DE UN ENFOQUE EN EL DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MÓVILES

Autor: Jhovanny Andrés Cañas Pino

Programa: Maestría en Ingeniería de Software.

Asesores: Fernán Alonso Villa, Juan Bernardo Quintero.

Universidad de Medellín

Medellín

2017

El desarrollo de software que funciones en dispositivos móviles presenta una serie de características y requerimientos únicos que afectan cada uno de las fases del ciclo de vida del proyecto. En la actualidad son utilizados diferentes métodos y prácticas para la construcción de software para móviles provenientes de metodologías tradicionales, enfoques agiles y propuestas específicas destinadas exclusivamente a este propósito. La heterogeneidad de enfoques representa una dificultad al momento de afrontar un proyecto de esta naturaleza, el entorno complejo, dinámico y competitivo del mercado de software móvil representa una presión adicional, sumado a la complejidad inherente de los proyectos de software constituyen un importante desafío tanto para la comunidad científica de la ingeniería de software, desarrolladores e industria.

El objetivo de esta tesis es desarrollar un marco de trabajo para el desarrollo de software orientado a dispositivos móviles, que sirva como un referente de solución para este tipo de proyectos .Permitiendo a desarrolladores, equipos y clientes la posibilidad de instanciar y adaptar el marco de trabajo según las necesidades de un proyecto móvil, orientando el camino adecuado en cada una de las fases del ciclo de vida que contribuya al desarrollo de software de calidad, con un óptimo esfuerzo de tiempo y recursos.

INTRODUCCIÓN

El rápido auge de los dispositivos móviles en la última década ha desencadenado una notable demanda de software especializado que funcione en ellos denominadas aplicaciones móviles o APPS. Estas aplicaciones aprovechan las crecientes capacidades tecnológicas de los dispositivos, el aumento en cantidad y calidad de las redes inalámbricas de datos y los avances en computación ubicua para proporcionar una rica variedad de servicios y funcionalidades a los usuarios. Los acelerados avances en la tecnología de computación móvil y la creciente demanda de software para dispositivos móviles, plantean desafíos en el desarrollo de este tipo de aplicaciones posicionándose como un nuevo campo de acción de la ingeniería de software.

El desarrollo de software para dispositivos móviles presenta una serie de características y requerimientos únicos que influyen en cada una de las etapas del ciclo de vida, las cuales serán descritas en mayor detalle en los siguientes capítulos ,las cuales se enmarcan en un contexto caracterizado por un nivel alto de competitividad y dinamismo, con el objetivo de construir software que funcione adecuadamente en entornos heterogéneos y con limitados recursos con que están provistos los dispositivos móviles, asegurando satisfacer las expectativas de desempeño, calidad y rendimiento de los usuarios finales, lo anterior añade mayor complejidad a la implícita, que sumado a la incertidumbre intrínseca de los proyectos de software representa una significativa dificultad al momento de iniciar un proyecto móvil para empresas y desarrolladores.

La propuesta de diseñar un marco de trabajo para ser utilizado durante el desarrollo de software que funcione en dispositivos móviles objeto de la presente tesis, intenta proveer un marco de abstracción general de la solución que pueda ser instanciado y particularizado según las necesidades particulares de un proyecto móvil, permitiendo guiar adecuadamente los esfuerzos hacia el desarrollo eficaz de software para móviles. El marco de trabajo se definió mediante el análisis de las características presentes en el software móvil y el examen de las principales propuestas metodológicas utilizadas en su desarrollo, para luego identificar las actividades que permiten afrontar los requerimientos del dominio en cada una de las fases del ciclo de vida.

El resto del documento está estructurado de la siguiente manera: el capítulo 1 contiene el marco metodológico de la presente tesis donde se detalla el planteamiento del problema, la hipótesis, se formulan los objetivos a alcanzar y la metodología utilizada, en el capítulo 2 se realiza una revisión de literatura asociada con la temática, el capítulo 3 se presenta un resumen de las aproximaciones metodológicas encontradas en la literatura para el desarrollo de software móvil y se identifican las principales características del dominio, el capítulo 4 se diseña y construye el marco de trabajo , el capítulo 5 se describe el experimento para evaluar

el marco de trabajo y se exponen los resultados de su ejecución, el capítulo 5 presenta las conclusiones y oportunidades de trabajos futuros derivados de la tesis.

CAPITULO1
MARCO METODOLÓGICO

1. PLANTEAMIENTO DEL PROBLEMA

El mercado de software para dispositivos móviles ha presentado una tendencia de fuerte y rápido crecimiento en la última década, los dispositivos móviles (tabletas, teléfonos inteligentes, relojes entre otros) son cada vez más provistos en potencia de procesamiento y almacenamiento, muchos de ellos cuentan con multiplicidad de prestaciones físicas tecnológicas expresada en la cantidad de sensores incorporados; que ha impulsado la demanda de software especializado que aproveche las capacidades de los dispositivos, y que por otro lado satisfaga las expectativas por parte de los usuarios. (Syer, Adams, Zou, & Hassan, 2011) (Dancer & Dampier, 2010) El lanzamiento del Iphone por la compañía Apple en 2007, desencadeno un crecimiento en la demanda de software para teléfonos inteligentes, los cuales anteriormente solo prestaban la funcionalidad de voz, mensajes de texto y en algunos casos con multimedia. Seguido a esto el mercado de dispositivos móviles creció de manera exponencial, cada proveedor inundo el mercado de dispositivos, muchos de estos fabricantes desarrollaron sus propias plataformas que soportaran sus tecnologías lo que dio a lugar a los sistemas operativos para dispositivos móviles, entre los que se destacan: iOS® de Mac, Android® de Google, BlackBerry® OS RIM, Symbian® de Nokia, windows mobile® de Microsoft. (Hammershøj, Sapuppo, & Tadayoni, 2010) (Lee & Lee, 2011)

Se estima que para el 2013 los dispositivos móviles superaran a los computadores de escritorio como los dispositivos usados comúnmente para acceder a la web; Smartphone y tabletas representaran un crecimiento neto del 90% como el nuevo dispositivo de adopción de 2012 a 2016. (Gartner, 2012)En consecuencia la demanda de software que suplan las necesidades de los usuarios es provocada por esta tendencia, por lo que las empresas y desarrolladores de software para móviles deben ser capaces de afrontar el reto de construir software de calidad a un ritmo acelerado.

El desarrollo de software para móviles presenta características y restricciones intrínsecas únicas que no se presentan frecuentemente en otros tipos de desarrollo, dadas principalmente por la naturaleza limitada de la tecnología de los dispositivos y por otro lado el entorno dinámico y de acelerados avances en que se encuentra. Lo anterior expuesto repercute en cada una de las fases de desarrollo de software y en consecuencia en el producto final esperado, software que funcione correctamente en dispositivos móviles.

La ausencia de un marco de trabajo para el desarrollo de software para dispositivos móviles que proporcione el camino adecuado para afrontar esta clase de desarrollos, representa una falencia en el momento de iniciar un proyecto de desarrollo de

software para dispositivos móviles, esto debido a el panorama del desarrollo móvil enmarcado en una diversidad de plataformas de desarrollo, cambios constantes acelerados de la industria de los dispositivos, sistemas operativos y las restricciones intrínsecas de los dispositivos, con lleva a la adopción de heterogeneidad de enfoques de desarrollo, que son adaptados durante el desarrollo móvil, ocasionando retrasos en cronogramas, sobrecostos y problemas de calidad en el producto final. (Gartner, 2010) (Forman & Zahorjan, 1994) (Dunlop, 2002)

1.1 PREGUNTAS DE INVESTIGACIÓN

Con base la literatura se detectaron vacíos conceptuales que dan pie a las siguientes preguntas:

- ¿Los enfoques utilizados en el desarrollo de software tradicional son eficaces en proyectos de desarrollo de software móviles?
- ¿Cuáles son las aproximaciones de tipo metodológicas existentes para el desarrollo de software para dispositivos móviles?
- ¿La ausencia de un marco de trabajo para el desarrollo móvil ocasiona problemas en los proyectos de desarrollo de software para dispositivos móviles?

1.2 HIPÓTESIS

El mercado de software para dispositivos móviles crece exponencialmente, en la actualidad muchas empresas de software y desarrolladores han volcado sus esfuerzos en suplir la necesidad de software móvil, caracterizado por su acelerado cambio y dinamismo en hardware y Sistemas operativos. Los usuarios y empresas demandan diferentes tipos de aplicaciones móviles con altos estándares de calidad, los proyectos de este tipo deben considerar las características del desarrollo móvil y adoptar uno o más enfoques de desarrollo para adaptarlo en el proceso de desarrollo, de esto último necesidad de contar con un proceso sistemático que sea un referente en el desarrollo de software para dispositivos móviles que permita disminuir los sobrecostos, los retrasos en tiempos de entrega y la falta de calidad del software para móviles. La adopción de un marco de trabajo para el desarrollo de software para dispositivos móviles contribuye a la disminución en los retrasos en los tiempos de entrega del producto, reducción de costos y aumento de la calidad del producto final, en este caso el software para dispositivo móvil.

1.3 JUSTIFICACIÓN

En la actualidad los dispositivos móviles han cambiado drásticamente las actividades cotidianas de las personas, esto debido a que los dispositivos han pasado de capacidades de comunicaciones básicas como voz y mensajes de texto cortos, aun gran número de funcionalidades y servicios que aprovechan las capacidades tecnológicas provistas en ellos, permitiendo a los usuarios disponer de una amplia variedad de software que les otorgan experiencias en su uso altamente valoradas, las cuales están cambiando la forma como interactúan con su entorno.

El continuo y rápido mejoramiento de las prestaciones tecnológicas de los dispositivos, acompañado de su amplia masificación en las personas ha disparado la necesidad de software que satisfaga las expectativas los consumidores, es por esto que el software móvil cada vez gana relevancia en la actualidad para empresas de software y desarrolladores.

Con un mercado de software móvil creciendo exponencialmente, la lucha por la cuota de usuarios de los diferentes sistemas operativos va creciendo, a la par en el aumento de aplicaciones desarrolladas en las diferentes plataformas con el objetivo de ser implementadas en los dispositivos móviles. El software móvil presenta características especiales al desarrollo tradicional como: variedad de plataformas fragmentadas cuya tendencia va en aumento cada una con consideraciones restrictivas en la plataforma, restricciones intrínsecas de los dispositivos móviles: tamaño, procesamiento, interfaz de usuario, variedad de aplicaciones: los dispositivos móviles cuentan con un gran número de prestaciones en su hardware: GPS, acelerómetro, termómetro, entre otras; (Graf & Jung, 2012) el panorama de desarrollo para este ambiente resulta ser un desafío para empresas y desarrolladores.

Con el anterior panorama, las organizaciones y desarrolladores enfrentan retos en el momento de tomar decisiones destinadas al desarrollo de soluciones de software móviles, en un mercado caracterizado por la constante innovación y velocidad, estas decisiones resultan cruciales para el éxito del proyecto. Desarrollar software para dispositivos móviles es más que la noción simple de un proyecto de software de nueva tecnología adaptado a un dispositivo de tamaño reducido, las características y restricciones del software móvil afectan a todos etapas del ciclo de vida del proyecto, por consiguiente el producto final, es entonces donde radica la importancia de desarrollar un marco de trabajo para el desarrollo de software para dispositivos móviles que contribuya al desarrollo eficaz de aplicaciones móviles.

1.4 OBJETIVOS

Objetivo general

Diseñar un marco de trabajo para el desarrollo de software para dispositivos móviles.

Objetivos específicos

1. Explorar los métodos de desarrollo utilizados en el desarrollo de software para dispositivos móviles en la literatura más relevante del medio.

2. Contrastar las metodologías de desarrollo para software para dispositivos móviles.

3. desarrollar el marco de trabajo para el desarrollo de software para dispositivos móviles.

4. Evaluar el marco de trabajo para el desarrollo de software para dispositivos móviles a través del enfoque de ingeniería de software experimental.

2. METODOLOGIA

Para el logro de los objetivos definidos, en la primera parte de este trabajo capítulos 2 y 3, se realiza un revisión de las diferentes aproximaciones disponibles en la literatura enfocada en el desarrollo de software para dispositivos móviles, las cuales incluyen, metodologías, técnicas, patrones de diseño y metamodelos.

Paso seguido se lleva a cabo un análisis de las características presentes en el dominio, en este caso el desarrollo de software que funciona en terminales móviles, que permitan obtener el conocimiento sobre los requerimientos y objetivos buscados en este tipo de proyectos.

Una vez obtenido las características presentes en el dominio, se procede a contrastar dichas características en relación con las propuestas metodológicas, que cuyo resultado permita identificar los requerimientos base del marco de trabajo buscado.

El componente empírico, los capítulos 4 y 5;corresponde al diseño y construcción del marco de trabajo que permita satisfacer los requerimientos del desarrollo de software

móvil, luego a continuación basado en un enfoque de ingeniería de software experimental se diseña y ejecuta un experimento con el objetivo de evaluar el aporte del marco de trabajo propuesto.

CAPITULO 2
MARCO TEORICO

1. INGENIERIA DE SOFTWARE

La ingeniería de software, en términos prácticos es la disciplina que involucra el diseño, desarrollo, mantenimiento y documentación de sistemas de software mediante la aplicación de técnicas de ingeniería. (Chaudron, y otros, 2002) En el medio coexisten diferentes opiniones de la definición de exacta sobre la ingeniería de software, a continuación se presenta algunas definiciones encontrada en la literatura:

La IEEE Computer Society proporciona la siguiente definición:

Es la aplicación de un enfoque sistemático, disciplinado y cuantificable al desarrollo, operación y mantenimiento de software, y el estudio de estos enfoques, es decir, la aplicación de la ingeniería al software. (IEEE, 1990)

Barry Bohem otorga siguiente definición:

Ingeniería de software es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software. (Bohem, 1976)

Fritz Bauer propone que la ingeniería de software es:

El establecimiento y uso de principios fundamentales de la ingeniería con objeto de desarrollar en forma económica software que sea confiable y que trabaje con eficiencia en máquinas reales. (Naur & Randall, 1969)

2. DESARROLLO DE SOFTWARE PARA DISPOSITIVOS MOVILES

El desarrollo de software que funciona en dispositivos móviles o la ingeniería de software de aplicaciones móviles representan un nuevo ángulo de investigación dentro de la ingeniera de software que aborda las necesidades y oportunidades en este campo emergente. En muchos aspectos, el desarrollo de aplicaciones móviles es similar a la ingeniería de software para otras aplicaciones embebidas (Wasserman, 2010).Algunas cuestiones comúnmente encontradas en este tipo de

desarrollos incluye integración con el hardware del dispositivo, desempeño, limitaciones de almacenamiento, confiablidad y seguridad. Otras características reportadas en la literatura que representan requerimientos adicionales que no se encuentran tan comúnmente en el desarrollo de software tradicional se detallan en la sección 2 del capítulo 3.

3. MARCO DE TRABAJO (FRAMEWORK)

Un marco de trabajo (*Framewok,* por su origen en inglés*)* proporciona un marco de abstracción de una solución para un número de problemas que tienen las mismas similitudes. Generalmente un marco de trabajo esboza o proporciona la idea general de los pasos o fases que se deben seguir en la implementación de una solución sin entrar en los detalles de cuales actividades son realizadas en cada fase. (Mnkandla, 2009)

En la tabla 2-1 se presentan algunos de los marcos de trabajo comúnmente utilizados en ingeniería de software.

Tabla 2-1.Marcos de trabajo (Framework) comúnmente utilizados en ingeniería de software.

Framework	Completo/Parcial	Área de Enfoque	Año de origen
Unified Process	Completo	Ciclo de vida de desarrollo	1988
Agile	Completo	Ciclo de vida de desarrollo	2001
Lean Software Development	Parcial	Administración del Ciclo de vida de desarrollo	2001
Adaptative Systems Development	Parcial	Administración del Ciclo de vida de desarrollo	1990
Rational Unified Process	Completo	Ciclo de vida de desarrollo	1998
PMBOK	Completo	Ciclo de vida del proyecto	1987
PRINCE2	Completo	Ciclo de vida del proyecto	1989
Microsoft Solutions Framework	Completo	Ciclo de vida de desarrollo	1993

4. METODOLOGIAS AGILES

Representan un relativo nuevo enfoque para el desarrollo de software que ofrecen una respuesta a la necesidad de responder al entorno impredecible y cambiante de los proyectos de software, surgen como una alternativa de desarrollo en relación a los enfoques tradicionales.

La motivación por un enfoque alternativo en el desarrollo de software dio origen al denominado movimiento ágil en la industria del software, donde en 2001 un grupo de académicos y expertos publican el manifiesto ágil para el desarrollo de software (Cunningham, 2015), el cual contiene las principales características de los métodos agiles que se derivan de una serie de principios; los valores y principios que reza en el manifiesto son:

Valores:

Individuos e interacciones sobre procesos y herramientas

Software funcionando sobre documentación extensiva

Colaboración con el cliente sobre negociación contractual

Respuesta ante el cambio sobre seguir un plan

Los principios que fundamentan los métodos agiles son:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.

- Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

En la última década el movimiento ágil ha crecido notablemente tanto en número de seguidores como en variantes metodológicas que han sido ampliamente difundas en el sector académico e industrial; a continuación se presenta un breve análisis de la selección de las principales metodologías agiles, cuyo criterio de inclusión fueron aquellas reconocidas como las abanderadas del movimiento ágil:

4.1 DSDM

DSDM (Dynamic System Development Method) es el resultado de la conformación de un consorcio de empresas de TI de la gran Bretaña en el año 1994 sin fines de lucro, motivadas por la necesidad de comprender las mejores prácticas en el desarrollo de aplicaciones, con el fin de enseñarlos e implementarlos ampliamente entre sus miembros, (DSDM Consortium, 2003) no obstante a partir del año 2007 el pleno apoyo de sus miembros decidieron hacer universalmente disponible DSDM para su lectura y uso. (DSDM Consortium, 2015)

El consorcio fue fundado por 17 miembros que abarcaban un amplio espectro relacionado con la industria de TI de Inglaterra, hoy en día el consorcio cuenta con cientos de miembros en diferentes países fuera de Inglaterra, con fuertes expectativas a su crecimiento en otras zonas geográficas.

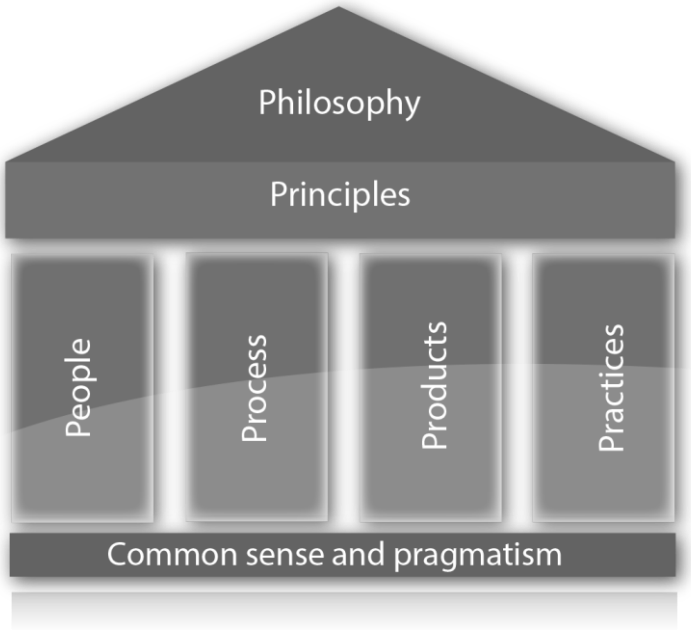
DSDM se concibe como un Framework para la entrega de proyectos que responde a las necesidades del negocio, proyectos DSDM se entregan dentro del cronograma, con el presupuesto y sin eliminar aspectos importantes.

El DSDM es un Framework genérico iterativo e incremental basado en la metodología RAD (Rapid Application Development), enfocado en el involucramiento constante del usuario y los involucrados, con la premisa de entrega constantes del producto, en adicción de otros principios considerados de la corriente de desarrollo ágil.

La filosofía del DSDM es:

“El mejor valor agregado para el negocio surge cuando se alinean los proyectos para aclarar los objetivos del negocio, entrega frecuente de productos e involucrar la colaboración de personas motivadas y empoderadas”.

La filosofía esta soportada por 8 principios que constituyen el pensamiento y el comportamiento para que la filosofía cobre vida (ver figura 2-1):



Ecuación 1

Figura 2-1. Composición de DSDM.

1. Enfoque en las necesidades del negocio
2. Entrega a tiempo
3. Colaboración.
4. Nunca comprometer la calidad
5. Construir incrementalmente desde bases solidas
6. Desarrollo iterativo
7. Comunicación clara y constante
8. Demostrar control

Estos principios están soportados por:

- Personas con roles y responsabilidades definidas
- Procesos agiles, ciclo de vida de desarrollo y de entregas iterativo e incremental
- Productos claros y definidos
- Practicas recomendadas para ayudar a lograr los mejores resultados

DSDM según el consorcio se basa en dos valores que subyacen todo el Framework que ellos han definido de la siguiente manera:

Sentido común

"*sound practical judgment independent of specialised knowledge or training; normal native intelligence*."

Pragmatismo:

"*action or policy dictated by consideration of the immediate practical consequences r ather than by theory or dogma*."

La última versión del DSDM proporciona una serie de procesos iterativos e incrementales, compuestos de cinco fases principales, las cuales están acompañadas de dos fases ubicados en el inicio y al final del proyecto, dando un total de siete fases, con una serie de productos definidos destinados a contribuir a la evolución de la solución y el correcto funcionamiento del marco ágil del proyecto, en la figura 2-2 se muestra las fases de DSDM; a continuación un breve descripción de cada una de las fases:

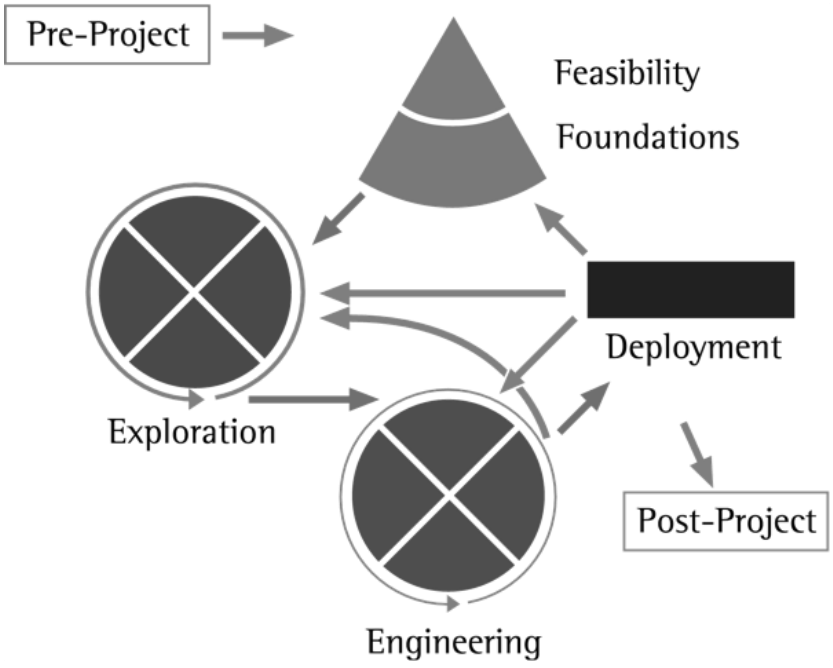


Figura 1-2. Proceso DSDM.

Pre-proyecto

Se asegura el inicio de aquellos proyectos adecuados y que se encuentran correctamente configurados, sobre la base de un objetivo claramente definido, dentro de sus tareas primordiales se encuentra:

- Describir el problema de negocio que se debe abordar
- Asegurar que el proyecto está alineado con la estrategia del negocio
- Planificar el alcance y los recursos de la fase viabilidad

Fase factibilidad

Se analiza la factibilidad técnica y económica del proyecto. El esfuerzo asociado a la factibilidad contribuye a decidir si el proyecto debe ser detenido o reevaluado con base a una investigación extra debido a la poca probabilidad de su viabilidad. Las tareas principales de la fase son:

- Establecer si existe una solución viable al problema de negocio descrito en la fase de pre proyecto.
- Identificar los beneficios que supondría la entrega de la solución propuesta
- Describir los aspectos organizativos y de gobierno del proyecto
- Planificación y financiación de la fase de fundamentos

Fase Fundamentos

El propósito de esta fase consta en determinar el alcance del trabajo, la forma en que se llevara a cabo, quien lo llevara a cabo, el cuándo y dónde. También se determina el ciclo de vida que tomara el proyecto, acordando como se aplicara el proceso DSDM a las necesidades específicas del mismo. Dentro de las actividades de esta fase se encuentran:

- Establecer una línea de base de alto nivel de los requerimientos para el proyecto, así como describir su prioridad y relevancia con las necesidades del negocio
- Detallar un caso de negocio para la solución
- Describir cuales necesidades serán soportadas por la solución
- Diseñar la arquitectura, e identificación de los componentes o elementos estructurales de la solución
- Definir las normas técnicas de implementación
- Establecer un calendario de actividades de desarrollo e implementación de la solución
- Descripción de cómo será el aseguramiento de calidad
- Describir, evaluar y gestionar los riesgos asociados a la solución

Fase exploración

Se investiga iterativa e incrementalmente los detallados requerimientos empresariales, para luego traducirlos en una solución viable. Dentro de sus principales actividades contempla:

- Explorar en detalle todas las necesidades del negocio y proporcionar los detallados requerimientos para evolucionar la solución
- Crear la solución funcional que satisfaga las necesidades del negocio.
- Proporcionar una visión generalizada de la solución a desplegar, su operación y mantenimiento.

Fase ingeniería

Se evoluciona de forma iterativa e incremental la solución preliminar determinada en la fase de exploración, enfocado principalmente en los requisitos no funcionales de la solución, las siguientes tareas son manejadas en esta fase:

- Refinar la evolución de la solución de la fase de exploración para cumplir con los criterios de aceptación acordados.
- Ampliar y perfeccionar los productos requeridos para operar y apoyar la solución en producción.

Fase despliegue

El objetivo principal es llevar la solución en evolución a un uso operativo. El despliegue o la liberación pueden ser el producto final o un subconjunto de la solución final. Las siguientes tareas son realizadas en esta fase:

- Desplegar la solución o un subconjunto del mismo en el entorno del negocio
- Capacitación y documentación de las operaciones de la solución a las personas responsables de apoyar y operar la solución.
- En esta fase se da formalmente el cierre del proyecto, el foco se centra en revisar de forma retrospectiva el desempeño general del proyecto, desde diferentes ángulos como técnicos, de procesos y comerciales

Fase post proyecto

Después de culminado el proyecto, las actividades se enfocan en evaluar los beneficios obtenidos del funcionamiento de la solución en relación con las necesidades del negocio, aun que pueden percibirse beneficios inmediatos, se continúa una evaluación constante por un periodo de tiempo normalmente consta de tres a seis meses.

El modelo de procesos describe una secuencia progresiva de fases desde el pre-proyecto hasta el post-proyecto, no obstante, también prescribe flechas que indican retorno dentro de un proceso, específicamente las que señalan hacia fundamentos y

despliegue evolutivo de desarrollo. El proceso muestra a su vez el Framework y las opciones disponibles, y luego cada proyecto deriva su propio ciclo de vida de este proceso. La dinámica anterior está determinada por factores como el número de incrementos en los proyectos e influencias externas como: estabilidad del entorno del negocio y la resistencia en las decisiones tomadas en la fase de fundamentos.

Fase de configuración y formalidad

DSDM permite acoplarse a las restricciones y la complejidad que se presentan en los entornos corporativos, manteniendo el foco en la agilidad que en términos de productividad y calidad en la solución del proyecto. Así que es posible calibrar y configurar los procesos para adaptarse a diferentes tipos de proyecto según tamaño y complejidad, comúnmente estas adaptaciones se logra mediante la configuración adecuada del ciclo de vida del proyecto específico y determina el nivel de formalidad en la definición, creación y aprobación de los productos.

4.2 SCRUM

Su primera aparición en 1995 (Schawer, 1995) como una mejora al ciclo de desarrollo iterativo e incremental orientado a objetos, en la actualidad SCRUM es concebido como un marco de trabajo (Framework) para el desarrollo y mantenimiento de productos complejos. (SCRUM ORG, 2015) Anteriormente variantes de SCRUM para el proceso de desarrollo de nuevos productos en industrias manufactureras enfocados a un alto rendimiento y rapidez conformados por grupos pequeños fue observado en una serie de empresas japonesas. (Takeuchi & Nokana, 1986). El termino SCRUM es derivado del juego Rugby el cual hace referencia a una formación apretada de delanteros juntos entre sí en especificas posiciones cuando se inicia un reinicio del juego o Scrummdown. (World Rugby, 2015)

El marco de trabajo está constituido de grupos de personas organizados llamados equipos SCRUM, los integrantes de los equipos SCRUM cumplen roles establecidos, los cuales participan en una serie de eventos con el propósito de desarrollar el sistema, durante la ejecución de los eventos como resultado es la entrega de artefactos, todas las actividades en SCRUM son determinada por una serie de reglas a seguir.

A continuación se detalla cada uno de los elementos de SCRUM (También ver figura 2-3):

Equipo Scrum (Scrum Team)

Los equipos Scrum están constituidos de un dueño de producto (Owner Scrum) quien es el responsable de la gestión de la lista de productos (Product Backlog) o listas de requerimientos del sistema a desarrollar. El equipo de desarrollo (Development Team) conformado por grupos de 4 a 9 integrantes responsables de

entregar el incremento o funcionalidad totalmente operacional del sistema al terminar cada bloque de tiempo (time-box) denominado sprint que no debe superar los 30 días. Los equipos Scrum son auto organizados y empoderados para tomar sus propias decisiones sobre el curso del desarrollo del incremento; multifuncionales, cuentan con una diversidad de especialidades para garantizar las habilidades necesarias para realizar el desarrollo, no hay distinción entre títulos, todos parten desde una base de igualdad y cooperativismo, las responsabilidades individuales no existen, todas recaen al equipo de desarrollo como un todo. Cada equipo Scrum se le asigna un líder (Scrum Master) quien es el encargado de asegurar que las prácticas y reglas de Scrum sean adoptadas en todo el proceso. El Scrum master funge como facilitador encargado de contribuir a remover las dificultades que se presenten durante el sprint y contribuir a que el equipo no pierda el enfoque en el éxito del incremento.

Eventos Scrum

Los eventos Scrum son bloques de tiempo con una duración máxima establecida que tiene como objetivo inspeccionar y adaptar cualquier aspecto durante el desarrollo, a continuación se detallan los eventos Scrum

Sprint

Es el núcleo de Scrum, es un bloque de tiempo de 30 días en los cuales el objetivo es realizar un incremento o funcionalidad totalmente operacional del sistema. Un sprint esta constituidos de las siguientes actividades:

Reunión de planificación del sprint (Sprint Planing Meting)

Se lleva a cabo la planificación del trabajo a realizar durante el Spring, esta reunión tiene destinada 8 horas máximo de duración. Como resultado de la reunión debe resultar la definición del incremento que será entregado al finalizar el sprint y la estimación del esfuerzo, los recursos necesarios para la entrega del incremento.

Objetivo del Sprint (Sprint Goal)

Define el objetivo del Sprint en términos de la lista de productos que se deben implementar. Se realiza durante la reunión de planificación del sprint y se presta una especial atención a los elementos seleccionados de la lista de productos los cuales serán desarrollados durante el sprint con el fin de satisfacer el objetivo del sprint.

Scrum diario (Daily Scrum)

Consiste en una reunión diaria de una duración de15 minutos de los integrantes del equipo Scrum, estas reuniones son pactadas en la misma hora y en el mismo lugar, permitiendo manejar la simplicidad y asegurar el transcurso de 24 horas entre cada sesión. Durante estas sesiones los miembros del equipo discuten que han alcanzado desde la última reunión, cuáles son sus planes desde la sesión hasta la próxima reunión, y cuales han sido los obstáculos identificados. El propósito de las reuniones es el de mantener y realizar seguimiento del progreso del equipo y propender por resolver los problemas que podrían afectar la dinámica del equipo. El Scrum master se asegura de que las reuniones se lleven a cabo dentro de las reglas establecidas, no obstante es el equipo responsable de dirigir estas reuniones.

Revisión del sprint (Sprint Review)

Se realiza terminado el Sprint y por lo general tiene una duración de 4 horas, en el cual se inspecciona el incremento producido entre todos los interesados. Una revisión general de las actividades realizadas durante el sprint, con base en la retroalimentación de todas las partes se determinan las acciones que permitan mejorar el desarrollo del siguiente Sprint, se actualiza la lista de producto, se describen los fallos y los aciertos que surgieron durante el Sprint finalizado y como se resolvieron dichos inconvenientes, nuevos requerimientos pudieran aparecer o cambios a los ya establecidos, lo cual ocasionara una actualización de la lista de producto , se define la lista de productos posibles a ejecutar en el siguiente sprint.

Retrospectiva del sprint (Sprint Retrospective)

Esta reunión es llevada a cabo antes de la siguiente reunión de planificación es una revisión general de todo lo realizado en el sprint inmediatamente anterior, que involucra personas, relaciones, herramientas y procesos, identificando los elementos que tuvieron éxito y las posibles mejoras para aquellos elementos donde se presentaron dificultades, el resultado es un plan de mejoras para ser implementado por todo el equipo Scrum. Aunque las mejoras se pueden realizar en cualquier momento, decidir realizar las en el siguiente sprint es la adaptabilidad a los cambios buscado en los equipos Scrum.

Artefactos

Lista de producto (Product Backlog)

Son los requisitos del sistema. Engloba todas las características, funcionalidades, mejoras y correcciones. La lista de producto se considera dinámica e incompleta, a medida que el producto evolucione la lista de producto también lo hará adaptándose a las circunstancias que se presenten. Cada uno de los elementos de la lista contiene los siguientes atributos: descripción, ordenación, estimación y el valor. Todas las estimaciones son responsabilidad del equipo. El refinamiento a la lista de producto

es el proceso continuo de añadir detalle, estimación y ordenación a cada uno de los elementos, en este proceso participan el dueño del producto y el equipo, los cuales son los encargados de contribuir en los detalles de la lista de producto. El resultado de este proceso es la ordenación de los elementos de la lista de productos, en consecuencia los elementos de la lista con mayor ordenación son más claros y con mayor detalle, lo cual permite una estimación con mayor precisión, cuanto menor sea el orden, menor es el detalle. Los elementos de los que se ocupa el equipo en un sprint son aquellos que tienen una granularidad mayor, lo que traduce que cualquier elemento puede ser desarrollado o terminado dentro del tiempo del sprint. Estos elementos son acuñados con los nombres de preparados o accionables, y son los candidatos a ser seleccionados en la próxima reunión de planificación del sprint.

Lista de pendientes (Sprint Backlog)

Concerniente a la planificación del conjunto de elementos de la lista de productos que formara parte del próximo sprint, por lo general acompañado de un plan de entrega del incremento y el objetivo a cumplir una vez finalizado el Sprint. Durante el desarrollo del sprint el equipo modifica la lista de pendientes, a la vez que lista va emergiendo durante el transcurso del mismo. Cuando sea necesario un trabajo nuevo se agrega a lista de pendientes del Sprint y cuando un trabajo sea considerado innecesario será desechado de la lista, en la medida que el trabajo se ejecuta seguidamente se procede con la actualización en la estimación del trabajo restante.

Incremento

Es el resultado de la suma de todos los requerimientos o lista de productos terminados en el Sprint con la adicción del valor de los incrementos de todos los anteriores Sprint. A este punto el incremento debe estar totalmente operativo para su liberación a los usuarios.

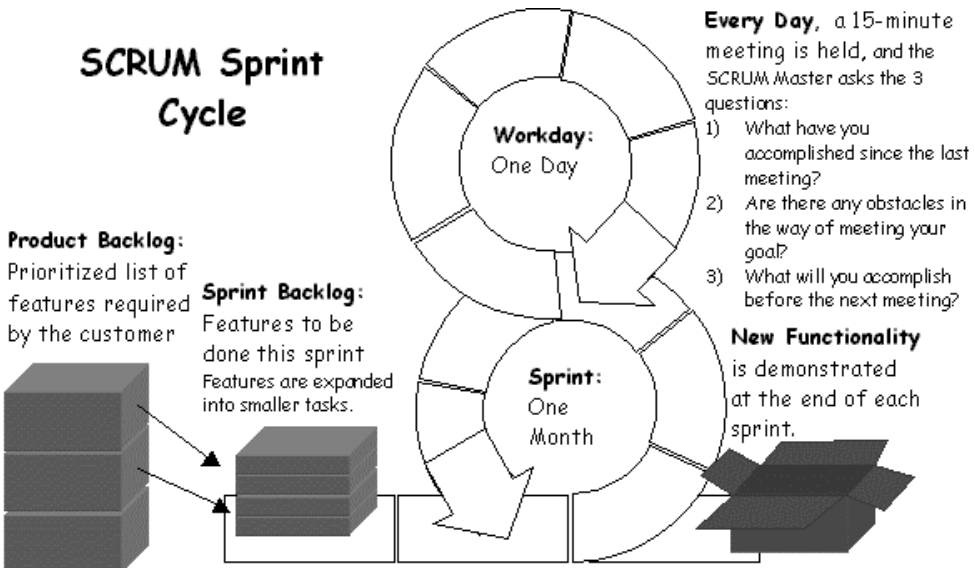


Figura 2-2. Proceso SCRUM. Fuente *(CodeProject, 2015)*

4.3 XP

La Extrem Programing (XP) desarrollada por Beck en 1996. (Wells, 2015)Propone una metodología ligera diseñada para grupos pequeños o medianos de desarrollo de software orientado a trabajar en un entorno donde los requerimientos del proyecto puedan ser vagos o cambiar rápidamente. (Beck, Extreme Programming Explained, 1999) Si bien, esta fue la idea principal subyacente de XP en sus inicios, el autor y sus adeptos consideran que XP trata de una disciplina de desarrollo de software que cubre todos los niveles del proceso de desarrollo. (Beck & Andres, Extreme Programming Explained: Embrace Change, Second Edition, 2004)

Una de las características sobresalientes de XP es su marcado enfoque en la satisfacción del cliente con la entrega constante de producto de software, en XP el trabajo en equipo es esencial, desarrolladores, gerentes y clientes conforman un equipo como iguales en un entorno simple y de colaboración que les permita ser altamente eficaces, auto organizado entorno al problema a resolver, buscando la mejor vía para realizarlo eficientemente.

El desarrollo de software en XP se fundamenta en una serie de prácticas, roles y valores que se muestran a continuación:

Valores

- Comunicación: Es un factor indispensable en el equipo de desarrollo de software, es el medio por el cual se establece la cooperación en el equipo, es la forma primordial de atender los problemas que surgen durante el proceso, la transmisión de conocimiento entre todos los integrantes resulta ser el primer paso a la resolución de los percances.

- Simplicidad: Concentrar los esfuerzos en los estrictamente necesario y simple en el trabajo a realizar. Esto exige constante evaluación del plan de trabajo y los objetivos planteados para el proyecto, si se identifica que el trabajo se torna complejo, se debe encontrar la manera de direccionar otra vez hacia la simplicidad.

- Retroalimentación: La mejor manera de afrontar los cambios que surgen durante el desarrollo es mediante la retroalimentación. Los equipos se esfuerzan en retroalimentar sus ideas lo más rápido posible, en relación con ciclos de desarrollo de software tradicionales donde las ideas eran compartidas en semanas o meses, en XP se trata de acortar el ciclo a minutos u horas. Los valores en XP están entrelazados unos ayudan a completar a

otros, en este caso la retroalimentación hace parte fundamental de la comunicación y contribuye a la simplicidad.

- Coraje: Se trata del compromiso de cada uno de los integrantes, en la forma como enfrentan los problemas que suceden en el desarrollo, del valor de comunicar las situaciones agradables y molestas con el objetivo de hacerlas de común conocimiento, contribuyendo a mejorar la comunicación, de tener la valentía de descartar soluciones ineficientes o complejas que van en contravía a la simplicidad, además del hecho de buscar la respuesta adecuada fomenta la retroalimentación.

- Respeto: Cada integrante de un equipo de desarrollo en XP es considerado importante y nadie es superior a otro en ningún aspecto. Las contribuciones de cada uno se consideran importantes y respetables.

Practicas

- Todos en sitio: El equipo completo de desarrollo es co-ubicado en el mismo sitio, procurando que el lugar de ubicación sea propicio para la comunicación y la productividad.

- Equipo completo: El equipo de desarrollo debe estar constituido por personas con las habilidades, capacidades y perspectiva que se requiere para el éxito del proyecto.

- Espacio de trabajo informativo: Mediante un pizarrón ofrecer a cualquier interesado del proyecto una visión rápida del estado del proyecto, el objetivo es suministrar información de los logros alcanzados y los problemas reales o potenciales con solo un vistazo.

- Trabajo energizado: El exceso de horas de trabajo solo genera cansancio e improductividad en los desarrolladores, el trabajo productivo y creativo solo se alcanza con las horas que cada integrante del equipo pueda sostener. Como máximo 40 horas a la semana trabajando en el sistema son aceptadas, si sobretiempo es necesario, es percibido como un problema a resolver por el equipo.

- Programación en parejas: La producción de código es generada por dos desarrolladores ubicados en el mismo sitio y en un computador.

- Historias: Los requerimientos del sistema son expresados mediante una historia que exprese de forma concisa y concreta la funcionalidad esperada denominada tarjeta de historia (Story Card), además sirve como base para el cálculo del esfuerzo necesario para su implementación.

- Ciclos semanales: Realizar los planes de trabajo una semana a la vez. En cada uno de los planes semanales se incluyen las tareas de revisar el estado actual del progreso, estimar el esfuerzo necesario de la semana y se priorizan las funcionalidades a desarrollar.

- Ciclos trimestrales: Revisión y planificación trimestral de todos los aspectos involucrados en el proyecto: equipo, progreso y seguimiento a los objetivos planteados.

- Construir en diez minutos: Generar el sistema en su totalidad y ejecutar las pruebas del mismo en diez minutos todos los días.

- Integración continúa: Paso inverso a la descomposición del problema, es la integración de sus partes, es necesario conocer los efectos de la integración, por tal motivo integrar y probar los cambios del sistema no puede tardar más de un par de horas.

- Pruebas unitarias: El desarrollo de software es dirigido por pruebas. Antes de la realización o cambio de cualquier código, el desarrollador crea la prueba unitaria y se encarga de la ejecución.

- Diseño incremental: El diseño del sistema es una tarea constante y persistente, los desarrolladores invierten en el diseño del sistema ajustado a los requerimientos que demande el día.

Roles

- Tester: Ayudan a los clientes a escribir las pruebas funcionales del sistema y son los responsables de ejecutarlas regularmente, el resultado de las pruebas es compartido al equipo.

- Diseñador de interacciones: Encargados de evaluar el uso del sistema, con el objetivo de corregir o identificar nuevos requerimientos, a su vez se encargan de traducir estas necesidades en historias.

- Arquitectos: Los arquitectos en un equipo equipos son los encargados de dividir el sistema, en pequeños sistemas, la idea es que probar la arquitectura

de un sistema pequeño resulta conveniente que realizar una a gran escala. Una vez la arquitectura ha sido probada, esta va siendo integrada al total de la solución. A si mismo se encarga de buscar y ejecutar refactorizaciones a gran escala, escriben las pruebas funcionales de la arquitectura del sistema e implementar las historias de la arquitectura.

- Administradores de proyecto: Son los facilitadores de la comunicación interna y externa de la situación del proyecto a todos los involucrados. Dentro de sus funciones se incluyen: recopilar, actualizar y transmitir la información del progreso del proyecto en la menor brevedad.

- Administradores de producto: Encargados de escribir y elegir los temas de las historias en los diferentes ciclos, responder a las preguntas sobre insuficiencias en la implementación de las historias en diferentes áreas.

- Executivos: Los ejecutivos son los responsables de proporcionar el coraje, confidencialidad y responsabilidad a los integrantes del equipo con el propósito de mantener articulado el rumbo hacia los objetivos.

- Escritores técnicos: Su trabajo consiste en escribir los documentos técnicos para el equipo XP, los cuales cumplen el rol de ser los medios de retroalimentación de las características del proyecto y de establecer relaciones estrechas con los usuarios.

- Usuarios: Los usuarios ayudan al equipo XP a escribir y elegir las historias, contribuyen en la toma de decisiones sobre el curso del proyecto durante el desarrollo.

- Desarrolladores: Los desarrolladores estiman el esfuerzo de las historias y tareas, escriben las pruebas unitarias, el código de los requerimientos, mantienen el código limpio sin errores y mejoran regularmente el diseño del sistema.

La siguiente sección contiene la descripción del proceso general de un proyecto XP como se muestra en la figura 2-4, el cual se enmarca en un conjunto de reglas y prácticas que son desarrolladas en las siguientes actividades estructurales:

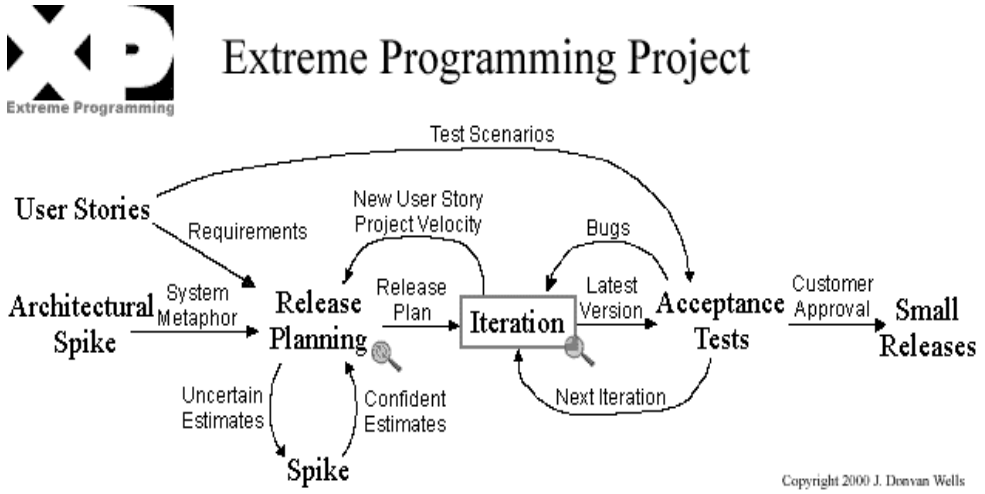


Figura 2-3. Proceso de desarrollo XP.

Planeación

La actividad de planeación comienza con la escritura de las historias de usuarios un símil de los casos de uso, pero que sirve a un diferente propósito en XP, son usadas, primero para la estimación del tiempo necesario para el desarrollo de cada una y segundo como el documento de las características y requerimientos del sistema. Las historias de usuario son escritas por los clientes que son la traducción de las necesidades que el sistema deberá cumplir; paso seguido se lleva a cabo la reunión de planeación de la liberación donde se establece el alcance general del proyecto con la creación del plan de liberación que es usado para estimar los planes de iteración de cada una de las iteraciones.

En esta actividad el equipo de desarrolladores estima el costo en tiempo cada una de las historias de usuario en términos de semanas, por lo general cada historia o iteración debe tener una longitud de entre 1 a 3 semanas, si la historia requiere más de este tiempo, se solicita una descomposición en historias más pequeñas. Los clientes determinan cual historia tiene mayor importancia o prioridad para ser implementada, luego las historias son copiadas en tarjetas o en un tablero para ser colocadas alrededor de una mesa creando una serie de historias para ser implementadas fijando así la primera liberación luego la siguiente y así sucesivamente.

En cada inicio de iteración se realiza una reunión de planeación de la iteración donde se realiza la programación de las tareas. Una iteración está fijada de una a tres semanas, se escoge un grupo de historias de usuario del plan de liberación para la primera entrega o incremento. Terminada la primera liberación el equipo de desarrollo determina la velocidad del proyecto, la cual proporciona información a los clientes y desarrolladores mejorando la estimación del esfuerzo necesario para las historias de usuario, esto puede conllevar a la actualización o inserción de nuevas historias al plan de liberación.

Administración

Los esfuerzos se concentran en mejorar la comunicación entre el equipo uno de los valores primordiales de XP; Esto se logra eliminado las barreras que impiden la iteración y la comunicación entre los integrantes, por lo que se recomienda que el equipo este ubicado en un lugar amplio, donde los computadores estén ubicados en la zona central, ninguno debe trabajar como una isla separada de los demás, esto contribuye a disponer la actitud de trabajar juntos mejorando la práctica de la programación en parejas, disponer de una mesa de conferencias para las discusiones grupales sobre el proyecto, contribuye a transmitir la información del estado del mismo a todos los integrantes, colocar tableros blancos para bocetos de diseños, notas importantes o historias de usuarios añaden un vital canal de información para el equipo.

Reunión diaria de pie: es una actividad representativa de XP, se realiza cada mañana y su rasgo característico es que es llevada por todo el equipo en posición levantada en círculo estimulando la concentración y evitando las discusiones largas. Su propósito es comunicar los problemas, logros obtenidos y promover el trabajo en equipo. La discusión de las reuniones a pie son dirigidas por tres preguntas: que se ha logrado desde la última reunión, que se espera lograr hasta la próxima y que problemas están ocasionado retrasos.

Mover a la gente es una práctica que busca la rotación de todos los integrantes con el objetivo que todos conozcan lo suficiente de la totalidad del sistema que se está desarrollando, evitando que las personas se aíslen o concentren todo el conocimiento de una parte del sistema, también como una medida de balanceo de las cargas, procurando no sobrecargar a los desarrolladores y distribuir a los desarrolladores en forma adecuada basado en los puntos críticos que se presenten.

Diseño

El diseño en XP es regido por el principio de mantener simple todo como sea posible, si algún aspecto es complejo este deberá ser reemplazado por algo simple, se diseña enfocado en la codificación solo las funcionalidades necesarias son implementadas y nunca se añaden funcionalidades que este fuera del diseño o de la programación.

Las metáforas del sistema como se denominan en XP es un diseño simple y sus cualidades, es una expresión del sistema desde el punto de vista de los involucrados en el proyecto y que cuenta cómo funciona el sistema. La idea del uso de las metáforas en XP es lograr explicar el diseño del sistema sin incurrir en costosa y voluminosa documentación a personas nuevas o externas al proyecto.es también el punto de partida en la definición de los nombres de las clases y métodos.

XP recomienda el uso de las tarjetas CRC(Clase, responsabilidad, colaborador), esta técnica permite al equipo desarrollador diseñar desde una perspectiva orientada a

objetos, además de permitir un mayor número de personas influyan en el diseño, las tarjetas CRC son el único producto de la actividad de diseño.

Si una parte del diseño es compleja se recomienda construir un prototipo o Spike solución, con el objetivo de disminuir el riesgo o explorar alternativas, que pudieran ser costosas en su futura implementación.

Re factorizar es un mecanismo clave del diseño simple, remover código innecesario, eliminar código duplicado, mantener legible y limpio el código contribuye a la calidad, en adicción permitirá que el diseño sea comprendido, modificado y extendido por cualquiera.

Codificación

Se define un estándar de codificación con el propósito de mantener el código legible y consistente para todos los miembros del equipo encargados de leerlo y re factorizarlo. Las pruebas unitarias son creadas antes que el código, la creación de las pruebas unitarias aporta valiosa información y otorga una rápida retroalimentación al desarrollador de que es lo que requiere para poder superar la prueba.

La programación en parejas es una de las características llamativas de XP, esta práctica es llevada por una pareja de programadores trabajando junto en un mismo computador, el resultado es aumento de la calidad del código producido.

En la medida de que las parejas de desarrolladores tengan código terminado y luego de haber superado con éxito las pruebas unitarias deben integrarlo al repositorio en pocas horas como sea posible, típicamente se destina un computador como repositorio de las liberaciones, además de contener la última versión del sistema actual. Permite a todo el equipo tener control de quienes están liberando y cuando. La integración continua permite identificar problemas de compatibilidad que pueden ser abordados rápidamente y no al final cuando su solución resulta más costosa y compleja.

Pruebas

Si bien las pruebas unitarias son creadas por los desarrolladores antes de la codificación, es necesario crear el mecanismo para crear un conjunto de pruebas unitarias automatizadas que posibiliten ejecutarlas en repetidas ocasiones.

Las pruebas unitarias se desarrollan en el repositorio junto con todo el código que aún no sea liberado, código que no supere la prueba unitaria no puede ser puesto en producción. Con la construcción de un conjunto de pruebas unitarias universal para el proyecto estimula las pruebas de validación y de regresión en la medida que se modifique el código. La adición de una nueva funcionalidad también conlleva un cambio en las pruebas unitarias para evidenciar la funcionalidad.

Las pruebas de aceptación en XP se derivan de las historias de usuario, durante cada iteración las historias seleccionadas son trasladadas en pruebas de aceptación. El cliente es el responsable de especificar los escenarios y las características para que una historia sea considerada correcta o satisfactoria.

La ejecución de una prueba de aceptación son pruebas de caja negra del sistema es la representación de un resultado esperado por el sistema; son los clientes los encargados de revisar las pruebas de aceptación, determinan cuales pruebas se consideran fallidas. Una historia de usuario que no supere la prueba de aceptación se considera no terminada.

4.4 ASD

Desarrollo adaptativo de software (ASD) introducido por James Highsmith a principios de 1990, es un Framework de desarrollo orientado a proyectos de software complejos y altamente cambiantes. Se fundamenta en la teoría de sistemas adaptativos complejos y experiencias de procesos de la metodología de desarrollo rápido de aplicaciones (RAD). (Highsmith, 1998)

El modelo de ciclo de vida propuesto por ASD llamado ciclo de vida de desarrollo adaptativo es un proceso iterativo compuesto de tres fases superpuestas de especular, colaborar, aprender que se muestran en la figura 2-5, a continuación se describe brevemente cada una de las actividades:

- Especular: la unificación de los esfuerzos y el enfoque necesario, en una específica misión, para la entrega de un conjunto de versiones de productos de la aplicación real, en cronogramas definidos en cada fase, las iteraciones pueden ser planeadas, no obstante bajo este enfoque están son impulsadas por el riesgo que supone el entorno complejo y lleno de incertidumbre, que proporciona una vía de exploración y de aprendizaje que una ruta predefinida.
- Colaboración: el diseño y la implementación de los componentes del sistema son resultado derivados de la activa participación y colaboración.
- Aprender: después de terminado cada ciclo, grupos de revisiones son llevados a cabo con el objetivo de que las personas involucradas aprendan de la experiencia y la autoevaluación, para ser implementadas en lecciones aprendidas del proceso.

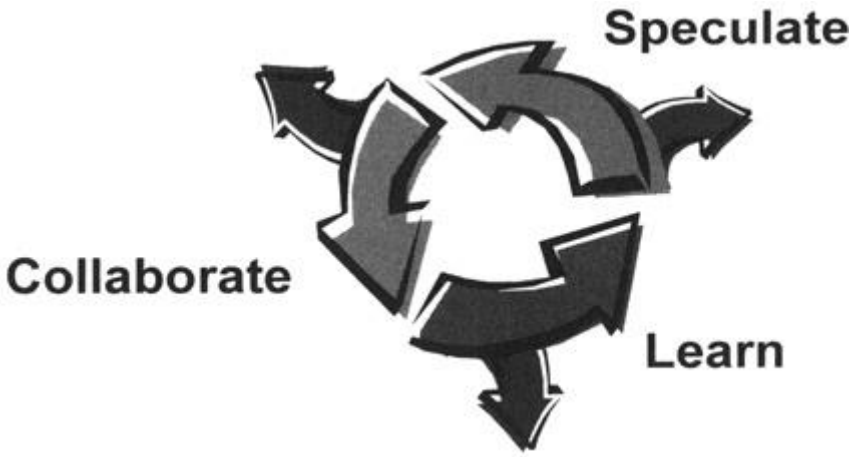


Figura 2-4. Ciclo de vida de desarrollo adaptativo.

En la figura 2-6 se muestra las cinco fases y el orden relativo correspondiente al ciclo de vida de especular-colaborar-aprender, sobresale las tres fases intermedias que constituyen el motor de desarrollo iterativo de ASD, a continuación se detallan brevemente cada una de las fases:

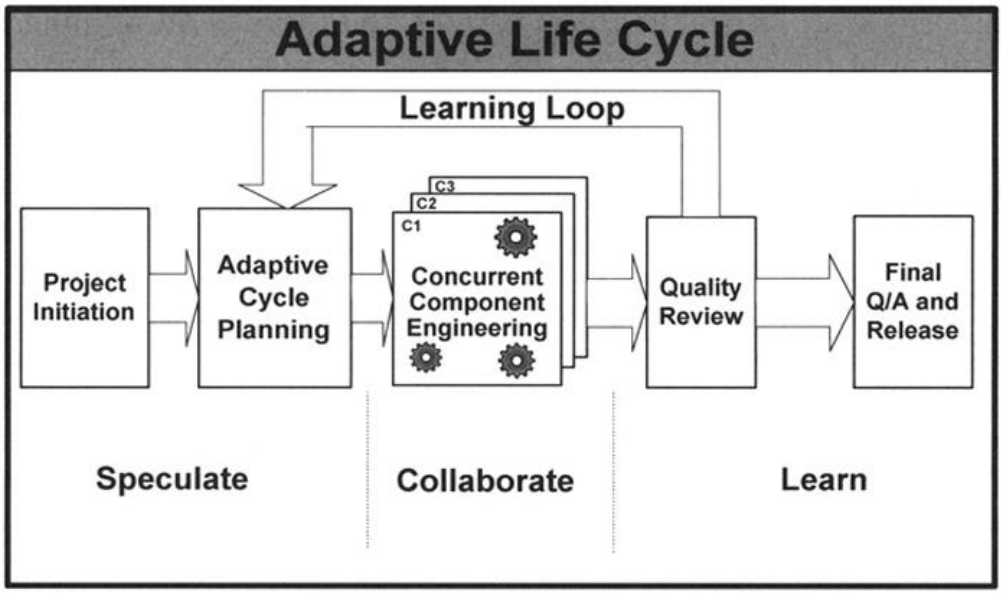


Figura 2-5. Ciclo de vida adaptativo detallado.

La fase especular se compone de la actividad de inicio del proyecto donde se enfoca en desarrollar la misión del proyecto, requisitos del sistema y planificarlos los recursos para la construcción o fabricación de los componentes.

La fase colaborar se divide en:

- Planeación adaptativa del ciclo donde se configura el marco de tiempo general del proyecto y sus ciclos. Se determina los componentes que serán desarrollados, se asigna cada componente a un ciclo y se programan las iteraciones. El plan puede ser revisado al inicio de cada iteración.
- Ingeniería concurrente de componente con el foco en el desarrollo de los componentes asignados para cada uno de los ciclos.
- Revisión de calidad se realizan grupos de revisión del componente desarrollados

La fase aprender se enfoca en la revisión de la calidad del producto desde el punto de vista del cliente, además del desempeño del equipo del proyecto, se compone de:

- Final Q/A liberación con el foco en validar del sistema producido y desplegarlo en un entorno de operación.

4.5 UP

Proceso unificado ágil introducido por Scott Ambler en 2006 es una versión ágil del proceso racional unificado para el desarrollo de sistemas de IBM 's (RUP). (Ambler S. , 2015) El proceso de UP consiste en las mismas fases de RUP, no obstante las tareas realizadas en cada fase son mucho más simples y las disciplinas se realizan de forma iterativa sin realizar todos los flujos de trabajos prescritos por RUP. En la figura 2-7 se muestra el proceso ágil unificado.

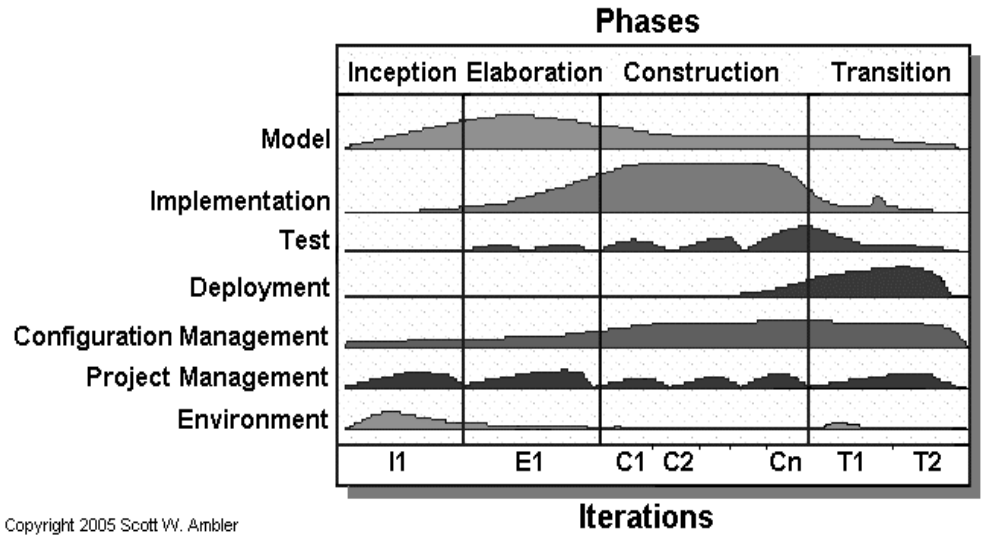


Figura 2-6. Proceso unificado ágil.

Las fases del proceso UP son:

1. Comienzo: con el objetivo de establecer el alcance del sistema, proponer una arquitectura candidata y lograr la aceptación y financiación de los involucrados.

2. Elaboración: con el objetivo de proporcionar la arquitectura que soportara el sistema a desarrollar.

3. Construcción: con el objetivo de producir software funcional de forma constante e incremental que satisfaga los requerimientos prioritarios definidos por los involucrados.

4. Transición: con el objetivo de validar y desplegar el sistema producido en un entorno de producción.

Las cuatros fases y las tareas realizadas en cada una de ellas se describen brevemente a continuación:

Comienzo

1. definición del alcance del sistema: se define desde una perspectiva de alto nivel que es lo que hará el sistema y más aún importante que no hará. Dentro de la definición se establecen las fronteras donde el sistema funcionara, generalmente lo anterior se traslada en forma de una lista de casos de uso general del sistema.

2. estimación de presupuesto y cronograma: se realiza una estimación general del presupuesto y cronograma para el proyecto.

3. definición de los riesgos: con el objetivo de identificar, documentar y clasificar los riesgos del proyecto según el nivel de severidad y materialización durante el desarrollo del proyecto.

4. determinación de la viabilidad del proyecto: se analiza desde diferentes perspectivas (económicas, técnicas, estratégicas) si el proyecto a construir tiene sentido para las partes involucrados, si como resultado es su no viabilidad el proyecto debe ser cancelado.

5. preparación del entorno para el proyecto: se enfoca en proporcionar todos los elementos necesarios para el desarrollo del proyecto, usualmente relacionados con: la conformación del equipo, el espacio de trabajo y las herramientas necesarias para su trabajo como el software y hardware.

Elaboración:

El equipo prepara un prototipo de la arquitectura del sistema con el objetivo de probar si cumple con los requisitos del sistema, se prioriza en el diseño e implementación de los casos de uso críticos, durante la ejecución de la prueba riesgos arquitectónicos son identificados y priorizados. Aquellos riesgos

arquitectónicos críticos son abordados con el propósito de mitigarlos durante la fase de elaboración, permitiendo adaptar la arquitectura a las necesidades del sistema.

Construcción

Se desarrolla el sistema hasta el punto en que este pueda ser sometido a pruebas de pre producción. A este punto los principales requisitos del sistema han sido identificados y la arquitectura que los soportara ha sido delineada, lo que permite cambiar el enfoque en actividades de codificación y pruebas, los casos de uso implementados pueden ser liberados en ambiente de producción internos o externos.

Transición

Se enfoca en entregar el sistema en un entorno de producción a los usuarios. Durante esta fase se da lugar a numerosas tareas como: pruebas beta del sistema, la integración continúa del sistema con sistemas existente, capacitación de los usuarios, la generación de la documentación.

4.6 FDD

Desarrollo basado en funcionalidades (FDD) fue introducido en 1999 por Peter Coad and Jeff De Luca en Java Modeling in Color with UML, (COAD, LEFEBVRE, & AND DE LUCA, 1999),si bien el propósito de los autores era describir la proposición de una nueva técnica de modelamiento llamada modelamiento en color, el último capítulo de la obra describe un particular proceso para la construcción de sistemas complejos, conformado por equipos grandes con distintas habilidades y experticia denominado Feature Driven Development (FDD).

FDD es un proceso de desarrollo de software adaptativo ágil enfocado en la entrega regular de tangibles resultados de trabajo, altamente iterativo, énfasis en la calidad en cada uno de sus pasos, proporciona información precisa del progreso y estado del proyecto a todos los involucrados (clientes, desarrolladores, usuarios finales). (Palmer & Felsing, 2002)

Un proyecto FDD inicia con la creación de un modelo de objetos del dominio por parte de los expertos de dicho dominio, con base en la información obtenida de actividades de requerimientos y de modelamiento, los desarrolladores crean una lista de funcionalidades, seguido de un plan estimativo de producción y de asignación de responsabilidades. Luego pequeños grupos de funcionalidades son implementados en iteraciones de diseño y codificación que no deben superar dos semanas.

El proceso de FDD consiste de cinco subprocesos que durante su desarrollo generan algunos entregables, los iniciales tres subprocesos están relacionados con el análisis de los requerimientos y la planeación del desarrollo secuencial al inicio del proceso, los dos subprocesos restantes se encargan de las actividades de diseño y la implementación de las funcionalidades.

El proceso general FDD y sus subprocesos se muestran en la figura 2-8, y se explican brevemente a continuación:

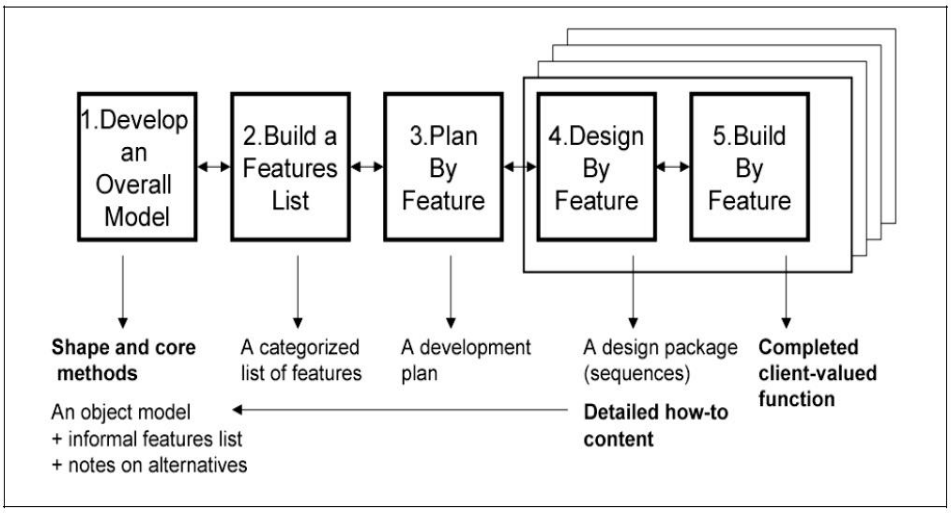


Figura 2-7. Proceso FDD y entregables.

Desarrollar un modelo general: el propósito es crear un modelo estructurado del problema y su contexto denominado modelo de objetos. El modelo consiste de varios modelos visuales como: diagrama de clases del dominio que contienen atributos, operaciones y restricciones, diagramas de secuencia (si se requiere) de los patrones de comportamiento o interacciones de los objetos del dominio.

Construir una lista de funcionalidades: con el objetivo de producir una lista de funcionalidades del sistema consiste de: una lista de las áreas donde funcionara el sistema, y las actividades realizadas en cada área, las funcionalidades son representadas como pasos en cada una de las actividades, el resultado de este subproceso es una jerárquica lista de funcionalidades categorizadas.

Plan según funciones: con el objetivo de construir el plan de desarrollo que consiste de: cronograma de implementación de las funcionalidades de, la asignación de los conjuntos de funcionalidades y de las clases del modelo de objetos a los desarrolladores. El proceso se considera iterativo donde el jefe de programadores puede considerar la secuencia de desarrollo adecuada luego reprogramar y reasigna las clases de las funcionalidades a implementar.

Los subprocesos 4 y 5 son iterativos que constituyen el motor de desarrollo de FDD, el jefe de programadores selecciona un grupo de funcionalidades objeto de ser implementadas en las próximas dos semanas, los subprocesos, en orden de ejecución realizados durante cada iteración son los siguientes:

Diseñar por funcionalidad: con el objetivo de producir un paquete de diseño totalmente inspeccionado, el jefe de programadores forma grupos de funcionalidades mediante la identificación de propietarios de clases o desarrolladores que podrían estar involucrados en el desarrollo de la funcionalidad seleccionada. El paquete de diseño puede consistir de: diagramas de secuencia detallados, acompañado del modelo de objetos el cual pudo haber sido objeto de adiciones o modificaciones de clases, propiedades y métodos.

Construir por funcionalidad: con el objetivo de codificar, probar e inspeccionar cada uno de los elementos de la funcionalidad, después de superar con éxito las inspecciones, los elementos implementados son integrados a la construcción principal.

4.7 CRYSTAL

Basado en la suposición de que ninguna metodología es la mejor y que todos los proyectos son diferentes por consiguiente requieren diferentes metodologías, Alistair Cockburn propone una familia de metodologías denominadas Crystal (Cockburn, Agile Software Development, 2000), el nombre de la metodología es una metáfora de los cristales geológicos los cuales tienen un diferente color y dureza con el tamaño y criticidad de los proyectos de desarrollo de software.

La familia de metodologías crystal hace énfasis en la entrega constante de producto de software y en la comunicación constante entre los integrantes involucrados; los proyectos crystal son categorizados según el nivel de criticidad y tamaño del sistema a desarrollar. La criticidad de un sistema está definida por el daño que podría presentarse por aquellos defectos no identificados, los cuales están clasificados en cuatro niveles de criticidad: Confort(C), Dinero discrecional (D), Dinero esencial (E), o Vida (L).

Las metodologías cristal son denominadas por color y este es definido según el nivel de complejidad y el número de personas que serán coordinadas del proyecto; color claro para proyectos de menor criticidad con equipos de 8 integrantes o menos co-ubicados, amarillo para equipos de 10 a 20 integrantes, naranja para equipos de 20 a 50 integrantes, rojo para equipos de 50 a 100 personas con un alto nivel de criticidad; la literatura respecto a las familias cristal menciona otra serie de familias de metodologías con mayor peso: granate, azul y violeta en orden ascendente según nivel de criticidad del proyecto. A continuación en la figura 2-9 se muestra la cuadricula de algunas de las familias de metodologías cristal:

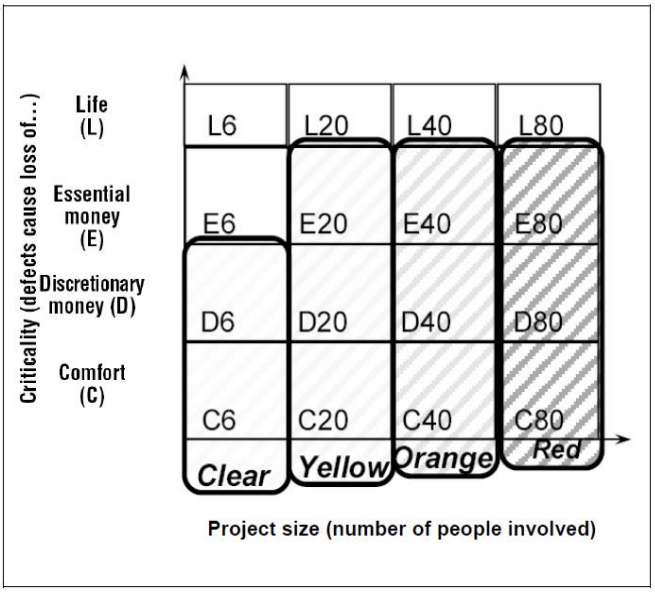


Figura 2-8. Tipos de proyectos en Crystal y la correspondiente metodología Crystal.

Proyectos grandes, por lo general involucran mayor número de personas, requieren mayor coordinación y mejor comunicación por consiguiente requieren metodologías más sólidas, proyectos con niveles altos de criticidad requieren de metodologías con enfoques más rigurosos, en aspectos como validación y verificación. Apoyándose en la imagen anterior, moviéndose hacia la derecha de la cuadricula corresponde a proyectos de gran tamaño donde es necesario mayor coordinación de personas, metodologías complejas son requeridas, mientras tanto moviéndose hacia arriba de la cuadricula corresponde a proyectos con alto nivel de criticidad donde metodologías rigurosas y con mucha ceremonia son requeridas.

Las metodologías cristal son iterativas-incrementales donde cada ciclo de entrega se produce un incremento que no supera los cuatro meses, es necesario que el equipo de desarrollo este co-ubicado no es compatible con equipos distribuidos, y hacen énfasis en un efectivo flujo de información y comunicación entre los integrantes involucrados para su éxito en la adopción.

Cada una de las metodologías cristal implica un marco de procesos de desarrollo constituidos de una serie de elementos de procesos como estándares, técnicas y estrategias muy generales, además de una serie de productos de trabajo a ser producidos. Las metodologías cristal proporcionan mecanismos para la adaptación de la metodología según las necesidades del proyecto, el equipo de desarrollo inicia el proyecto con una metodología base, a medida del progreso del proyecto la metodología es mejorada y refinada, es frecuente que los desarrolladores tomen prestada prácticas de otras metodologías, esta técnica es la que permite la adaptación de la metodología de desarrollo a los variables niveles de criticidad y ser resiliente ante las complicaciones derivadas. El seguimiento y control se realiza

mediante el uso de sesiones de revisión llamadas talleres de reflexión cuyo propósito es revisar la metodología de desarrollo, los planes del proyecto y la calidad de los incrementos entregados, al final se toman medidas correctivas si son necesarias.

La literatura disponible de las metodologías cristal alude a un grupo de metodologías, no obstante solo algunas de ellas han sido definidas y utilizadas en proyectos reales, las restantes están pendientes por desarrollar. Las metodologías cristal documentadas encontradas son: Crystal Claro, Crystal Naranja y Crystal Web Naranja.

El ciclo de vida de un proyecto Crystal, en este caso Crystal Claro consiste de las siguientes fases (Cockburn, Crystal Clear A Human-Powered Methodology for Small Teams, 2004):

Chartering: su duración no supera unas pocas semanas e involucra las actividades de desarrollo del equipo, análisis preliminar de factibilidad, desarrollo del plan del proyecto y formar y refinar la metodología.

Entrega cíclica: es el núcleo del motor de procesos que consiste de dos o más ciclos de entrega que pueden tomar de una semana a tres meses, durante el cual el equipo de desarrollo actualiza y refina el plan de liberación, se implementan un subconjunto de requerimientos mediante la iteraciones de pruebas e integración. Cada uno de los entregables es integrado a la solución final, constantemente el equipo realiza revisiones de la metodología y el plan de proyecto, las iteraciones en un ciclo de entrega están compuestos de ciclos de integración.

Envolver: se realizan las actividades se clausura o post implementación, el producto de software es desplegado en el entorno de operación de los usuarios, seguido se llevan a cabo actividades de revisión y lecciones aprendidas después del despliegue.

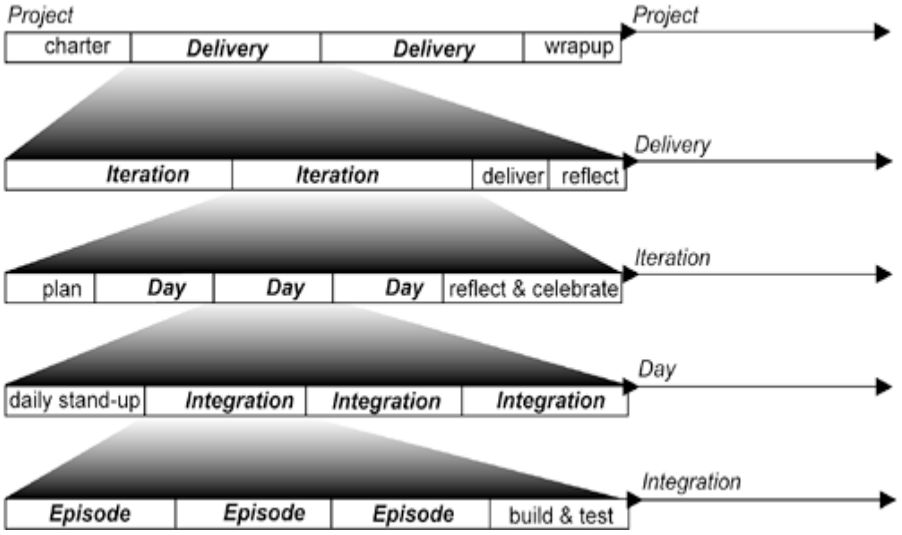


Figura 2-9. Ciclos expandidos y sus específicas actividades Crystal.

5. PATRONES DE PROCESOS

Los patrones de procesos son el resultado de la aplicación de abstracción de recurrentes procesos y sus componentes, permitiendo obtener los medios para desarrollar metodologías mediante la instanciación de un patrón adecuado. El termino patrones de procesos deriva de dos términos muy extendidos dentro del campo de la ingeniería que son procesos y patrones, el primero termino básicamente hace referencia a una serie de establecidas acciones en las cuales una o varias entradas son utilizadas para producir una o más salidas, mientras que el termino patrones hace alusión a una solución general a un problema o temática común, por media de la cual una solución especifica es derivada.

En 1994 James Coplien introduce una definición del término patrones de proceso como "los patrones de actividad dentro de una organización (y por lo tanto dentro de su proyecto) se llama un proceso". (Coplien, 1994)

Scott Ambler en 1998 propone la siguiente definición de patrón de procesos como "un patrón que describe un probado, enfoque y/o serie de acciones exitosos para el desarrollo de software" (Ambler S. W., Process Patterns: Building Large-Scale Systems Using, 1998),el autor en otra de sus publicaciones introduce el termino patrón de procesos orientado a objetos como "Un patrón de proceso describe un conjunto generales de técnicas, acciones y/o tareas para desarrollar software orientado a objetos" (Ambler S. W., 2015).

Según Ambler los patrones de proceso son de tres tipos, los cuales están categorizados según el nivel de abstracción, a continuación se describe brevemente los tipos de proceso en orden ascendente de nivel de abstracción:

Patrón de proceso de tareas: describe los pasos detallados para realizar una tarea específica, en la figura 2-11 se presenta el patrón de proceso de revisión técnica:

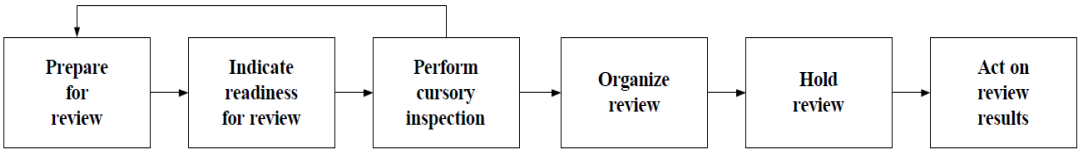


Figura 2-10. Patrón de proceso para revisión técnica.

Patrón de proceso etapa: describe los pasos que se realizan iterativamente en una fase del proyecto; y está compuesto generalmente de varios patrones de procesos de tareas, en la figura 2-12 se muestra el patrón de proceso de la etapa de programación:

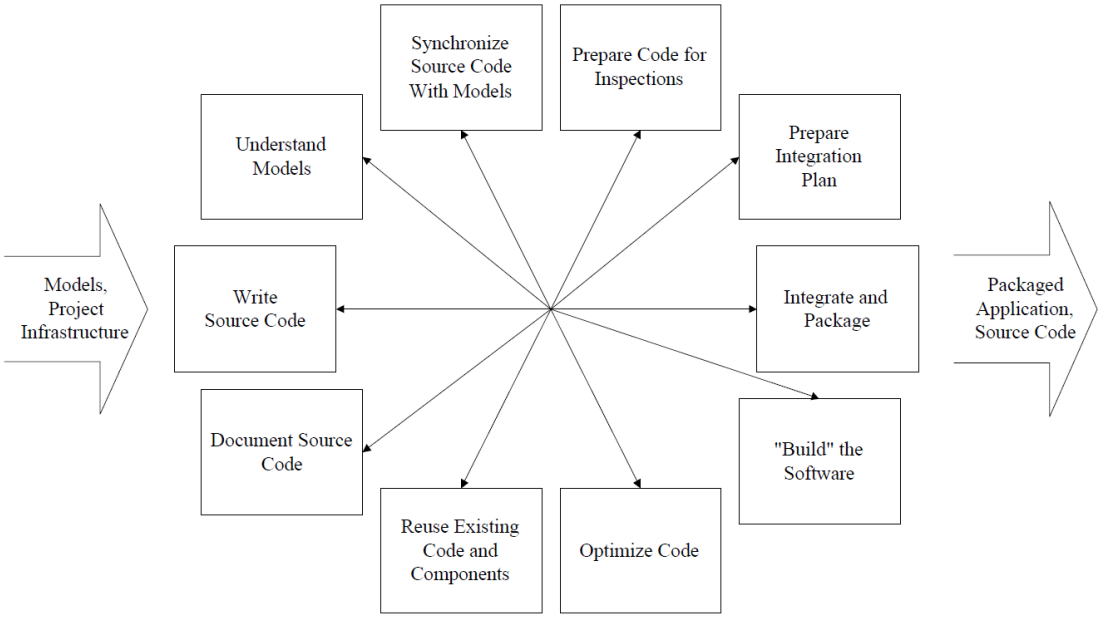


Figura 2-11. Patrón de proceso de la etapa programación

Patrón de proceso fase: describe las interacciones entre los patrones de proceso de etapa en una sola fase del proyecto en las que pertenecen. Generalmente las fases en un proyecto son realizadas de forma serial en desarrollos estructurados y orientados a objetos; la figura 2-13 presenta el patrón de proceso para la fase construcción:

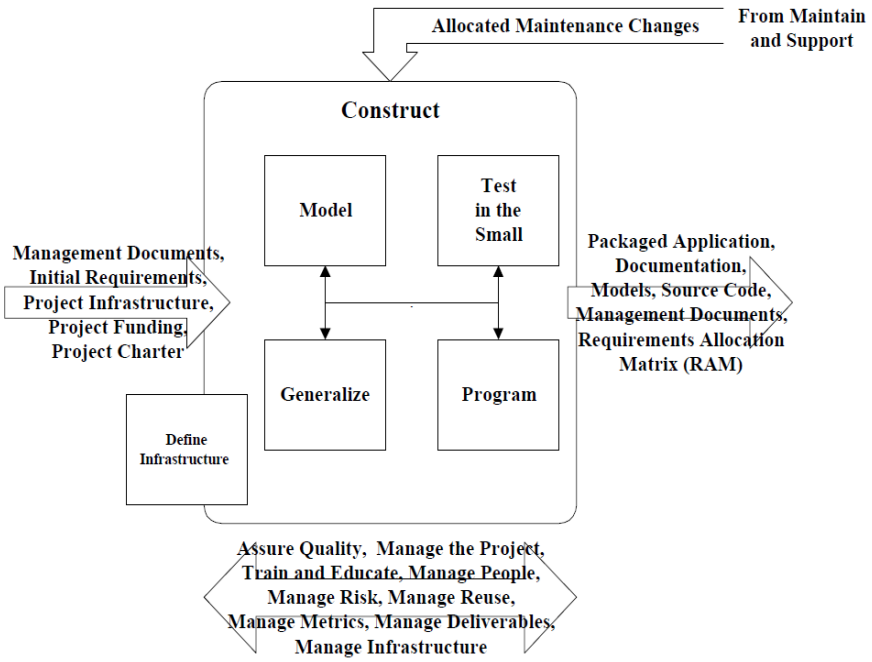


Figura 2-0-12. Patrón de procesos de la fase de construcción.

El trabajo de Ambler relacionado con patrones de cada tipo es prolijo, lo cual se ve reflejado en varios de sus libros donde proporciona detallados pasos y directrices para integrar y formar los patrones dentro de un proceso (Ambler S. W., More Process Patterns: Delivering Large-Scale Systems, 1999), basado en sus librerías de patrones el autor ha propuesto un proceso general para el desarrollo de software denominado el proceso de software orientado a objetos compuesto de cuatro seriales fases, cada una realiza una serie de etapas, y a su vez cada etapa consiste de una serie de tareas como se muestra en la figura 2-14:

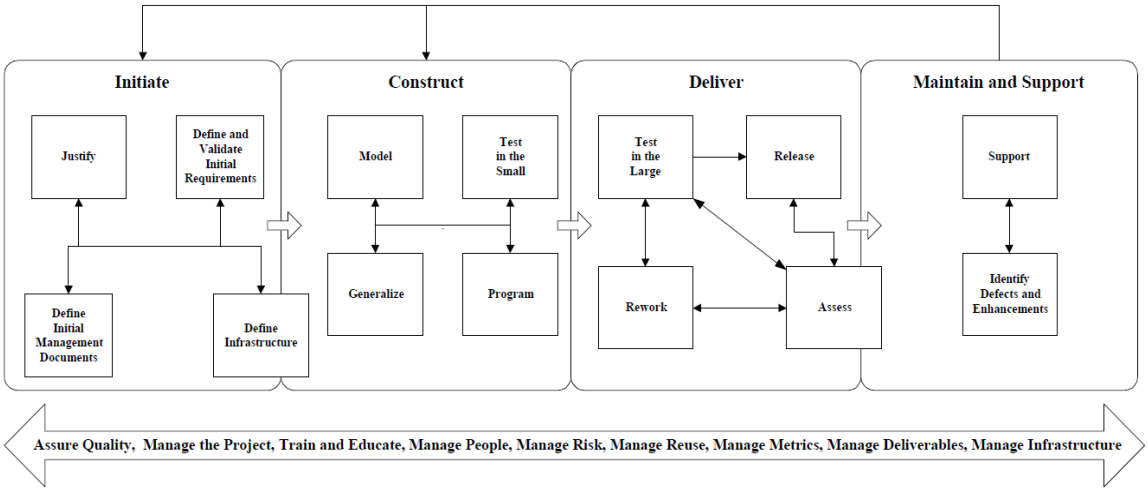


Figura 2-13. Proceso de software orientado a objetos.

6. METAMODELOS DE PROCESOS

En 2002 la Object Management Group (OMG) publica el estándar SPEM (Software and System Process engineering Metamodel) con el propósito de establecer el estándar para la industria para la representación de los modelos de procesos en ingeniería de software y de sistemas (OMG, 2002). SPEM se basa en conceptos relacionados con el modelo dirigido por arquitectura (MDA) (Schuppenies & Steinhauer); a su vez también es influenciado por el metamodelo de procesos unificado (USPM) este último pensado inicialmente como un metamodelo para el proceso RUP, en consecuencia el modelamiento de procesos en SPEM se apoyan principalmente en UML (Verdugo & Ruiz, 2008).

La representación de un metamodelo en SPEM se logra mediante la colaboración de entidades activas denominadas roles encargados de realizar las operaciones especificas denominadas actividades en un conjunto de objetos tangibles denominados productos de trabajo para producir artefactos definidos y completos; recapitulando SPEM describe la estructura de un proceso de desarrollo de software como un conjunto de roles responsables de productos de trabajo y de las actividades que realizan a los productos de trabajo(ver figura 2-15).

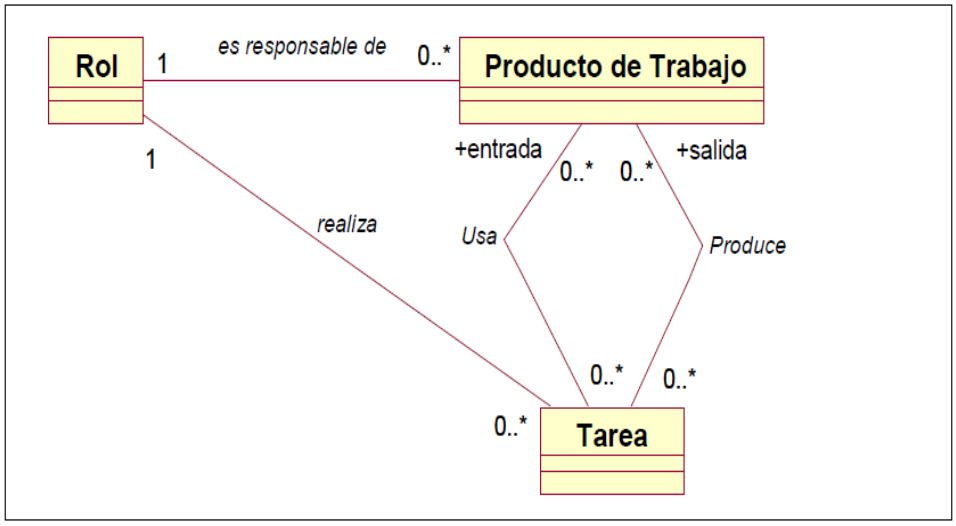


Figura 2-14. Estructura de un proceso de desarrollo en SPEM.

Actualmente SPEM se encuentra en su versión 2, y si bien el proceso anteriormente descrito es simple, el proceso completo de SPEM es mucho más complejo y granular; SPEM también adiciona elementos para definir el orden de ejecución de actividades y el ciclo de vida del proceso como: iteración, fase y ciclo de vida, a su vez se adiciona elementos como pre y post condiciones, que comúnmente están asociados a segmentos importante dentro de un ciclo de vida como lo son los hitos y sus correspondientes productos de trabajo. SPEM no proporciona elementos para el modelado del comportamiento, no obstante proporciona mecanismos provenientes de UML para definir comportamiento del proceso.

CAPITULO 3
ANTECEDENTES Y REVISION DE LITERATURA

1. ANTECEDENTES

En la siguiente sección, las diferentes propuestas metodológicas existentes orientadas específicamente al desarrollo de software para dispositivos móviles son estudiadas y analizadas.

1.1 MODELO DESARROLLO MOVIL EN ESPIRAL

Ann Nosseir et al, utilizaron el modelo de desarrollo en espiral como base, con la adopción de un enfoque de usabilidad dirigida por modelos. El modelo en espiral es un refinamiento del clásico modelo de ciclo de vida de desarrollo de software en cascada. El modelo en espiral tiene la ventaja de combinar el enfoque de desarrollo iterativo de varios prototipos con los sistemáticos y controlados aspectos del modelo en cascada, donde continuamente se refinan los requerimientos, seguidos de diseño e implementación. El modelo es el segundo entre otros cuatro modelos en integrar usabilidad en la práctica habitual en el proceso de desarrollo. El concepto es que mediante varias iteraciones mejorar la calidad del software y reducir los errores de usabilidad. (Nosseir, Flood, Harrison, & Ibrahim, 2012)

Los autores proponen que el modelo es apropiado para largos, costosos y complicados proyectos. Agregan que el modelo intenta ser un modelo para reducción del riesgo. Es un modelo de desarrollo iterativo centrado en el usuario que se basa en la premisa de que los usuarios podrán estar involucrados durante el ciclo de vida del diseño. El modelo en espiral tiene varias iteraciones cada una produciendo prototipos que cuentan con la ayuda de la evaluación de los usuarios.

El modelo es espiral está constituido de 5 iteraciones, cada una con 4 partes denominadas: análisis y definición de requerimientos, diseño, implementación y pruebas y planeación de próxima iteración (ver figura 3-1). A medida que el progreso en espiral avance, podrán aparecer más detalles de usabilidad. La medición de usabilidad en el modelo es iterativa y usa trazabilidad en el tiempo para comparar los resultados y modificar y mejorar las métricas. A continuación se describe cada una de las iteraciones:

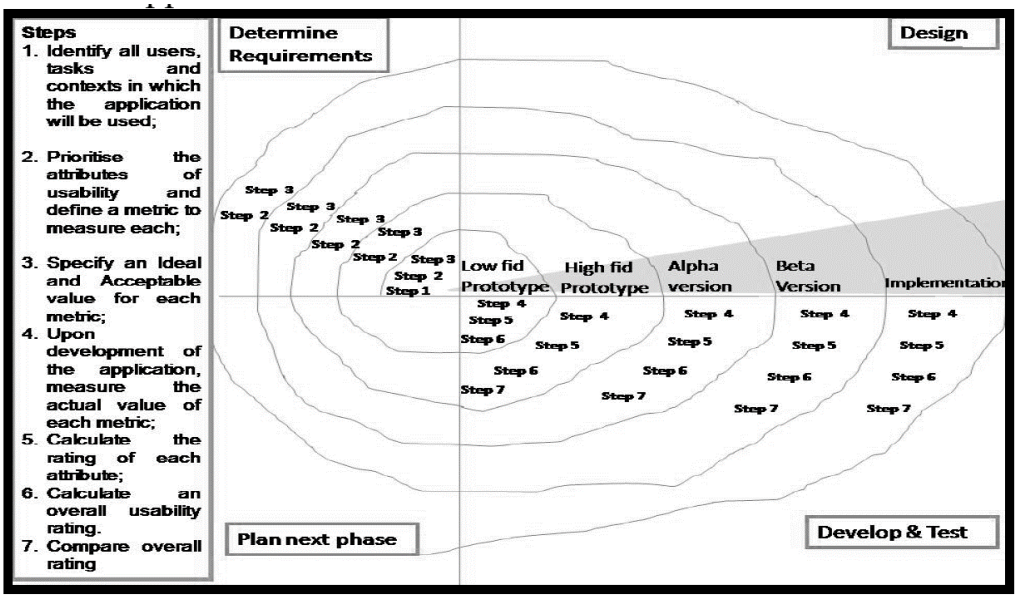


Figura 3-1. Proceso de desarrollo en espiral móvil. Fase 1. Determinación de requerimientos.

Primera Iteración:

Requerimientos: recolección de requerimientos del sistema, identificación de usuarios, tareas y el contexto donde la aplicación va ser utilizada. Se definen y priorizan los atributos de usabilidad e identificando una métrica para medir cada atributo, donde además se especifica un ideal y aceptable calor de cada una.

Diseño: esbozo y desarrollo de un prototipo de baja fidelidad de la interface de la aplicación.

Pruebas: se realizan diferentes técnicas de usabilidad para medir el actual valor de cada atributo y calcular la calificación de cada atributo.

Plan: próxima iteración

Segunda iteración:

Requerimientos: el equipo de desarrollo deberá tener una mejor idea de cuáles son los requerimientos de usabilidad para el sistema aunque los requerimientos de diseño pueden no estar completos. Se recolecta más información y requerimientos de la app. Entonces los atributos de usabilidad son redefinidos y priorizados. El resultado es la modificación de las métricas para acomodar la adición de requerimientos y de nuevo se especifican valores ideales y aceptables para cada métrica.

Diseño: se comienza a desarrollar un prototipo de alta fidelidad enfocado a la interfaz.

Pruebas: durante el desarrollo se aplican diferentes técnicas de usabilidad del actual valor de cada atributo podrá ser medido y calificado de cada atributo y podrá ser calculado y comparado con los resultados de la iteración previa

Plan: siguiente iteración

Tercera iteración:

Requisitos: los requisitos de diseño están siendo determinados, se evalúan nuevamente todos los tributos de usabilidad.

Diseño: se desarrolla todo el sistema la versión Alpha es realizada

Pruebas: se aplican las técnicas de usabilidad utilizadas anteriormente y se comparan los resultados con la iteración previa.

Plan: siguiente iteración

Cuarta iteración:

Requisitos: con base a los resultados de la anterior iteración son usados para identificar y priorizar los atributos de usabilidad y las respectivas métricas

Diseño: la versión beta del sistema es desarrollada y liberada para ser evaluada.

Pruebas: nuevamente se aplican las técnicas de usabilidad para medición de cada atributo de usabilidad y se compara con la anterior iteración:

Plan: siguiente iteración

Quinta iteración:

Requisitos: los resultados de las anteriores iteraciones son utilizados para identificar y priorizar requisitos de usabilidad.

Diseño: el producto final es desarrollado

Pruebas: se realiza la medición de cada atributo de usabilidad y se compara con las iteraciones pasadas.

Plan: informe final de los resultados

El proceso de desarrollo móvil en espiral es un enfoque dirigido por usabilidad, altamente influenciado por el modelo en espiral. Este enfoque busca reducir los altos riesgos introducidos por las restricciones de los dispositivos móviles. Las cinco iteraciones aseguran que los requerimientos son apropiadamente dirigidos y validados contra los requerimientos. Para reducir errores de usabilidad, el enfoque en espiral usa métricas para evaluar el desarrollo del desarrollo de móviles

aplicaciones en un número de iteraciones. Se identifican una serie de técnicas de usabilidad y estas son incorporadas in cada iteración para evaluar la aplicación.

1.2 DISEÑO DE METODOLOGIA HIBRIDA

Rahimian y Ramsin, proponen que las metodologías agiles han demostrado ser apropiadas para el desarrollo de software para dispositivos móviles. Basados en este supuesto identificaron los requerimientos específicos para una metodología de desarrollo de software móvil. Basado en esto un nuevo método ágil es diseñado usando el enfoque de diseño de metodología hibrida. Afirman que su metodología y el enfoque utilizado para su construcción facilitan la aplicación de un enfoque de ingeniería de software en la construcción de software móvil. (Rahimian & Ramsin, 2008)

La nueva metodología fue desarrolla mediante la aplicación de un enfoque de método de ingeniería (ME).el enfoque utilizado fue el Diseño de metodología hibrida, se utiliza para desarrollo iterativo e incrementales de metodologías basadas en una serie de requisitos predefinidos y de conocimiento adquirido de existentes metodologías, procesos, patrones y meta modelos. Elicitaron requerimientos de alto nivel del entorno de desarrollo de software móvil mediante revisión de literatura, Ideas de desarrollo adaptativo de software (ASD) y desarrollo de nuevos productos (NPD) se utilizaron en la construcción de la metodología.

La metodología se construyó mediante cuatro iteraciones del diseño hibrido, el genérico ciclo de vida de desarrollo de software fue utilizado como base del proceso (como se puede ver en la figura 3-2). En la primera iteración la metodología fue elaborada mediante el uso de patrones genéricos basados en riesgo, centrado en la arquitectura y desarrollo basado en pruebas. La fase de análisis fue dividida en pre análisis y análisis detallado con el fin de mitigar los riesgos en el desarrollo. La fase de diseño fue dividida en diseño arquitectural y diseño detallado. La implementación y los subprocesos de pruebas fueron combinadas para acomodar el desarrollo basado en pruebas. Hasta el momento los patrones incluidos son frecuentemente utilizados en las metodologías agiles. El foco principal de la metodología de desarrollo de software móvil está enfocada en el desarrollo del producto, los autores incorporaron la fase de comercialización. El enfoque utilizado en el diseño en la iteración fue principalmente instanciación usando metamodelos incluido SPEM y OPF, y ciclos de vida generales orientada a objetos como OOSP.

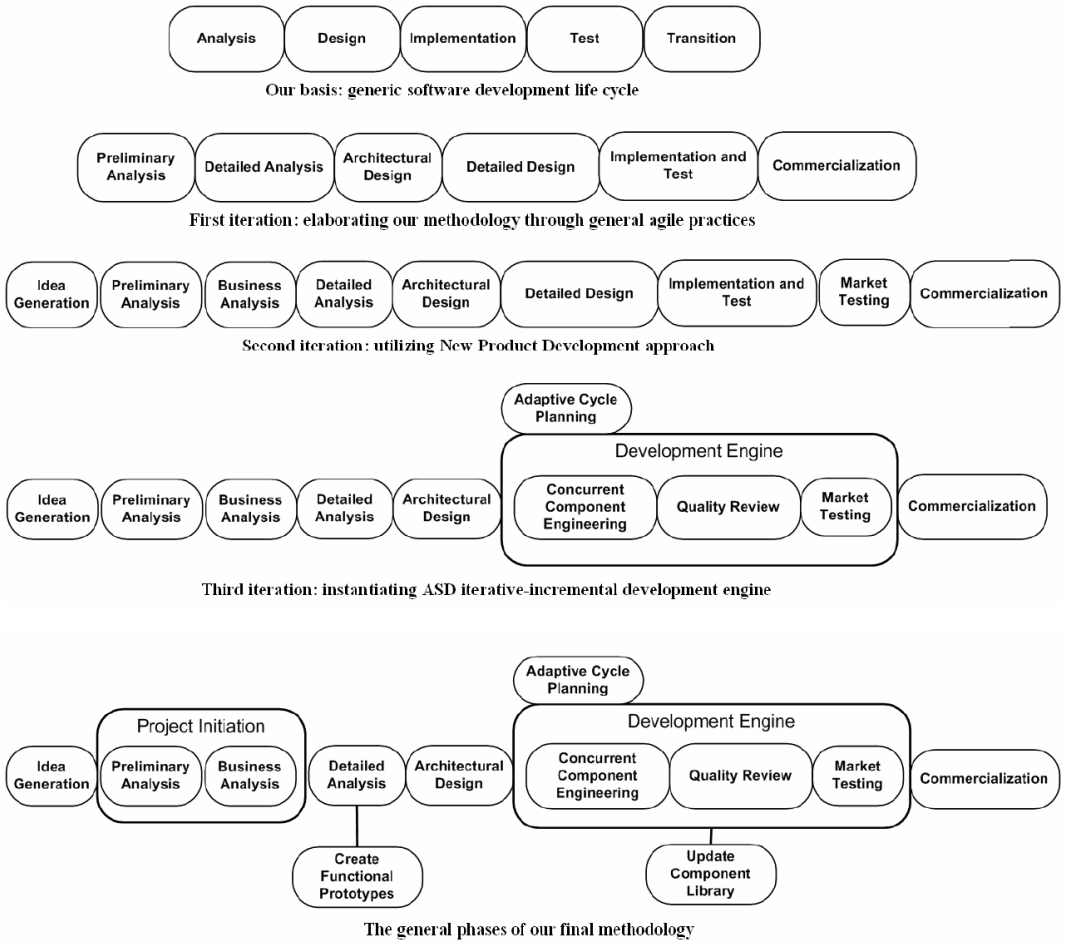


Figura 3-2. Refinamiento gradual de la metodología ágil para el desarrollo durante las iteraciones del proceso de diseño de metodología hibrida.

En la segunda iteración el foco se centró en el mercado, actividades del desarrollo de nuevos productos fueron incorporados. Estas fueron generación de la idea la cual fue incorporada en el inicio de las fases del proceso y prueba del mercado fue integrado antes de la comercialización del software. El enfoque de diseño en la segunda iteración fue integración usando reusables partes de Procesos de desarrollo de nuevos productos.

La tercera iteración fue mejorado los procesos de desarrollo con la incorporación de ideas provenientes del desarrollo adaptativo de software ASD, el cual es componente basado en metodología agiles especialmente fuerte en la medición de aseguramiento de la calidad. El ciclo de especular colaborar aprender que forma parte de los procesos básicos de ASD proporcionan los medios para enfrentar las incertidumbres del desarrollo de software. Usando el enfoque de integración, el resultado de la iteración es la incorporación de sesiones de revisión dentro de la metodología. Se mejoraron el soporte de procesos de reutilización con el beneficio que ofrece la práctica de desarrollo basada en componentes.

Los autores considerando los potenciales riesgos tecnológicos en el desarrollo de software móvil, la última iteración se enfocó principalmente con la adicción de prototipo al proceso. La prototipacion facilita la rápida especificación de la arquitectura física. En esta iteración los procesos fueron refinados moviendo las actividades de análisis preliminar y de negocios dentro de iniciación del proyecto. El enfoque de diseño usado principalmente fue composición con integración usados cuando se reusaron ideas de la metodología ASD.

1.3 MOBILE-D.

Abrahamsson et al., Desarrollaron un método ágil llamado Mobile-D. El modelo desarrollado es basado en las prácticas de desarrollo de Extreme Programing (XP) , metodologías Crystal como método para la escalabilidad y la cobertura del ciclo de vida proporcionado por el proceso unificado RUP (Abrahamsson, y otros, 2004).

El enfoque es optimizado por un grupo de desarrolladores no superior a 10 trabajando co-ubicados en el mismo lugar, con el objetivo de entregar una completa y funcional aplicación móvil en corto tiempo, menos de 10 semanas.

Mobile D está dividido en cinco iteraciones: exploración, inicialización, producción, estabilización y pruebas. A su vez cada fase consiste de tres tipos diferentes de día: día de planeación, día de trabajo y día de liberación (ver figura 18). Si el proyecto es abordado por múltiples grupos, un día de integración as necesario. Las practicas utilizadas en las diferentes fases son: ajustes en fase y estimulación, línea base de arquitectura, desarrollo dirigido por pruebas(TDD), integración continua, programación por pares, métricas, mejoras al proceso de desarrollo, cliente en sitio y un enfoque centrado en el usuario.

A continuación se detallan cada una de las fases de Mobile D:

Fase de exploración:

Se establece un plan de proyecto y los conceptos relacionados. Los autores de la metodología recalcan la importancia de la identificación de los involucrados y su participación en esta fase.

Fase de inicialización:

Los desarrolladores identifican y preparan todos los recursos necesarios, se preparan los planes para las siguientes fases y se estable el entorno técnico (capacitación y entrenamiento del equipo).Los autores de Mobile-D afirman que su contribución al desarrollo ágil se centra fundamentalmente en esta fase, en la investigación de la línea arquitectónica. Esta acción se lleva a cabo durante el día de planificación.

Fase de producción:

Se repite la programación de tres días (planificación-trabajo-liberación) se repite iterativamente hasta implementar todas las funcionalidades. Primero se planifica la iteración de trabajo en términos de requisitos y tareas a realizar. Se preparan las pruebas de la iteración de antemano (de ahí el nombre de esta técnica de Test Driven Development, TDD). Las tareas se llevarán a cabo durante el día de trabajo, desarrollando e integrando el código con los repositorios existentes. Durante el último día se lleva a cabo la integración del sistema (en caso de que estuvieran trabajando varios equipos de forma independiente) seguida de las pruebas de aceptación.

Fase de estabilización:

Se llevan a cabo la integración de los subsistemas en un solo producto de software. Esta será la fase más importante en los proyectos multi-equipo con diferentes subsistemas desarrollados por equipos distintos; luego Todo el esfuerzo se dirige a la integración del sistema. Adicionalmente se puede considerar en esta fase la producción de documentación.

Fase prueba y reparación del sistema:

Tiene como meta la disponibilidad de una versión estable y plenamente funcional del sistema. El producto terminado e integrado se prueba con los requisitos de cliente y se eliminan todos los defectos encontrados.

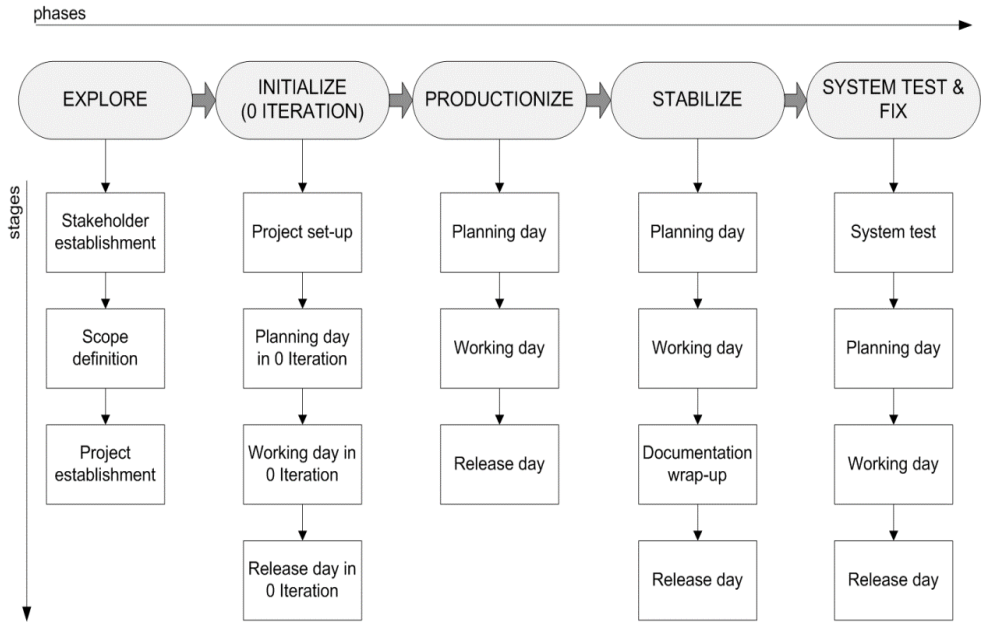


Figura 3-3. Mobile D Fases y etapas. Fuente *(ELECTRONICS, 2015)*.

1.4 RaPiD7.

Dooms et al (Dooms & Kylmäkoski, 2005), propusieron un método cuyo propósito es mejorar la documentación generada durante el proceso de desarrollo sin afectar la calidad y cantidad de la documentación lo cual como resultado produce software de mejor calidad; el método es denominado RaPiD7 (rápido proceso de documentación, siete pasos). El enfoque de rapid7 es crear documentación pertinente con la realidad empleando el menor esfuerzo posible; una mejor interacción humana y documentación de calidad contribuyen a la calidad del software, para ello RaPid7 propone realizar talleres con el propósito de facilitar la documentación y el plan de interacción de los integrantes del proyecto, cada taller sigue una estricta planeación que consta de siete fases: primero preparación de la fase, segundo iniciarla correctamente (fase Kick-off), tercero fase de reunión de la idea, cuarta buscar la solución del problema en la fase de análisis, resultado de la anterior fase, quinto las decisiones son tomadas en la fase de decisión, sexto luego estas decisiones son especificadas en la fase de diseño detallado y por último los resultados del talleres son verificados y los próximos pasos a seguir en la fase de cierre. Los pasos de un taller se muestra en la siguiente imagen:

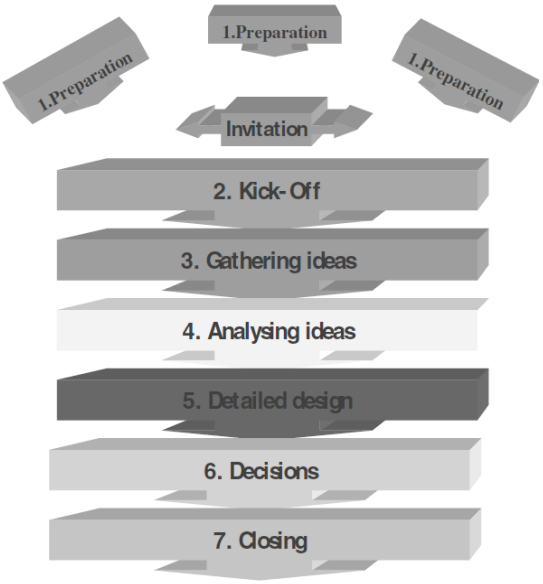


Figura 3-4. Siete pasos de un taller en RaPid7.

RaPid7 promueve la agilidad con la adopción de dos prácticas comunes del movimiento que son hacer la cosa más simple que trabaja y todo el equipo. El método proporciona una estructura de tres capas del proceso:

Capa de proyecto: describe como la interacción humana y la toma de decisiones conjunta es prevista para los proyectos de software.

Capa de caso: describe como los casos seleccionados como los documentos son creados en consecutivos talleres.

Capa de talleres: describe como el trabajo actual se lleva a cabo en forma de talleres facilitadores.

1.5 MASAM.

Un enfoque ágil para el desarrollo de software de dispositivos móviles es propuesto por Jeong et al (Jeong, Lee, & Shin, 2008), basado en ideas derivadas de Extreme Programing (XP), Proceso unificado ágil, RUP y metamodelo para procesos de ingeniería para sistemas y software (SPEM).El método es denominado MASAM (Mobile Application Software Agile Methodology) según sus autores es recomendada para empresas pequeñas dedicadas al desarrollo de este tipo de aplicaciones, es descrito acordado con las especificaciones de SPEM el cual define tres principales procesos:

Tabla 3-1. Procesos de MASAM.

Kind	Description	Name
Role	It defines a set of related skills, competencies and responsibilities of an individual(s).	Planner, Manager, UI designer, Developer, Development team, Initial development team, Tester, User
Task	It is an assignable unit of work assigned to a specific role. The granularity of a task is generally a few hours to a few days and usually affects one or only a small number of work products.	Product Summary, Initial Planning, User Definition, Initial Analysis, Select Resource, Select Process, Establish Environment, Write Story Card, UI Design, Define Architecture, Planning, Iteration plan, Face-to-face Meeting, Incremental Design, TDD, Refactoring, Release Plan, Feedback, Pattern Manage, Pair Programming, Integration, Acceptance Test, User Test Figure
Work Product	It is a general term for task inputs and outputs. There are three types of work product.	Product Summary, Project Planner, UI Sample, UI Model, UI pattern, Architecture Pattern, Application Pattern, Story Card, Task Card, Architecture Model, Component Model, Test Case.

La metodología está compuesta de cuatro fases, se destaca la adopción de muchas prácticas provenientes de XP en la fase de desarrollo, a continuación en la figura 3-2 muestra los elementos que constituyen las fases y una descripción breve de cada una de ellas:

Tabla 1-2. Fases de MASAM.

Phase	Activity	Task
Preparation Phase	Grasping Product	Product summary
		Pre-planning
	Product Concept Sharing	User Definition
		Initial product analysis
	Project Set-up	Development process coordination
		Project resource coordination
		Pre study
Embodiment Phase	User Need Understanding	Story card workshop
		UI design
	Architecting	Non-functional requirement analysis
		Architecture definition
		Pattern management
Development Phase	Implementation & Preparation	Environment setup
		Development Planning
	Release Cycle	Release Planning
		Iteration Cycle
		Release
Commercialization Phase	System Test	Acceptance Test
		User Test
	Product Selling	Launching Test
		Product Launching

Fase preparación para el desarrollo: se identifica el objetivo de la aplicación por parte del cliente, luego se establece y se comparte la visión del producto entre todos los interesados, si la visión no es comprendida, entonces la empresa lo aprenderá en forma de ensayo y error durante el ciclo de vida.

Fase de realización: Los requisitos de la aplicación son representados en forma concreta, puede ser apoyado por el uso de prototipos o simuladores para confirmar que lo realizado cumple las expectativas del cliente. Mediante el uso de conocimiento del dominio proporciona patrones de aplicaciones disponibles en su biblioteca, los cuales incluyen algoritmos, el modelo de arquitectura y modelo de interfaz gráfica, para ayudar al cliente a identificar la aplicación buscada.

Fase desarrollo de producto: se construye gradualmente la aplicación con liberaciones regulares al cliente; durante cada iteración el grupo de desarrollo escoge una tarjeta de usuario a implementar, luego de ser discutida la tarjeta es dividida en varias tareas entre el equipo para su realización, al terminar el equipo integra el trabajo, practicas agiles de XP son adoptadas en esta fase.

Fase de comercialización: la aplicación es desplegada en el dispositivo y entorno de los usuarios, se enfatiza en la usabilidad de la aplicación como indicador de éxito, pruebas de aceptación son realizadas.

1.6 SLESS.

Otro enfoque de tipo ágil es el propuesto por Cunha et al denominado SLeSS que integra la gestión de proyectos propuesto por la metodología ágil Scrum y la mejora de procesos de Lean Six Segma (LSS) (Cunha, Dantas, & Andrade, 2011). Se ha documentado su utilización en proyectos de software embebido para teléfonos móviles con un enfoque experimental de investigación y desarrollo sobre terminales de proveedores simulando clientes.

La premisa de SLeSS es la adopción del enfoque Scrum en la organización, luego con la aplicación de LSS proporciona un Framework de calidad que complementa todo el proceso de desarrollo de Scrum. El modelo SLeSS se describe en cinco fases que se presentan a continuación:

Tabla 3-3. Fases de SLeSS.

	Phases	Backlog Items
1.	Definition Phase	LSS Project Contract
		Initial Analysis
2.	Measurement Phase	SIPOC Diagram (Supplier, Inputs, Process, Outputs, Customers)
		Process Map
		Cause and Effect Diagram
		Cause and Effect Matrix
		Impact Effort Matrix
		Initial Capability
		Measurement and Inspection Systems
3.	Analysis Phase	FTA (Fault Tree Analysis)
		Analysis of critical inputs of the processes
4.	Improvement Phase	Action Plan
		SIPOC
		Process Map
		Piloted Solution
		Final capability of the processes
5.	Control Phase	Control Plan

Los autores manifiestan que el enfoque mejora la productividad notablemente, la calidad del proceso y reduce los costos por su mejor forma de adaptarse a los cambios de los requisitos, el cumplimiento de los plazos acordados, la disminución de trabajo extra y la entrega constante de versiones con minoría en defectos o fallos. El método también aloja el mejoramiento del proceso de desarrollo y de gestión, mediante estadísticas de control y la alineación de los objetivos y expectativas para satisfacer las necesidades del cliente.

1.7 ENFOQUE FLEXIBLE PARA EL DESARROLLO DE SOFTWARE MOVIL.

Flora et al (Flora, Wang, & Chande, 2014). Realizaron una encuesta investigativa con el propósito de obtener información relacionada con diversas practicas emergentes utilizadas en el proceso de desarrollo de software para móviles, el cuestionario fue aplicado a empresas de desarrollo móviles, equipos de desarrollo, experto en movilidad, investigadores y personas involucradas en el sector móvil. Los resultados obtenidos fueron utilizados como el inicio para identificar los requerimientos para la ingeniería de software para móviles. El autor basado en el análisis de las respuesta propone un enfoque flexible para el desarrollo de software para móviles compuesto de cuatro fases comprobadas como eficaces en este tipo de desarrollo según expertos móviles, las cuales cubren desde la idea o concepción hasta la liberación del producto final, a continuación se describen las cuatro fases propuestas:

Fase envisión:

Etapa de análisis, se identifica el propósito o motivación de la aplicación y su público objetivo. Puede ir a acompañado de análisis de los requerimientos del negocio y el conjunto de características de la aplicación ofrecerá a los objetivos del negocio.

Etapa de planeación, se estima el presupuesto, los recursos, el cronograma y se desarrolla la visión inicial del proyecto; cuestiones de tecnología son abordadas: plataforma, dispositivos y navegadores. Planeación estratégica se define el objetivo, metas, hitos y la viabilidad del proyecto. Esta etapa viene acompañada de otras tareas como:

Diseño de la experiencia de usuario mediante diferentes técnicas (bocetos, dibujos, prototipos).

Desglose de tareas de trabajo de las funcionalidades en pequeños historias de usuario o tareas.

Fase solución:

Diseño: hace énfasis en el diseño visual de la aplicación, se desarrollan bocetos, maquetas y prototipos visuales delos escenarios de la aplicación, después de aprobado la interfaz de usuario por los clientes, se define la arquitectura y el modelo de seguridad de la aplicación. Se acompaña de planes de pruebas que incluyen pruebas de código, escritura de cartas de historia.

Desarrollo: el proceso es realizado en iteraciones, se adopta la práctica de Sprint proveniente de Scrum para el desarrollo del código de los módulos y el diseño de la base de datos.se realizan pruebas en dispositivos objetivo de despliegue, en ellos pruebas unitarias son ejecutadas, los errores detectados son corregidos. Terminado el desarrollo el producto se pone a disposición de los usuarios finales para pruebas y opiniones.

Fase aseguramiento de la calidad.

Pruebas: se somete a pruebas los requerimientos, especificaciones y diseños creados en las fases anteriores. Las pruebas consisten en la definición de casos de prueba, pruebas automatizadas y pruebas en emulador procurando abarcar una amplia variedad de dispositivos.

Cambio: los cambios y correcciones sugeridas por los usuarios son realizados, luego se lleva a cabo pruebas de usuario final y pruebas de aceptación del cliente.

Fase Liberación de producto:

Despliegue: se realizan actividades enfocadas en la distribución de la aplicación: presentarla en una tienda de aplicaciones, actividades de mercadeo, suscripción, captación de nuevos usuarios y el lanzamiento de la primera versión de la aplicación creada.

Soporte y mantenimiento: después de liberado el producto a los usuarios se presentan errores o ausencia de funcionalidades, proporcionar el medio de retroalimentación por parte de los usuarios es vital, acompañado de reportes automatizados de desempeño de la aplicación. El mantenimiento y el soporte debe ser realizado en frecuentes iteraciones de liberaciones a través de la tienda de aplicaciones o el canal de distribución, y debe poder proveer todo lo relacionado con el soporte que incluye: mejoras, actualizaciones, adición de nuevas funcionalidades y características de la aplicación.

2. CUESTIONES, CARACTERÍSTICAS Y DESAFÍOS DEL DESARROLLO DE SOFTWARE QUE FUNCIONA EN MÓVILES.

En la siguiente sección se presenta una lista de algunas de las principales características reportadas en la literatura como únicas presentes en el desarrollo de software móvil; la construcción de la lista se realizó mediante la unificación de las características comunes encontradas.

Cuestiones técnicas del dispositivo: capacidades limitadas en procesamiento, almacenamiento y funcionamiento que afectan el desempeño de las aplicaciones. (Heyes, 2002) (Abrahamsson P. , Agile software development of mobile information systems, 2007) (Joorabchi, Mesbah, & Kruchten, 2013) (Wasserman, 2010).

Entorno altamente volátil: El entorno es marcado por un alto nivel de incertidumbre y dinamismo; reflejado en la constante publicación de versiones en las plataformas de desarrollo y lanzamientos de números dispositivos donde desplegar las aplicaciones como: teléfonos inteligentes, tabletas, relojes, entre otros; el gran número de desarrolladores y empresas en el sector es otro factor que lo hacen altamente competitivo. (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005) (Abrahamsson, y otros, 2004) (Wasserman, 2010)

Incompletos, faltantes o cambiantes requerimientos: cambios, modificaciones o adicción de requerimientos son frecuentes durante el desarrollo de una aplicación móvil. (Abrahamsson P. , Agile software development of mobile information systems, 2007) (Joorabchi, Mesbah, & Kruchten, 2013)

Interfaz de usuario: Los aspectos de la experiencia del usuario son definidos por las directrices de interfaz de usuario de cada plataforma, muchas de ellas se encuentran implementadas en el Kit de desarrollo (SDK) (Wasserman, 2010). (Holler, 2006)

Usabilidad o cuestiones de experiencia de usuario: restricciones intrínsecas del tamaño del dispositivo introducen restricciones de usabilidad, dadas generalmente en dificultades de lectura en la pantalla, resolución de texto, imágenes, multimedia; afectación de la interacción del usuario con la aplicación. Algunas de los desafíos de usabilidad de las aplicaciones móviles han sido descritos: (Zhang, Adipat, & B, 2005) (Holler, 2006) (Wasserman, 2010):

- Contexto de movilidad: los usuarios de las aplicaciones móviles no permanecen estáticos o en un solo sitio, la interacción con otros elementos del entorno como objetos o personas pueden ocasionar distracción del usuario.
- Conectividad: cuando la conectividad es lenta o deficiente impacta el desempeño de la aplicación.

- Tamaño reducido de la pantalla: el despliegue de la información es drásticamente reducido en comparación con un computador de escritorio.
- Resolución de vista: la resolución de la pantalla de un dispositivo móvil, puede no ser buena en relación con otros dispositivos ocasionando distorsión o baja calidad en imágenes o video.
- Métodos de entrada de datos: Los métodos de entrada disponibles de los dispositivos móviles son limitados, cada uno de ellos requiere de cierta experticia, lo cual ha propiciado el incremento del ingreso de datos erróneos y la disminución de entrada de datos.

Portabilidad y cuestiones asociadas a compatibilidad entre plataformas: una aplicación para un dispositivo móvil puede operar en diferentes plataformas, donde la característica principal es la fragmentación en la tecnología de la plataforma y los dispositivos. (Heyes, 2002) (Wasserman, 2010)La fragmentación presenta serios desafíos en el desarrollo de software en móviles, la consolidación de algunas de ellas ha dado pie para aumentar esta diferencia entre ellas, las cuales se presenta en (Joorabchi, Mesbah, & Kruchten, 2013):

- Fragmentación entre plataformas: La heterogeneidad de las plataformas se distingue principalmente en los lenguajes de programación, API/SDK, herramientas que soportan, interfaz de usuario, experiencia de usuario, expectativas de usuario entre otras.
- Fragmentación en la misma plataforma: En una misma plataforma, pueden existir varios dispositivos con diferencias en sus propiedades como capacidad de procesamiento, resolución entre otros, también puede presentarse en la versión del sistema operativo.

Tamaño del equipo de desarrollo: en su mayoría el desarrollo de las aplicaciones para dispositivos móviles es realizado por micro empresas o pequeños equipos de desarrollo (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005). No obstante se ha observado cambios en el tamaño de los equipos de desarrollo, empresas y organizaciones medianas y grandes de la industria del software están incursionando cada vez más en el desarrollo de este tipo de aplicaciones motivadas por la atractiva cuota de mercado que representa (Harleen, Flora, & Swati).

Time to market: el tiempo para distribución y salida al mercado son estrictos, es común en este tipo de desarrollo plazos apretados de entrega dispuesto generalmente por departamentos de Marketing. (Heyes, 2002) (Abrahamsson P. , Agile software development of mobile information systems, 2007)

Sistemas pequeños: El tamaño del software para móviles es variable, por lo general su tamaño es menor a 10000 líneas de código. (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005)

Complejidad de Pruebas: El soporte para pruebas automatizadas de aplicaciones móviles se encuentra limitado, muchas herramientas y emuladores no cuentan con

características importantes del entorno móvil: movilidad, sensores, servicios de localización, gestos y los diferentes tipos de entrada de datos de los dispositivos. Un reducido porcentaje de pruebas son realizadas de forma automatizada, mientras la práctica de pruebas manuales en emuladores y dispositivos físicos es mayoría; en muchos de los casos son los mismos desarrolladores los encargados de ejecutarlas (Joorabchi, Mesbah, & Kruchten, 2013) (Wasserman, 2010).

Ciclos de desarrollo cortos: Los tiempos de desarrollo de las aplicaciones móviles generalmente suele estar comprendido entre 1 a 6 meses. (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005)

Entorno de desarrollo con tecnología orientada a objetos: Plataformas populares de desarrollo de software móvil son orientadas a objetos, entre las que se encuentran: JAVA, Object C, C++. (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005)

Identificación de los usuarios finales o consumidores: Potencialmente ilimitado o no cuantificable su número de usuarios finales (Abrahamsson P. , Agile software development of mobile information systems, 2007) (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005)

Software críticos no seguro: Un alto porcentaje de las aplicaciones disponibles en los mercados de aplicaciones son para propósitos de entretenimiento (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005).Sin embargo; actualmente, existen aplicaciones móviles con niveles de criticidad alto, como las aplicaciones que manejan información financiera y monitoreo constante de salud (Harleen, Flora, & Swati).

Cuestiones de seguridad y privacidad: Las plataformas móviles disponibles son abiertas, lo que permite la instalación de software malintencionado o "Malware" que pueden afectar el funcionamiento del dispositivo, incluyendo manipulación indebida o acceso a información local del dispositivo (Wasserman, 2010).

Nivel de aplicación del software: La mayoría de las aplicaciones para dispositivos móviles son stand-alone (Abrahamsson P. , Keynote: Mobile software development - the business opportunity of today, 2005).No obstante, En la actualidad las aplicaciones móviles interactúan con otros sistemas de forma colaborativa o con otros elementos como hardware. (Harleen, Flora, & Swati). (Wasserman, 2010)

3. ANALISIS DE RESULTADOS.

Después de la revisión de las propuestas de tipo metodológico existentes que abordan el desarrollo de software para dispositivos móviles, principales características, dificultades y desafíos comunes que presenta el entorno móvil fueron identificados y descritos, en la sección anterior.

Para lograr contrastar las diferentes aproximaciones metodológicas para el desarrollo de software para móviles fue utilizada la lista de características identificadas como significativas en el entorno móvil anteriormente descrita como criterio de comparación, un punto importante de la actividad es obtener los requerimientos iniciales del marco de trabajo objetivo de la presente tesis, mediante la revisión de la comparación permita obtener un entendimiento de las fortalezas y debilidades encontradas, permitiendo luego definir un set de requerimientos para el diseño posterior del marco de trabajo.

El resultado de aplicar la comparación de las propuestas metodológicas para el desarrollo de software para dispositivos móviles es presentado en la siguiente tabla:

Tabla 3-4. Comparación de metodologías móviles y características del entorno móvil.

	METODO EN ESPIRAL	**DESARROLLO HIBRIDO MOVIL**	**MOBILE-D**	**RaPiD7**	**MASAM**	**SLESS**	**ENFOQUE FLEXIBLE**
Cuestiones técnicas del dispositivo		X					X
Entorno altamente volátil		X					
Incompletos, faltantes o cambiantes requerimientos		X	X		X	X	X
Interfaz de usuario					X		X
Usabilidad	X						
Portabilidad y cuestiones asociadas a compatibilidad entre plataformas							X
Tamaño del equipo de desarrollo			X		X	X	
Time to market		X				X	X
Sistemas pequeños						X	
Complejidad de Pruebas			X		X		X

Ciclos de desarrollo cortos			X			X	
Entorno de desarrollo con tecnología orientada a objetos						X	
Identificación de los usuarios finales o consumidores		X					X
Software críticos no seguro							
Cuestiones de seguridad y privacidad							X
Nivel de aplicación del software						X	

CAPITULO 4
DISEÑO Y DESARROLLO DEL MARCO DE TRABAJO

El diseño y construcción del marco de trabajo es iniciado de la identificación de las principales características del desarrollo de software en dispositivos móviles de la sección 2 del capítulo 3, los cuales serán el insumo principal del marco de trabajo objetivo. Si bien, el desarrollo de aplicaciones móviles presenta una serie de características únicas, muchas de las actividades, roles y productos involucrados a lo largo del transcurso del ciclo de vida de desarrollo son similares a otros tipos de desarrollo de software.

Los elementos que constituyen el marco de trabajo propuesto constan de dos principales elementos: primero de un ciclo de vida que Indica cómo influye el marco de trabajo al inicio y fin de un proyecto, generalmente un ciclo de vida de un proyecto de desarrollo de software sigue un patrón similar independiente del tipo de software a ser construido; segundo de las actividades o pasos propuestos a realizar en cada una de las fases. El funcionamiento del marco de trabajo será realizado mediante una serie de fases, las cuales a su vez están constituidas de una serie de actividades que son realizadas iterativamente por un grupo de roles o responsables.

Una vez definidos los elementos que constituyen el Framework, el paso siguiente consistió en establecer su objetivo, para la presente propuesta es proveer una abstracción de alto nivel de los objetivos a alcanzar para un correcto desarrollo de una aplicación que funcione en dispositivos móviles. Que satisfaga las expectativas y necesidades de los clientes y usuarios, y que estas se una vez desplegadas funcionen correctamente en su contexto conformado por la plataforma y los dispositivos. El marco de trabajo para el desarrollo de software móvil propone una referencia genérica del proceso de desarrollo de software para dispositivos móviles que sirve como medio para satisfacer los requerimientos tradicionales presentes en el dominio.

Después de definido el objeto del marco, es necesario establecer la restricción del marco de trabajo propuesto, el cual no debe considerarse como un enfoque de solución para todos los desarrollos de este tipo o como se denomina en el medio *"una talla para todo"* que debe ser aceptada por cualquiera con intereses en estos proyectos, por consiguiente es un marco de trabajo que recomienda cómo se debe construir software para dispositivos móviles.

El diseño del marco de trabajo objetivo se inició definiendo la estructura de alto nivel del marco de trabajo, en este caso la base corresponde al armazón del genérico ciclo de vida de desarrollo de software, el cual consiste de una serie de subprocesos o fases genéricas secuenciales de análisis, diseño, implementación, pruebas y transición (Pressman, 2010), cada una agrupa una serie de actividades básicas a realizar.

Una vez establecido el marco base se procede a realizar una descripción de alto nivel de cada una de las actividades recomendadas con base a los requerimientos del dominio correspondientes a cada una de las fases, el orden propuesto de ejecución de las actividades es discrecional, a continuación se amplía la descripción de cada una de las fases y sus correspondientes actividades del marco de trabajo (ver figura 4-1):

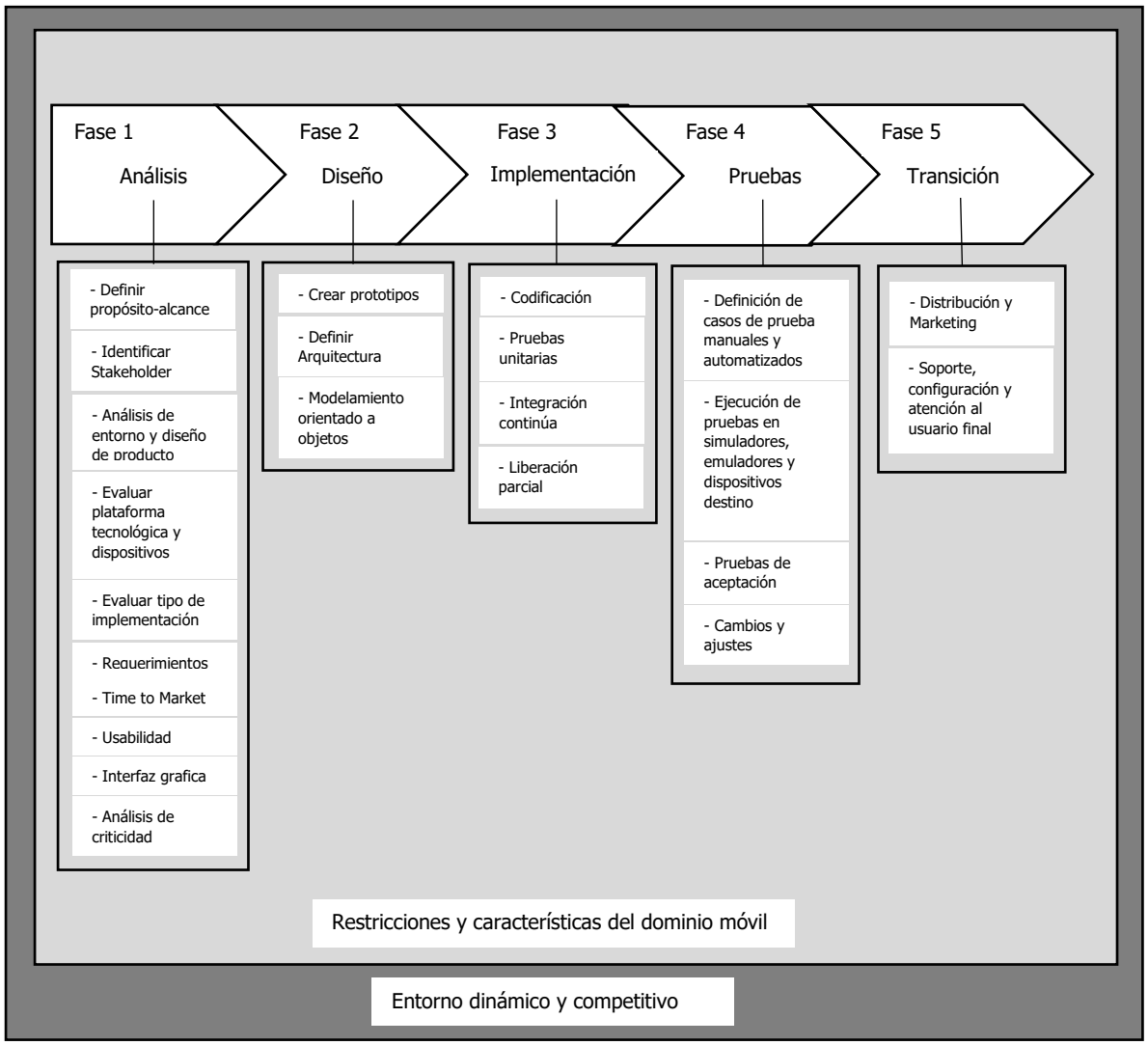


Figura 4-1. Marco de trabajo para el desarrollo de software para dispositivos móviles.

Fase Análisis

El objetivo de esta fase es determinar la motivación o la necesidad de desarrollar la aplicación, responde a la pregunta ¿Por qué se necesita construir una aplicación móvil? ¿Cuál es el usuario objetivo? ¿Qué plataforma y dispositivos se desplegara? Varios factores cruciales en el desarrollo móvil son examinados en esta fase,

inicialmente se parte de un entorno marcado por un alto dinamismo y volatilidad que añade complejidad en el desarrollo, a continuación se describe en detalle cada uno de las características identificadas para esta fase:

En cualquier proceso de desarrollo de software de cualquier tipo se inicia con la identificación del propósito, motivación u objetivo de la aplicación móvil a desarrollar, en este caso las personas involucradas deben tener absoluta comprensión del alcance de la aplicación, muchos enfoques para lograrlo se encuentran descritas de forma detalladas en metodologías tradicionales y agiles, algunos de los más difundidos son Rational Unified Process (IBM, 2015) y DSDM respectivamente.

Después de ganar la comprensión de cuál es la finalidad de la aplicación se debe identificar el público objetivo o potenciales usuarios. Las aplicaciones móviles sirven a diferentes propósitos, análogamente también a diferentes públicos, determinar el público objetivo de la aplicación y el número de potenciales usuarios presenta dificultades dadas en gran medida por la masificación de los dispositivos en los usuarios, se sugiere realizar actividades de establecimiento e identificación de Stakeholder de la aplicación, priorizadas en el usuario final o consumidor de la aplicación.

El entorno dinámico, acelerado y competitivo que caracteriza el desarrollo de software móviles, es quizás una de las principales dificultades que plantea un proyecto de esta naturaleza, en algunos casos estos proyectos parten de iniciativas enmarcadas en un ambiente de total incertidumbre, cuya característica es crear productos totalmente nuevos e inexistentes o que buscan atender nuevos nichos de mercado, estas clases de proyectos son categorizados como innovadores y disruptivos, generalmente a este tipo de iniciativas se les denomina Startup. (Blank & Dorf, 2012) Algunas estrategias y métodos ampliamente aceptados en la academia e industria que pueden contribuir al desarrollo de nuevos productos es el proceso de diseño de nuevos productos(NPD), (Ulrich & Eppinger, 2012) para afrontar el entorno dinámico y competitivo, el enfoque de generación de modelos de negocio o modelo de lienzo conocido comúnmente como el modelo Canvas, (Osterwalder & Pigneur, 2011) resulta ser una herramienta indispensable en esta fase, que puede contribuir a disminuir en alto grado la incertidumbre mediante una serie de análisis de los factores que influyen en la creación de productos, por otro lado un enfoque que recopila prácticas y métodos enfocados a la creación de productos exitosos y también enfatizando en la mitigación de la incertidumbre es el método Lean Startup. (Startup, 2012)

La compatibilidad multiplataforma presenta serios desafíos en el desarrollo móvil, las plataformas de desarrollo, software, herramientas, hardware y la tecnología disponibles del desarrollo móvil son de naturaleza fragmentada; lo anterior se traduce en mayor inversión de tiempo y recursos en la construcción de la aplicación.

La elección de la forma de implementar dentro del abanico existente para el desarrollo de las aplicaciones móviles supone una decisión crucial que determinara en gran medida las actividades a realizar en cada fase del desarrollo, dentro del

abanico de implementación se distinguen las aplicaciones web para móviles que aprovechan los navegadores web preinstalados en los dispositivos para su funcionamiento, que utilizan las tecnologías web estándar con algunas optimizaciones para la visualización de contenido, sin embargo no pueden acceder a las funcionalidades provistas en los dispositivos móviles como sensores que limitan la potencialidad de este tipo de implementaciones, para contrarrestar esta falencia en el mercado existen compañías que ofrecen alternativas de solución multiplataforma o implementaciones hibridas, basadas en la reunión de tecnologías utilizadas en los lenguajes de programación Web y partes de código permiten acceder a las interfaces de programación de aplicaciones nativas(API) (Palmieri, Singh, & Cicchetti, 2012), no obstante en la actualidad la tendencia es optar por el desarrollo nativo de aplicaciones debido al mejor desempeño, velocidad de procesamiento, aprovechamiento del hardware del dispositivo y la rica interfaz gráfica altamente valorada por los usuarios que proporcionan cada una de las plataformas de desarrollo (Voice of the Next-Generation Mobile Developer, Appcelerator / IDC, 2015). Estrategias para afrontar este desafío son reconocer las ventajas y desventajas que ofrecen cada una de las plataformas de desarrollo, para luego elegir cual(es) son adecuadas a los requerimientos de la aplicación, tiempo y presupuesto, restringir una aplicación en una sola plataforma supone seria limitaciones en el alcance en el público. Seguidamente y con estrecha conexión es recomendable realizar el ejercicio de explorar el mayor número posible de dispositivos que serán objetos de la aplicación, algunas de las cuestiones técnicas que se deben tener en cuenta son: versión del sistema operativo, características físicas tecnologías soportada por el dispositivos(poder de procesamiento, capacidad de almacenamiento, sensores disponibles, resolución y tamaña de la pantalla, entre otros) que permita conocer el posible desempeño de la aplicación en ejecución en los dispositivos.

Los requerimientos de las aplicaciones móviles cambian rápidamente, a menudo sobre el tiempo de entrega; definir y especificar cuáles son las funcionalidades de la aplicación móvil resulta ser una actividad esencial, quizás la más importante y critica de todo el ciclo de vida de desarrollo (Brooks, 1987). la obtención de requisitos o la ingeniería de requisitos intenta proveer alternativas de solución a uno delos principales problemas que enfrenta la ingeniera de software que es responder la pregunta de qué se va a construir, en este ámbito se distinguen dos tipos de requisitos: requisitos funcionales relacionados con lo que el sistema debe hacer y los requisitos no funcionales los cuales se subdividen en: desempeño, restricciones de diseño, Interfaz externa y los atributos de calidad (Pohl, 1993). La naturaleza cambiante y ambigua de los requerimientos sugiere la adopción de una estrategia enfocada a la adaptación y el cambio, características que es uno de los pilares que soportan las metodologías agiles, que proponen una mayor intervención de los involucrados en la definición de los requerimientos, apoyados por un proceso iterativo e incremental enfocado en rápidos y constantes revisiones de los requerimientos. Una vez se han definido y especificado los requerimientos de la aplicación por lo general son descompuestos, algunos enfoques conocidos son los

casos de uso de RUP y UP, las historias de usuario utilizadas en XP o la descomposición en funcionalidades de FDD.

El Time to Market como se denomina en Marketing, o el tiempo presupuestado entre la concepción de la idea de construir la aplicación móvil hasta su distribución, suelen ser limitados en su horizonte de tiempo, por lo general con pocos meses de plazo para entrega, por consiguiente es conveniente definir un cronograma tareas, recursos y presupuesto dentro del cual estará enmarcado el desarrollo del proyecto, para contrarrestar esta presión la adopción de prácticas agiles como XP, ASD Y SCRUM son recomendadas, uno de las características importantes de estas metodologías es su énfasis en la constantes integración y construcción de incrementos de software.

Analizar el nivel de criticidad de la aplicación a construir es necesario, ya que determina el nivel de exhaustividad de las actividades futuras de las siguientes fases, además de condicionar la complejidad de las pruebas a realizar. El examen que debe realizar los responsables de la aplicación es determinar el nivel de tolerancia a una falla no detectada, que pueden ir desde un nivel aceptable que solo repercuta en la experiencia de uso del usuario hasta el extremo de representar consecuencias perjudiciales como pueden presentarse en las aplicaciones bancarias o encargadas de monitorear la salud.

Las restricciones físicas intrínsecas presentes en los dispositivos móviles plantean serios desafíos de usabilidad, definir una estrategia de valoración de usabilidad en esta fase sea evidenciado como altamente beneficiosa, así lo evidencia el modelo en espiral para el desarrollo móvil proponen definir una serie de métricas de usabilidad y el examen regular de cada una en el transcurso del ciclo de vida del desarrollo. Los aspecto físicos como el limitado tamaño de visualización de contenido y los diferentes medios de entrada de datos de los dispositivos representan desafíos para los desarrolladores que deben replantear estrategias para un lado determinar el contenido estrictamente necesario que cubra las necesidades de los usuarios y por otro lado determinar los métodos de entrada más eficiente dentro del amplio rango disponible como teclado físico, teclado táctil, los cada vez más populares entradas basadas en voz; otros mecanismos de entrada puede darse por otras fuentes como los derivados de las funcionalidades provistas en los dispositivos que interactúan con el entorno como los sensores que pueden recibir información de las redes o lectura de información de imágenes por medio de la cámara. (Williamson, 2012)

La Interfaz gráfica de usuario identificada como requisito primordial en la experiencia de usuario de las aplicaciones móviles, resultante entonces importante crear diseños preliminares de la aplicación mediante dibujos, bocetos o prototipos en papel con suficiente nivel de detalle, que posibiliten traducirlos en prototipos de interfaz gráfica de baja fidelidad en la plataforma de desarrollo, para obtener mayor exactitud en la definición de la Interfaz gráfica. Uno de los aspectos conexos a la interfaz gráfica que resulta obvio en las aplicaciones es el factor de interacción y visualización significativamente reducido que ofrecen comparado con otro tipo de software. Los dispositivos móviles usualmente proveen un área comprendida en pocas pulgadas para presentar el contenido de la aplicación, por lo anterior la forma

de presentar el contenido en las aplicaciones móviles debe ser optimizado y priorizado con la información relevante que el usuario necesite en el punto donde se encuentre dentro de la aplicación. (Williamson, 2012)

Fase Diseño

la fase de diseño se preocupa de la especificación detallada de los componentes de la aplicación móvil, si bien a este punto ya debería estar totalmente establecido los componentes se ha evidenciado que el prototipado contribuye a mejorar el diseño, crear un prototipo visual de las pantallas de la aplicación y algunas interacciones básicas de la navegación dentro de ella, otorgando la posibilidad de posibles usuarios interactúen con ella, proporciona valiosa información en el refinamiento del diseño, que pude reflejarse en ahorro de recursos y tiempo en la fases posteriores como la codificación.

La definición de la arquitectura juega un principal rol en esta fase, su propósito es definir los componentes y sus interacciones que conforman la aplicación, la vía recomendada para alcanzarlos es abogar por el principio del diseño simple proveniente de XP, que permita diseñar una simple pero adecuada arquitectura para la aplicación móvil, la correcta descomposición modular de las funcionalidades permitirá que estas puedan ser consistente entre plataformas, el propósito es que los módulos que sean construidos sea idénticos en comportamiento, lógica y reglas de negocio que se ajusten a las características impuestas por cada plataforma.

Las plataformas de desarrollo para software móviles son en su mayoría orientadas a objetos, la tecnología orientada a objetos es uno de los paradigmas para analizar y construir software ampliamente difundido en las últimas décadas, es totalmente coherente utilizar diseño orientado a objetos en el desarrollo de las aplicaciones móviles, una de las ventajas de trabajar bajo este paradigma es la de disponer de un lenguaje de modelado de sistemas de común acuerdo por la comunidad como el lenguaje unificado de modelamiento (UML) (Object Management Group, 2015), que permite especificar mediante modelos visuales la estructura, el comportamiento y la arquitectura de la aplicación.

Fase Implementación

La fase de implementación tiene como propósito codificar la aplicación, dentro de las características identificadas se distinguen que las aplicaciones móviles son desarrolladas por pequeños grupos, el tamaño de la aplicación es pequeño y en ciclos cortos, características análogas que se encuentran en las metodologías agiles y que pueden ser adoptadas en el proceso de implementación, como el desarrollo iterativo e incremental, integración continua y entrega regular de incrementos o funcionalidades de la aplicación. Algunas prácticas clave de algunas metodologías agiles podrían aportar gran apoyo en la fase de construcción, la descomposición de las funcionalidades de la aplicación como prescribe FDD, ciclos de desarrollo en

Sprint proveniente de Scrum, las prácticas de desarrollo de XP son valiosas en algunos aspectos claves como las pruebas unitarias, la optimización del código mediante la refactorización, la integración continua del código que posibilite la liberación parcial de incrementos de la aplicación que puedan estar disponibles para revisión y pruebas de los clientes.

Fase Pruebas

Su objetivo es probar el software desarrollado ¿La aplicación funciona correctamente en los dispositivos esperados? ¿Probar que lo desarrollado cumple con los requerimientos definidos?

El entorno móvil presenta una serie de características que infieren en el funcionamiento de una aplicación de este tipo, algunas dadas por el entorno como las redes móviles de transmisión de datos, otras por cuestiones de físicas técnicas y aquellas intrínsecas como el tamaño del dispositivo, lo que repercute en complejidad al momento de ejecutar las pruebas por los múltiples factores a considerar durante su ejecución, un escenario típico de pruebas para software móvil podrían darse el siguiente curso de acción: definición los diferentes casos de prueba manuales y en lo posible casos de pruebas automatizados; a este punto del desarrollo se deben haber sido superados con éxito todos las pruebas unitarias de la fase anterior, por consiguiente se debe complementar con la realización de casos de prueba a los módulos, luego a la integración de los módulos y finalmente al conjunto total de la solución.

Una de las limitaciones encontradas es el poco soporte de pruebas automatizadas en las plataformas, no obstante, en la medida que sea posible ejecutar pruebas automatizadas; después es necesario someter la aplicación a pruebas en su funcionamiento, primero en un ambiente simulado en los emuladores y simuladores disponibles de cada plataforma, que ofrecen una manera rápida, configurable y económica de comprobar el funcionamiento, segundo en un ambiente que refleja las condiciones normales de despliegue que son los dispositivos, en ellos podremos probar un mayor número de características técnicas entre las que encontramos todo el conjunto de sensores soportados en los dispositivos(GPS, Giroscopio, Acelerómetro, Termómetro, Rotación, Navegación), conectividad a las redes de datos y ubicación, cuestiones de seguridad, desempeño, funcionalidad, interoperabilidad entre los dispositivos y versiones de sistemas operativos, adicional pueden adelantarse pruebas de usabilidad, finalmente como paso final pruebas de aceptación de los usuarios finales, regularmente después de realizarlas, suele venir acompañado de sugerencias como cambios o correcciones, los cuales una vez realizados deben pasar nuevamente por pruebas de integración, en emuladores y físicas, para terminar nuevamente en la pruebas de aceptación del usuario final.

Fase Transición

Su objetivo es asegurar el correcto funcionamiento de la aplicación una vez entregada a los usuarios y determinar el canal de distribución. ¿Cómo se distribuye la aplicación? ¿La aplicación presenta mecanismos para notificar el funcionamiento o fallas?

El software en esta fase se encuentra totalmente construido y ha sido aprobado por el cliente, dos grupos principales actividades se realizan en esta fase; la primera refiere a distribución y marketing, y la segunda corresponde a soporte y manteamiento de la aplicación una vez ha sido puesta al servicio de los usuarios.

La distribución de las aplicaciones móviles puede darse por medio de dos principales canales, uno es orientado a la masificación y obtención del mayor número de suscriptores que son las tiendas de aplicaciones, las acciones realizadas a este medio de distribución son mayormente enfocadas a cuestiones de marketing, seguimiento de suscriptores, campañas entre otros con el propósito de aumentar la cuota de suscriptores, otro medio corresponde a las aplicaciones desarrolladas a la medida o empresariales, donde cumplen una específica función para el cliente que la solicita, suele caracterizarse por ser privativa en su uso y condiciona una relación estrecha entre el cliente y el grupo desarrollador.

Una vez la aplicación ha sido puesta al servicio de los usuarios es necesario continuar con todas las acciones para monitorear el funcionamiento de la aplicación, las siguientes son estrategias sugeridas para contribuir a un eficaz soporte de la aplicación: optimizar el uso de la memoria utilizada en el funcionamiento de las aplicaciones, las limitaciones físicas de los dispositivos pueden ocasionar fallos en la ejecución, proporcionar mecanismos de retroalimentación a los usuarios, habilitar el reporte de daños o sugerencias de los usuarios en las tiendas de aplicaciones o las páginas web de las compañías desarrolladoras, incorporar la generación de reportes automatizados del desempeño de la aplicación que sean notificados al grupo de desarrolladores. Es común que las aplicaciones móviles son consta mente mejoradas ya sea por correcciones de fallos, la incorporación de nuevas funcionalidades o actualizaciones derivadas de cambio de versión de la plataforma ,mediante la liberación frecuente de nuevas versiones, es importante realizarlas utilizando las configuraciones de actualización que proporcione la plataforma, y replicarlos a través de la tienda de aplicaciones o en el caso de las aplicaciones a la medida asegurar la actualización de todos los dispositivos. (Flora, Wang, & Chande, 2014)

CAPITULO 5
EVALUACIÓN DE LA PROPUESTA

1. DESCRIPCIÓN DEL EXPERIMENTO

Con base en los métodos empíricos para la evaluación de aportes en Ingeniería de software denominados ingeniería de software experimental detallado en (Basili, The Experimental Paradigm in Software Engineering, 1993) (Basili, The Role of Controlled Experiments in Software Engineering Research, 2007) (Easterbrook, Singer, Storey, & and Damian, 2007), se propone realizar un experimento para evaluar el marco de trabajo propuesto.

El experimento consistió en la realización de un caso de estudio con la participación de estudiantes avanzados de la carrera de Gerencia de sistemas de información en salud de la universidad de Antioquia; donde el propósito era generar una aplicación orientada al sector salud, la cual está vinculada a un programa en ejecución de la facultad nacional de salud pública, adscrita a la universidad de Antioquia, donde se planteó construir un prototipo funcional de una aplicación móvil que permita consultar y recoger información de los pacientes en campo por parte del personal sanitario encargado de realizar seguimiento dentro del programa. Se estableció que el experimento no forma parte del plan de estudios del programa, por consiguiente no tiene ningún tipo de repercusión en el curso y su calificación en los estudiantes, por otro lado el programa accedió a apoyar la realización del experimento aportando el material necesario para ejecutar las actividades y la información necesaria para generar los requerimientos esenciales del prototipo. Los productos generados después de la ejecución del experimento fueron almacenados en el repositorio general de proyectos de la facultad, los cuales por reglamentación y restricciones intrínsecas del programa no pueden ser adjuntados en el presente documento.

La presentación, coordinación y ejecución del experimento estuvo a cargo de un profesor de ingeniería de software, el asesor de la tesis, el autor de la tesis y un monitor de sistemas, para un total de 4 participantes responsables de guiar todo el proceso; el experimento fue presentado en la modalidad de curso-taller a los estudiantes como un complemento de tipo profundización en desarrollo de software móvil.

Los grupos participantes del experimento fueron conformados por estudiantes del programa de Gerencia de Sistemas de Información en Salud de la Universidad de Antioquia de la cohorte correspondiente al segundo semestre del 2014, provenientes del curso de ingeniería de software del sexto semestre de carrera. Se conformaron dos grupos: el primer grupo de 25 estudiantes corresponden al curso de ingeniería de software 1, el segundo grupo se conforma de 22 estudiantes del curso de ingeniería de software del mismo semestre pero del grupo 2.

Una vez definidos los grupos que participaran el experimento, se realizó una sección introductoria a ambos grupos donde como propósito fue introducir la finalidad de la aplicación, una vez familiarizada la motivación de la aplicación, se procede a describir a los grupos de estudiantes la aplicación a desarrollar, se comparten instrucciones básicas para el proceso de desarrollo, se delimita un cronograma base y se establecen algunos recursos disponibles para su elaboración.

Habiendo definido los grupos, recursos y cronograma, se dio paso a iniciar el proceso de desarrollo, a continuación se describe cada uno de los pasos realizados con cada uno de los grupos:

Grupo 1

1. Se comparte instrucciones sobre la metodología o método a utilizar durante todo el desarrollo, los estudiantes participantes contaban con fundamentos teóricos en varios metodologías de desarrollo provenientes del curso de ingeniería de software impartido, entre los que se distingue RUP, aprovechando que los estudiantes poseen conocimientos sobre esta metodología se decidió por proponer como metodología oficial para el desarrollo de la aplicación la versión ágil de RUP denominada UP, posteriormente se realizó un breve sección introductoria sobre UP.
2. Una vez definida la metodología los estudiantes desarrollan la aplicación según los pasos que prescribe UP.
3. Terminada la aplicación utilizando el enfoque UP, se realiza una sección introductoria con el propósito de explicar el marco de trabajo para el desarrollo de software para dispositivos móviles.
4. Luego se solicita al grupo desarrollador realizar de nuevo la aplicación móvil, pero en este caso siguiendo las indicaciones del marco de trabajo.
5. Terminada la aplicación bajo el enfoque del marco de trabajo, se solicitó a los integrantes del grupo diligenciar un cuestionario con el fin obtener información sobre los dos enfoques utilizados durante las sesiones de desarrollo.

Grupo 2

1. Se realiza una sección de introducción donde se explica el marco de trabajo para el desarrollo de software móvil, una vez generalizado en el grupo, se proporciona la indicación de realizar la aplicación basada en el marco de trabajo propuesto.
2. Los integrantes del grupo desarrollan la aplicación siguiendo las actividades recomendadas del marco de trabajo para el desarrollo de software móvil.
3. Terminada la aplicación utilizando el enfoque del marco de trabajo, se realiza una sesión introductoria con el propósito de explicar la metodología de desarrollo UP, aprovechando los conocimientos en la metodología RUP que poseen los estudiantes.
4. Los integrantes desarrollan la aplicación móvil siguiendo las indicaciones que prescribe UP.

5. Terminada la aplicación bajo el enfoque UP, se solicita a los integrantes del grupo diligenciar un cuestionario con el fin de obtener información sobre los dos enfoques utilizados durante las sesiones de desarrollo.

2. RESULTADOS DE LA APLICACIÓN DEL EXPERIMENTO

Con el objetivo de conocer el aporte del marco de trabajo propuesto en el desarrollo de software para dispositivos móviles, una encuesta fue conducida entre los integrantes de los dos grupos participantes, el cuestionario aplicado consta de 15 preguntas directamente enfocadas en los principales objetivos a alcanzar en el desarrollo de aplicaciones móviles (Ver anexos), a continuación se muestran los resultados obtenidos para los dos grupos para cada una de las preguntas:

En la pregunta ¿El marco de trabajo propuesto es fácil de entender?, según la figura 5-1 y la tabla 6, los integrantes del primer grupo considera en un 72% que es totalmente entendible; mientras que en un 24% considera que es entendible, con posibilidades de mejora.

Tabla 2. Resultados pregunta 1.

El marco de trabajo propuesto Es fácil de entender	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	1	4
4. De acuerdo	6	24
5. Fuertemente de acuerdo	18	72
Total	25	100

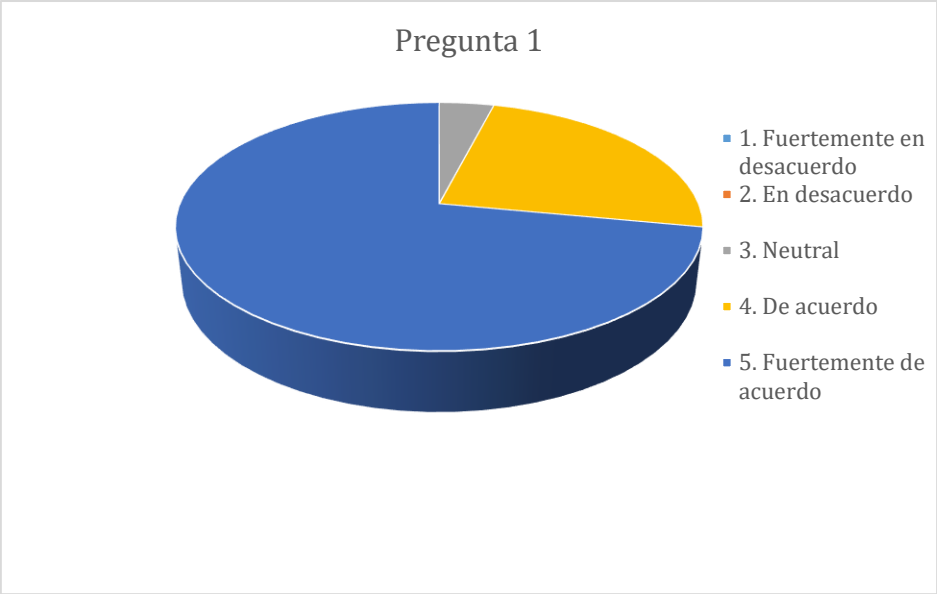


Figura 5-1. Resultados pregunta 1.

En la pregunta ¿El marco de trabajo propuesto es fácil de utilizar?, según la figura 5-2 y la tabla 7, los integrantes del primer grupo considera en un 68% que es totalmente fácil de utilizar; mientras que en un 32% considera que es fácil de utilizar, pero con posibilidades de mejora.

Tabla 3.Resultados pregunta 2.

El marco de trabajo propuesto Es fácil de utilizar	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	8	32
5. Fuertemente de acuerdo	17	68
Total	25	100

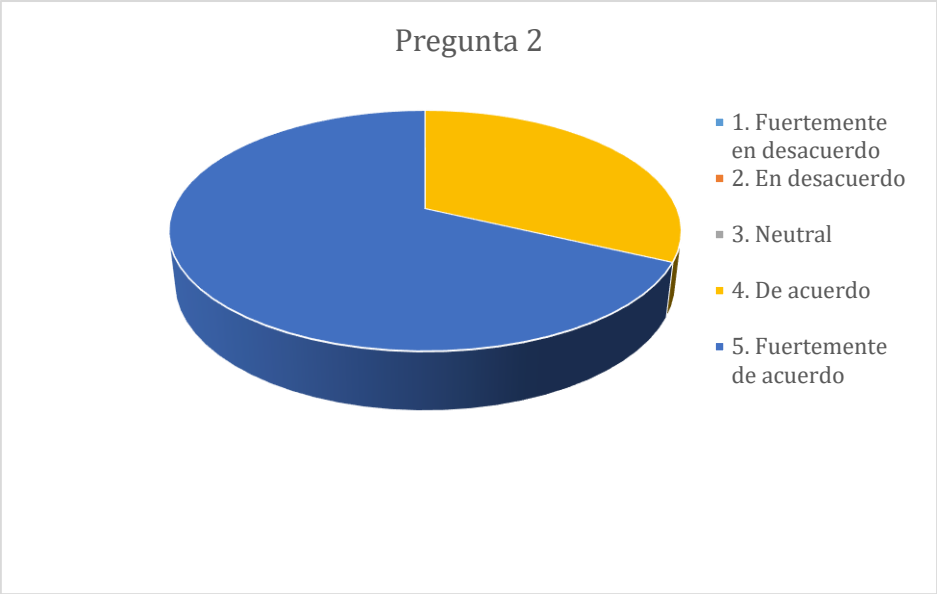


Figura 5-2. Resultados pregunta 2.

En la pregunta ¿El marco de trabajo propuesto cubre todos los aspectos del ciclo de vida de un desarrollo de software móvil?, según la figura 5-3 y la tabla 8, los integrantes del primer grupo considera en un 92% que da cobertura total del ciclo de vida de desarrollo; mientras que en un 8% considera que presta cobertura del ciclo de vida de desarrollo, pero con posibilidades de mejora.

Tabla 4. Resultados pregunta 3.

El marco de trabajo propuesto Cubre todos los aspectos del ciclo de vida de un desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	2	8
5. Fuertemente de acuerdo	23	92
Total	25	100

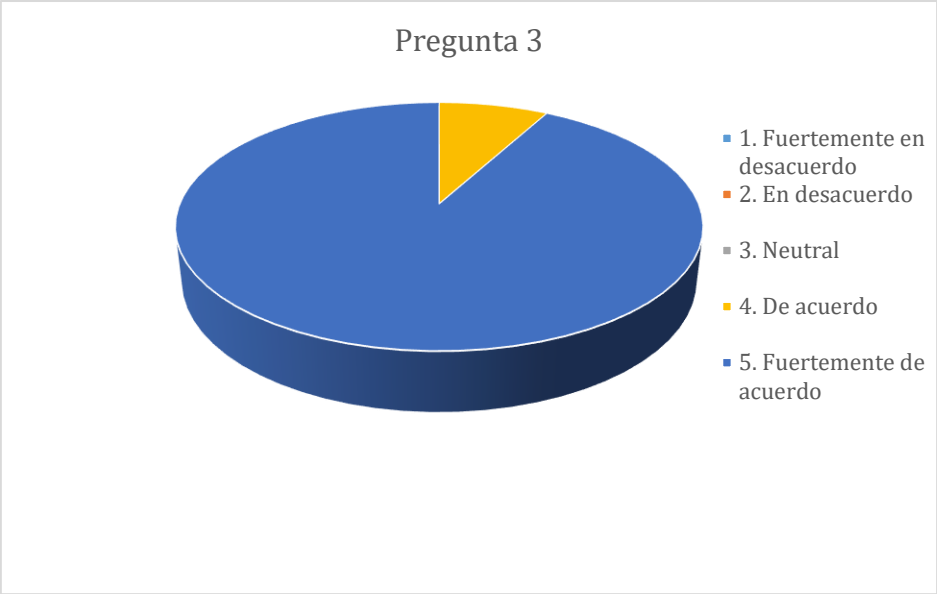


Figura 5-3. Resultados pregunta 3.

En la pregunta ¿El marco de trabajo propuesto proporciona estrategias que contribuyen a mejorar la calidad de la aplicación?, según la figura 5-4 y la tabla 9, los integrantes del primer grupo considera en un 60% que el marco proporciona totalmente estrategias enfocadas a la calidad; mientras que en un 40% considera que están en acuerdo con las estrategias propuestas, pero con posibilidades de mejora.

Tabla 5. Resultados pregunta 4.

El marco de trabajo propuesto Proporciona estrategias que contribuyen a mejorar la calidad de la aplicación	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	10	40
5. Fuertemente de acuerdo	15	60
Total	25	100

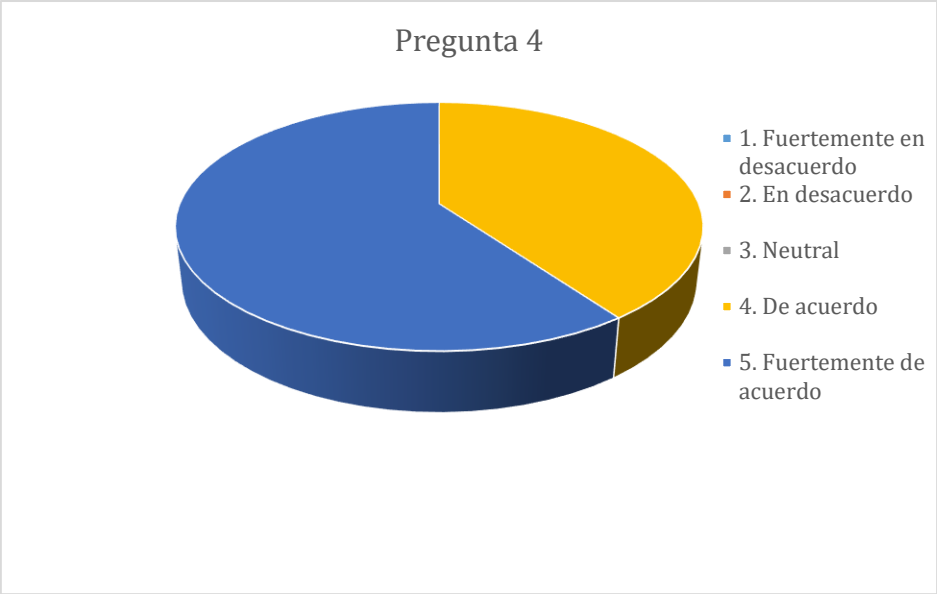


Figura 5-4. Resultados pregunta 4.

En la pregunta ¿El marco de trabajo propuesto puede ser extendido o adaptado a cualquier desarrollo de software móvil?, según la figura 5-5 y la tabla 10, los integrantes del primer grupo considera en un 84% que el marco es totalmente extensible o adaptable en cualquier proyecto de desarrollo móvil; mientras que en un 16% considera que es adaptable o extensible, pero con posibilidades de mejora.

Tabla 6. Resultados pregunta 5.

El marco de trabajo propuesto Puede ser extendido o adaptado a cualquier desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	4	16
5. Fuertemente de acuerdo	21	84
Total	25	100

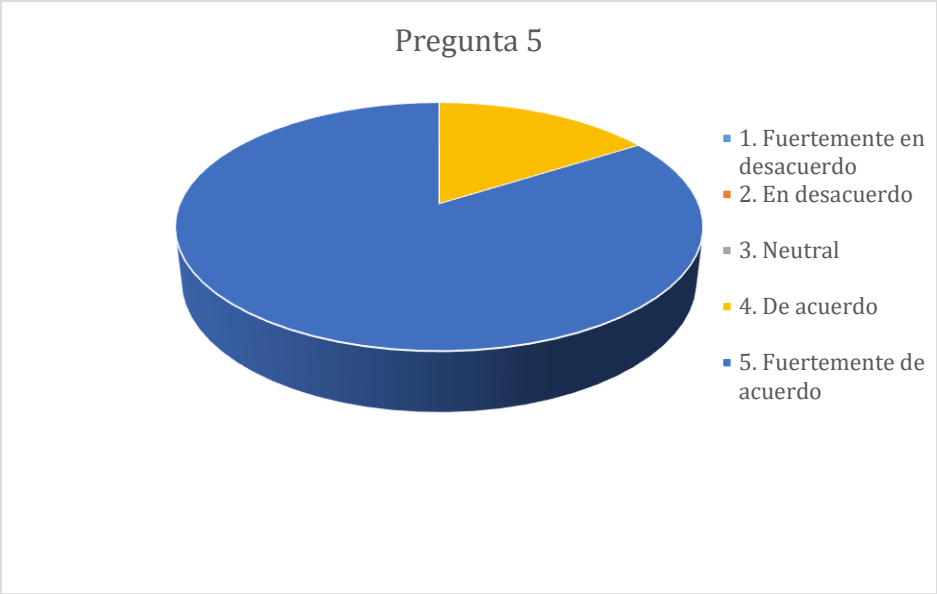


Figura 5-5. Resultados pregunta 5.

En la pregunta ¿El marco de trabajo propuesto basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que el marco de trabajo propuesto complemento o apoyo el método de desarrollo propuesto para la construcción de la aplicación?, según la figura 5-6 y la tabla 11, los integrantes del primer grupo considera en un 84% el marco complemento totalmente el método desarrollo propuesto; mientras que en un 16% considera que lo complemento, pero con posibilidades de mejora.

Tabla 7. Resultados pregunta 6.

El marco de trabajo propuesto Basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que el marco de trabajo propuesto complemento o apoyo el método de desarrollo propuesto para la construcción de la aplicación	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	4	16
5. Fuertemente de acuerdo	21	84
Total	25	100

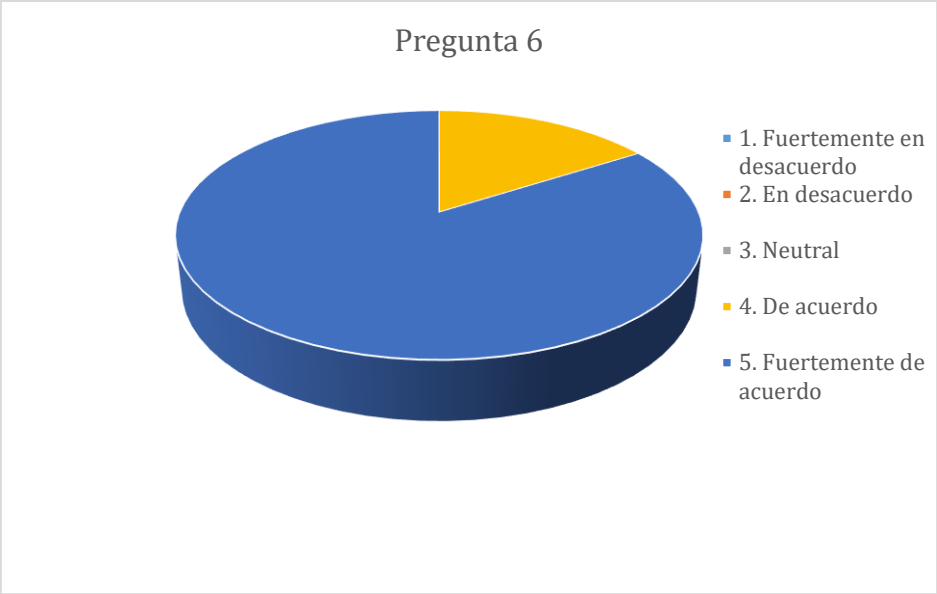


Figura 5-6. Resultados pregunta 6.

En la pregunta ¿El marco de trabajo propuesto contribuye a disminuir el tiempo de entrega de la aplicación móvil?, según la figura 5-7 y la tabla 12, los integrantes del primer grupo considera en un 80% que contribuye totalmente a disminuir el tiempo de entrega de una aplicación móvil; mientras que en un 20% considera que contribuye en la eficiencia del tiempo de entrega, pero con posibilidades de mejora.

Tabla 8. Resultados pregunta 7.

El marco de trabajo propuesto Contribuye a disminuir el tiempo de entrega de la aplicación móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	5	20
5. Fuertemente de acuerdo	20	80
Total	25	100

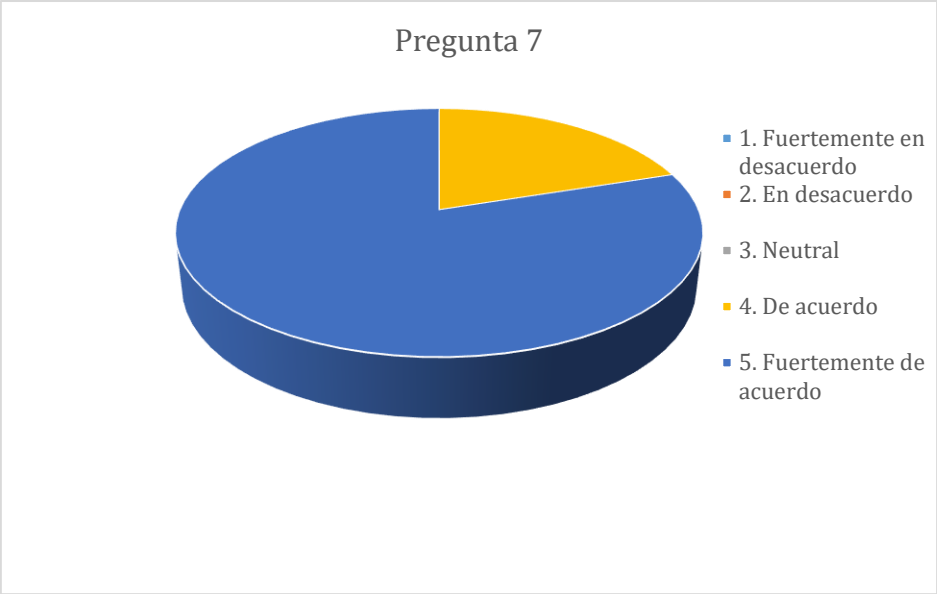


Figura 5-7. Resultados pregunta 7.

En la pregunta ¿El marco de trabajo propuesto las actividades recomendadas para el desarrollo de software para móviles son entendibles y claras?, según la figura 5-8 y la tabla 13, los integrantes del primer grupo considera en un 64% que las actividades propuesta son totalmente entendibles y claras; mientras que en un 36% considera que son entendibles, pero con posibilidades de mejora.

Tabla 9.Resultados pregunta 8.

El marco de trabajo propuesto las actividades recomendadas para el desarrollo de software para móviles son entendibles y claras	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	9	36
5. Fuertemente de acuerdo	16	64
Total	25	100

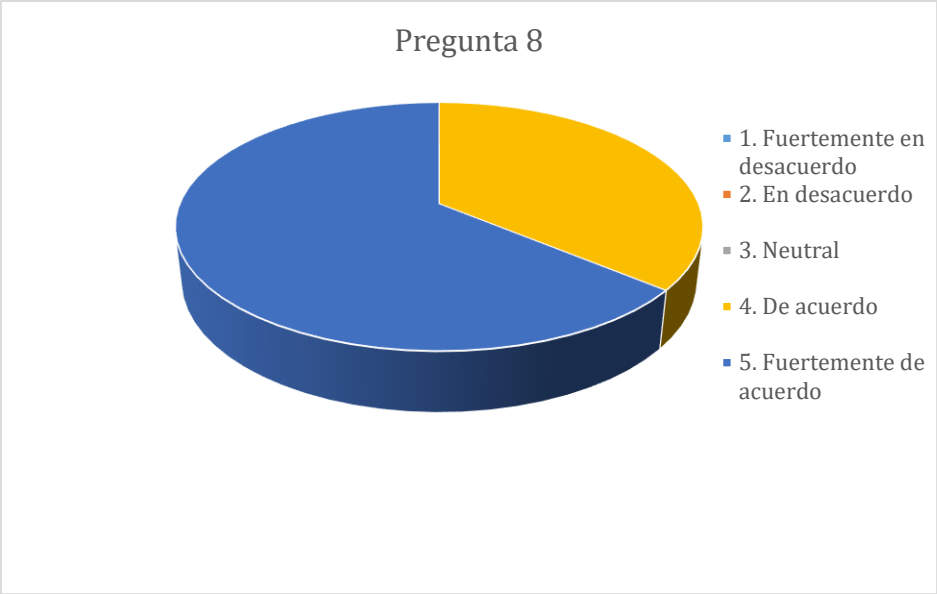


Figura 5-8. Resultados pregunta 8.

En la pregunta ¿El marco de trabajo propuesto proporciona conocimiento sobre el dominio de desarrollo de software para dispositivos móviles?, según la figura 5-9 y la tabla 14, los integrantes del primer grupo considera en un 72% que el marco proporciona un total entendimiento sobre el dominio de software para móviles; mientras que en un 28% considera que proporciona conocimiento sobre el dominio, pero con posibilidades de mejora.

Tabla 10. Resultados pregunta 9.

El marco de trabajo propuesto Proporciona conocimiento sobre el dominio de desarrollo de software para dispositivos móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	7	28
5. Fuertemente de acuerdo	18	72
Total	25	100

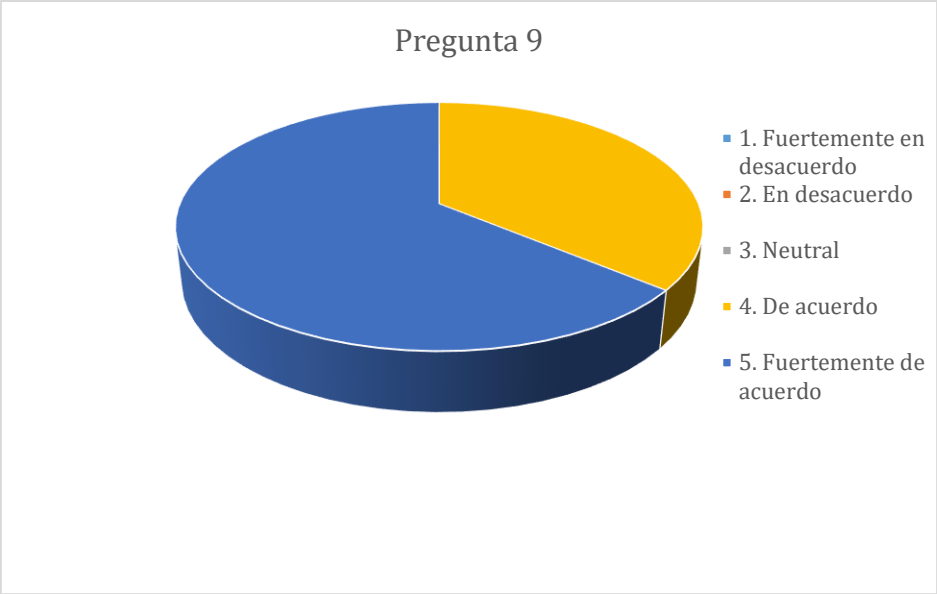


Figura 5-9. Resultados pregunta 9.

En la pregunta ¿El marco de trabajo propuesto las actividades propuestas para alcanzar los principales objetivos de una aplicación móvil son los adecuados?, según la figura 5-10 y la tabla 15, los integrantes del primer grupo considera en un 68% que las actividades propuestas son totalmente adecuadas para alcanzar los objetivos esperados en el desarrollo móvil; mientras que en un 32% considera que las actividades son adecuadas, pero con posibilidades de mejora.

Tabla 11. Resultados pregunta 10.

El marco de trabajo propuesto las actividades propuestas para alcanzar los principales objetivos de una aplicación móvil son los adecuados	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	8	32
5. Fuertemente de acuerdo	17	68
Total	25	100

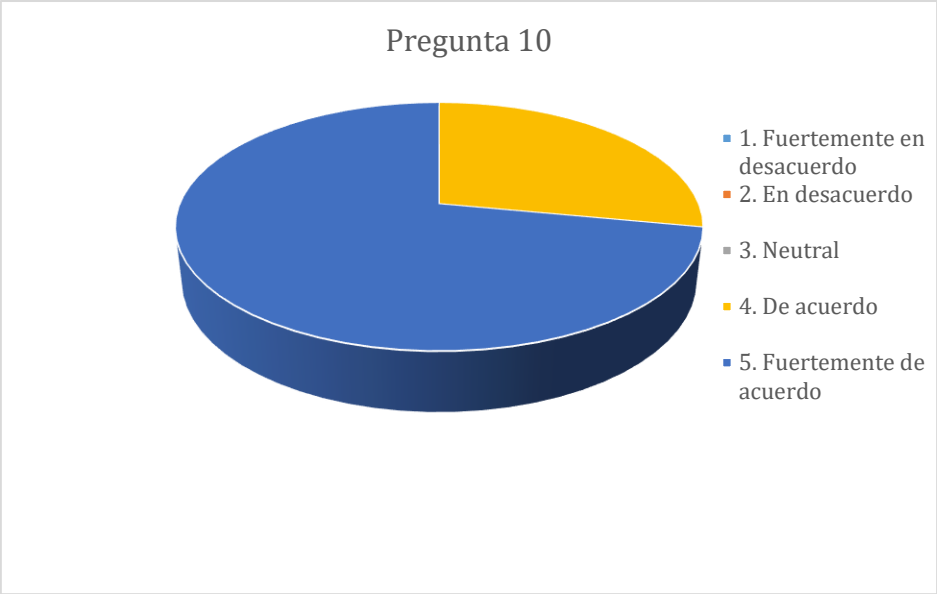


Figura 5-10. Resultados pregunta 10.

En la pregunta ¿El marco de trabajo propuesto presenta mayor conveniencia en relación a otras propuestas para el desarrollo de software móvil?, según la figura 5-11 y la tabla 16, los integrantes del primer grupo considera en un 84% que resulta totalmente más conveniente en relación a otras propuestas para el desarrollo móvil; mientras que en un 16% considera que es conveniente frente a otras propuestas, pero con posibilidades de mejora.

Tabla 12. Resultados pregunta 11.

El marco de trabajo propuesto Presenta mayor conveniencia en relación a otras propuestas para el desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	4	16
5. Fuertemente de acuerdo	21	84
Total	25	100

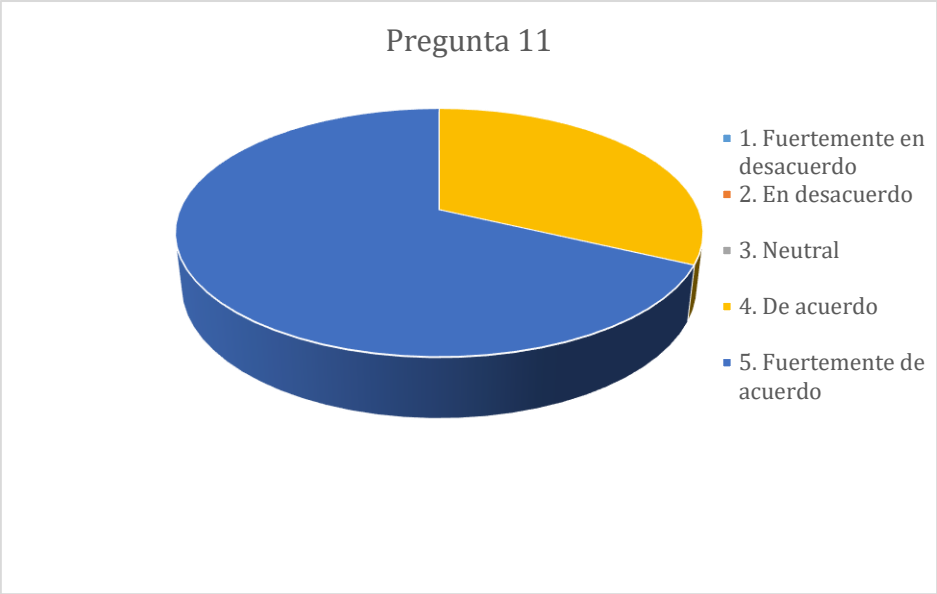


Figura 5-11. Resultados pregunta 11.

En la pregunta ¿El marco de trabajo propuesto es un sustito de otras propuestas para el desarrollo de software móvil?, según la figura 5-12 y la tabla 17, los integrantes del primer grupo considera en un 76% que es un total sustituto en relaciona otras propuestas de desarrollo; mientras que en un 24% considera que es sustituto, pero con posibilidades de mejora.

Tabla 13. Resultados pregunta 12.

El marco de trabajo propuesto Es un sustito de otras propuestas para el desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	6	24
5. Fuertemente de acuerdo	19	76
Total	25	100

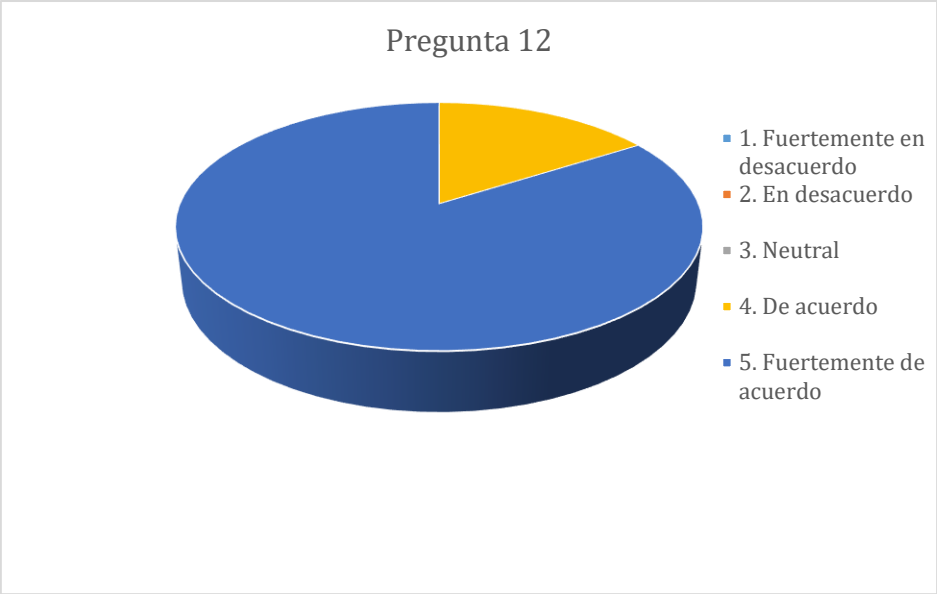


Figura 5-12. Resultados pregunta 12.

En la pregunta ¿El marco de trabajo propuesto contribuye a mejorar el desempeño de los desarrolladores que construyen software para dispositivos móviles?, según la figura 5-13 y la tabla 18, los integrantes del primer grupo considera en un 88% que contribuye a mejorar totalmente de mejorar el desempeño de los desarrolladores que construyen software para móviles; mientras que en un 12% considera que contribuye en el desempeño, pero con posibilidades de mejora.

Tabla 14. Resultados pregunta 13.

El marco de trabajo propuesto Contribuye a mejorar el desempeño de los desarrolladores que construyen software para dispositivos móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	3	12
5. Fuertemente de acuerdo	22	88
Total	25	100

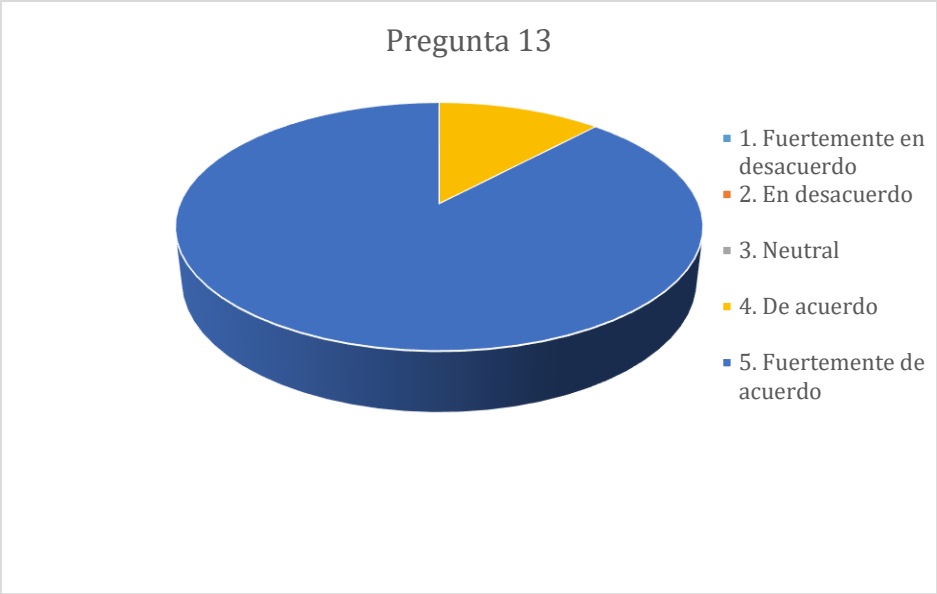


Figura 5-13. Resultados pregunta 13.

En la pregunta ¿El marco de trabajo propuesto complementa a otros enfoques de desarrollo utilizados en este tipo de desarrollo o el utilizado en la caso de estudio?, según la figura 5-14 y la tabla 19, los integrantes del primer grupo considera en un 88% que complementa totalmente a otros enfoques utilizados en el desarrollo de software para móviles; mientras que en un 12% considera que es un complemento, pero con posibilidades de mejora.

Tabla 15. Resultados pregunta 14.

El marco de trabajo propuesto Complementa a otros enfoques de desarrollo utilizados en este tipo de desarrollo o el utilizado en la caso de estudio	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	3	12
5. Fuertemente de acuerdo	22	88
Total	25	100

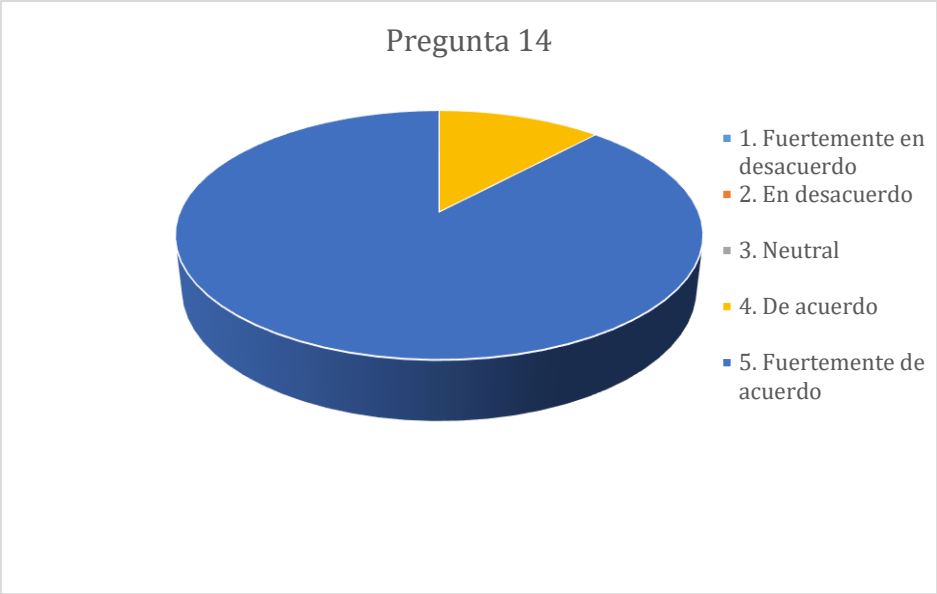


Figura 5-14. Resultados pregunta 14.

En la pregunta ¿El marco de trabajo propuesto es recomendable para su adopción por desarrolladores de software para móviles?, según la figura 5-15 y la tabla 20, los integrantes del primer grupo considera en un 92% que es totalmente recomendable para su adopción por parte de los desarrolladores de software móvil; mientras que en un 8% considera que es recomendable su adopción, pero con posibilidades de mejora.

Tabla 16. Resultados pregunta 15.

El marco de trabajo propuesto Es recomendable para su adopción por desarrolladores de software para móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	2	8
5. Fuertemente de acuerdo	23	92
Total	25	100

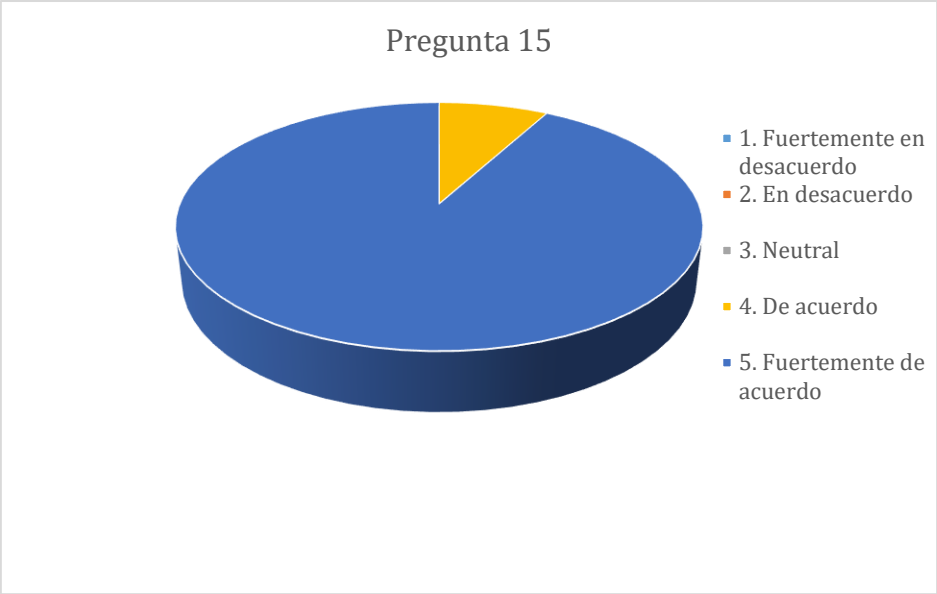


Figura 5-15. Resultados pregunta 15.

En la siguiente sección se muestran los resultados de la aplicación de la encuesta al segundo grupo:

En la pregunta ¿El marco de trabajo propuesto es fácil de entender?, según la figura 5-16 y la tabla 21, los integrantes del segundo grupo considera en un 86% que es totalmente entendible; mientras que en un 14% considera que es entendible, con posibilidades de mejora.

Tabla 17. Resultados pregunta 1.

El marco de trabajo propuesto Es fácil de entender	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	3	14
5. Fuertemente de acuerdo	19	86
Total	22	100

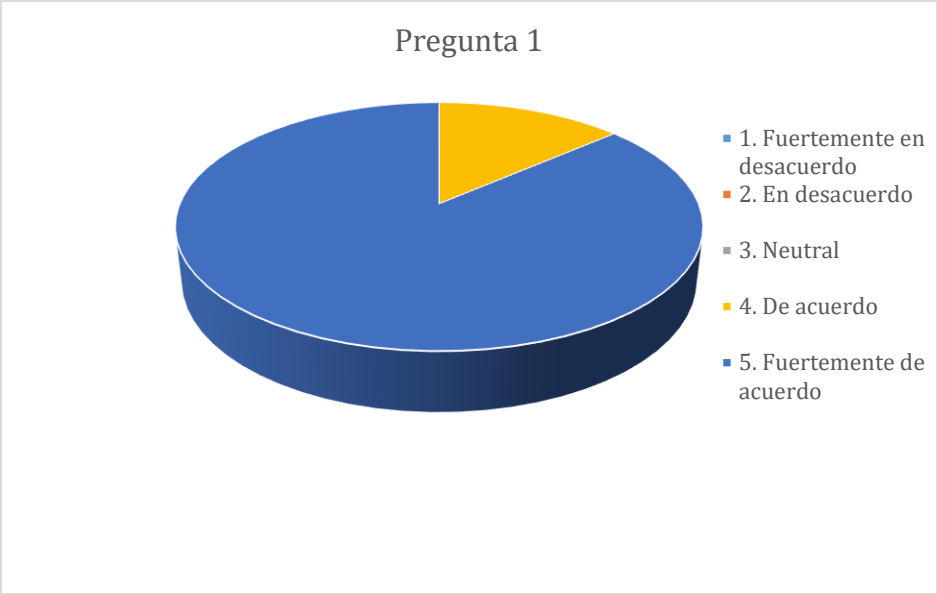


Figura 5-16. Resultados pregunta 1.

En la pregunta ¿El marco de trabajo propuesto es fácil de utilizar?, según la figura 5-17 y la tabla 22, los integrantes del segundo grupo considera en un 77% que es totalmente fácil de utilizar; mientras que en un 23% considera que es fácil de utilizar, pero con posibilidades de mejora.

Tabla 18. Resultados pregunta 2.

El marco de trabajo propuesto es fácil de utilizar	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	5	23
5. Fuertemente de acuerdo	17	77
Total	22	100

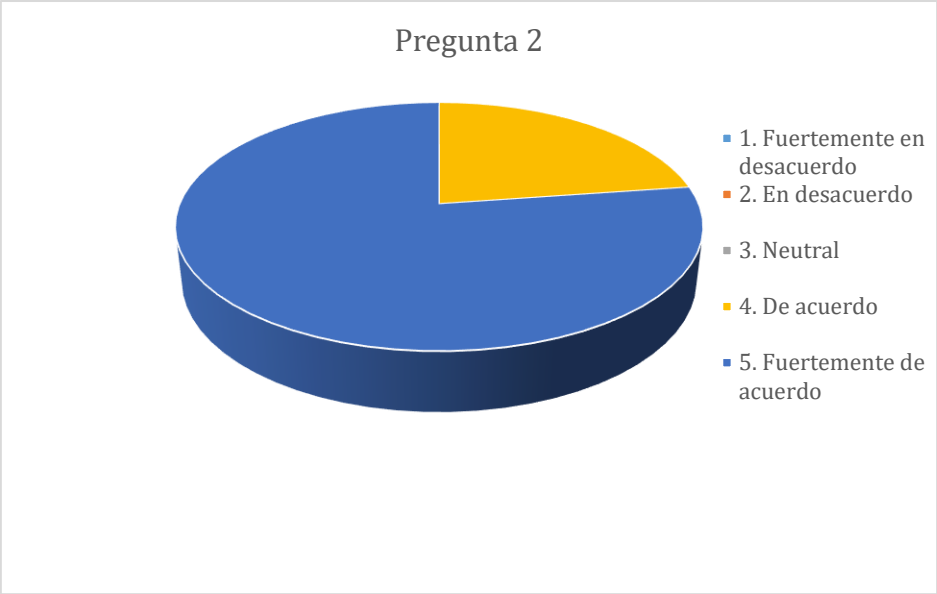


Figura 5-17.Resultados pregunta 2.

En la pregunta ¿El marco de trabajo propuesto cubre todos los aspectos del ciclo de vida de un desarrollo de software móvil?, según la figura 5-18 y la tabla 23, los integrantes del segundo grupo considera en un 68% que da cobertura total del ciclo de vida de desarrollo; mientras que en un 32% considera que presta cobertura del ciclo de vida de desarrollo, pero con posibilidades de mejora.

Tabla 19. Resultados pregunta 3.

El marco de trabajo propuesto cubre todos los aspectos del ciclo de vida de un desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	7	32
5. Fuertemente de acuerdo	15	68
Total	22	100

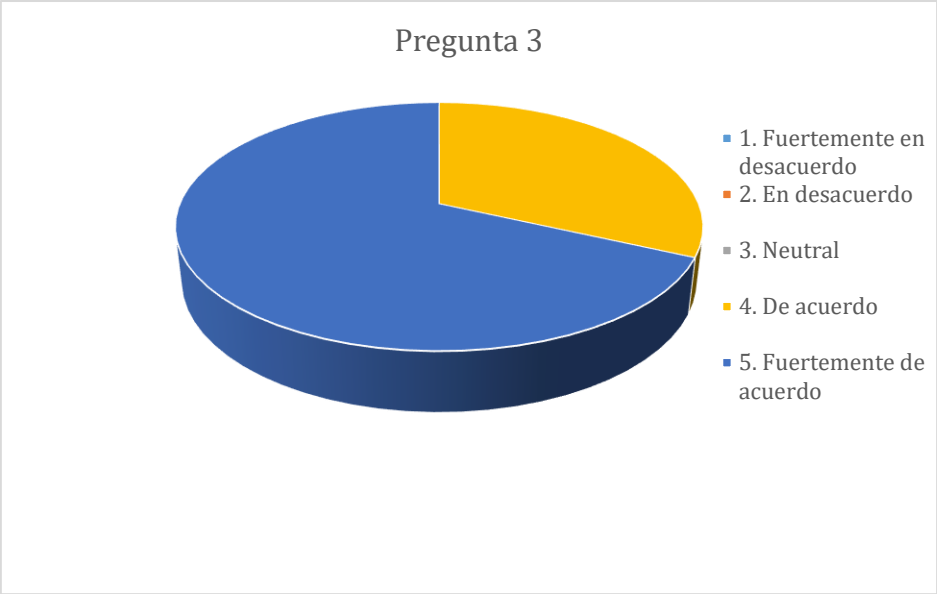


Figura 5-18. Resultados pregunta 3.

En la pregunta ¿El marco de trabajo propuesto proporciona estrategias que contribuyen a mejorar la calidad de la aplicación?, según la figura 5-19 y la tabla 24, los integrantes del segundo grupo considera en un 73% que el marco proporciona totalmente estrategias enfocadas a la calidad; mientras que en un 27% considera que están en acuerdo con las estrategias propuestas, pero con posibilidades de mejora.

Tabla 20. Resultados pregunta 4.

El marco de trabajo propuesto estrategias que contribuyen a mejorar la calidad de la aplicación	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	6	27
5. Fuertemente de acuerdo	16	73
Total	22	100

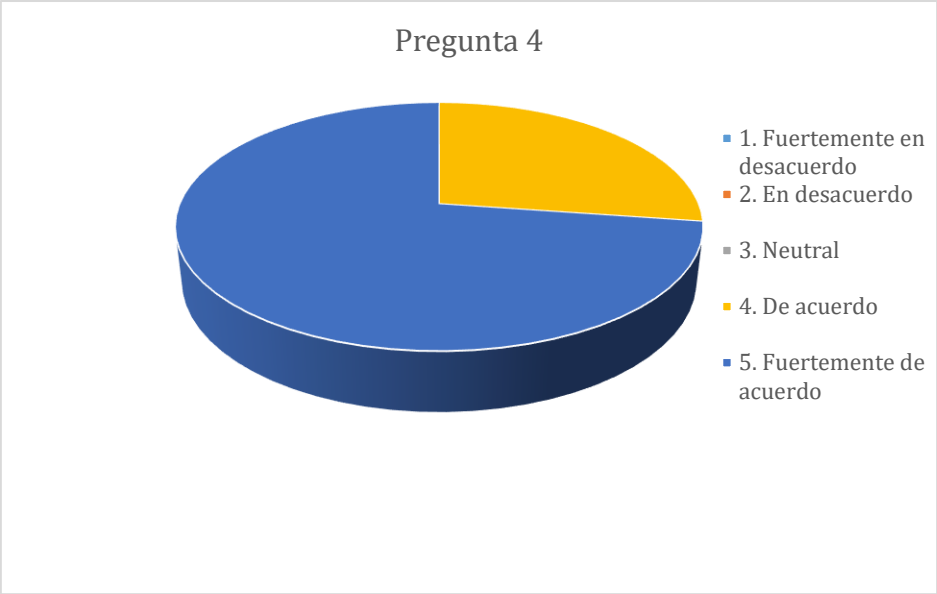


Figura 5-19. Resultados pregunta 4.

En la pregunta ¿El marco de trabajo propuesto puede ser extendido o adaptado a cualquier desarrollo de software móvil?, según la figura 5-20 y la tabla 25, los integrantes del segundo grupo considera en un 73% que el marco es totalmente extensible o adaptable en cualquier proyecto de desarrollo móvil; mientras que en un 27% considera que es adaptable o extensible, pero con posibilidades de mejora.

Tabla 21. Resultados pregunta 5.

El marco de trabajo propuesto puede ser extendido o adaptado a cualquier desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	6	27
5. Fuertemente de acuerdo	16	73
Total	22	100

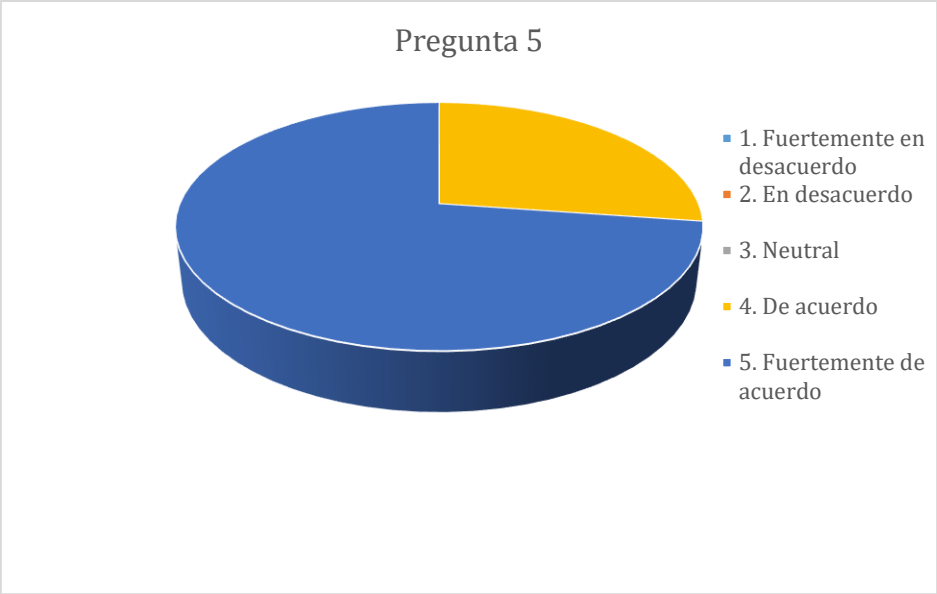


Figura 5-20. Resultados pregunta 5.

En la pregunta ¿El marco de trabajo propuesto basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que la metodología de desarrollo propuesta para el ejercicio complemento o apoyo el marco de trabajo propuesta para la construcción de la aplicación móvil?, según la figura 5-21 y la tabla 26, los integrantes del segundo grupo considera en un 36% que el método de desarrollo complemento totalmente al marco de trabajo; mientras que en un 64% considera que lo complemento, pero con posibilidades de mejora.

Tabla 22. Resultados pregunta 6.

El marco de trabajo propuesto basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que la metodología de desarrollo propuesta para el ejercicio complemento o apoyo el marco de trabajo propuesta para la construcción de la aplicación móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	14	64
5. Fuertemente de acuerdo	8	36
Total	22	100

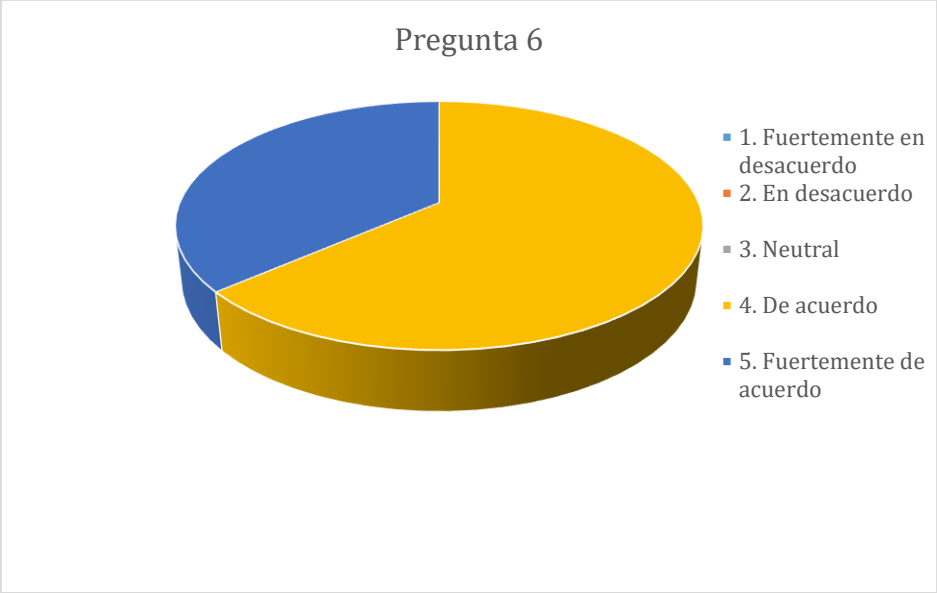


Figura 5-21. Resultados pregunta 6.

En la pregunta ¿El marco de trabajo propuesto contribuye a disminuir el tiempo de entrega de la aplicación móvil?, según la figura 5-22 y la tabla 27, los integrantes del segundo grupo considera en un 77% que contribuye totalmente a disminuir el tiempo de entrega de una aplicación móvil; mientras que en un 23% considera que contribuye en la eficiencia del tiempo de entrega, pero con posibilidades de mejora.

Tabla 23. Resultados pregunta 7.

El marco de trabajo propuesto contribuye a disminuir el tiempo de entrega de la aplicación móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	5	23
5. Fuertemente de acuerdo	17	77
Total	22	100

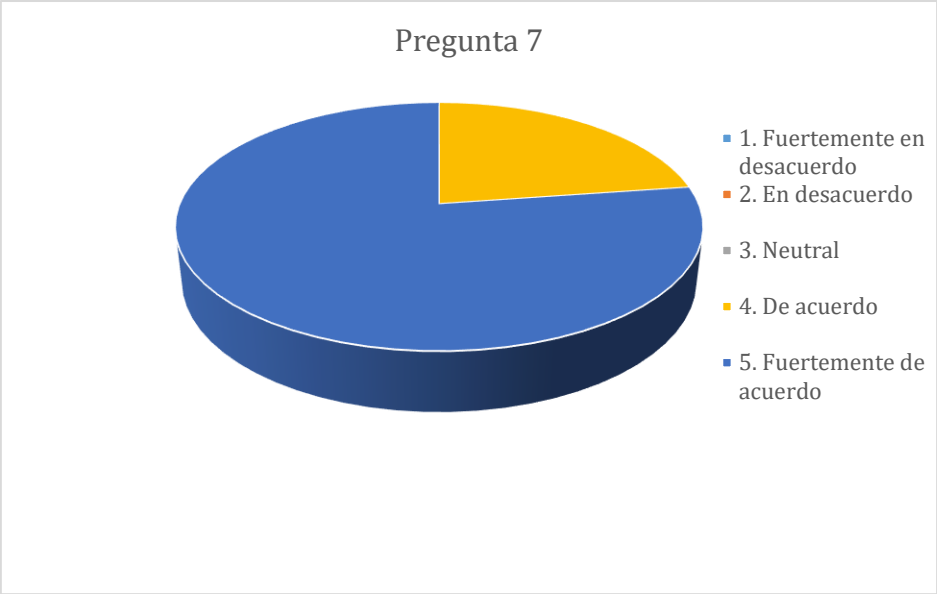


Figura 5-22. Resultados pregunta 7.

En la pregunta ¿El marco de trabajo propuesto las actividades recomendadas para el desarrollo de software para móviles son entendibles y claras?, según la figura 5-23 y la tabla 28, los integrantes del segundo grupo considera en un 77% que las actividades propuesta son totalmente entendibles y claras; mientras que en un 23% considera que son entendibles, pero con posibilidades de mejora.

Tabla 24. Resultados pregunta 8.

El marco de trabajo propuesto las actividades recomendadas para el desarrollo de software para móviles son entendibles y claras	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	5	23
5. Fuertemente de acuerdo	17	77
Total	22	100

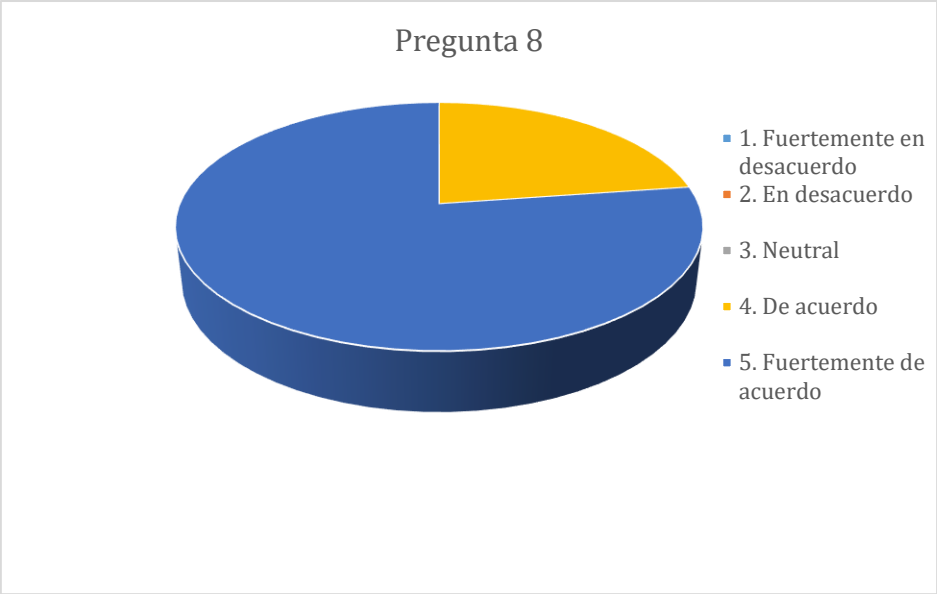


Figura 5-23. Resultados pregunta 8.

En la pregunta ¿El marco de trabajo propuesto proporciona conocimiento sobre el dominio de desarrollo de software para dispositivos móviles?, según la figura 5-24 y la tabla 29, los integrantes del segundo grupo considera en un 68% que el marco proporciona un total entendimiento sobre el dominio de software para móviles; mientras que en un 32% considera que proporciona conocimiento sobre el dominios, pero con posibilidades de mejora.

Tabla 25. Resultados pregunta 9.

El marco de trabajo propuesto proporciona conocimiento sobre el dominio de desarrollo de software para dispositivos móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	7	32
5. Fuertemente de acuerdo	15	68
Total	22	100

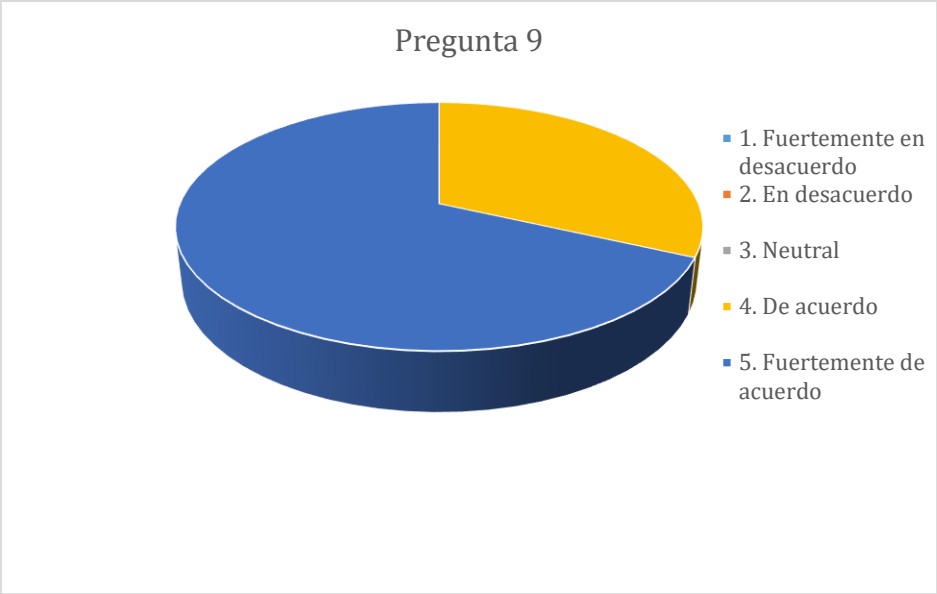


Figura 5-24. Resultados pregunta 9.

En la pregunta ¿El marco de trabajo propuesto las actividades propuestas para alcanzar los principales objetivos de una aplicación móvil son los adecuados?, según la figura 5-25 y la tabla 30, los integrantes del segundo grupo considera en un 77% que las actividades propuestas son totalmente adecuadas para alcanzar los objetivos esperados en el desarrollo móvil; mientras que en un 23% considera que las actividades son adecuadas, pero con posibilidades de mejora.

Tabla 26. Resultados pregunta 10.

El marco de trabajo propuesto las actividades propuestas para alcanzar los principales objetivos de una aplicación móvil son los adecuados	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	5	23
5. Fuertemente de acuerdo	17	77
Total	22	100

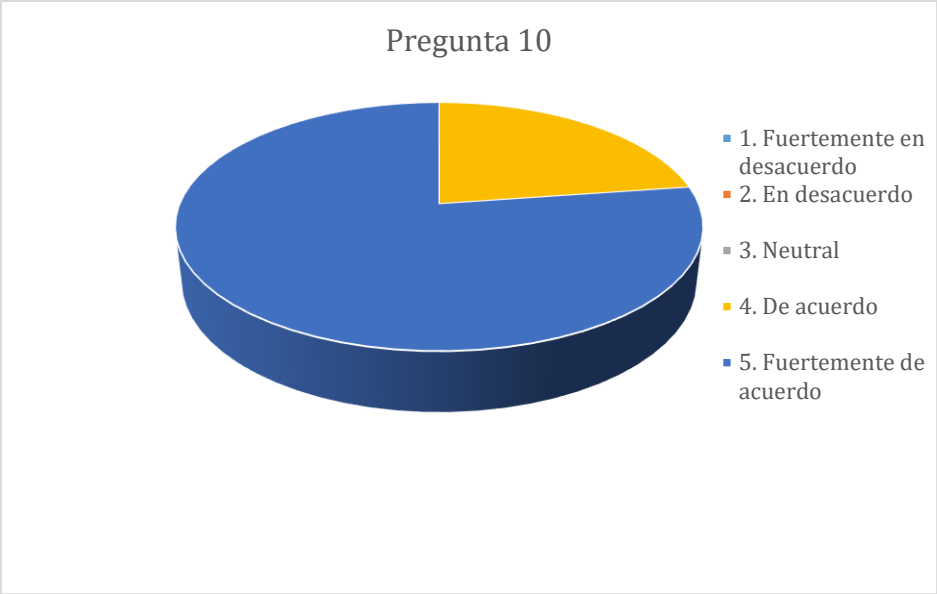


Figura 5-25. Resultados pregunta10.

En la pregunta ¿El marco de trabajo propuesto presenta mayor conveniencia en relación a otras propuestas para el desarrollo de software móvil?, según la figura 5-26 y la tabla 31, los integrantes del segundo grupo considera en un 59% que resulta totalmente más conveniente en relación a otras propuestas para el desarrollo móvil; mientras que en un 41% considera que es conveniente frente a otras propuestas, pero con posibilidades de mejora.

Tabla 27. Resultados pregunta 11.

El marco de trabajo propuesto presenta mayor conveniencia en relación a otras propuestas para el desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	9	41
5. Fuertemente de acuerdo	13	59
Total	22	100

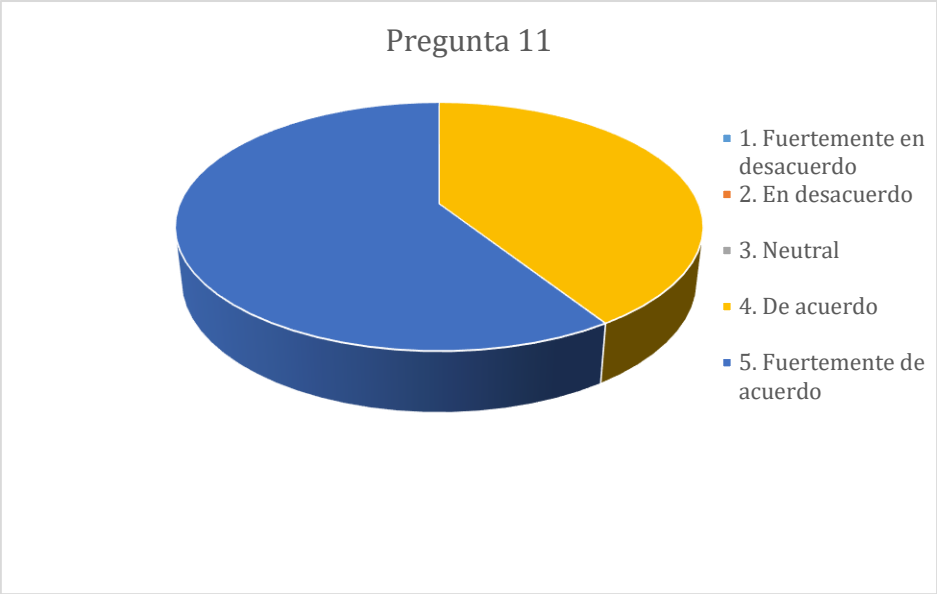


Figura 5-26. Resultados pregunta 11.

En la pregunta ¿El marco de trabajo propuesto es un sustito de otras propuestas para el desarrollo de software móvil?, según la figura 5-27 y la tabla 32, los integrantes del segundo grupo considera en un 86% que es un total sustituto en relaciona otras propuestas de desarrollo; mientras que en un 14% considera que es sustituto, pero con posibilidades de mejora.

Tabla 28. Resultados pregunta 12.

El marco de trabajo propuesto es un sustito de otras propuestas para el desarrollo de software móvil	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	3	14
5. Fuertemente de acuerdo	19	86
Total	22	100

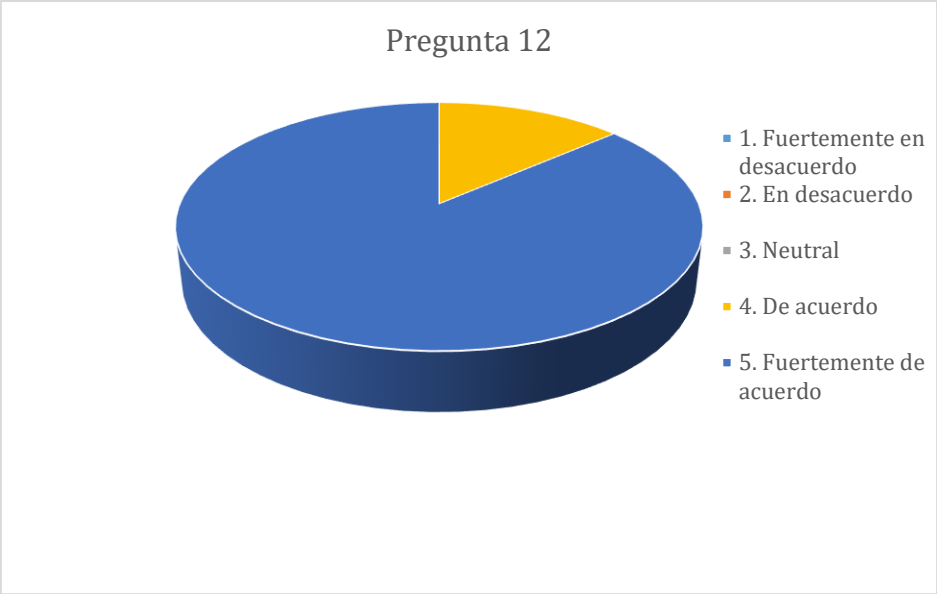


Figura 5-27. Resultados pregunta 12.

En la pregunta ¿El marco de trabajo propuesto contribuye a mejorar el desempeño de los desarrolladores que construyen software para dispositivos móviles?, según la figura 5-28 y la tabla 33, los integrantes del segundo grupo considera en un 91% que contribuye a mejorar totalmente de mejorar el desempeño de los desarrolladores que construyen software para móviles; mientras que en un 9% considera que contribuye en el desempeño, pero con posibilidades de mejora.

Tabla 29. Resultados pregunta 13.

El marco de trabajo propuesto contribuye a mejorar el desempeño de los desarrolladores que construyen software para dispositivos móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	2	9
5. Fuertemente de acuerdo	20	91
Total	22	100

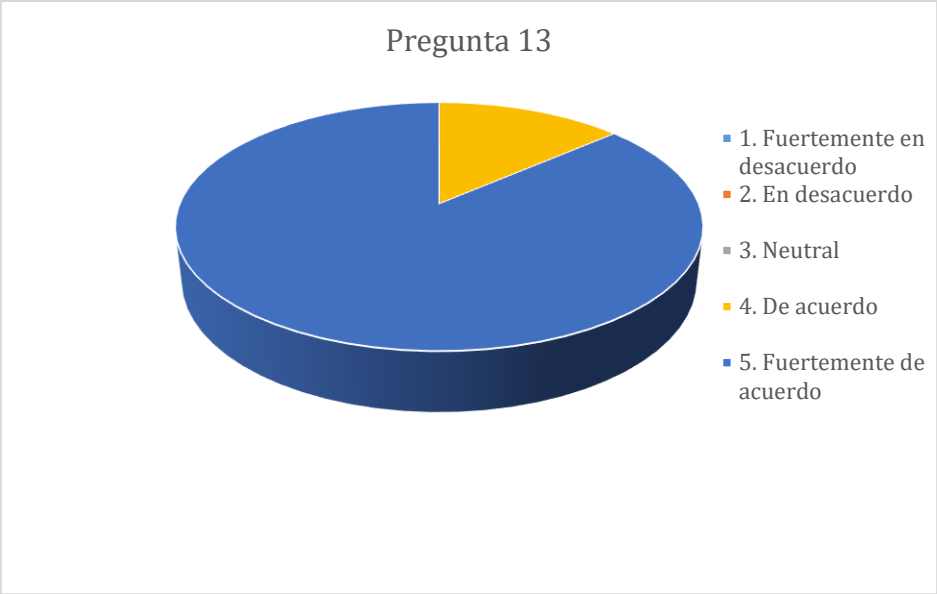


Figura 5-28. Resultados pregunta 13.

En la pregunta ¿El marco de trabajo propuesto complementa a otros enfoques de desarrollo utilizados en este tipo de desarrollo o el utilizado en la caso de estudio?, según la figura 5-29 y la tabla 34, los integrantes del segundo grupo considera en un 86% que complementa totalmente a otros enfoques utilizados en el desarrollo de software para móviles; mientras que en un 14% considera que es fácil de utilizar, pero con posibilidades de mejora.

Tabla 30. Resultados pregunta 14.

El marco de trabajo propuesto complementa a otros enfoques de desarrollo utilizados en este tipo de desarrollo o el utilizado en la caso de estudio	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	3	14
5. Fuertemente de acuerdo	19	86
Total	22	100

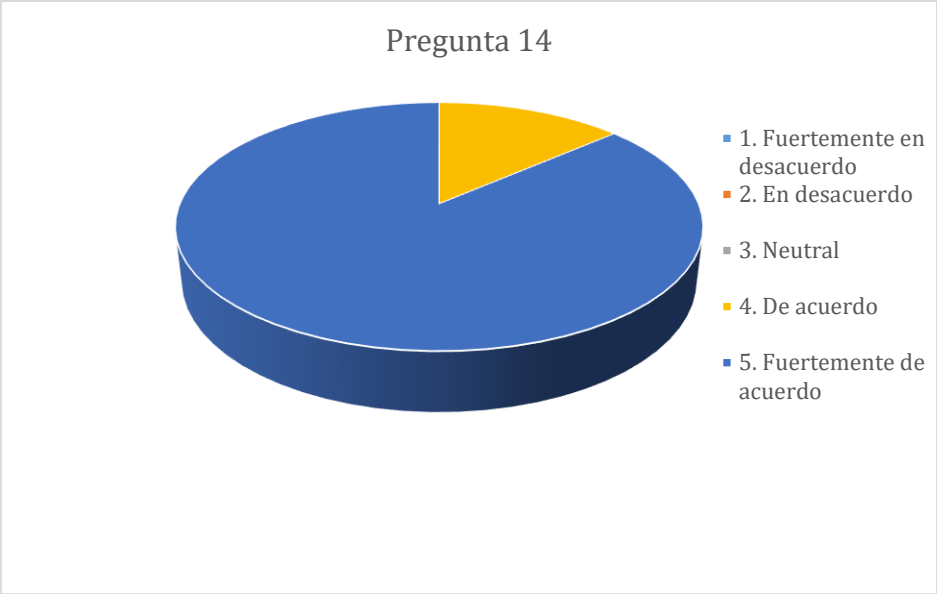


Figura 5-29. Resultados pregunta 14

En la pregunta ¿El marco de trabajo propuesto es recomendable para su adopción por desarrolladores de software para móviles?, según la figura 5-30 y la tabla 35, los integrantes del segundo grupo considera en un 95% que es totalmente recomendable para su adopción por parte de los desarrolladores de software móvil; mientras que en un 5% considera que es recomendable su adopción, pero con posibilidades de mejora.

Tabla 31. Resultados pregunta 15.

El marco de trabajo propuesto es recomendable para su adopción por desarrolladores de software para móviles	**n**	**%**
1. Fuertemente en desacuerdo	0	0
2. En desacuerdo	0	0
3. Neutral	0	0
4. De acuerdo	1	5
5. Fuertemente de acuerdo	21	95
Total	22	100

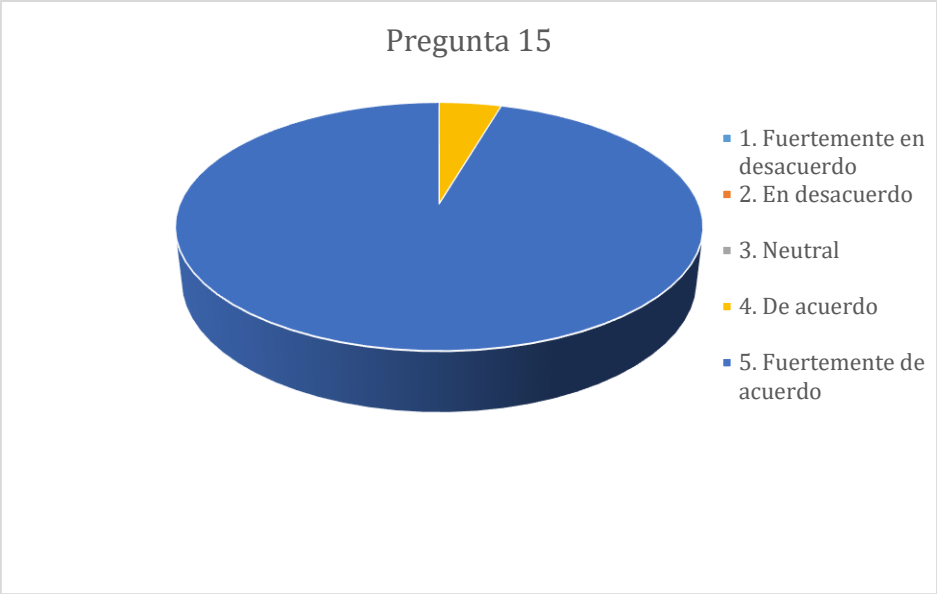


Figura 5-30. Resultados pregunta 15.

3. DESCRIPCION DE LOS RESULTADOS

En la siguiente sección se analizan algunos resultados sobresalientes obtenidos en ambos grupos después de la ejecución del experimento.

En general los estudiantes de ambos grupos consideran que el marco de trabajo les resulta fácil de entender y aplicar en un contexto de desarrollo de software para dispositivos móviles.

Para ambos grupos el marco de trabajo proporciona un adecuado cubrimiento del ciclo de vida, lo que confirma que el marco propuesto asegura el cumplimiento de la cobertura del ciclo de vida.

Cuando se preguntó a los alumnos la experiencia obtenida después de utilizar los dos enfoques propuestos, el primer grupo manifiesta que el marco de trabajo apoya o complementa en alto grado al método de desarrollo propuesto inicialmente; mientras que para el grupo 2, el método de desarrollo no presento un significativo apoyo al marco de trabajo según la valoración entregada.

Los alumnos perciben que el marco de trabajo contribuye a disminuir el tiempo de desarrollo de las aplicaciones, según lo manifestado en ambos grupos.

La mayoría de los alumnos esta en total acuerdo en que el marco de trabajo resulta recomendable para su adopción por los desarrolladores para proyecto de software móvil y que este además contribuye a mejorar el desempeño.

En ambos grupos valoraron positivamente el marco de trabajo propuesto en relación con otras alternativas existentes de desarrollo móvil.

Un alto porcentaje de los encuestados en ambos grupos manifestaron haber adquirido un mayor conocimiento sobre el dominio cuando utilizaron el marco de trabajo.

La pregunta relacionada con la contribución del marco de trabajo a mejorar la calidad del producto, si bien los resultados fueron positivos, se puede observar que este aspecto no fue tan altamente valorado como los otros ítems.

CAPITULO 6
CONCLUSIONES

La principal contribución de esta tesis es una propuesta de marco de trabajo para el desarrollo de software para dispositivos móviles, construida en base a un análisis de las principales características del dominio (requerimientos) y la unificación de las mejores prácticas identificadas en la literatura que basada en la evidencia sean identificado como adecuadas para el logro de los objetivos esperados en el desarrollo móvil. El marco de trabajo propuesto es una abstracción de alto nivel de la solución que esboza o provee una serie de actividades recomendadas para cada una de las fases del ciclo de vida de desarrollo que pueden ser instanciada y ajustada a las particularidades en un desarrollo de software móvil.

Se han presentado las principales características presentes en el desarrollo de software móvil que han sido reportado por diferentes autores como únicas en este aspecto, en adición se han presentado diferentes propuestas de tipo metodológico específicas para el desarrollo de software para dispositivos móviles, que luego fueron contrastadas permitiendo una visión general de las fortalezas y debilidades de cada una en relación con los objetivos esperados para este tipo de desarrollo.

El experimento realizado bajo un enfoque de ingeniería de software experimental permitió evaluar el aporte del marco de trabajo propuesto, los resultados obtenidos permitieron conocer la contribución de la propuesta, algunos de los principales resultados obtenidos de la ejecución del experimento mostraron que en términos generales los participantes manifestaron que el marco de trabajo propuesto presenta una serie de beneficios que lo hacen útil al momento de afrontar un proyecto de desarrollo para móviles.

Entre las principales limitaciones de la propuesta, es la ausencia de cobertura a la gestión o administración del proyecto, el enfoque utilizado en la construcción del marco de trabajo fue orientado al producto, en este caso la aplicación móvil. Gran parte de las actividades o prácticas que componen el marco de trabajo, son altamente influenciadas por las metodologías agiles, si bien, se ha reportado como convenientes para este tipo de proyectos, no se puede desconocer el aporte que pudo haber significado el conocimiento proveniente de metodologías estructuradas y orientadas a objetos, las cuales por cuestión de alcance no pudieron ser abordadas a profundidad en esta tesis.

Por consiguiente cada uno de los objetivos específicos se cumplió del siguiente modo:

El objetivo número uno se planteó:

"Explorar los métodos de desarrollo utilizados en el desarrollo de software para dispositivos móviles en la literatura más relevante del medio"

Para dar cumplimiento a este objetivo se realizó la búsqueda de las diferentes aproximaciones de corte metodológico propuestas para el desarrollo de software para dispositivos móviles disponibles en la literatura; luego para cada una de las aproximaciones se realizó una breve descripción de los principales componentes constitutivos, entre los que se distinguen: ciclo de vida o proceso de desarrollo propuesto, las motivaciones que

fundamentan la metodología, las características principales que responden a los requerimientos del desarrollo de software móvil. Los resultados del ejercicio explicado anteriormente se encuentran en el capítulo 3, donde se presenta la síntesis de cada una de las aproximaciones.

Para el objetivo numero dos

"Contrastar las metodologías de desarrollo para software para dispositivos móviles"

Con base en el conocimiento adquirido del logro del objetivo anterior, el paso siguiente consistió de identificar las principales características o requerimientos presentes en el dominio; se procedió a realizar una búsqueda en la literatura, dando como producto una lista de las características reportadas por diferentes autores como únicas que impactan en el desarrollo de este tipo de aplicaciones; paso seguido se agruparon las características similares o con estrecha relación en una lista de requerimientos; la cual proporciona una breve definición de cada requerimiento. Una vez la lista de requerimientos o características del dominio ha sido establecida, cada una fue comparada entre las diferentes metodologías identificadas del objetivo número uno, dando como resultado una tabla que muestra el resultado de contrastar las metodologías y los requerimientos del software móvil.

El objetivo número tres:

"desarrollar el marco de trabajo para el desarrollo de software para dispositivos móviles"

Una vez identificados los requerimientos que debe satisfacer el marco de trabajo para software móvil, se procedió a identificar las principales actividades o practicas adecuadas que abordan cada uno de los requerimientos, apoyado en el conocimiento, practicas e ideas de diferentes aproximaciones metodológicas obtenido de la revisión de ellas en la presente tesis, fueron adoptados en la construcción del marco de trabajo. La descripción del proceso y el resultado se encuentra explicado en el capitulo4.

El cuarto objetivo

"Evaluar el marco de trabajo para el desarrollo de software para dispositivos móviles a través de un enfoque de ingeniería de software experimental"

Mediante el enfoque de ingeniería de software experimental fue llevado a cabo un experimento que permitió evaluar el aporte o contribución del marco de trabajo propuesto; los detalles del experimento son descritos ampliamente en el capítulo 5.

Como trabajo a futuro, se propone continuar con el proceso de refinamiento y profundidad en un nivel de abstracción más bajo del marco de trabajo, que permita especificar patrones de procesos, actividades, tareas, productos de trabajo, roles y políticas enfocados que mediante un diseño conduzca a la construcción de un método formal de desarrollo de software móvil, que de tanto cobertura al producto y al proceso por igual. El marco de trabajo puede ser puesto en prueba en un ambiente de producción industrial de software móvil.

Para concluir, el alcance de la presente tesis es limitado, dado el potencial de investigación y desarrollo que representa este campo de la ingeniería de software es apenas explorado, por consiguiente el autor continuara su refinamiento y profundización sobre la temática en aras de lograr mayor exhaustividad y formalización que permitan la proposición de un método formal o metodología altamente valoradas por la comunidad científica de la ingeniería de software.

BIBLIOGRAFIA

Abrahamsson, P. (2005). Keynote: Mobile software development - the business opportunity of today. *Proceedings of the International Conference on Software Development*, (págs. 20-23). Reykjavik, Iceland.

Abrahamsson, P. (2007). Agile software development of mobile information systems. *Advanced Information Systems Engineering*.

Abrahamsson, P., Hanhineva, A., Hulkko, H., Ihme, T., Jäälinoja, J., Korkala, M., . . . Salo, O. (2004). Mobile-D: An Agile Approach for Mobile Application Development.

Ambler, S. (10 de 04 de 2015). *The Agile Unified Process (AUP)*. Obtenido de http://www.ambysoft.com/unifiedprocess/agileUP.html

Ambler, S. W. (1998). *Process Patterns: Building Large-Scale Systems Using.* New York.

Ambler, S. W. (1999). *More Process Patterns: Delivering Large-Scale Systems.* New York: Cambridge University Press.

Ambler, S. W. (10 de 05 de 2015). Obtenido de http://www.ambysoft.com/downloads/processPatterns.pdf

Basili, V. (1993). The Experimental Paradigm in Software Engineering. *Lecture Notes Computer Software*, 1-7.

Basili, V. (2007). The Role of Controlled Experiments in Software Engineering Research. *Lecture Notes in Computer Software*, 33-37.

Beck, K. (1999). *Extreme Programming Explained.*

Beck, K., & Andres, C. (2004). *Extreme Programming Explained: Embrace Change, Second Edition.* Addison Wesley Professional.

Blank, S., & Dorf, B. (2012). *The Startup Owner's Manual: The Step-by-Step Guide for Building a Great Company* .

Bohem, B. (1976). Software Engineering. *IEEE TRANSACTIONS ON COMPUTERS*, 1226-1240.

Brooks, F. (1987). No Silver Bullet: Essence and Accidents of Software Engineering. *Computer*, 10-19.

Carreño, A., & Aristizabal L, C. C. (2012). Design of a Framework to generate applications oriented to biosignals processing. *IEEE*.

Chaudron, M., Groote, J., van Hee, K., Hemerik, K., Somers, L., & Verhoeff, T. (2002). Software Engineering Reference Framework.

COAD, P., LEFEBVRE, E., & AND DE LUCA, J. (1999). *Java Modeling in Color with UML: Enterprise Components and Process.* Prentice-Hall, Englewood Cliffs.

Cockburn, A. (2000). *Agile Software Development.*

Cockburn, A. (2004). *Crystal Clear A Human-Powered Methodology for Small Teams.* Addison-Wesley Professional.

CodeProject. (03 de Mayo de 2015). *What is SCRUM*. Obtenido de http://www.codeproject.com/Articles/4798/What-is-SCRUM

Coplien, J. (1994). A Development Process Generative Pattern Language.

Cunha, T., Dantas, V., & Andrade, R. (2011). SLeSS: A Scrum and Lean Six Sigma Integration Approach for the Development of Software Customization for Mobile Phones.

Cunningham, W. (2 de mayo de 2015). *Manifiesto for agile development*. Obtenido de http://agilemanifesto.org/

Dancer, F., & Dampier, D. (2010). A Platform Independent Process Model for Smartphones Based on Invariants. *Systematic Approaches to Digital Forensic Engineering (SADFE), 2010 Fifth IEEE International Workshop* (págs. 56-60). IEEE.

Dooms, K., & Kylmäkoski, R. (2005). Comprehensive documentation made agile – experiments with RaPiD7 in Philips.

DSDM Consortium. (2003). *DSDM: Busines Focused Development.* Addison-Wesley.

DSDM Consortium. (20 de Febrero de 2015). Obtenido de DSDM Consortium From Page: http://www.dsdm.org/

Dunlop, M. (2002). The Challenge of Mobile Devices for Human Computer Interaction. *Personal and Ubiquitous Computing*, 235-236.

Easterbrook, S. M., Singer, J., Storey, M.-A., & and Damian, D. (2007). Selecting Empirical Methods for Software Engineering Research. *Proceeding ASE '07 Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering.* Springer.

ELECTRONICS, V. (24 de 03 de 2015). *MOBILE-D*. Obtenido de http://virtual.vtt.fi/virtual/agile/mobiled.html

Flora, H., Wang, X., & Chande, S. (2014). An Investigation into Mobile Application Development Processes: Challenges and Best Practices.

Forman, G., & Zahorjan, J. (1994). The Challenges of Mobile Computing . 39-46.

Gartner. (2010). A guiding framework for the development of a mobile application strategy.

Gartner. (2012). Mobile Applications and Interfaces: New Approaches for a Multichannel Future. 1-8.

Graf, H., & Jung, K. (2012). The Smartphone as a 3D Input Device: Using Accelerometer and Gyroscope for 3D Navigation with a Smartphone. *IEEE Second International Conference on Consumer Electronics .* Berlin.

Hammershøj, A., Sapuppo, A., & Tadayoni, R. (2010). Challenges for Mobile Application Development. *IEEE*.

Harleen, K., Flora, D., & Swati, V. (s.f.). A review and anaysis on mobile application development processes using agile methodlogies.

Heyes, I. (2002). Just Enough Wireless Computing.

Highsmith, J. (1998). *Adaptive software development: a collaborative approach to managing complex systems.* Dorset House Publishing Co.

Holler, R. (2006). Mobile Application Development: A Natural Fit with Agile Methodologies.

IBM. (15 de 06 de 2015). *Rational Unified Process: Best Practices for Software Development Teams*. Obtenido de http://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

IEEE. (1990). IEEE Standart Glossary of Software Engineering Terminology . *IEEE Std*.

Jeong, Y.-J., Lee, J.-H., & Shin, G.-S. (2008). Development Process of Mobile Application SW Based on Agile Methodology.

Joorabchi, M., Mesbah, A., & Kruchten, P. (2013). Real Challenges in Mobile App Development. *Empirical Software Engineering and Measurement.*

Lee, J. K., & Lee, J. Y. (2011). Android programming techniques for improving performance. *Awareness Science and Technology (iCAST), 2011 3rd International Conference* (págs. 27-30). IEEE.

Mnkandla, E. (2009). About Software Enginieering Frameworks and Methodologies. *IEEE AFRICON*, 23-25.

Naur, P., & Randall, B. (1969). Software Engineering. *A Report on a Conference Sponcered By NATO Science Committee.*

Nosseir, A., Flood, D., Harrison, R., & Ibrahim, O. (2012). Mobile Development Process Spiral. *IEEE*.

Object Management Group. (4 de 06 de 2015). *Unified Modeling Language (UML).* Obtenido de http://www.uml.org/

OMG. (2002). *Software & Systems Process Engineering Meta-Model Specification.* Object Management Group (OMG).

Osterwalder, A., & Pigneur, Y. (2011). *Generación de modelos de negocio: una manual para visionarios, revolucionarios y retadores.* Grupo Planeta.

Palmer, S., & Felsing, J. (2002). *A Practical Guide to Feature-Driven Development.* Prentice Hall PTR.

Palmieri, M., Singh, I., & Cicchetti, A. (2012). Comparison of Cross-Platform Mobile Development Tools. *International Conference on Intelligence in Next Generation Networks .* IEEE.

Pohl, K. (1993). The Three Dimensions of Requirements Engineering. *Fifth International Conference on Advanced Information Systems Engineering* , (págs. 275-292).

Pressman, R. S. (2010). ingeniería del software: Un enfoque práctico. Mexico: Mc Graw Hill.

Rahimian, V., & Ramsin, R. (2008). Designing an Agile Methodology for Mobile Software Development: A Hybrid Method Engineering Approach. *IEEE*.

Schawer, K. (1995). SCRUM Development Process.

Schuppenies, R., & Steinhauer, S. (s.f.). Software Process Engineering Metamodel.

SCRUM ORG. (10 de Marzo de 2015). *Scrum.orh The Home of SCRUM*. Obtenido de https://www.scrum.org/

Startup, E. m. (2012). *Ries, E.* Grupo Planeta.

Syer, M., Adams, B., Zou, Y., & Hassan, A. (2011). Exploring the Development of Micro-apps: A Case Study on the BlackBerry and Android Platforms. *Source Code Analysis and Manipulation (SCAM), 2011 11th IEEE International Working Conference on* (págs. 55-64). IEEE.

Takeuchi, H., & Nokana, I. (1986). The new new product development game. *Harvard Business Review*.

Ulrich, K., & Eppinger, S. (2012). *Product Design and Development.* McGraw-Hill.

Verdugo, J., & Ruiz, F. (2008). Guía de Uso de SPEM 2 con EPF Composer version 3.0.

Voice of the Next-Generation Mobile Developer, Appcelerator / IDC. (12 de 06 de 2015). Obtenido de http://www.appcelerator.com.s3.amazonaws.com/pdf/Appcelerator-Report-Q3-2012-final.pdf

Wasserman, A. (2010). Software Engineering Issues for Mobile Application Development. *ACM*.

Wells, D. (15 de Marzo de 2015). *Extreme Programming: A gentle introduction*. Obtenido de http://www.extremeprogramming.org/

Williamson, L. (2012). A mobile application development primer:A guide for enterprise teams working on mobile application projects. *IBM Corporation*, 1-11.

World Rugby. (12 de Marzo de 2015). Obtenido de http://www.worldrugby.org/

Zhang, D., Adipat, & B. (2005). Challenges, methodologies, and issues in the usability testing of mobile applications. *International Journal of Human-Computer Interaction*.

ANEXOS

MARCO DE TRABAJO PARA EL DESARROLLO DESOFTWARE MOVIL

CUESTIONARIO DIRIGIDO A LOS ESTUDIANTES

El presente cuestionario tiene como propósito obtener información de los estudiantes que participaron en los ejercicios, sobre las bondades y dificultades en el uso del marco de trabajo para el desarrollo de software para dispositivos móviles propuesto.

Su nombre y las respuestas son confidenciales y serán utilizadas estrictamente para propósitos de la investigación.

Gracias por su participación.

Fecha:
Nombre: Cédula:
Profesor: Grupo:
Cuestionario

1. Marque una "X" en la columna que mejor represente su posición sobre cada una de las siguientes afirmaciones:

Afirmación: El marco de trabajo propuesto por(Cañas Jhovanny)…	**Fuertemente de acuerdo**	**De acuerdo**	**Neutral**	**En desacuerdo**	**Fuertemente en desacuerdo**
Es fácil de entender					
Es fácil de utilizar					
Cubre todos los aspectos del ciclo de vida de un desarrollo de software móvil					
Proporciona estrategias que contribuyen a					

Afirmación: El marco de trabajo propuesto por(Cañas Jhovanny)…	Fuertemente de acuerdo	De acuerdo	Neutral	En desacuerdo	Fuertemente en desacuerdo
mejorar la calidad de la aplicación					
Puede ser extendido o adaptado a cualquier desarrollo de software móvil					
(Contestar si usted participo del primer grupo) Basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que el marco de trabajo propuesto complemento o apoyo el método de desarrollo propuesto para la construcción de la aplicación					
(Contestar si usted participo del segundo grupo) Basado en la experiencia obtenida en el ejercicio del desarrollo de la aplicación móvil Considera que la metodología de desarrollo propuesta para el ejercicio complemento o apoyo el marco de trabajo propuesta para la					

Afirmación: El marco de trabajo propuesto por(Cañas Jhovanny)…	Fuertemente de acuerdo	De acuerdo	Neutral	En desacuerdo	Fuertemente en desacuerdo
construcción de la aplicación móvil					
Contribuye a disminuir el tiempo de entrega de la aplicación móvil					
las actividades recomendadas para el desarrollo de software para móviles son entendibles y claras					
Proporciona conocimiento sobre el dominio de desarrollo de software para dispositivos móviles					
las actividades propuestas para alcanzar los principales objetivos de una aplicación móvil son los adecuados					
Presenta mayor conveniencia en relación a otras propuestas para el desarrollo de software móvil					
Es un sustito de otras propuestas para el desarrollo de software móvil					
Contribuye a mejorar el desempeño de los					

Afirmación: El marco de trabajo propuesto por(Cañas Jhovanny)…	**Fuertemente de acuerdo**	**De acuerdo**	**Neutral**	**En desacuerdo**	**Fuertemente en desacuerdo**
desarrolladores que construyen software para dispositivos móviles					
Complementa a otros enfoques de desarrollo utilizados en este tipo de desarrollo o el utilizado en la caso de estudio					
Es recomendable para su adopción por desarrolladores de software para móviles					