Guía práctica para la selección de una metodología ágil respecto al tipo de proyecto de desarrollo de software entre SCRUM, XP y KANBAN

Jackeline Centeno Mena, Daniel Retana Ruiz

Escuela de Ingeniería,
Universidad Latinoamericana de Ciencia y Tecnología,
ULACIT, Urbanización Tournón, 10235-1000
San José, Costa Rica
[jcentenom753,dretanar179@ulacit.ac.cr]
http://www.ulacit.ac.cr

Resumen La administración de proyectos es un tema ampliamente estudiado, en el caso de desarrollo de software, se cuenta con más de 30 años de experiencia y aún existe un alto porcentaje de fracaso en los proyectos tecnológicos. En este artículo se tratará de profundizar sobre los tipos de proyectos y plantea una guía para los líderes de este propósito, basado en metodologías ágiles, que han sido probadas en empresas de desarrollo como Google.inc., con la finalidad de hacer que el proyecto tecnológico que se desarrolle sea de éxito.

En esta guía se investiga tres metodologías ágiles, Xtreme Programing, Scrum y Kanban. A pesar de que existen otros sistemas se resolvió basarse en estas tres metodologías, debido a que registran en su uso, grandes casos de éxito, Kanban por ejemplo es el modelo que utiliza Toyota.

Sin embargo para poder brindar una guía es necesario poder categorizar qué tipos de proyectos tecnológicos coexisten, incluye desde implementación de paquetes hasta proyectos de desarrollo. Además se requiere conocer los detalles de cada una de las metodologías ágiles, roles que definen, técnicas de trabajo y el papel que tiene el cliente en cada una de ellas. Y así poder recomendar qué metodología se adapta mejor a cada tipo de proyecto.

Una vez que se recopila toda esta información, se procede a analizar las condiciones que se necesitan para cada tipo de esquema, el cual debe ser desarrollado, de manera que se discrimina entre las metodologías elegidas, según roles de trabajo o funciones de las mismas, algunos proyectos requieren mayor cantidad de iteraciones, otros necesitan mayor control de tiempo.

En los resultados se detalla en qué radica la efectividad de cada metodología para cada tipo de proyecto y por último, en las conclusiones se especifica los datos clave para equiparar cada tipo de plan y buscar una metodología ágil que se adapte a las condiciones del mismo.

Keywords: SCRUM, XP, KANBAN

1. Introducción

Las Metodologías Ágiles de desarrollo de Software surgen aproximadamente en los años 90, ya que, la metodología tradicional no se adaptaba a las condiciones cambiantes que presenta el mundo actual, esta eran predictiva y enfocada a procesos, mientras que las metodologías ágiles son adaptables, flexibles y se orientan a las personas.

El uso de este tipo de metodologías no se limita a proyectos pequeños o medianos, sino que es compatible con planes de gran tamaño debido a que es incremental y a su vez iterativo, lo que permite tener una retroalimentación del proceso y los usuarios. Esto permite eliminar posibles problemas como falta de eficiencia y validez, a su vez, disminuir el tiempo de desarrollo con un diseño adaptable que permita mantener la calidad durante todo el proceso.

Con la finalidad de poder establecer la metodología de investigación para el desarrollo de software se realizan la las siguientes interrogantes:

- ¿Qué beneficios se obtienen al desarrollar un proyecto mediante sistemas de gestión ágiles ?
- ¿Cuál metodología de desarrollo ágil es mejor para un proyecto de software, a corto, mediano o largo plazo?
- ¿Por qué es importante considerar la implementación de una metodología para el desarrollo de un proyecto?
- ¿A cuáles etapas del proyecto aplican las metodologías ágiles?

Para poder contestar a todas las interrogantes indicadas se plantean los siguientes objetivos específicos:

- Definir de forma clara la estructura de cada metodología.
- Especificar qué tipos de proyectos se adaptan a cada metodología, según experiencias y documentación recopilada en la investigación.
- Precisar las ventajas y desventajas, que confrontan cada una de las tecnologías en casos típicos de desarrollo de software.
- Establecer una guía que permita a los gerentes de TI, PMP, o PMO elegir la metodología idónea para su proyecto entre SCRUM, XP y KANBAN.

Con el fin de dar respuesta a los puntos indicados se propone investigar metodologías ágiles para el desarrollo de software enfocados a la gestión de proyectos, tales como: SCRUM, XP y KANBAN, y así mostrar las ventajas y desventajas competitivas, en casos reales mediante la ejemplificación; de esta manera se ofrecerá una guía o recomendación general que le permita a la gerencia de TI seleccionar la metodología, la cual se adapte a sus necesidades.

La razón por la que se decide desarrollar este proyecto fue para poder comparar las tres metodologías mencionadas anteriormente desde una perspectiva de Gestión de proyectos de Software, teniendo la delimitación de poca documentación local, por lo que la técnica de recolección de datos utilizada sería recolección de información en Internet sobre artículos relacionados, libros, blogs y material multimedia.

2. Marco Teórico

En el presente trabajo se plantea una guía que facilite la selección de una Metodología Ágil, la cual depende del tipo de proyecto de Software a desarrollar entre SCRUM, XP y KANBAN, consideradas las metodologías más populares en la actualidad (Bowes, 2015). Esto, debido a que no hay periodos de software terminados, sino periodos estables, mientras se realiza un nuevo lanzamiento con mejoras en el software inicial (McGee, 2015), por lo que es importante determinar cuál Metodología Ágil se adapta mejor al proyecto.

Estas metodologías nacen para sustituir a las tradicionales, caracterizadas por ser rígidas y muy documentadas(Fau, s.f.), en las cuales se valoran los siguientes puntos (Fau, s.f.; Scotland, 2009):

- Las personas son el principal factor de éxito, ya que pueden construir excelentes equipos de trabajo y estos un entorno en el cual desarrollar el software.
- No se debe documentar de forma excesiva, esta debe ser corta y centrarse en los puntos más importantes del desarrollo del software.
- El cliente no es solo un punto de contacto, tiene que interactuar constantemente durante toda la elaboración del proyecto.
- No se debe ser rígidos con los procesos alineados y definidos al inicio del proyecto, sino responder rápidamente a los cambios que surjan durante su elaboración.

Presenta 12 principios definidos del Manifiesto Ágil: (Fau, s.f.; Scotland, 2009; Gimson, 2012):

1. La principal prioridad es satisfacer al cliente mediante entregas tempranas y continuas de software que aporten valor.
2. Requisitos cambiantes.
3. Entregar el software trabajado frecuentemente, con preferencia en escalas de tiempo corto.
4. Los desarrolladores y el cliente deben trabajar, en conjunto, durante todo el proyecto.
5. Construir proyectos mediante personas motivadas. Ofrecer un ambiente adecuado, la confianza y el apoyo para el desarrollo de sus tareas.
6. El método más eficiente y eficaz de transmitir información desde y dentro de un equipo de desarrollo es la conversación cara a cara.
7. La medida principal de progreso es el software funcional.
8. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deberían ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.
10. La simplicidad es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.

12. En intervalos regulares, el equipo reflexiona sobre cómo llegar a ser más eficaz, rítmicos y ajustar su comportamiento en consecuencia.

Las metodologías que se analizan (Cuadro 1) en esta investigación nacen de la siguiente manera:

Scrum	XP- Extreme Programming	Kanban
La primera referencia se da en un artículo en Japón para el año de 1986 por Ikujiro Nonaka e Hirotaka Takeuchi, mediante un enfoque holístico o de rugby, en el cual este término se deriva de una estrategia del juego: en el cual, se debe "conseguir un fuera de juego de la pelota en el juego" (Pekka Abrahamsson, 2002).	El término "extremo" fue creado por Kent Beck, en el año 1996,(Gimson, 2012), el cual consiste en utilizar las prácticas y principios habituales pero a niveles extremos. (Pekka Abrahamsson, 2002)	Se originó en Japón, adoptado en 1962, por Taiichi Onho como control de producción entre procesos y la implementación "Just in Time" (JIT) para las plantas de manufactura de Toyota. (Scotland, 2009)

Cuadro 1. Origen de las Metodologías Ágiles.

A continuación se detalla la forma en la que se encuentra estructurada cada metodología y los procesos que se realizan:

2.1. XP (eXtreme Programming)

Las practicas utilizadas no son nuevas en comparación a las metodologías tradicionales, sin embargo, estos se ejecutan a niveles extremos (Agile Alliance, 2002; Pekka Abrahamsson, 2002) para acoplarse a las prácticas cambiantes actuales.En el cuadro 2 se detallan las 5 fases en la que consiste XP.

Para que las etapas mencionadas, en el cuadro 2, se pueden llevar a cabo es necesario que se ejecuten por medio de los siguientes roles (Agile Alliance, 2002; Pekka Abrahamsson, 2002; Fau, s.f.):

- Programador (Programmer): Encargados de escribir las pruebas y dar mantenimiento al código de la forma más clara y simple posible. Un factor de vital importancia es la comunicación entre todos los integrantes del equipo.
- Cliente (Customer): Indica el requerimiento (características) que debe tener el sistema, así como las pruebas funcionales a realizar y además, indicar cuando este se haya cumplido. También, muestra el orden que debe tener la realización de cada requerimiento.
- Encargado(a) de las pruebas (Tester): Se encarga de realizar-ayudar las pruebas con el cliente, a nivel de funcionalidades, resultados y mantenimiento de las herramientas.

- Encargado (a) de brindar seguimiento (Tracker): Se faculta a alguien para medir el trabajo realizado por el equipo de desarrollo, las iteraciones, limitantes en los recursos o tiempo estimados, así como también indicar la mejora de los procesos-trabajos realizados hasta ese momento y retroalimentar al grupo de encargo.
- Entrenador (Coach): Se encarga de guiar al equipo a que siga el proceso establecido.
- Consultor (Consultant): Es externo a la empresa. Tiene conocimiento técnico y se encarga de guiar al grupo de trabajo para solucionar problemas en específico.
- Gerente (Manager): Toma todas las decisiones del proceso mediante la guía de su equipo de trabajo. Tiene que identificar deficiencias o debilidades en el proceso, si las hubiera, para corregirlas lo más pronto posible.

El equipo de trabajo debe ser integrado como mínimo por tres integrantes y como máximo veinte.

Las características de XP son las siguientes:

- Juego de planificación: Es la interacción entre los programadores y el cliente, los cuales miden el esfuerzo necesario para la realización, y la definición del alcance y comunicación entre ambos.
- Iteraciones cortas: Las entregas tienen que ser producidas rápidamente, no deberían de tardar 3 meses.
- Metáfora: Definidas por el cliente y los programadores para describir cómo funciona el sistema
- Diseño simple: La solución brindada al cliente debe ser lo más sencilla posible, si algo es complejo o tiene código innecesario, debe ser removido.
- Pruebas: Hay 2 tipos de pruebas:
  1-las unitarias desarrolladas antes que el código y
  2- las funcionales que describe el cliente.
- Refactorización: Se busca estructurar mejor el sistema, eliminando la duplicación, simplificando el sistema y mejorando la comunicación.
- Programación en parejas: Esta pareja trabaja en la misma computadora, uno de ellos se encarga de desarrollar/programar el código y el otro de revisar y soportar al compañero.
- Propiedad colectiva: El código puede ser cambiado por cualquier persona en el momento que lo considere necesario.
- Integración continua: El código es integrado apenas se confirme que esté listo, por lo que hay integraciones constantes a lo largo del proyecto, hasta tener el código final.
- A la semana se deben de trabajar 40 horas, no más que estas, sí de forma consecutiva se trabajan horas extra, por ejemplo en un periodo de 2 semanas, se analiza, pero si fue solo por 1 día no es necesario.
- Cliente en el sitio: Siempre tiene que estar disponible el cliente para trabajar con el equipo de desarrollo.

- Normas de codificación: La comunicación también se da a través de líneas de código, por lo que se espera que estas sigan las reglas establecidas y fácilmente entendibles por los programadores.
- Espacios abiertos de trabajo: Los programadores deben de estar en el centro de la habitación, esta tiene que ser grande y dividida en pequeños cubículos.

2.2. SCRUM

Se compone de 3 fases detalladas en el cuadro 3. Fases SCRUM.

Esta metodología tiene cinco roles principales, los cuales se detallan en el cuadro 4 Roles SCRUM.

Al igual que XP, Scrum tiene sus propias prácticas o técnicas, las cuales se detallan a continuación:

- Pila de producto: En una técnica para definir todos los requerimientos y necesidades para la realización del proyecto final, identificados por una lista de prioridades y constantes actualizaciones, con base a los requisitos técnicos y solicitudes del negocio, como por ejemplo: mejoras solicitadas, actualizaciones de tecnologías, corrección de errores, entre otros. En esta parte participa el equipo del proyecto, el cliente, el encargado de ventas y mercadeo y el administrador. El encargado del mantenimiento de la pila de producto es el dueño del producto.
- Estimación de esfuerzo: Es un proceso iterativo realizado por el dueño del producto y el Scrum team para estimar si hay una demora o si el progreso actual del proyecto es el esperado.
- Iteración, conocida como Sprint: Es el proceso de adaptación cambiante por factores como: recursos, tecnologías, tiempo, entre otros, el cual tiene una duración aproximada de 30 días para que el equipo de Scrum lo organice. Las herramientas de trabajo utilizadas por el grupo son: reuniones, planificación, revisión de pedidos pendientes, y reuniones diarias Scrum.
- Reunión de planificación: En esta participan todos los integrantes del equipo y el cliente para definir los objetivos y la funcionalidad de los Sprint. En la segunda fase solo participa el Scrum Master y el equipo, ellos se centran en incrementar la implementación del producto.
- Pila de iteración : Son elementos identificados para ser realizados en el siguiente Sprint. Estos se definen en la reunión de planificación y son estables debido a que hasta que se complete el Sprint no se continúa con otra tarea (Sprint).
- Reunión diaria Scrum: En estas reuniones se mide el proceso del proyecto, se evalúa el cumplimiento de los puntos mencionados en la reunión anterior, problemas que hayan presentado, entre otros aspectos que se consideren importantes. El Scrum Master es quien la dirige y tiene una duración aproximada de 15 minutos.
- Reunión de Revisión: Es una reunión informal que se realiza al final del Sprint para verificar los resultados obtenidos y analizar las posibles acciones para mejorar en el siguiente Sprint.

- Pizarra Scrum: Permite mostrar el avance que ha tenido el proyecto de una forma visual para el equipo.

2.3. KANBAN

Tiene un sistema de trabajo similar a Scrum por el hecho de que se maneja por etapas, sin embargo ambas metodologías son muy distintas entre sí.

Kanban utiliza una pizarra o muro que divide en bloques, en estos coloca tarjetas con las labores que se deben de realizar y a su vez, a cada bloque (columna) le coloca un nombre que ilustre la etapa en la que se encuentra para poder medir el progreso del trabajo realizado.

Esta metodología limita la cantidad de procesos que se pueden ejecutar en las diferentes etapas, ya sea por un acuerdo entre el equipo de trabajo o porque el límite quedó establecido en la pizarra, por ejemplo: Si tenemos una columna llamada entrega y a esa columna le colocamos el número 2, eso quiere decir que solo se pueden realizar dos entregas, al mismo tiempo, las demás se harán una vez que estas concluyan, para tener en orden las tareas completadas por cada etapa.

También, esta metodología brinda la opción de medir las etapas por tiempo(Skarin, 2010), se espera que este sea corto y lo más predecible posible para calcular el tiempo estimado del proyecto.

Kanban se caracteriza por ser muy adaptativo (Skarin, 2010), es decir, tiene menos reglas a seguir, esto debido a que no es tan rígido como los sistemas tradicionales y permite adaptarse fácilmente al cambio.

Como principales normas a cumplir, Kanban maneja 8:

- Visualizar el flujo de trabajo
- Limitar el tiempo de trabajo en proceso (work in progress)
- Reducir multitarea
- Mejorar el enfoque
- Mejorar ciclos de tiempo
- Aumentar Feedback
- Mejorar la calidad
- Mejorar la madurez equipos

Estas 8 normas permiten tener un mejor control de lo que se está realizando en el proyecto y de las posibles acciones a tomar en caso de notar una demora o un proceso pendiente de completar.

En el caso de Kanban, no tiene roles definidos(Skarin, 2010) como en XP o Scrum, ya que esta metodología permite establecer los roles que se consideren necesarios, siempre y cuando estos generen valor y no se perciban como un conflicto con algún elemento del proceso. Tampoco hay iteraciones con un tiempo establecido o fijo, en este caso, etapas como la de planeación o revisión de procesos se realizan conforme se considere necesario.

Para determinar qué procesos hay que mejorar y de los cuales tener retroalimentación se tienen dos métricas (Skarin, 2010):

1. Medio tiempo de espera: en esta métrica se mide el proceso alcanzado, por lo que si el objetivo/actividad/labor ya fue completada esta se debe de actualizar en la pizarra para tener visibilidad de los objetivos pendientes.
2. Cuellos de Botella: Se revisa la forma en que están distribuidas las labores en la pizarra para asegurarnos que no tengamos columnas muy cargadas y otras vacías.

Por el gran avance que han tenido las tecnologías durante los últimos años es muy difícil restringir o apartar al cliente del desarrollo de sus procesos-programas-proyectos, por lo que estas metodologías muestran una cultura de colaboración entre el cliente y el equipo de trabajo de forma constante, en el cual la documentación tiene que ir muy enfocada al proyecto de software realizado sin agregar documentación adicional que no sea de utilidad o provecho (Keith, 2002).

Cuando el modelo de trabajo que se implementa es muy rígido, es difícil adaptarlo a los cambios del mercado actual, donde se quiere que todo salga lo más rápido posible, que sea amigable con el usuario y que sea un proceso organizado, las metodologías que explican en el desarrollo de este trabajo pretende ofrecer una herramienta que sea de utilidad para próximos proyectos.

En la siguiente sección se expone posibles categorizaciones según el tipo de proyecto de software a implementar:

2.4. Tipos de proyectos de Software

Aún no existe una clasificación de proyectos de software de la cual exista consenso, los proyectos de software se caracterizan por ser artesanales y esto dificulta su gestión, y así mismo condiciona el éxito. Una clasificación para los proyectos tecnológicos que se adapta muy bien a los proyectos de software, a continuación se enumeran las diferentes clasificaciones (Archibald, 2003):

1. Por tamaño de proyecto(Velasco, 2004): para definir el tamaño del proyecto se debe de tomar en cuenta la cantidad de dinero a invertir, el tiempo que pueda tomar, recursos y habilidades requeridas, por supuesto todo esto debe de ser evaluado por un experto en desarrollo de software.
2. Por complejidad: en este caso se debe de tomar en cuenta la interacción entre diferentes entidades ya sea terceros o si estamos en una empresa de tipo corporativo, la interacción multitarea, el financiamiento del mismo, si requiere habilidades especializadas que conlleve contratación de terceros.
3. Basada en el cliente: se define(Archibald, 2003) dos posibilidades: cliente interno y cliente externo, naturalmente dependiendo del consumidor así será el grado de riesgo, sabemos que el conflicto es menor si estamos en un ambiente interno, donde podemos justificar atrasos y demás, ya que el equipo desarrollador es parte de la empresa, mas sin embargo, cuando el cliente es externo mediante los contratos de nivel de servicio, (SLA por sus siglas en inglés) los equipos deben de ser eficientes y apegarse a fechas de entrega, pues una entrega tardía suele convertirse en un proyecto ruinoso.

4. Basada en el conocimiento que tenga el cliente en el proyecto: es una perspectiva donde el autor del libro demuestra su experiencia, y un cliente envuelto en el proyecto obtiene la garantía de éxito, dado que conoce el negocio y brindaría a los desarrolladores datos clave para cumplir el objetivo. Lo anterior es verdadero, sin embargo (Archibald, 2003) cuando el cliente está envuelto en el proyecto genera atrasos que deben de ser considerados en la planeación, ya que esto involucra dinero y tiempo en el proyecto, además detalla que el cliente dentro del proyecto ejerce un control de calidad exhaustivo que dará un producto más pulido pero más costoso, en todos los ámbitos del proyecto.

Otro enfoque(Velasco, 2004)distinto al anterior, más enfocado al desarrollo de software, se fundamenta en la experiencias que se han obtenido a través de varios años en la dirección de proyectos bajo las buenas prácticas establecidas en el PMBoK; A continuación se enumera cada una de ellas:

1. Modelos DES: Desarrollo de aplicaciones estructurado, el autor Bualo Velasco lo describe como el modelo más clásico de desarrollo, en donde se hacen fases y una vez que una fase está concluida y cumple con los requerimientos del cliente se procede con la siguiente, a este modelo también se le llama de cascada.
2. Modelo DDO: Desarrollo de aplicaciones orientada a objetos; dentro de esta clasificación caben los desarrollos a la medida de una aplicación para una función específica, basado en el paradigma moderno de orientación a objetos (modelo de programación que se adapta más a la vida real, similar a como vemos las cosas en la vida cotidiana) tiene fases como el modelo DES, pero en este canon algunas fases pueden ser iterativas, lo que brinda al cliente versiones del software que se van optimizando con el tiempo que tarda el proyecto.
3. Modelo MAN: Modelo de mantenimiento de aplicaciones, este modelo se refiere al trabajo sobre software ya existente, entiéndase de mantenimiento o mejora del mismo, y puede ser de mantenimiento correctivo o bien de mantenimiento evolutivo. El caso de mantenimiento correctivo debido a que la aplicación o software no funciona como debería en condiciones definidas, y mantenimiento evolutivo en el caso de que el software necesite agregar más funcionalidades debido a que el negocio lo requiere. Suele gestionarse por fases como los dos modelos anteriormente mencionados.
4. Modelo BIN: Bussiness Intelligence[1], o bien inteligencia de negocio en español. Son proyectos donde las empresas buscan obtener información valiosa para el negocio a partir de la información que ya se tiene mediante el uso de tecnologías de software como Datawarehouse[2], minería de datos y un concepto popular en nuestros días el big data. Con la suma de las tecnologías mencionadas es posible seleccionar, preparar y analizar la información para

[1] Proceso para convertir datos en información relevante para el negocio
[2] Base de datos que integra información de varias fuentes para procesarla posteriormente

saber qué necesitan los clientes de la empresa y que rumbo tomar a nivel de negocio. En este modelo el sistema de cascada y fases que se menciona en los modelos anteriores cambia y las iteraciones son múltiples según la información que se va obteniendo con el avanzar del proyecto.

5. Modelo PKS (Package Selection): Este modelo de proyecto consiste en elegir qué paquete de software se adapta mejor a las necesidades de la empresa, para conjugarlo con el software ya existente. Este tipo de proyectos suele justificarse en la necesidad de pertenecer y adaptarse al mercado global de estándares transaccionales para ser más ágiles en los negocios, es aquí donde los desarrollos a la medida quedan de lado y toman relevancia los paquetes de software. Este proyecto se divide en tres fases:
   a) Planificación
   b) Definición de Requerimientos
   c) Selección del Software
6. Modelo PKI (Package Implementation): implementación de paquete. Posterior al PKS continua el modelo PKI, consiste en implementar paquetes de software como ERPs[3] y CRMs[4] en las empresas e integrarlos con los desarrollos propios de la empresa cliente. El modelo de desarrollo se divide en tres fases, planificación diseño e implantación.
7. Modelo OUT: modelo outsourcing o modelo de contratación externa. Este tipo de proyecto consiste en tomar el control parcial o total de alguna de las tareas de TI en una empresa con el fin de optimizar las tareas y reducir los costos para el negocio, va desde mantenimiento de infraestructura hasta desarrollo de aplicaciones. Se define como proyecto y no como proceso porque tiene una duración conocida generalmente definida en un contrato.

3. Desarrollo

Posterior al detalle del origen de cada metodología, se identifica la utilidad de cada una de ellas para entender el enfoque que poseen y de esta manera poder aplicarlas correctamente, como se muestra en el cuadro 5. Utilidad de las Metodologías Ágiles.

Las imágenes de Scrum (Figura 1) y Kanban (Figura 2) ejemplifican la forma en que se podría utilizar estas metodologías.

Ambas pizarras se ven muy similares sin embargo tienen una gran diferencia, en Scrum se maneja por intervalos de tiempo mientras que en Kanban por procesos a realizar.

En la imagen 2 2 debajo de las columnas de análisis y desarrollo, aparecen los número 2 y 3, eso quiere decir que únicamente se pueden realizar 2 o 3 tareas en ese momento, hasta que esas sean completadas y se mueven a la otra columna se podrán trabajar otras nuevas. En el caso de Scrum, no tiene un limite de tareas por proceso, lo que lo limita es el tiempo que establezcan durante los Sprint para

[3] Sistema de gestión empresarial para automatizar procesos de la empresa
[4] Software para gestionar la relación con los clientes de una empresa

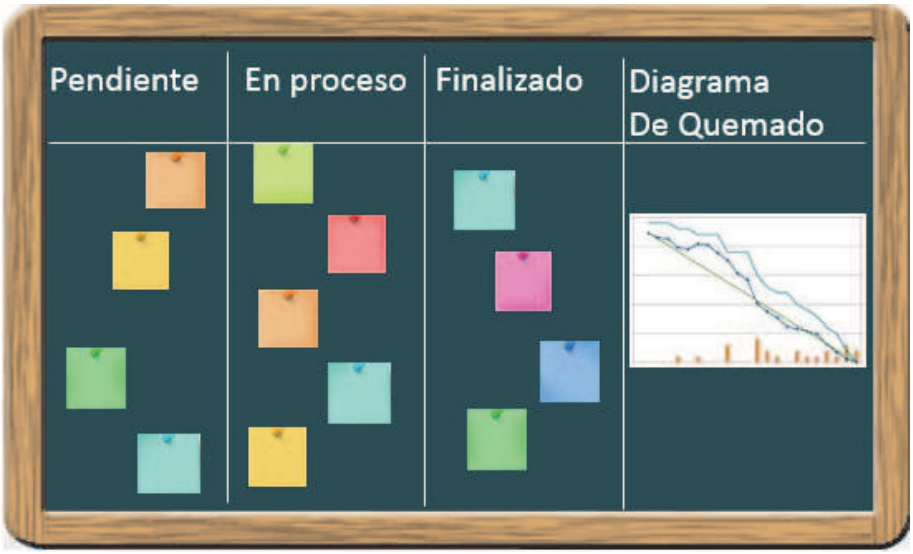

Figura 1. Ejemplo de una pizarra Scrum detallada por tareas y Diagrama de Quemado

la realización y revisión de los mismos, por eso es que cada columna puede tener la cantidad de tareas que ellos consideren necesarias.

Además Scrum presenta un diagrama de quemado, que muestra las tareas pendientes en relación con el tiempo para su ejecución.

Al momento de implementar estas metodologías y que se desarrollen de la manera correcta es necesario tener claro los roles establecidos para su ejecución, los cuales son:

Según el análisis propuesto se busca brindar una guía de metodologías ágiles según el tipo de proyecto, hasta el momento tenemos dos vertientes de la cuales se elige las mejores seis (Cuadro 7), ya que algunas categorías de las mencionadas se pueden englobar dentro de una sola, debido a similitudes de las mismas:

Teniendo en cuenta lo anterior se procede a identificar qué metodología ágil corresponde a los tipos de proyecto.

3.1. Proyectos por Tamaño

El tamaño y la complejidad de un proyecto, requiere evaluación de un experto en la materia, en cualquiera de los campos de TI que abarque el proyecto. Para esta investigación no se entrará en detalles de esa evaluación, ya que depende de variables como tiempo, cantidad de personal disponible y el cliente, que serán datos muy propios de cada proyecto. A continuación de definen las tres subdivisiones y su metodología correspondiente:

Proyecto pequeño: los proyectos pequeños se caracterizan por realizarse en periodos cortos de tiempo, con una cantidad limitada eh incluso reducida de personal, o bien en estos casos las empresas brindan tiempos parciales de

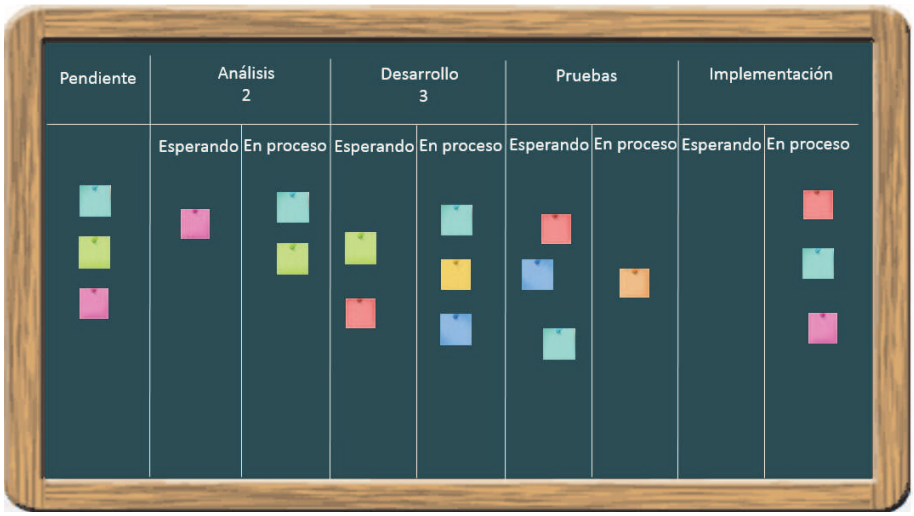

Figura 2. Ejemplo de una pizarra Kanban detallada por tareas

sus empleados para el desarrollo del entregable, sin embargo por ser proyectos pequeños se esperan resultados rápidos, en este caso se elige Kanban debido a que no tiene roles definidos, y tomando en cuenta que el equipo es pequeño es probable que los roles deben de rotar entre los miembros del grupo, anteriormente se mencionó que se busca que el tiempo sea corto, por lo que la medición de tiempos que hace Kanban y la orientación a entrega de procesos, brinda avances en corto tiempo al cliente.

Proyecto Mediano: para este tipo de proyectos se eligió SCRUM, ya en esta clasificación se dispone de más personal y se pueden asignar roles como el Scrum Master y el Product Owner, además el cliente está más involucrado en este tipo de proyectos, debido a que la inversión es mayor, se presentan avances de manera continua. Parte importante de los proyectos medianos que se adapta con Scrum es que permite cambios en todas las etapas del proyecto y ya que el cliente estará más involucrado es posible que esta variable se de, por lo que los sprints de scrum se hacen necesarios en esta subclasificación.

Proyectos Grandes: XP cuenta con etapas de exploración y planificación de manera más específica, lo cual es uno de los grandes errores en proyectos de esta índole, los planes grandes pueden tardar años en lograrse y XP busca la calidad del producto mediante el trabajo en parejas de desarrollo, sin hacer jornadas extensivas para los empleados, aparte de lo mencionado en el desarrollo de software está normado, esto es importante debido a que se busca que todo el equipo logre comprender el código. Las iteraciones son cortas lo que mantiene al cliente lo suficientemente cerca para que vea el trabajo realizado, pero no tan cerca como para atrasar el proyecto. Para este caso Scrum puede ser funcional dependiendo de las variables que se tengan, Scrum administrado correctamente brinda seguimiento con el cliente, iteraciones cortas, e incluso si es necesario

puede manejar equipos de desarrollo de más de 8 personas. Por las razones anteriores, XP y Scrum son funcionales para proyectos grandes.

3.2. Tipo de Cliente

En este apartado se dan a conocer dos divisiones: En la primera se describe el comportamiento del cliente interno: el cual muy relativo a la empresa, este suele estar enterado de los movimientos globales de la empresa e inmerso total o parcialmente en el proyecto, aunque no sea competencia de su área y no es tan riguroso en los tiempos de entrega; sin embargo quiere el producto muy bien pulido, que se adapte de manera justa a las necesidades de los colaboradores del proceso, esto brinda una referencia de qué modelo debemos usar para cliente interno.

Basado en lo anterior a nivel interno se puede utilizar Scrum dado que el formato que ofrece es rápido, permite cambios, brinda documentación concisa y definición de figuras de liderazgo que suelen existir a lo interior de las empresas.

Se suma también el detalle de los sprints de scrum, el cual es básico para la clasificación de cliente interno: suele suceder en las empresas que el usuario interno acostumbra solicitar constantes cambios en la marcha, conforme se va probando con los usuarios el sistema desarrollado, es esta la razón de mayor peso por la que se elige Scrum para este modelo.

La segunda división es el cliente externo, en este caso la administración del cliente suele ser más formal, las entregas se dan completamente documentadas y probadas. La etapa de planificación procura ser lo más detallada posible para omitir grandes cambios en medio del desarrollo del proyecto, cuando el cliente es externo este no está involucrado en el desarrollo a tiempo completo y básicamente busca un producto que funcione y que esté terminado lo más pronto posible para agilizar las operaciones de su empresa.

Para clientes externos se puede utilizar XP, debido a que de las tres metodologías define etapas, lo que familiariza al usuario con su conocimiento previo de proyectos, y le hace confiar en la empresa. Entonces existe una primera etapa de exploración para conocer al cliente, para continuar con el periodo de planificación, el cliente externo necesita saber por el bienestar de su negocio de manera clara la ruta que se trazara para llegar al objetivo, todo lo anterior se da en un tiempo razonable de uno o dos meses. Luego, comienza el desarrollo con sus respectivas iteraciones, antes de la liberación de la versión o parte de la solución tecnológica. En la etapa de producción se realizan las pruebas con el cliente y este puede pedir cambios, se continúa realizando iteraciones, pero de manera más rápida, y se genera un porcentaje de operatividad al negocio que mejora el software. XP dentro de sus características norma la codificación del programa, esto hace que todo el equipo tenga la posibilidad de entender los problemas o fallas en el código y corregirlas en pro del cliente. Todo lo anterior define que el producto sea de calidad y entregado en un tiempo razonable según como lo solicitó el interesado.

3.3. Involucramiento con el Cliente

Es prácticamente imposible predecir qué tan involucrado está el cliente en el proyecto si no lo conocemos, sin embargo es una variable que se puede definir desde el principio. Esto define múltiples tipos de proyectos y de cliente, desde el cliente que no se involucra y quiere un resultado funcional y probado, por ejemplo desarrollo de aplicaciones móviles de ventas de mercancías hasta el cliente que no quiere perder ningún detalle del desarrollo, queriendo corregir fallas del programa e incluir mejoras de ser necesario para la funcionalidad del entregable, bajo esta condición las metodologías de desarrollo ágil, se adaptan según sea el caso.

Para el cliente que busca un resultado final y operacional en el menor tiempo posible Kanban es la elección, porque entre sus características el funcionamiento por etapas y la medición del tiempo de entrega de cada proceso, punto importante para poder saber cuánto se tarda desde que se hace la solicitud de parte del cliente hasta que se entrega el producto terminado.

Cuando el cliente que trabaje con el grupo de desarrollo quiera involucrarse medianamente en el proyecto o más se recomienda Scrum y XP debido a que toman al cliente como parte del proyecto.

Por su parte, Scrum involucra al cliente desde el pre-juego, en la planificación, y por su adaptación a cambios, el usuario forma parte de estos y de las iteraciones. Suelen involucrarse mediante el rol de dueño de producto, el cual puede estar hasta el cien por ciento del tiempo con el equipo de desarrollo y en la parte final se entrega el producto al cliente y este debe de aprobarlo. Scrum está hecho para aceptar cambios, como se menciona en el marco teórico. Cuando el cliente está involucrado suele solicitar muchos cambios a lo largo de todo el proceso, en esta parte el cuadro negro de Scrum toma valor. Si vemos en detalle el cliente está inmerso en todas las etapas del proceso, forma parte de los sprint y reuniones periódicas que defina el Scrum Master.

Por otra parte, tenemos XP, aquí el usuario está involucrado desde la planificación del proyecto y en conjunto con el programador define la metáfora, en las siguientes fases se ve involucrado al cliente en la etapa de pruebas, donde él como usuario experto tiene que aprobar cada iteración o cada entregable. XP es muy recomendable cuando el interesado está muy involucrado en el proyecto, ya que tiene como variable "cliente en sitio" hace al interesado, uno más del equipo de desarrollo, de pruebas y este debe dedicar un cien por ciento al proyecto.

3.4. Fases del Proyecto

La selección de esta clasificación fue sencilla, debido a que el desarrollo de un pequeño o gran proyecto por fases es lo más conocido y estudiado, todos los tecnólogos conocen el funcionamiento de un diagrama de flujo, o básicamente una acción es consecuencia de una operación anterior, en el caso de los administradores de proyectos, se han visto obligados a estudiar el PMBoK, que define fases consecutivas para lograr el éxito del proyecto, se suele decir que la informalidad de las metodologías ágiles, no se adapta de ninguna manera al desarrollo por

fases, sin embargo como se puede ver en el marco teórico de esta investigación eso es completamente falso.

Para el desarrollo por fases se adapta excepcionalmente Xtreme Programming, a pesar de que existen periodos no consecuentes en el modelo, la mayoría lleva un orden muy claro, e incluso los equipos de trabajo también están claramente definidos. La primera fase es la de planificación, sin ella no se puede avanzar, XP va un paso más allá y la divide en dos: Exploración y Planificación, en consecuencia las iteraciones para las primeras entregas, aquí es donde podría existir una leve diferencia, debido a que según el tipo de proyecto hay fases que se pueden ejecutar paralelas con el modelo de XP; sin embargo el Manager puede definir cambios consecutivos agregando al modelo de fases las iteraciones necesarias para que el producto (liberación) sea lo más limpio posible. Una vez que se han brindado las liberaciones del producto, se comienza la producción, la cual suma una mejora utilizando XP, puesto que la entrada a producción de un programa o producto suele darse al concluir el proyecto, lo que genera incertidumbre en el plan o hasta su fracaso, en esta etapa se corren pruebas con los usuarios finales y se realizan las modificaciones necesarias, en la última fase se da mantenimiento al aplicativo desarrollado y se cierra el proyecto.

3.5. Contrato Externo de Mantenimiento

A pesar de que se sabe que las metodologías ágiles son aplicables a múltiples proyectos, su mayor enfoque es al desarrollo de software debido a la complejidad que maneja y la poca experiencia que se tiene en relación con otros tipo de proyectos en ingeniería, sin embargo, día a día, se desarrollan proyectos tecnológicos un poco más alejados del desarrollo de software y ligados a la selección e implementación de paquetes de software, instalación de redes activas o bien implementación de servidores en todas sus modalidades, en este punto el panorama cambia y nos enfrentamos a un modelo de producción más parecido al de Toyota, donde se busca los mejores resultados en el menor tiempo posible y no requiere de tantas iteraciones como lo requiere el desarrollo de software. Para esta clasificación la que mejor se adapta es Kanban.

Kanban trabaja por etapas divididas en bloques, muy coherente con la implementación de paquetes de software, infraestructura y redes, de esta forma se tiene control de que se hace y el tiempo que se utiliza en cada una de las tareas para asignar el personal necesario y evitar lo cuellos de botella, define también tareas paralelas que están especificadas en la pizarra de Kanban, según número de entregas que se pueden dar al mismo tiempo. Para el administrador del proyecto de mantenimiento es fácil controlar quién y lo que se está haciendo, esto le permitirá brindar al cliente informes periódicos de tareas, puesto que aquí ya no se mide tanto el rendimiento de un producto, si no la cantidad de trabajo realizada.

Kanban a diferencia de Scrum y XP tiene muy pocas normas lo cual facilita la tarea del coordinador del grupo y hace sencillo mostrar al cliente el trabajo realizado. Se puede mostrar al cliente de manera inmediata el avance del proyecto, trabajos que se están realizando y los concluidos en la pizarra de Kanban,

que se convierte en un escritorio; que a nivel gerencial permite tomar decisiones, sin tener que leer datos extensos.

3.6. Inteligencia de Negocio

El modelo de proyectos de inteligencia de negocio tiene variables un poco diferentes a las demás, se suele trabajar con bases de datos de cualquier índole para obtener o filtrar la información que el negocio, en este caso sinónimo de cliente requiere, lo que necesita múltiples iteraciones, de los tres modelos que estudia esta investigación los que brindan mayor importancia a las iteraciones son XP y Scrum. XP por un lado, vela mucho por la calidad del producto, los programadores trabajan en parejas para tener el código lo más sencillo y depurado posible; sin embargo, el costo a nivel económico de esta Metodología ágil es uno de los más altos, y para esta clasificación de proyecto, la prioridad no es precisamente depurar el software, si no realizar múltiples iteraciones para obtener datos específicos. Por otro lado, esta Scrum que cuenta con un modelo bastante sólido de iteraciones que permite de manera más cómoda para el cliente, obtener lo que busca: información. Otra ventaja que presenta Scrum frente a esta clasificación es que el usuario está a lo largo de todo el proyecto, así que, él mismo puede indicar que requiere en cada sprint e ir obteniendo la información que necesita y llevarla a su empresa para que le brinden un tratamiento comercial, mercadológico o investigativo.

4. Factores de riesgo al elegir Metodologías Ágiles

Si bien la investigación va dirigida hacia la promoción de las metodologías ágiles, esto no es garantía absoluta de éxito. Como se expuso en el marco teórico las metodologías ágiles son métodos que han sido probados, depurados que funcionan, sin embargo existe una serie de riesgos que siempre se debe de tomar en cuenta:

Definición de un nuevo paradigma: se conoce que los paradigmas antiguos de desarrollo de proyectos generan más fracasos que éxitos en los proyectos, sin embargo a muchos de los tecnólogos se les adoctrino bajo la teoría de que los modelos convencionales funcionan y son la única forma de llevar a buen fin los proyectos. Es aquí donde las metodologías ágiles encuentran resistencias y pueden ser boicoteadas por cualquiera de los integrantes del grupo, desde el cliente hasta el líder del proyecto. Esto debido a que las Metodologías Ágiles son innovadoras, buscan el bienestar del cliente y del empleado, lo cual a menudo se confunde con vagancia.

Incorrecta utilización: Las metodologías ágiles funcionan, las tres que se evalúan en esta investigación tienen ejemplos de éxito en empresas de renombre o bien que tomaron renombre después de la implementación de una metodología ágil, sin embargo si se entiende mal el concepto por alguno o algunos de los participantes del grupo de desarrollo de la Metodología Ágil, puede fracasar, porque va a desviar los objetivos del proyecto.

Confianza del cliente:para los que conocen las metodologías ágiles es sencillo confiar en una empresa que las utiliza, sin embargo para muchos otros la informalidad de trabajo o la pérdida parcial o total de uno de sus recursos, les brinda desconfianza acerca del desarrollador del proyecto, ya que le hace pensar que necesita personal de experiencia del cliente, ya que la empresa carece del recurso. A este apartado se puede agregar que la documentación puede ser diferente a como el cliente acostumbra a recibirla, según proyectos anteriores en los que ha trabajado.

Este documento al ser una guía para elegir metodologías ágiles pretende también brindar las herramientas necesarias para que los proyectos que se desarrollen en el futuro disminuyan el riesgo y optimicen los recursos disponibles, sin embargo las variables mencionadas anteriormente deben de mantenerse bajo control.

5. Conclusiones y recomendaciones

Como se ha venido explicando en el presente documento, estas tres metodologías ágiles se pueden adaptar a diferentes tipos de proyectos de software, e incluso podrían ser adecuados a otros tipos de propósitos, que requieran organizar sus tareas y tiempo de una forma más adaptable a los cambios constantes solicitados por el interesado.

Sin embargo, la solicitud constante de cambios por el cliente en algunos casos podría significar una desventaja, ya que, por ejemplo en Scrum podría solicitar más cambios de los esperados, y exigir más funcionalidades y procesos que no estaban documentados. XP es un ejemplo, en el cual tiene como ventaja que su cliente interactúe durante todo el proceso, lo que permite que el producto desarrollado sea igual o muy similar a lo que esperaba este, a diferencia de Kanban que la interacción es mínima y su atención se centra en las tareas por desarrollar.

También es importante que la comunicación sea fluida entre los miembros del grupo, lo cual es una ventaja que presentan estas 3 metodologías, lo que permite, por ejemplo en XP tener una retroalimentación constante, en Scrum logra revisar las áreas de mejora y buscar soluciones prontas a los problemas presentados y en Kanban, aunque los roles no son tan definidos como en las otras 2 metodologías, si la comunicación no fluye, la revisión de procesos sería complicada para completar cada una de las tareas.

Otra ventaja, es la documentación realizada, ya que se centra en evidenciar lo principal, no todo lo que se hace y que no genere un valor agregado al final, aunque esto también se podría tomar para algunas personas como desventaja, ya que prefieren que cada proceso y tarea realizada esté debidamente documentada y detallada, aunque no sea de uso diario o genere mucha utilidad.

Cada una de estas metodologías tiene una estructuración diferente, por ejemplo: en el caso de XP se utilizan iteraciones para llevar a cabo revisiones de procesos mientras que en Srum estos se denominan Sprints.

Se dice que usualmente estas metodologías no se emplean un cien por ciento a como se espera, sino que se toma lo mejor de cada una y se complementa con otra metodología de ser necesario, para obtener el control del proceso y objetivos esperados. Todo va a depender del tipo de proyecto, ya sea enfocado a clientes, por fases, tamaño o por inteligencia del negocio, entre otros. Por lo que en conclusión, aunque los proyectos de software son muy diferentes entre sí, hay metodologías que les permitirían mejorar la estructura actual y aprovechamiento del tiempo, para los requerimientos cambiantes de los clientes actuales, favoreciendo los cambios y soluciones de una manera oportuna y eficaz.

Como recomendaciones, para que esas metodologías se acoplen lo mejor posible al modelo de desarrollo realizado, es importante tener claro el objetivo del proyecto, la meta a alcanzar, los integrantes del equipo, la labor que cada miembro del equipo va a realizar, conocer al cliente con el que trabajamos y detallar los procesos que se requieren para el desarrollo del Software. Con esta información se facilita identificar la metodología correcta que mejor se adapte a nuestro proyecto y que brinde un valor agregado importante en la estructuración del mismo.

También se recomienda entender el tipo de proyecto a trabajar, por ejemplo si este es un proyecto pequeño, se espera que este se realice de una forma corta y que no involucre mucho personal, por lo que Kanban en este caso sería el más indicado para estructurar el proyecto y documentar el avance de los requerimientos solicitados por el cliente.

Otra recomendación sería, trabajar en equipo, tener una buena comunicación tanto interna como externa y un líder que pueda brindar ejemplo al equipo, que realice las reuniones semanales o diarias y que pueda encaminar el proyecto de la mejor manera para tener una conclusión exitosa. Todos los roles son distintos en las metodologías, pero unos con otros se complementan para la elaboración correcta del producto solicitado por el cliente.

6. Referencias Bibliográficas

Referencias

Agile Alliance, B. (2002). *Comparison of agile methodologies.* Descargado de `http://www.exa.unicen.edu.ar/catedras/agilem/comp.pdf` pages 4, 23

Archibald, R. D. (2003). *Managing high tecnology programs and projects* (third ed.). Hoboken NJ: John Wiley & sons .Inc. pages 8, 9

Bowes, J. (2015, jul). *Kanban vs scrum vs xp – an agile comparison.* Descargado de `http://manifesto.co.uk/kanban-vs-scrum-vs-xp-an-agile-comparison/` pages 3

Fau, C. (s.f.). *Xp agil.* Descargado de `http://www.carlosfau.com.ar/nqi/nqifiles/XP_Agil.pdf` pages 3, 4, 23

Gimson, L. L. (2012). *Metodologías ágiles y desarrollo basado en conocimiento* (Tesis de Master no publicada). UNIVERSIDAD NACIONAL DE LA PLATA, Argentina, Los Hornos. pages 3, 4

Keith, E. R. (2002, dec). Agile software development processes a different approach to software design. , *1*. pages 8

McGee, M. (2015, nov). *What is adaptive software development?* Descargado de `http://www.wisegeek.com/what-is-adaptive-software-development.htm` pages 3

Pekka Abrahamsson, J. R. . J. W., Outi Salo. (2002). *Agile software development methods* (first ed.). Centre, Finland: VTT publications 478. pages 4, 23

Scotland, K. (2009, jul). *Kanban, flow and cadence.* Descargado de `http://availagility.co.uk/wp-content/uploads/2009/04/kfc-development-spa2009.pdf` pages 3, 4

Skarin, H. K. . M. (2010). *Kanban y scrum – obteniendo lo mejor de ambos* (first ed.). Estados Unidos: C4Media Inc. pages 7, 23

Velasco, L. B. (2004). *Clasificación de los proyectos informáticos y establecimiento de sus estructuras de descomposición de trabajo* (Tesis de Master no publicada). UNIVERSITAT OBERTA DE CATALUNYA, España, Cataluña. pages 8, 9

Exploration (Exploración)	En esta etapa el cliente indica las características que debe tener el programa, a la vez, que el equipo de proyecto se familiarice con la tecnología y herramientas indicadas por el cliente. Esta etapa tarda pocas semanas y depende de la medida en la que los programadores se adapten con la tecnología.
Planning (Planificación)	Establece el orden y la prioridad de cada característica (requerimiento) indicado por el cliente. Los programadores deben estimar el esfuerzo y tiempo para la ejecución del proyecto la cual no debería de excederse de 2 meses.
Iterations to first release (Iteraciones para la primera liberación)	Se debe de realizar varias iteraciones antes de entregar el proyecto final. En la etapa de planificación se tiene que definir la cantidad de iteraciones (de 1 a 4 semanas), ya que debe realizarse conforme las prioridades indicadas por el cliente. Las pruebas a nivel de funcionamiento se deben ejecutar cuando concluya cada iteración para visualizar el momento en el que este proyecto sea concluido.
Production(Producción)	Verifica el funcionamiento correcto del sistema mediante pruebas adicionales para luego poder hacer la entrega completa al cliente. En esta etapa aún se pueden realizar modificaciones, pero se necesitaría que las iteraciones sean más aceleradas.
Maintenance and Death (Mantenimiento y muerte)	En esta etapa el cliente ya utiliza el sistema, pero se le brinda mantenimiento en caso de alguna corrección adicional. De igual forma se manejan iteraciones para el avance en curso: cambiar la estructuración del equipo y en caso de ser necesario incluir nuevo personal al módulo de trabajo. La fase de "muerte" se puede dar por tres situaciones: 1- El cliente ya no requiere algún otro cambio o modificación en el sistema, 2- El sistema no está funcionando de la forma esperada o 3- Genera un costo muy elevado para continuar ejecutándose.

Cuadro 2. Fases XP

Pre-Juego (Pre-game)	Incluye subfases: 1- Planificación: En esta subfase se detalle el sistema que se está desarrollando. Contiene todos los requisitos a partir del cliente, ventas, mercadeo, comunicación con el consumidor y el desarrollo del software, los cuales son priorizados para su ejecución. También se define: el grupo de trabajo, herramientas, recursos, entrenamientos, riesgos, entre otros. En cada iteración, el equipo de Scrum debe revisar el producto que va atrasado para entregar la iteración completa en el siguiente entregable. 2-Arquitectura: Se planifica con base a los avances actuales y en las pilas de trabajo que se tienen. En caso de una posible mejora al sistema actual, es necesario conocer lo elementos que presenta así como los problemas que se podrían causar si se modifica. Se realiza una reunión para la revisión del diseño y obtener las propuestas de la aplicación y tomar las esta decisiones revisatorias.
Desarrollo (Development)	También conocida como fase de juego, se caracteriza por ser el enfoque ágil de Scrum, el cual se ve como un "cuadro negro" donde lo impredecible es esperado. En esta fase las variables técnicas como recursos o requisitos y las ambientales se identifican en Scrum como cambiantes durante el proceso de desarrollo, por lo que son observados y se controlan por medio de diversas prácticas Scrum en los Sprints. Esto se hace ya que en lugar de solo ser considerados al inicio del desarrollo del sistema, Scrum considera que se tienen que controlar de forma constante durante todo el proyecto con el fin de poder adaptarse fácilmente a los posibles cambios que puedan surgir. En esta fase el sistema se desarrolla por medio de Sprints, estos son ciclos iterativos en el cual la funcionalidad del sistema se desarrolla o mejora para brindar nuevos incrementos. Cada Sprint va a incluir las fases tradicionales como: requerimientos, diseño, análisis, desarrollo y entrega. Tiene una duración de 1 semana a un mes.
Postgame	Es el cierre de la entrega, en la que se han completado todos los requerimientos solicitados por el cliente. En esta fase ya no pueden agregar más requerimientos, el sistema está completo, se incluye pruebas de integración y documentación.

Cuadro 3. Fases SCRUM

Scrum Master	Es el encargado del desarrollo del proyecto, asegurándose que se cumplan acorde a las prácticas y reglas de Scrum. Así como también de quitar posibles obstáculos para asegurar la efectividad en la productividad tanto como sea posible. Debe de interactuar con el cliente y el equipo de desarrollo en todo momento.
Product Owner (Dueño del producto)	Es escogido por el Scrum Master, el cliente y el administrador para ser responsable del proyecto, encargándose de las decisiones finales en las tareas relacionadas a las pilas del producto, participa también en la estimación del esfuerzo del desarrollo y se asegura de que la demora detectada en el proyecto se desarrolló lo más pronto posible.
Scrum Team	Es el equipo de trabajo del proyecto, decide sobre las medidas necesaria para alcanzar los objetivos definidos en cada Sprint.
Cliente	Participa a lo largo del desarrollo del proyecto, tiene una interacción directa con el Scrum Master y el Product Owner.
Management (Administrador)	Se encarga de tomar las decisiones finales según los estatutos, convenios y normas establecidas en el proyecto. También se encarga de establecer objetivos y requisitos, por ejemplo: midiendo el progreso del proyecto.

Cuadro 4. Roles SCRUM

Scrum	XP - Extreme Programming	Kanban
Se implementa en proyectos que puedan llegar a presentar cambios rápidos o inclusión de nuevos requerimientos a los inicialmente planteados (Fau, s.f.), concentrándose en la forma en que los miembros del equipo deben de trabajar para producir un sistema de manera flexible que involucra variables tanto ambientales como técnicas durante todo el proceso (Pekka Abrahamsson, 2002). Además, Scrum permite mejorar prácticas de ingeniería ya existentes en la compañía para identificar posibles deficiencias en el desarrollo así como en la práctica utilizada (Pekka Abrahamsson, 2002).	Fomenta el trabajo en equipo. Se preocupa por el aprendizaje de los desarrolladores y brinda un buen clima de trabajo (Fau, s.f.). Retroalimentación continua entre el equipo de desarrollo y el cliente, buena comunicación entre todos los participantes, soluciones simples (Fau, s.f.). Se adecua aproyectos con requisitos imprecisos, cambiantes y donde hay alto riesgo técnico (Fau, s.f.).	La cantidad de elementos con que se trabaja debe ser limitada, ya que mide tiempo de espera y se llevan las entregas de una forma detallada por procesos, según el avance en los tiempos de espera para la ejecución de cada etapa (Skarin, 2010).

Cuadro 5. Utilidad de las Metologias Agiles

Scrum	XP-Extreme Programming	Kanban
Los principales roles son (Agile Alliance, 2002): Product Owner (dueño de proceso) Scrum Team (Equipo Scrum) Scrum Master (Encargado Scrum) Pero además de estos se mencionan también (Pekka Abrahamsson, 2002) los roles de: Customer/Stakeholder (cliente) Management (Director o Gestor)	En algunos materiales didácticos se menciona que los roles principales son (Agile Alliance, 2002): Coach (entrenador) Programmer (programador) Tester (Encargado de pruebas) Customer (cliente) Consultant (consultor) Sin embargo, hay otros autores (Pekka Abrahamsson, 2002) que además de los roles indicados anteriormente también agregan el de Tracker (encargado de seguimiento) y Manager (Gerente)	No hay roles definidos o establecidos pero esto no implica que se puede definir alguno si fuera necesario (Skarin, 2010)

Cuadro 6. Roles de las Metodologías Ágiles

Por tamaño de proyecto	Este caso en particular se subdivide en tres categorías: pequeño, mediano y gran proyecto, por lo que es muy posible, que se elijan metodologías ágiles diferentes, según cada subdivisión. En este apartado también se tomará en cuenta la complejidad del proyecto para definir el tamaño del mismo, por lo general proyectos grandes, los cuales son complejos.
Por tipo de cliente	En este sumario se presentan dos variables: cuando el cliente es externo y cuando el cliente es interno.
Por involucramiento del cliente	Se define basado en cuan inmerso está el cliente durante el desarrollo del proyecto.
Por fases de desarrollo	Uno de los enfoques más conocidos según las metodologías tradicionales de administración de proyectos, donde se entrega fases del plan y no se continúa, hasta que se haya terminado la anterior, esta se elige debido a que actualmente se aplica en las empresas que utilizan fases como lo establece el PmBoK
Por contrato externo de mantenimiento	Es otra modalidad muy utilizada por las empresas para dar soporte a proyectos de TI, se incluyen proyectos de desarrollo de software, de redes e incluso de infraestructura, se elige debido a que abarca todas las áreas de TI. Dentro de este grupo caben la instalación de paquetes de software
Por inteligencia de negocio	Este modelo es una herramienta de ámbito múltiple, que hace más de diez años usan las empresas para obtención de datos claves para el negocio.

Cuadro 7. Tipo de Proyecto