UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas
Carrera de Ingeniería en Sistemas Computacionales

BENCHMARKING METODOLOGÍAS HÍBRIDAS PARA EL DESARROLLO DE SOFTWARE

PROTOTIPO SISTEMA DIDÁCTICO

AUTOR

Mery Elizabeth Mesa Andrango

DIRECTORA

MSc. Daisy Imbaquingo

Ibarra – Ecuador

2017

UNIVERSIDAD TÉCNICA DEL NORTE BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

IDENTIFICACIÓN DE LA OBRA

La UNIVERSIDAD TÉCNICA DEL NORTE dentro del proyecto Repositorio Digital Institucional, determina la necesidad de disponer de textos completos en formato digital con la finalidad de apoyar los procesos de investigación, docencia y extensión de la Universidad.

Por medio del presente documento dejo sentada mi voluntad de participar en este proyecto, para lo cual pongo a disposición la siguiente información.

DATOS DEL CONTACTO	
CÉDULA DE IDENTIDAD:	1004013338
NOMBRES Y APELLIDOS:	MESA ANDRANGO MERY ELIZABETH
DIRECCIÓN:	PANAMERICANA ANTIGUA – LOS OVALOS ALTO
EMAIL:	elizabethmesa16@gmail.com
TELÉFONO MÓVIL:	0993524592
DATOS DE LA OBRA	
TÍTULO:	“BENCHMARKING METODOLOGÍAS HÍBRIDAS PARA EL DESARROLLO DE SOFTWARE” PROTOTIPO “SISTEMA DIDÁCTICO”.
AUTOR:	MESA ANDRANGO MERY ELIZABETH
FECHA:	
PROGRAMA:	PREGRADO
TÍTULO POR EL QUE OPTA:	INGENIERO EN SISTEMAS COMPUTACIONALES
DIRECTOR:	MSC. DAISY IMBAQUINGO

AUTORIZACIÓN DE USO A FAVOR DE LA UNIVERSIDAD

Yo, Mery Elizabeth Mesa Andrango, con cédula de identidad Nro. 1004013338, en calidad de autor y titular de los derechos patrimoniales del trabajo de grado descrito anteriormente, hago entrega del ejemplar respectivo en formato digital y autorizo a la Universidad Técnica del Norte, la publicación del trabajo en el Repositorio Digital Institucional y uso del archivo digital en la Biblioteca de la Universidad con fines académicos, para ampliar la disponibilidad del material y como apoyo a la educación, investigación y extensión; en concordancia con la Ley de Educación Superior Artículo 144.

CONSTANCIAS

El autor manifiesta que la obra de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, es el titular de los derechos patrimoniales, por lo que asume la responsabilidad sobre el contenido de la misma y saldrá en defensa de la Universidad en caso de reclamación por parte de terceros.

(Firma).....

Nombre: Mery Elizabeth Mesa Andrango

Cédula: 100401333-8

Ibarra, Mayo del 2017

UNIVERSIDAD TÉCNICA DEL NORTE

CESIÓN DE DERECHOS DE AUTOR

A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

Yo, Mery Elizabeth Mesa Andrango, con cédula de identidad Nro. 1004013338, manifiesto mi voluntad de ceder a la Universidad Técnica del Norte los derechos patrimoniales consagrados en la Ley de Propiedad Intelectual del Ecuador, artículos 4, 5, 6, en calidad de autor del trabajo de grado denominado "Benchmarking Metodologías Híbridas para el desarrollo de software." "Prototipo Sistema Didáctico" que ha sido desarrollado para optar por el título de Ingeniero en Sistemas Computacionales, en la Universidad Técnica del Norte, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En mi condición de autor me reservo los derechos morales de la obra antes citada, aclarando que el trabajo aquí descrito es de mi autoría y que no ha sido previamente presentado para ningún grado o calificación profesional.

En concordancia suscribo este documento en el momento que hago entrega del trabajo final en formato impreso y digital a la biblioteca de la Universidad Técnica del Norte.

(Firma): .....

Nombre: Mery Elizabeth Mesa Andrango

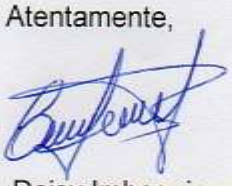
Cédula: 100401333-8

Ibarra, Mayo del 2017

CERTIFICACIÓN

Certifico que la Tesis previa a la obtención del título de Ingeniero en Sistemas Computacionales con el tema: **"Benchmarking Metodologías Híbridas para el desarrollo de software" "Prototipo Sistema Didáctico"**,ha sido desarrollada y terminada en su totalidad por la Srta. Mery Elizabeth Mesa Andrango con C.C. 100401333-8 bajo mi supervisión para lo cual firmo en constancia.

Atentamente,



MSc. Daisy Imbaquingo

DIRECTOR DE TESIS

DEDICATORIA

En primer lugar, doy gracias a Dios por darme la vida y la oportunidad de haber llegado a cumplir mi objetivo principal.

Este trabajo está dedicado a mis padres por el gran esfuerzo que hicieron en todos estos años de apoyo, quienes incondicionalmente siguieron paso a paso hasta ver mi objetivo cumplido.

A mis hermanas y hermano que estuvieron apoyándome en los buenos y malos momentos de mi carrera Universitaria.

Y al resto de mi familia que son parte importante en mi vida que estuvieron brindándome apoyo en todo este tiempo deseándome lo mejor hasta ver con mi meta cumplida.

AGRADECIMIENTO

A mis padres, hermanas y hermano que estuvieron apoyándome con todos los recursos necesarios, su constante amor y confianza que me brindaron en los momentos más difíciles de mi carrera universitaria.

A todos mis compañeros de carrera que fuimos cada día creciendo más y más en conocimientos y en personalidad.

Un agradecimiento a la MSc. Daisy Imbaquingo directora de tesis, Ing. Marco Pusda, Ing. Mauricio Rea, quien con su experiencia y suficientes conocimientos supieron guiarme de la mejor manera en este Trabajo.

A cada uno de mis maestros quienes infundieron conocimientos necesarios para llegar a culminar la carrera con la mejor de las enseñanzas.

Un agradecimiento especial a la Universidad Técnica del Norte, a la carrera de Ingeniería en Sistemas Computacionales, que representó mi segundo hogar en la cual se quedan los mejores momentos de mi carrera universitaria y me llevo las mejores enseñanzas de mis maestros.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En este Trabajo de Titulación se realiza un Benchmarking para comparar la calidad de vida del software, entre dos metodologías híbridas: EssUP (Essentials Unified Process), una combinación entre Scrum, RUP y Scrum/XP una mezcla de las mejores prácticas entre estas dos metodologías; en base a la Norma ISO/IEC 12207, que tiene a su disposición 17 procesos los cuales están subdividos en procesos principales, procesos de apoyo, procesos de organización, de los cuales a su vez se subdividen en 73 parámetros de comparación, que a partir de análisis realizado se determina a una de las dos metodologías como la mejor opción en el desarrollo de software, con la finalidad de aplicarla en el diseño del sistema para enseñanza de inglés para niños que describe cada una de las etapas de desarrollo del sistema web. Contiene una breve introducción a la realización de este trabajo, en el que contiene el problema, objetivos, alcance y justificación que fueron planificados para el desarrollo. En el capítulo uno contiene información acerca de las tres metodologías para desarrollo de software: tradicionales, agiles e híbridas, se conoce contenido más a fondo para poder realizar la comparativa entre estas dos metodologías híbridas. El capítulo dos contiene los parámetros de comparación en base a la Norma ISO/IEC 12207, con el objetivo de definir cuál de las dos metodologías se desarrollará el sistema. El capítulo tres contiene el desarrollo del sistema para enseñanza de inglés, aplicando la metodología que salió a partir del Benchmarking. Finalmente se añade análisis de impactos, conclusiones, recomendación y referencias bibliográficas.

SUMMARY

In this work of titration is carried out a Benchmarking to compare the quality of life of software, between two hybrid methodologies: EssUP (Essentials Unified Process), a combination of Scrum, RUP and Scrum/XP a blend of best practices between these two methodologies; based on the standard ISO/IEC 12207, having at your disposal 17 processes which are subdividos in main processes, support, processes of organization processes, which in turn are subdivided into 73 parameters of comparison, that analysis is determined by one of the two methodologies as the best option in software development , with the purpose of its application in the design of the system for teaching English to children describing each of the stages of development of the web system. It contains a brief introduction to the realization of this work, which contains the problem, objectives, scope and rationale that were planned for development. In chapter one contains information about the three methodologies for software development: traditional, agile and hybrid, referred to content more thoroughly in order to make the comparison between these two hybrid methodologies. Chapter two contains the parameters of comparison based on the standard ISO/IEC 12207, aiming to define which of the two methodologies will develop the system. Chapter three contains the development of the system for teaching English, by applying the methodology that came out from the Benchmarking. Finally add analysis of impacts, conclusions, recommendation and references.

INTRODUCCIÓN

1.1 Tema

“BENCHMARKING METODOLOGÍAS HÍBRIDAS PARA EL DESARROLLO DE SOFTWARE, PROTOTIPO SISTEMA DIDÁCTICO”

1.2 Problema

1.2.1 Antecedentes

Metodologías tradicionales son muy conocidas en el medio porque llevan una documentación extensa y a veces innecesaria, sus costos son muy elevados cuando se necesita realizar algún cambio, y al no poder dar una solución adecuada al proyecto que está en desarrollo. Se caracterizan por tener un ciclo de vida de software, en cascada y en espiral que son utilizadas por este tipo de metodologías. Las Metodologías Agiles fueron diseñadas para realizar software de manera rápida y que se vaya adaptando a los cambios que pueden surgir en el transcurso del desarrollo.

Las Metodologías hibridas retoman las ventajas de las Metodologías Tradicionales y Ágiles, para así lograr una combinación de las mejores prácticas que posteriormente vendría a ser utilizadas en el desarrollo de software.

La técnica está en mezclar las metodologías tradicionales con las ágiles para poder obtener una metodología hibrida más sencilla de manejar, que no demande de mucho tiempo para su realización y no exceda costos más allá de los que el cliente pueda cubrir, lo que se trata es que se pueda tener un producto de calidad en el menor tiempo posible y se vaya adaptando a los cambios que se presenten.

1.3 Situación Actual

1.3.1 De la Investigación

En la actualidad es un paradigma que las empresas de desarrollo de software utilizan metodologías tradicionales y agiles, porque desconocen la utilización de metodologías híbridas porque no existe información suficiente acerca de este tema, pero en otros países está en marcha la implementación de este tipo de metodologías para demostrar que sus aplicaciones pueden ser desarrolladas en menor tiempo, los costos no son tan elevados como implican las metodologías expuestas anteriormente y la documentación se desarrolla solo la necesaria.

1.3.2 Del Prototipo

En la actualidad el método de enseñanza en escuelas es tradicional, se lo realiza en base a libros, manuales, revistas, pero con el desarrollo de este sistema didáctico para la enseñanza del inglés se quiere mejorar la calidad de la educación haciéndola más llamativa y con un mejor entendimiento y no crear un ambiente de aburrimiento dentro del aula.

1.4 Prospectiva

1.4.1 De la Investigación

Mediante el análisis que posteriormente realizaremos, se obtendrá una documentación necesaria para todos aquellos que quieran y necesiten desarrollar software en base a las metodologías híbridas, de esta manera se quiere poner en práctica en la Universidad Técnica del Norte Facultad de Ingeniería en Ciencias Aplicadas en la Escuela de Ingeniería en Sistemas Computacionales para que posteriormente pueda ser conocida dentro y fuera de la provincia.

1.4.2 Del Prototipo

Con el desarrollo del sistema didáctico se demostró que este tipo de metodologías son factibles en el desarrollo de software a corto plazo.

1.5 Planteamiento del problema

¿Cómo el benchmarking de las metodologías híbridas plantea ayudar en el desarrollo de software?

En el Ecuador existe desconocimiento sobre este tipo de metodologías híbridas que son utilizadas para el desarrollo de software, es por eso que con el estudio a realizarse se quiere difundir información sobre la misma obteniendo ventajas como reducir costos y tiempo en su posterior desarrollo.

¿Por qué realizar un sistema didáctico?

Con el fin de aportar con un granito de arena en la educación de los niños, para que mejoren su aprendizaje visualizando desde otra perspectiva influyendo en su lenguaje de enseñanza tanto de parte de los docentes como del estudiante.

1.6 Objetivos

1.6.1 Objetivo General

Realizar un estudio comparativo de metodologías híbridas con el fin de conocer su utilidad, funcionalidad y empleo en el desarrollo software.

1.6.2 Objetivos Específicos

- Recolectar de información sobre metodologías tradicionales, agiles e híbridas
- Procesar y analizar de metodologías híbridas
- Determinar parámetros de comparación entre las metodologías híbridas
- Desarrollar el sistema didáctico para la enseñanza de inglés.

1.7 Alcance

1.7.1 De la Investigación

El presente estudio se analizó las características, ventajas, desventajas de cada una de las metodologías que se estudió en el desarrollo de software realizando una comparativa para determinar cuál de las metodologías es más factible para su desarrollo para ello se comparó las siguientes metodologías:

EssUP y Scrum/XP

Metodologías híbridas Ventajas

Tabla 1: Metodologías a comparar

ESSUP	**MÉTODO ÁGIL HÍBRIDO Scrum/XP**
Respuesta rápida a los cambios	Construcción basada en características
Flexibilidad	Documentación Ágil
Menor tiempo	Ciclos cortos de trabajo
Costos bajos	
Roles que se deben manejar	

Fuente: Propia

1.7.2 Del Prototipo

Para su respectiva aplicación posteriormente se desarrolló el sistema didáctico para enseñanza de inglés para niños aplicando una de las metodologías que salgan a partir de la comparativa.

El prototipo corresponde a un sistema de enseñanza para escuelas orientado a niños comprendidos en edad de 7 años, que permitirá una interacción entre la herramienta

informática, docentes y niños que tienen la necesidad de aprender inglés mediante este tipo de herramientas que ayudan a mejorar la calidad de la educación.

1.7.3 Arquitectura Cliente - Servidor

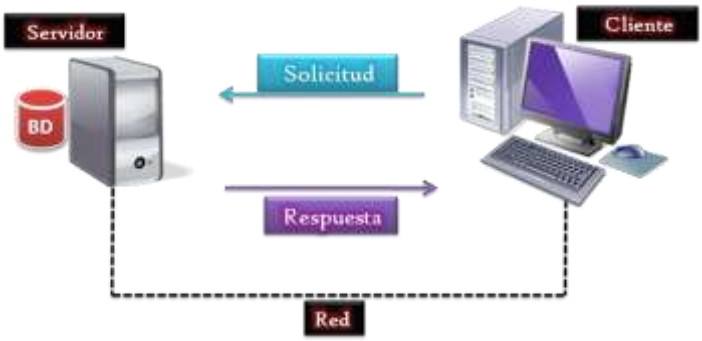


Figura 1: Arquitectura Cliente – Servidor
Fuente: (Ulyla, 2010)

1.7.4 Aplicativo

El aplicativo se diseñó con la finalidad de hacer un reconocimiento de las palabras básicas del inglés que ayuden a mejorar el entendimiento en los niños.

- Reconocimiento de alfabeto en inglés
- Reconocimiento sobre animales en inglés
- Reconocimiento sobre números en inglés
- Reconocimiento sobre palabras básicas en inglés

1.8 Justificación

1.8.1 De la investigación

Con la utilización de las Metodologías Híbridas se espera hacer un reconocimiento de la importancia de la misma a todos los sectores de la industria del software proporcionando mayor información acerca de este tipo de metodologías, conociendo las ventajas y desventajas de la misma y su potencialidad para desarrollo de software.

Se obtuvo suficiente documentación para la orientación de estudiantes y docentes que conocen sobre esta nueva tendencia en el área de ingeniería de software, que los estudiantes puedan seguir investigando sobre diferentes tipos de metodologías híbridas y su aplicación en casos reales para su desarrollo.

1.8.2 Impacto económico y social

Con la implementación de las metodologías híbridas se quiere reducir costos a nivel de desarrollo de software para así lograr cubrir con las necesidades del cliente sin salirnos fuera del presupuesto analizado en un inicio.

En lo social difundir información acerca de este de este tipo de metodologías para lograr un cambio en el desarrollo de software eligiendo esta nueva tendencia en el área de Ingeniería del software.

1.8.3 Del Aplicativo

En la actualidad el método de enseñanza es tradicional, se lo realiza a base libros. Se quiere llegar con esta herramienta informática a tener un aprendizaje interactivo con los niños para así lograr un entendimiento y captación mejor de lo que se necesita que los niños aprendan y así lograr una educación de calidad.

En la actualidad no existe software desarrollado o que este por desarrollarse en este tipo de metodologías en nuestro país.

1.9 Contexto

Luego de haber revisado los proyectos de tesis en la Biblioteca en la Universidad Técnica del Norte específicamente de la Facultad de Ingeniería en Ciencias Aplicadas carrera Ingeniería Sistemas Computacionales no existe algún tema que haga referencia al estudio que se haya revisado entes de la presentación del Anteproyecto de trabajo de grado.

CAPÍTULO I

2.1 MARCO TEÓRICO

El presente capitulo detalla a profundidad las bases esenciales sobre los conocimientos previos para la posterior realización del trabajo de investigación comparativo propuesto o llamado también benchmarking, por ejemplo, las metodologías tradicionales, metodologías agiles, metodologías híbridas para el desarrollo de software, sus principales definiciones, concepto de benchmarking y elementos a utilizar.

2.2 Benchmarking

“Es un proceso sistemático y continuo para evaluar los productos, servicios y procesos de trabajo de las organizaciones que son reconocidas como representantes de las mejores prácticas, con el propósito de realizar mejoras organizacionales”.(Campos, 2016)

Los aspectos principales que sobresalen en el benchmarking son:

- La calidad
- La productividad
- El tiempo

El benchmarking es aplicable en todos los procesos, que necesiten ser mejorados o actualizados, viene a ser una herramienta de mejoramiento continuo, es decir que debe estar en constante estudio de sus propios procesos, para llegar a cumplir una meta u objetivos planteados en el inicio del benchmarking, es una muy buena forma de recolectar información en el área de trabajo que se esté aplicando.

2.3 Ingeniería del software

Se realizará una pequeña definición acerca de la ingeniería del software, ya que trabajo investigativo está relacionado con los elementos principales para el desarrollo de software.

2.3.1 Software

La Ingeniería del software es un conjunto métodos y procedimientos que buscan la mejorar la calidad del software, resolviendo problemas orientados a cualquier área satisfaciendo necesidades de los clientes.

El Software se compone de tres elementos principales que son:

Los programas, procedimientos y documentación que vienen relacionados con el desarrollo de los sistemas informáticos. Según lo que define al autor del libro (Pressman, 2010), los elementos son los siguientes:

Programas. Son instrucciones que deben proporcionar funcionalidad sobre alguna función creada, en lenguajes de programación.

Procedimientos. Los programas necesitan de los datos para poder ser manipulados, de la misma manera para realizar el mantenimiento y las pruebas necesarias.

Documentos. Son los manuales de administración y de usuario que deben ser un complemento importante para poder realizar el mantenimiento del mismo.

La ingeniería de software consta de actividades, acciones y tareas que se ejecutan cuando va a crearse algún producto.

La estructura del proceso consta de cinco actividades:

Comunicación. Se busca entender los objetivos de los participantes respecto al proyecto y reunir los requerimientos que ayuden a definir las características y funciones del software.

Planeación. Llamado plan de proyecto se software, que define el trabajo de ingeniería de software al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades a realizarse.

Modelado. Crear modelos a fin de entender mejor los requerimientos del software y el diseño que los asistirá.

Construcción. Combina la generación del código y las pruebas que se requieren para descubrir errores en este.

Despliegue. El software se entrega al consumidor que lo evalúa y que le da retroalimentación, misma que se basa en dicha evaluación.

2.4 Metodologías

“Metodologías para el desarrollo de software es un modo sistemático de realizar gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxitos. Viene a ser procesos a seguir sistemáticamente para idear, implementar y mantener un producto software

desde que surge la necesidad del producto hasta que se cumple con el objetivo para el cual fue creado".(Urs.cod, 2007).

2.4.1 Metodologías Tradicionales

Las metodologías tradicionales son las que necesitan tener más énfasis en la planificación y control del proyecto que se lo está realizando, es por eso que reciben el nombre de metodologías tradicionales o pesadas ya que deben contar con una especificación precisa de requisitos y modelado.

En este tipo de metodologías se impone mucha disciplina en el trabajo sobre su proceso, con el único fin de lograr software de calidad y eficiente. Es por ello que se pone mayor énfasis en la planificación, ya que es desde este punto donde se va a llevar a cabo todas las actividades muy bien detalladas de todo el trabajo a realizarse, luego de esto se centra en lo que es el control del proceso mediante la definición de roles, actividades para su posterior modelado y una documentación muy detallada, las metodologías tradicionales no se ajustan a cualquier cambio, es por eso que no son muy adecuadas cuando se necesita trabajar en entornos que sean variantes ya que sus procedimientos son muy rígidos.

Existen diferentes modelos de desarrollo de software, cada modelo tiene distinto proceso de trabajo, pero deben llegar a satisfacer las necesidades del cliente.

2.4.1.1 Modelo Cascada

Llamado también ciclo de vida clásica, se debe seguir un enfoque sistemático y secuencial para el desarrollo de software, que comienza con la especificación de los requerimientos por parte del cliente, avanza a través de la planeación, modelado, construcción y despliegue para concluir con el apoyo del software terminado.(Pressman, 2010)

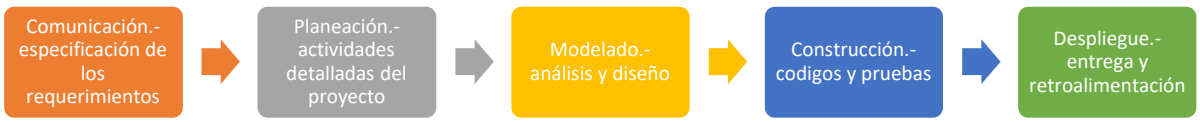


Figura 2: Fases Método Cascada
Fuente: Propia

Existen problemas que surgen al aplicar el modelo cascada los que menciona (Pressman, 2010) y son los siguientes:

1. Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo. Aunque el modelo lineal acepta repeticiones, lo hacen en forma indirecta. Como resultado, los cambios generan confusión conforme el equipo del proyecto avanza.

2. Es difícil para el cliente enunciar en forma explícita todos los requerimientos. El modelo cascada necesita que se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.

3. El cliente debe tener paciencia. No se dispondrá de una función del programa hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento.

Otros problemas que pueden surgir al aplicar este modelo son los estados de bloqueo, es decir que se debe esperar a que otros terminen las tareas que son independientes para seguir con el proceso.

El modelo en cascada es más orientado cuando se tiene requerimientos fijos, entonces el trabajo se avanza en forma lineal hasta llegar a terminar el proyecto.

2.4.1.2 Modelos de proceso incremental

Al combinar elementos de los flujos de proceso lineal y paralelo, se refiere al proceso del software, la estructura y las actividades sombrillas que son las que se llevan a cabo durante todo tiempo que dure el proyecto, se encargan de administrar, controlar el avance, el cambio y el riesgo.

Cuando se utiliza el modelo incremental, es frecuente que el primer incremento sea el producto fundamental, es decir se abordan los requerimientos básicos, pero no se proporcionan muchas características suplementarias. El cliente usa el producto fundamental o lo somete a una evaluación detallada como resultado del uso o evaluación se desarrolla un plan para el incremento que sigue. El plan incluye la modificación del producto fundamental para cumplir mejor las necesidades del cliente, así como la entrega de características adicionales y más funcionalidad. Este proceso se repite después de entregar cada incremento, hasta terminar el producto final.(Pressman, 2010)

En el modelo incremental se van entregando incrementos que deben estar operando, es así como el cliente puede ir evaluando para obtener un producto de calidad.

El proceso del desarrollo es igual al modelo en cascada la diferencia es que en el modelo incremental se debe ir entregando incrementos, que hagan que el cliente vaya quedando satisfecho y así evitar retrasos en la entrega del proyecto.

2.4.1.3 Modelo de Proceso Evolutivo

Los modelos evolutivos son iterativos. Se caracteriza por la manera en que permiten desarrollar versiones cada vez más completas del software.

Proceso evolutivo

Hacer prototipos. Los prototipos son de mucha ayuda en los casos donde los requerimientos no estén muy claros.

Se empieza por la comunicación, es aquí donde se van a definir los objetivos generales con los demás participantes, identifica los requerimientos, a partir de esto se planifica una iteración para hacer el prototipo a continuación se lleva a cabo el modelado, en esta fase inicial se diseña la interfaz de usuario y pantallas de salida como por ejemplo una pantalla de ingreso de datos, luego de realizar este prototipo se presenta a los participantes para que puedan hacer una retroalimentación para mejorar los requerimientos. La iteración ocurre a medida de que el prototipo es afinado para satisfacer las necesidades de los diferentes participantes y al mismo tiempo permite entender mejor de lo que se necesita hacer.(Pressman, 2010)

Para entender mejor el proceso visualizar la **Figura. 3**.

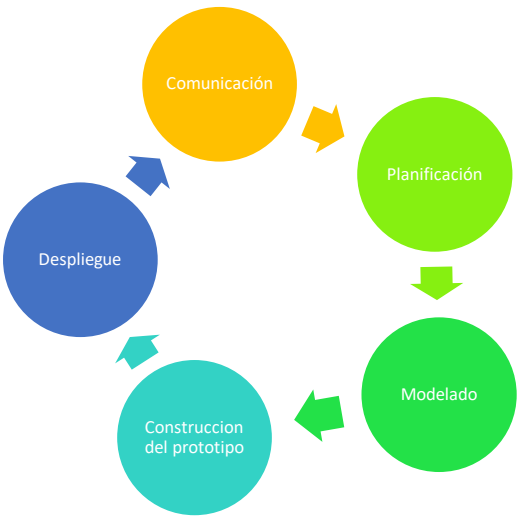


Figura 3: Proceso Evolutivo
Fuente: Propia

Algunos prototipos son desechables, pero otros se convierten en evolutivos, y llegan a transformarse en el sistema real, las reglas deben estar claras desde el momento en que se inicia con el proyecto, es una clave muy importante, a partir de aquí deben estar de acuerdo todos los participantes con el prototipo que debe servir para tener claros los requerimientos del cliente y posteriormente presentar un producto de calidad.

2.4.1.4 Modelo Espiral

El modelo espiral es un enfoque realista para el desarrollo de sistemas y de software a gran escala. Como el software evoluciona a medida que el proceso avanza, el desarrollo y el cliente comprende para reaccionar mejor ante los riesgos en cada nivel de evolución. El modelo espiral usa los prototipos como mecanismo de reducción de riesgos, pero más importante permite aplicar el enfoque de hacer prototipos como mecanismo de reducción de riesgos, pero lo más importante, permite aplicar el enfoque de hacer prototipos en cualquier etapa de la evolución del producto. Mantiene el enfoque de escalón sistemático sugerido por el ciclo de vida clásico, pero lo incorpora en una estructura iterativa que refleja al mundo real en una forma más realista.(Pressman, 2010)

Figura 4: Fases Modelo Espiral
Fuente:(Apr.com, 2015)

El primer circuito alrededor de la espiral da como resultado el desarrollo de una especificación del producto; las vueltas sucesivas se usan para desarrollar un prototipo y luego versiones cada vez más sofisticadas del software. Cada paso por la región de la planeación da como resultado ajustes en el plan de proyecto. El costo y la programación de actividades se ajustan con base en la retroalimentación obtenida del cliente después de la entrega.(Pressman, 2010)

2.4.2 Tipos de metodologías tradicionales

Entre las metodologías tradicionales más reconocidas se mencionan a las siguientes:

- RUP (Rational Unified Procces)
- MSF (Microsoft Solution Framework)
- Win-Win Spiral Model
- Iconix

2.4.2.1 Metodología RUP (Rational Unified Process)

Rational Unified Process que en español significa Proceso Unificado Racional (RUP), es un proceso de desarrollo de software el cual implica asignar tareas y responsabilidades, es una de las metodologías más utilizadas por su análisis, implementación y documentación, donde el objetivo principal es presentar software de calidad y satisfacer las necesidades de los usuarios cumpliendo con el tiempo y costos estimados desde el inicio del proyecto.

Este tipo de metodologías trabaja con iteraciones e incrementos, poniendo más énfasis en las tareas donde requieran más atención realizando los casos de usos y la arquitectura que van definiendo la forma y función para el desarrollo del proyecto.

RUP va de la mano con lenguaje de Modelado UML, este tipo de lenguaje es una técnica de modelado que ayuda a definir muy bien las acciones de los individuos por ejemplo como, cuando y porque hacerlo a la vez consiste en vistas, diagramas, símbolos y reglas.

Una iteración es volver a repetir el mismo proceso con el fin de ir puliendo el producto.

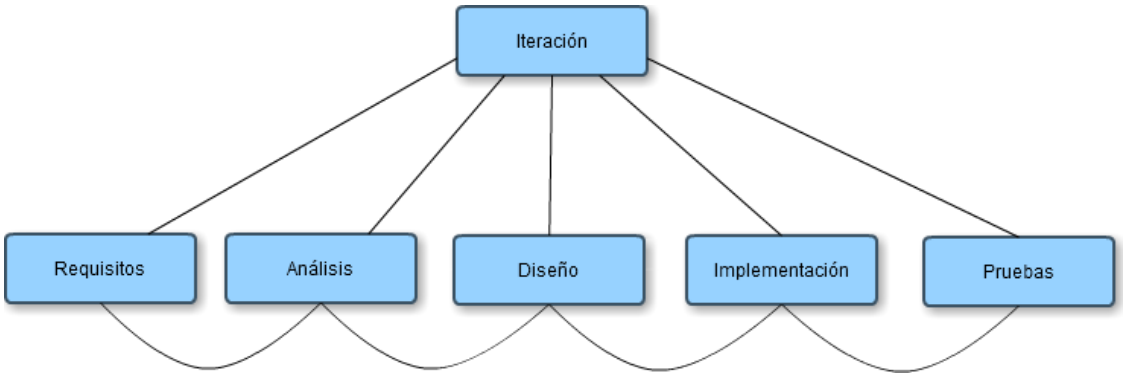


Figura 5: Proceso Iteración
Fuente: Propia

Cada iteración terminada es analizada, para determinar si existen nuevos requerimientos o cambiaron los que ya estaban planteados anteriormente, afectando a las siguientes iteraciones haciendo una retroalimentación para ir ajustando al objetivo principal.

RUP cuenta con cuatro fases de desarrollo, dentro de las cuales deben realizarse las iteraciones.

Ciclo de vida

RUP divide el proceso de desarrollo en ciclos, teniendo un producto final después de haber finalizado cada ciclo.

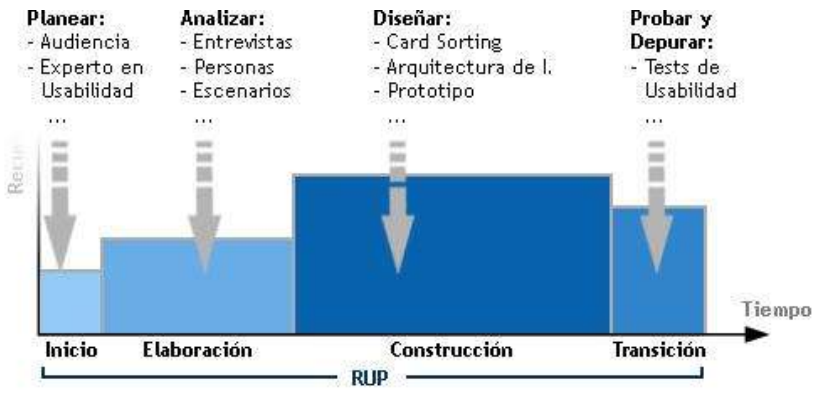


Figura 6: Ciclo de vida RUP
Fuente:(Barco-Camaronero, 2009)

La fase de inicio se define el alcance del proyecto, para lograr esto se debe identificar los actores principales y los casos de usos, se necesita desarrollar un plan para determinar los recursos que deben ser asignados al proyecto. Según (Ponce, 2011), manifiesta que el resultado en la fase inicio es el siguiente:

- Requerimientos básicos del proyecto, características dominantes y de las restricciones principales.
- Un modelo inicial de casos de uso
- Glosario inicial del proyecto
- Una estimación de riesgo inicial
- Un plan de proyecto, demostrando fases e iteraciones
- Un modelo de negocio, en caso de necesidad
- Uno o más propósitos

Fase de elaboración

El propósito de esta fase es de analizar el dominio del problema, desarrollar el plan del proyecto y eliminar los elementos de riesgo, a partir de aquí ya se puede construir un prototipo ejecutable de una o más iteraciones.(Ponce, 2011). Según (Ponce, 2011), manifiesta que el resultado en la fase de elaboración es el siguiente:

- Un modelo de caso de uso por lo menos en 80% completo
- Una descripción de la arquitectura del software
- Un prototipo arquitectónico ejecutable
- Una lista revisada del riesgo y un caso de negocio revisado
- Un plan de desarrollo para el proyecto total, demostrando iteraciones y los criterios de evaluación para cada iteración
- Un manual preliminar de usuario

Fase de Construcción

Durante esta fase de la construcción, todos los componentes y características restantes se desarrollan, se integran en el producto y se prueban a fondo, este proceso es el de fabricación donde el énfasis se pone en manejar los recursos y controlar las operaciones para optimizar costos, tiempos y calidad.(Ponce, 2011)

Según (Ponce, 2011), manifiesta que el resultado en la fase construcción es el siguiente:

- El producto de software integrado n las plataformas adecuadas
- Los manuales de usuario
- Una descripción de la versión

Fase de Pruebas

El propósito de esta fase es la transición del producto de software al ambiente de producción. Una vez que el producto se haya entregado al usuario final, surgen algunos temas que llevan al desarrollo de nuevas versiones, a corregir errores o a terminar algunas características que habían sido pospuestas.(Ponce, 2011)

Se ingresa a esta fase cuando el producto está lo suficientemente maduro para comenzar a pasar a producción. Esto requiere que un cierto subconjunto del sistema se encuentre en un nivel aceptable de la calidad y que la documentación del usuario está disponible de modo que la transición proporcione resultados positivos para todas las partes.(Ponce, 2011)

Según (Ponce, 2011)concluye en:

- La prueba beta para validar el nuevo sistema contra las expectativas del usuario.
- Operación en paralelo con un sistema anterior que el nuevo sistema este sustituyendo.
- La conversión de las bases de datos operacionales
- Entrenamientos y capacitación de los usuarios y la gente de mantenimiento.

2.4.2.2 Metodología MSF

Es la unión del modelo cascada y espiral, es una metodología flexible e interrelacionada con una serie de conceptos, modelos y prácticas de uso, que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos.(Vergara, 2014)

MSF se centra en los modelos de procesos y de equipo dejando en un segundo plano las elecciones tecnológicas. MSF es un compendio de las mejores prácticas en cuanto a administración de proyectos, MSF es una serie de modelos que pueden adaptarse a cualquier proyecto de tecnología de información.(Vergara, 2014)

Según (Vergara, 2014), describe las siguientes características para MSF

MSF tiene las siguientes características:

Adaptable. Tiene similitud a un compás, usando en cualquier parte como un mapa, del cual su uso es limitado a un específico lugar.

Escalable. Organiza equipos pequeños de 3 o 4 personas, y también equipos grandes que pueden superar las 50 personas dependiendo de qué tan grande sea el proyecto.

Flexible. Se ajusta a diferentes ambientes de desarrollo para cualquier cliente

Tecnología Agnóstica. Puede ser diseñado sobre cualquier base tecnológica, es decir que puedes ser desarrollada en cualquier tecnología cumpliendo con los requisitos necesarios.

MSF se compone de varios modelos encargados de planificar las diferentes partes implicadas en el desarrollo de un proyecto:

Modelo de Arquitectura del Proyecto, Modelo de Equipo, Modelo de Proceso, Modelo de Gestión del Riesgo, Modelo de Diseño de Proceso y finalmente el modelo de Aplicación. Según (Vergara, 2014), las fases son las siguientes:

Modelo de Arquitectura del Proyecto. Diseñado para acortar la planificación del ciclo de vida. Este modelo define las pautas para construir proyectos empresariales a través del lanzamiento de versiones.

Modelo de Equipo. Fue diseñado para que el equipo de desarrollo pueda mejorar el rendimiento.

Modelo de Proceso. Mejora el control del proyecto, en el que debe minimizar los riesgos y a la vez aumentar la calidad del producto, en el menor tiempo posible para realizar la entrega. Teniendo una relación de comunicación con el Modelo del Equipo.

Modelo de Gestión del Riesgo. Diseñado para ayudar al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlar las emergencias que puedan surgir. Este modelo proporciona un entorno estructurado para la toma de decisiones y acciones valorando los riegos que puedan provocar.

Modelo de Diseño de Proceso. Está diseñado para distinguir entre los objetivos empresariales y las necesidades del usuario. Proporciona un modelo centrado en el usuario para obtener un diseño eficiente y flexible a través de un enfoque iterativo. Las fases de diseño conceptual, lógico y físico proveen tres perspectivas diferentes para los tres tipos de roles: los usuarios, el equipo y los desarrolladores.

Modelo de Aplicación. Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software. Los servicios utilizados en este modelo son escalables y pueden ser usados en un solo ordenador o incluso en varios servidores.

Visión y Alcances. El equipo de trabajo debe tener una visión común y clara de todo lo que se va a realizar para satisfacer las necesidades del cliente, aquí se podrá definir a los líderes del proyecto y responsables, objetivos y metas que son parte importante para empezar en el desarrollo.

Planificación. Es una fase muy importante, se definen especificaciones funcionales, planes de trabajo, diseño de la solución, estimación de costos y cronogramas de los diferentes entregables del proyecto.

Desarrollo. El equipo se dedica a la construcción de los diferentes componentes, se denomina componentes a la documentación y el código.

Estabilización. Test de evaluación sobre la solución en la que el equipo de trabajo lo está realizando, los cuales deben enfocarse en resolver los errores y posteriormente preparar una solución para su lanzamiento.

Implantación. El quipo trabaja en la implementación de la tecnología base y sus componentes relacionados, traspasando el proyecto al personal de soporte y operaciones

para finalmente obtener la aprobación del cliente. Según (Vergara, 2014), describe las siguientes fases de la metodología MSF

Fases de la Metodología MSF

Estrategia y Alcance

En esta fase se encarga de la elaboración del documento de alcance para su posterior aprobación, en este documento deben quedar bien claras las funcionalidades y servicios para llegar a la solución definitiva que vaya a ser implantada.

Se realiza la formación del equipo de trabajo y la distribución de responsabilidades, se definen las áreas de trabajo como el Diseño de la Arquitectura, pruebas, documentación, logística y coordinación.

Elaboración del plan de trabajo. Deben marcar fechas contenidos para esta fase y las siguientes. Los mecanismos y protocolos de intercambio de información y coordinación deben quedar bien establecidos.

Se realiza la elaboración de la matriz de los principales riesgos detectados, y también debe contar con un plan de contingencia.

Planificación y prueba de concepto

En esta fase se realiza la planificación bien detallada de todas las funcionalidades y el diseño de la arquitectura, que son los elementos principales, luego de tener este documento se debe proceder a la aprobación para poder contar con una guía de las tareas a realizarse.

Documentos de plan de laboratorio, prueba de concepto. Descripción de contextos en diversos escenarios, criterios de validez, control de incidencias y las métricas de calidad, son objetivos generales de este tipo de documentos.

Estabilización

La solución que fue diseñada e implantada en una maqueta, debe pasar a entornos real de explotación.

Hitos y objetivos

Un hito es hecho muy importante que marca un punto de referencia, a una meta que cumplir un objetivo hacia donde llegar.

Selección del entorno de prueba piloto: no es el despliegue propiamente, sino una fase de observación en la que es absolutamente crítico establecer unos causes efectivos para la trata de errores.(Vergara, 2014)

Gestión de incidencias

Resolución de problemas, que depende mucho de la fase anterior, depende mucho de la recogida de incidencias es decir como un helpdesk.

Revisión de la documentación final de la Arquitectura

La alteración depende de los resultados de las anteriores fases, donde el documento debe ser aprobado, dando por terminado los trabajos de diseños.

Elaboración de la documentación de formación y operaciones

Se debe desarrollar la documentación necesaria es decir los manuales de usuarios y administración paso a paso para tener un sustento del trabajo realizado.

Elaboración del plan despliegue

Debe fijarse una fecha final para la culminación de la fase piloto, asegurándose de la calidad del producto que va ser entregado al cliente para poder iniciar con el despliegue.

Elaboración del plan de formación

Debe existir una compatibilidad y acuerdos con el plan despliegue.

Despliegue

Se llevan a cabo los planes diseñados en las anteriores fases, además la implantación de la plataforma, puesta en servicio de todas las funciones, formación a los usuarios y administradores.

- Continuación en las labores tratamiento y clasificación de incidencias.
- Registro de mejoras, sugerencias o novedades para realizar incorporaciones y desarrollar un buen proyecto
- Revisión de guías y manuales de usuarios, rectificando errores
- Entrega de documentos definitivos
- Revisión de la matriz de riesgos, métricas de la calidad, estableciendo estándares de calidad.
- Finalmente entrega y cierre de proyecto con o sin apertura.

2.4.2.3 Metodología Win-Win Spiral

Es una adaptación del modelo de espiral que se hace hincapié explícitamente situados en la participación del cliente en un proceso de negociación en la génesis del desarrollo de productos. Sería una buena comunicación entre el desarrollador y el cliente porque el cliente expondría lo que debe ser necesario en el sistema, pero es algo muy complicado hacerlo y no se puede llegar a buenos acuerdos en términos de funcionalidad, rendimiento, costos, entre otros.

El modelo Win-Win se deriva del objetivo de dichas negociaciones que significa ganar-ganar. El cliente recibe el producto que satisface la mayoría de sus necesidades y el desarrollador trabaja para alcanzar presupuestos y fechas de entrega. Para lograr este objetivo, el modelo define un conjunto de actividades de negociación al principio de cada paso alrededor de la espiral.

Las actividades se definen así:

- Identificación de los actores del sistema
- Determinación de las partes interesadas de ganar condiciones
- Las negociaciones de la victoria de las condiciones de las partes interesadas a que reconciliarlos en un conjunto de ganar-ganar para todos los interesados.(José, 2011)

Figura 7: Ciclo de vida de Win-Win Spiral
Fuente: (Toborda & Gonzales, 2015)

El modelo también presenta tres etapas del proceso que vienen a ser puntos de anclaje que ayudan a establecer la realización de un ciclo alrededor de la espiral y proporcionar los hitos de decisión. Según (Páez, 2011) continuación se mencionarán las siguientes ciclos:

Objetivos de ciclo de vida (Life Cycle Objectives - LCO). Se define una serie de objetivos para cada actividad e software más importantes, un conjunto de actividades relacionadas con la definición de los principales requisitos de nivel de producto.

Arquitectura del ciclo de vida (Life Cycle Architecture - LCA). Establece los objetivos que deben cumplirse así como la arquitectura de software.

Capacidad operativa inicial (Initial Operational Capability). Representa un conjunto de objetivos asociados a la preparación del software para la instalación y distribución, la preparación previa a las instalaciones del sitio, asistencia requerida por todas las partes que se utilizara o soporte técnico del software.

Características

- Mejorar los ciclos de vida clásicos y prototipos
- Combina modelos de desarrollo como cascada y evolutivo
- Debe contar con personal cualificativo es decir que tenga mucha experiencia en el área en la que se va desempeñar
- Debe incorporar objetivos de calidad y gestión de riesgos
- Permite iteraciones, vuelta atrás y finalizaciones rápidas

Ventajas

Las ventajas que expone (Grupo-Espiral-Php, 2009)

- Puede adaptarse y aplicarse a lo largo de vida del software
- Reaccionar mejor ante riesgos en cada nivel, puesto que la evolución del software avanza a medida que va en ascenso los procesos.
- Permite la construcción de prototipos que ayudan a tener bosquejos del sistema en real.

Desventajas

Las desventajas que expone (Grupo-Espiral-Php, 2009)

- No es fácil convencer al cliente de que los enfoques evolutivos se puedan controlar
- Este tipo de modelo es de complejidad alta, es por ello que no es recomendable utiliza en sistemas pequeños.(Benítez, 2014)
- El tiempo de desarrollo es elevado

2.4.2.4 Metodología Iconix

Características de Iconix

- Iterativo o incremental. Identificación de casos de usos.
- Trazabilidad. Cada paso esta referenciado por algún requisito. Hay que considerar a la trazabilidad como la capacidad de seguir una relación entre los diferentes artefactos productos.
- Dinámica de UML. Uso de diferentes casos que presenta UML para definir acciones y actores.

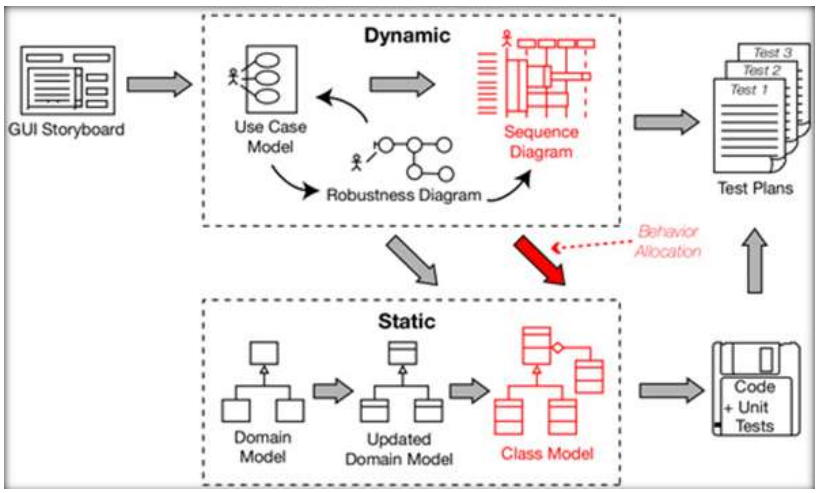


Figura 8: Ciclo de vida de Iconix
Fuente (Córazon, García, Luna, Padilla, & Abigail, 2011)

Consta de 4 fases para su implementación

Fase de Análisis de requisitos

Dentro de la cual podemos citar tres actividades de esta fase:

Modelo de dominio

Abstracción de los objetos y las relaciones de agregación y generalización que existen entre ellos. Utiliza el diagrama de casos de alto nivel.se hace un relevantamiento de todos los requisitos que deben estar en el sistema y con estos construir un diagrama de clases, que representan las agrupaciones funcionales que estructuran el sistema en desarrollo.(Benítez, 2014)

Prototipación rápida

Presentar un prototipo de las interfaces del sistema para ayudar al cliente a comprender mejor el sistema propuesto, para así conocer si el cliente está o no satisfecho con lo que se va a desarrollar de no ser el caso se debe acatar los nuevos cambios que el cliente especifique, así es como los clientes interactúan durante el desarrollo de todo el proyecto.(Benítez, 2014)

Modelo de casos de uso

Realizar los casos de usos del sistema, identificar a los actores que van a estar involucrados en el sistema, definir las acciones que se le atribuirán a cada uno de ellos que dan lugar al comportamiento del sistema, luego de terminar estos casos se procede a revisar los requisitos funcionales que deben estar de acorde al desarrollo de los casos de usos a esto se lo llama Trazabilidad.(Benítez, 2014).

Fase de Análisis y diseño preliminar

Según (Benítez, 2014) determina las siguientes actividades

Descripción de casos de usos

Se describen los casos de usos con un flujo principal de acciones y posibles flujos alternos y de excepción.

Diagrama de robustez

Se deben ilustrar las iteraciones que se vayan a ir realizando entre los objetos que están representados en los casos de usos.

Fase de Diseño

Encontramos las siguientes actividades:

Diagrama de secuencia

Se necesita tener un diagrama de secuencia y un modelo estático.

El diagrama de secuencia muestra iteraciones entre objetos como una vista temporal y es el núcleo del modelo dinámico, la realización del diagrama de secuencia ayuda a tener una forma más clara las interacciones entre objetos y actores dinámicamente.

Completar el modelo estático

En este punto se tiene la necesidad de terminar el modelo estático, añadiendo detalles del diseño en el diagrama de clases y verificar si el diseño satisface todos los requisitos identificados.

Fase de Implementación

Se encuentran las siguientes actividades:

Diagrama de componentes

Se debe diseñar este tipo de diagrama para mostrar la distribución de los elementos del sistema que están siendo desarrollados, describiendo elementos físicos de la estructura interna.

Generar código

Se inicia escribiendo código con la ayuda de cualquier herramienta de lenguajes de programación.

Realización de pruebas

Se hace muchas pruebas de unidad o de integración, corrigiendo errores y luego finalmente se espera la aceptación del cliente.

2.4.3 Metodologías de desarrollo Ágil

Las metodologías agiles son de rápido desarrollo, un equipo ágil es diestro y capaz de responder de manera apropiada a los cambios. El cambio es lo que trata el software a medida.(Pressman, 2010).

La agilidad puede aplicarse a cualquier proceso del software, sin embargo, para lograrlo es esencial que este se diseñe en forma que permita al equipo del proyecto adaptar las tareas y hacerlas directas, ejecutar la planeación de manera que entienda la fluidez de un enfoque ágil del desarrollo, eliminar todos los productos del trabajo excepto los más esenciales.(Pressman, 2010).

Al aplicar las metodologías agiles en un proyecto de desarrollo de software hay que tomar en cuenta que se deben ir entregando incrementos o prototipos del sistema y si hay algún cambio se lo haga de la manera más conveniente y en el menor tiempo posible con el equipo de trabajo, haciendo una retroalimentación necesaria de los cambios a realizare, entonces tenemos una comunicación con el equipo de trabajo y por parte del cliente.

Una de las características de las metodologías agiles es sacar software funcionando en mucho menos tiempo que se hace con las metodologías tradicionales, estaríamos hablando de un tiempo mínimo de dos meses para el desarrollo de sistemas.

Las metodologías agiles son más flexibles a los cambios, es por ello que en este tiempo han tenido una gran acogida por sus diferentes formas de trabajo, no se necesita seguir procedimientos rigurosos ni tampoco asignar muchos roles al equipo de trabajo como también puede adaptarse a las nuevas tecnologías.

En los tiempos en los que nos encontramos ya no se utilizan mucho las metodologías tradicionales por sus rigurosos procedimientos, y que no son adaptables a cualquier cambio que surja, pero no hay que dejar atrás tampoco que al aplicar estas metodologías garantizamos la calidad y eficiencia del producto. Ahora han surgido las metodologías agiles donde hacen más rápido el desarrollo de los sistemas en menor tiempo y costos, los procesos se los va realizando por medio de iteraciones, haciendo una retroalimentación es decir aportando nuevas ideas, en cada uno de ellos, eliminando los que no ayudan al éxito del mismo y así no se arriesga la calidad del producto final.

La comunicación con el cliente es muy esencial y es una de la principales características de las metodologías agiles siendo el mismo el que ira evaluando el producto cada vez que se haga iteraciones, se debe mantener dicha comunicación para el intercambio de información que ayude a mejorar el proceso de desarrollo, dando lugar a que el cliente sea un miembro más del equipo y no se mantenga fuera del proyecto, ya que es el dueño del negocio y por lo tanto debe velar por los interese de su empresa o negocio, según como vaya avanzando el proceso de desarrollo y las iteraciones el cliente podrá ir representando beneficios económicos y reducción de costos para el negocio.

Tipos de metodologías ágiles

- XP (Extreme Programming)
- Scrum
- Crystal
- DSDM (Dynamic System Development Method)
- FDD (Feature Driven Development)
- Proceso Unificado De Desarrollo

2.4.3.1 Metodología XP

La Metodología XP Extreme Programming es liviana para el desarrollo de software, es flexible en entornos que son cambiantes, se acerca más al cliente ya que pueden seguir surgiendo nuevos requerimientos durante el desarrollo del proyecto.

Roles XP

Los roles que especifica la metodología XP, que deben cumplir los miembros del equipo

Figura 9: Roles XP
Fuente: (Espinosa, 2015)

Imponiendo los siguientes valores

Valores XP

- Comunicación.
- Simplicidad.
- Coraje.
- Respeto.

Fases de la metodología XP

Fase de Exploración

Los clientes plantean a grandes rasgos las historias de usuario que son de iteres para las entregas del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas de desarrollo que se utilizan en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema, construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.(Morales de la Torre, 2013)

Fase de Planeación

El equipo XP establece la prioridad de cada historia de usuario y dividiendo el sistema en módulos, además se diseña un cronograma de actividades para optimizar los tiempos de

desarrollo del sistema. Se toma acuerdos sobre el contenido de la primera etapa y se asigna tareas por cada historia de usuario. Una entrega debería darse en más o menos tres meses. Esta fase dura pocos días.(Morales de la Torre, 2013)

Fase de iteraciones

Esta fase incluye varias iteraciones sobre el sistema ante de ser entregado. El programador responsable sabe cumplir con las tareas asignadas de acuerdo al cronograma de actividades, y no exceder de tres meses en cada iteración o entrega. En la primera iteración se desarrolla las tareas de cada historia de usuario, después de realizar las pruebas de aceptación al final de cada iteración el sistema estará listo para presentarlo al usuario final.(Morales de la Torre, 2013)

Producción

Requiere de pruebas adicionales y revisiones de rendimientos entes de que el sistema sea trasladado al entorno real. Al mismo tiempo se toma las decisiones sobre las características del servidor de aplicaciones a implementar.(Morales de la Torre, 2013)

2.4.3.2 Metodología Scrum

El objetivo general de la metodología Scrum es la de optimizar al máximo la productividad del equipo de trabajo, las actividades orientadas a producir software funcional y produce resultados en cortos tiempos.(Toapanta, 2012).

La metodología Scrum fue diseñada por Jeff Sutherland y Schwaber, para refinar y extenderla, llegando a ser muy conocida como una herramienta de hyperproductividad, Schwaber se dio cuenta que un proceso necesita aceptar el cambio, en lugar de esperar predictibilidad, está enfocado en el hecho de que procesos definidos y repetibles solo funcionan para atacar problemas, con personas y ambientes definidos y repetibles.(Toapanta, 2012).

Herramienta que ayuda a gestionar proyectos, mantiene un equipo de trabajo muy organizado y dedicado a sacar un producto de calidad con responsabilidad y compromiso, cumpliendo con el tiempo estimado en cada iteración.

Elementos de la metodología Scrum

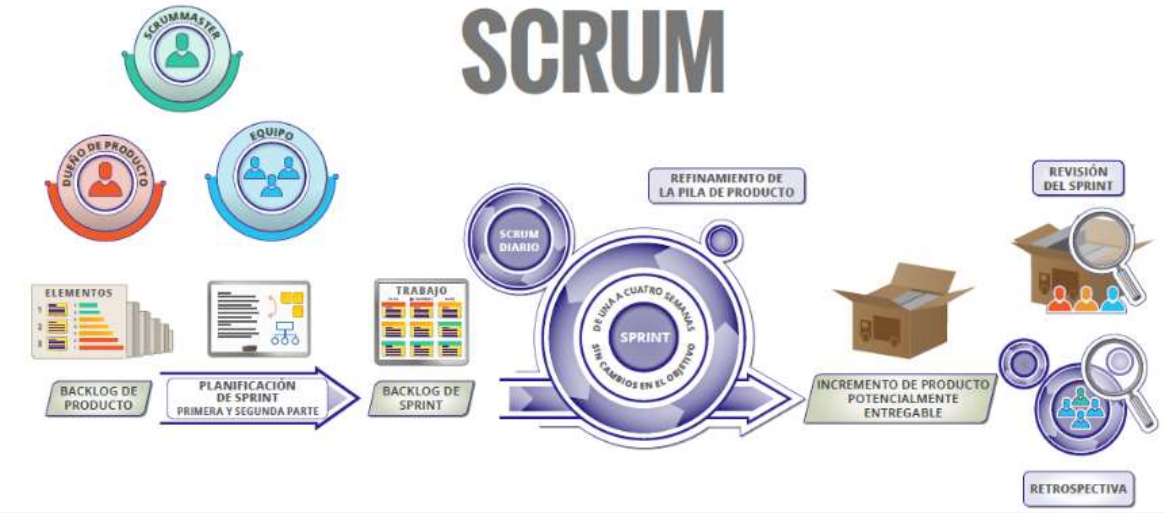


Figura 10: Elementos principales del SCRUM
Fuente:(Párraga, 2014)

Roles

Los roles que define Scrum:

Dueño del producto, equipo, Scrum master, tienen un papel muy importante en la metodología porque tienen que garantizar el correcto avance del proyecto, con la debida responsabilidad para llegar a tener éxito con el producto y satisfacer las necesidades del cliente.

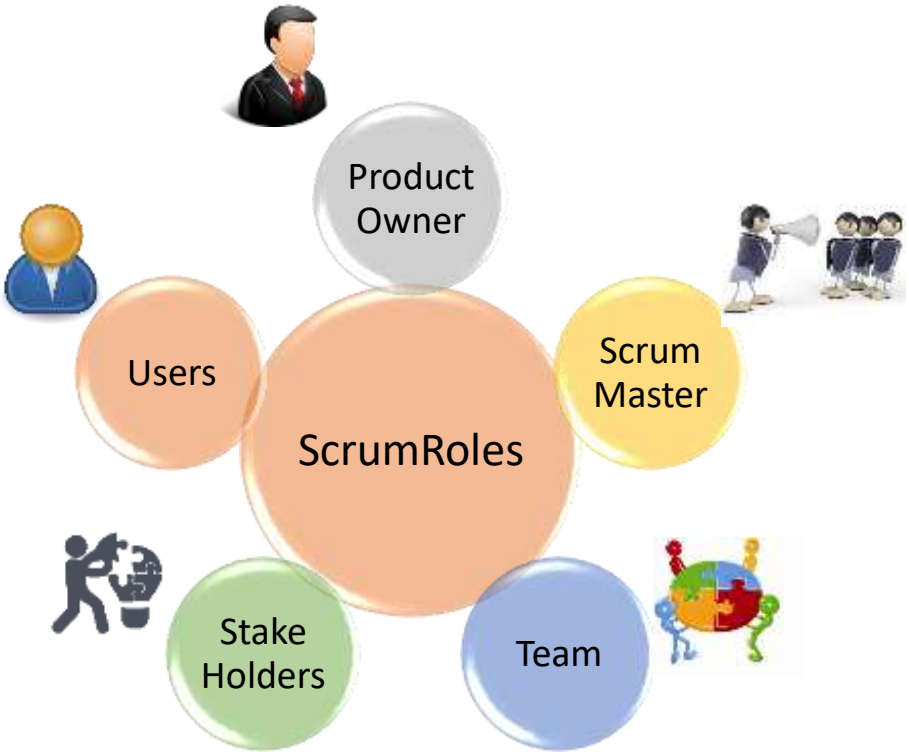


Figura 11: Roles Scrum
Fuente: Propia

Según como específica (Toapanta, 2012) los procesos para el desarrollo son los siguientes:

Poda de requerimientos

Consiste en desarrollar solo lo que el cliente necesita.

Se debe equipar una lista con los requerimientos del sistema, posteriormente irán a evaluación para ver si realmente son necesarios, finalmente sacar los más principales y empezar a trabajar en ellos para que se presente en la fecha de entrega.

Product Backlog

Es un documento dinámico que incorpora constantemente las necesidades del sistema. Por lo tanto, nunca llegará a ser una lista completa y definitiva. Se mantiene durante todo el ciclo de vida y es responsabilidad del Product Owner.

Sprint

Es el núcleo de la metodología, viene a ser un subconjunto de los requerimientos anteriormente obtenidos del sistema, en el Product Backlog, estos deben ejecutarse entre 1 a 4 semanas.

Características del Sprint

No debe superar los 30 días sino en un plazo de 15 días

Cuando se está ejecutando el sprint no debe cambiar el trabajo que se encuentra dentro del Backlog. En el caso de que se necesite cambiar el rumbo, se debe abortar todo el proceso del Sprint, por las siguientes razones:

- Cuando la tecnología escogida no funcione
- Cuando los ambientes del negocio han variado
- Cuando el equipo de trabajo tenga interrupciones para cumplir con sus responsabilidades.

Planificación

Una planificación muy detallada de cada inicio del Sprint, tomando en cuenta que los objetivos no pueden variar durante ese proceso.

Para ello se debe tomar en cuenta los siguientes aspectos:

- Una visión general del alcance
- Tener estimaciones del esfuerzo que demanda tener durante el proceso
- Reuniones diarias para tratar asuntos de planificaciones y revisión.

Roles del Scrum Master durante la planificación

- Dirigir la planificación
- Facilitar acuerdos entre el equipo y el Product Owner del proyecto
- Registro de problemas y riesgos detectados durante la planificación
- Registrar las tareas, asignaciones y estimaciones
- Iniciar el Backlog del Sprint

Sprint Backlog

Tareas asignadas a realizar en un Sprint

- Tener un producto funcional
- Cada tarea se asigna a un número determinado de horas de trabajo que puede oscilar entre 1 y 20
- Cada tarea debe actualizarse a diario

Scrum diario

Es necesario estar inspeccionando frecuentemente, realizando seguimientos de las tareas que fueron asignadas y así el equipo de trabajo no caiga en dilemas y los proyectos se retrasen mucho tiempo.

Cuando se realiza los focos es para determinar los avances, y riesgos que llegaran a existir, que pudieran retrasar los procesos, lo importante es que ningún problema se quede sin resolver o iniciada alguna acción.

Estimaciones

Se lo realiza cuando se va a dar inicio al primer sprint, luego cada tarea se re-estima se registran los cambios en el Backlog del Sprint, esto se lo debe realizar antes o después del Scrum diario.

Builds continuos y pruebas básicas

- Los encargados del desarrollo se basan según el Backlog del Sprint y cada que finaliza se debe notificar al integrador.
- El integrador es la persona que toma el código ya desarrollado para seguir integrándolo al producto.
- Se realiza la compilación y un testing del software para poder verificar si se puede ir avanzando en la versión.
- En el caso de que se encuentren errores en el código se lo devuelve al desarrollador.
- Cuando no existe errores se da la notificación al equipo de trabajo que la nueva versión está estable.

Revisión del Sprint

Consiste en presentar los avances del producto desarrollado por el equipo a los usuarios, estas reuniones sirven para poder detectar anomalías que deban ser resueltos en el siguiente sprint. Se reúnen todos los integrantes del equipo de trabajo esto incluye al Scrum Master, al Product Owner entonces se realiza demos con todos los avances presentados del proyecto.

Reuniones retrospectivas

Se las realizan con el fin de obtener los puntos positivos y negativos para obtener una mejora en los Sprint que vienen.

Valores

Foco

Determinar al equipo de trabajo con el elemento principal para el desarrollo del proyecto.

Comunicación

Tener una comunicación directa con el cliente lo hace que sea un integrante más del equipo de trabajo, esto ayuda aclarar dudas y se va observando el avance del proyecto día a día.

Respeto

Delegar responsabilidades a cada integrante del equipo, y trabajar conjuntamente para cumplir una meta en común.

Coraje

Toma de decisiones importantes, teniendo muy en cuenta los errores y llegar a resolverlos

2.4.3.3 Metodología CRYSTAL

CRYSTAL pertenece a la familia de las metodologías ágiles, su creador fue Alistair Cockburn, marca una diferencia única ya que maneja colores para para determinar la complejidad de la metodología, es decir que se debe utilizar métodos pesados, mientras más oscuro sea el método requiere de más personal, si el estado del sistema es crítico se necesita mucho más rigor. Crystal tiene un cambio continuo por lo cual no tiene establecido en procedimiento concreto, las personas que conforman el grupo de trabajo son netamente responsables del éxito o el fracaso del proyecto.

Cada color representa el nivel de complejidad y el número de personas que integraran los equipos de trabajo para llegar a cumplir la meta.

Clear o Claro: Grupos de 3 y 8 personas o podrían ser menos

Yellow o Amarillo: Grupos de 10 y 20 personas

Orange o Naranja: Grupos de 20 a 50 personas

Red o Rojo: Grupos de 50 a 100 personas

Blue o Azul: Grupos de 100 a 200 personas

Figura 12 : Filosofía de CRYSTAL
Fuente:(Powered, 2016)

Cada carácter tiene su significado según la criticidad:

C: indica pérdida debido a una falla generada en el sistema

D: representa una pérdida económica que no significaría nada en la producción

E: representa una pérdida de dinero esencial que puede representar un riesgo para la organización.

L: representa una pérdida de vida

El número que va junto al carácter quiere decir la persona que va hacer afectada en el proyecto

Todo esto está centrado en las personas y la comunicación que debe fluir, el equipo puede reducir el trabajo a medida que vaya avanzando la generación del código

Se debe respetar las reglas que tiene la familia Crystal:

- La primera indica que los ciclos donde se crean los incrementos no deben exceder cuatro meses.(Powered, 2016)
- La segunda es necesario realizar un taller de reflexión después de cada entrega para afinar la metodología.(Powered, 2016)

Principios de CRYSTAL

- Se hace una entrega incremental que se las debe realizar en dos meses y como máximo 4 meses
- Debe haber un seguimiento de hitos
- El usuario juega un papel muy importante en el desarrollo del sistema
- Se realiza pruebas de funcionalidad
- Por cada incremento se realiza talleres de producto para ajustar a los principios.

Una de sus características es mantener a un usuario activo cuando se vaya a realizar los test de validación de las interfaces, dicho usuario debe ser un integrante más para establecer los requerimientos funcionales y no funcionales del sistema.

Las siete propiedades de Crystal

Según como explica (Chicaiza, 2007)

Entregas frecuentes. Se realiza en pocas semanas después de haber iniciado el proceso, liberando código que ya está en modo ejecutable y posteriormente será testeable con usuarios reales, es así como el cliente va observando los avances del producto, que va a satisfacer las necesidades a medida que vaya avanzando surgirán errores que deben ser corregidos en su debido momento.

Mejora reflexiva. Se lo realiza una vez cada tres meses con el fin de comparar muchos aspectos, hábitos en los cuales haya que mejorar la reunión tendrá una duración de media hora, hora o un medio día.

Comunicación Osmótica. Es una comunicación de alto nivel que no debe durar más de 30 segundos para hacer escuchar o ver los problemas que se tenga a la persona que está encargada de ayudar a resolverlo, es una manera de contactarse de una forma más personal y ágil.

Seguridad personal. Ayuda a fomentar confianza y respeto entre los integrantes de equipo, para cumplir con las responsabilidades que le competen a cada uno cumpliendo con los objetivos anteriormente propuestos.

Enfoque. Identificar los elementos que tengan más prioridad y se debe fundamentar en lo siguiente:

El trabajo exige alta concentración que evitar cualquier tipo de errores en el desarrollo.

Cuando no encuentra la concentración necesaria puede perder minutos muy valiosos que pueden afectar al trabajo de los demás integrantes del equipo.

Cuando la concentración es al 100 por ciento se puede llegar a resolver mucho más rápido los errores que se generen y si no es el caso puede llegar a demorar una hora o más.

Acceso fácil a usuarios expertos. No debe superar un máximo de tres días alguna duda que surja del sistema para que el experto pueda aclarar o resolverla.

Ambiente de desarrollo con pruebas automáticas, administración de la configuración e integración frecuente. El sistema debe estar funcionando con las pruebas necesarias, así no sería necesario que el experto tenga que estar físicamente presente porque es una pérdida de tiempo. Crystal no necesita llevar una técnica exacta para el desarrollo de proyecto, sino que se debe seguir las pautas que plantea:

Exploración 360. Establecer los requerimientos necesarios, y obtener la información necesaria del campo en cual se va aplicar el sistema.

Victoria temprana. Mostrar un código funcional que vaya satisfaciendo poco a poco las necesidades por las cuales se lo esté desarrollando.

Esqueleto caminante. Podría ser un pequeño prototipo que esté funcionando, que utilice los componentes más importantes de la arquitectura principal.

Arquitectura incremental. Mientras el sistema esté en ejecución se puede ir modificando la arquitectura sin afectar el objetivo principal.

Radiadores de información. Publicaciones que deben ser visualizados y muy entendibles por todos los integrantes del equipo, esta debe ser modificada periódicamente.

Roles

Los roles que propone la metodología Crystal a continuación se muestra en la **Figura. 13**

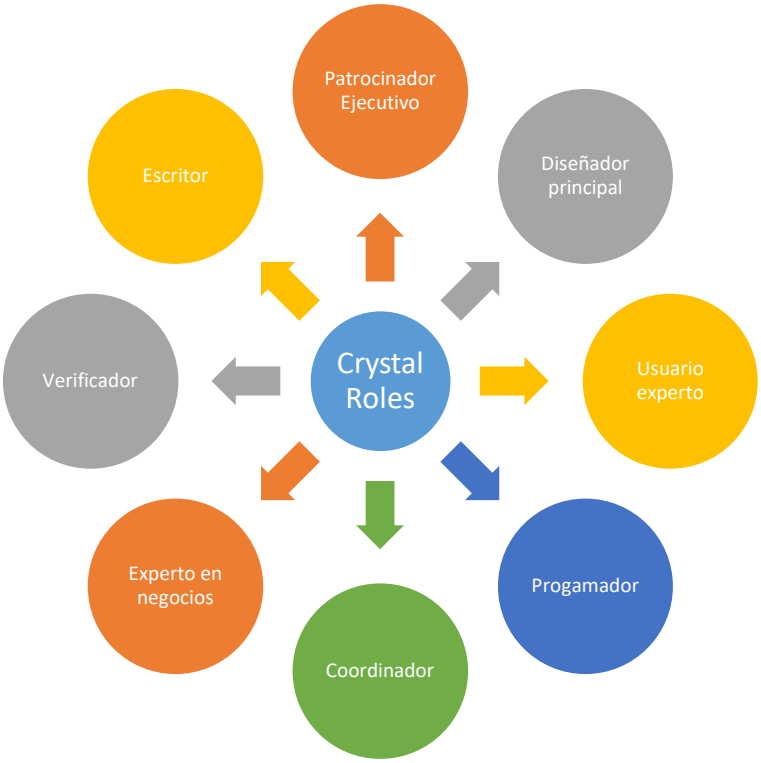


Figura 13: Roles Metodología Crystal
Fuente: Propia

2.4.3.4 Metodología DSDM (Dynamic Systems Development Method)

Determina un marco de trabajo, se realiza controles para el desarrollo de aplicaciones que se determina como RAD (Rapid Application Development).

Su objetivo principal es determinar el tiempo y costo, definición de requerimientos.

MoSCoW son las reglas con las que trabaja DSDM:

Musthave (debe). Son los requerimientos más importantes del proyecto.

Shouldhave (debería). Requerimientos esenciales para los cuales deberán ser resueltos en el menor tiempo.

Couldhave (podría). Requerimientos que no tengan solución podrían salir fuera del proyecto.

Want to have (desear). Requerimientos que son valorados pero que por momento no son tomados en cuenta.

Fase de la metodología DSDM

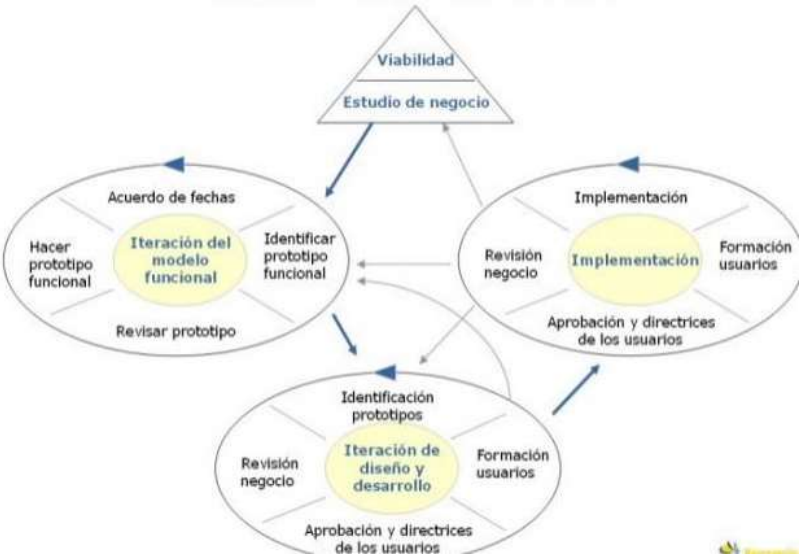


Figura 14: Diagrama de Proceso DSDM
Fuente: (IDS, 2015)

Estudio de Viabilidad. Se identifican las técnicas y riesgos, se definen los artefactos como los reportes de factibilidad y un plan llamado sumario para el desarrollo, cuando no se define muy bien la tecnología se realiza un pequeño prototipo en el que se realizara un test.

Estudio del negocio. Se determina las características y la tecnología a utilizarse, se tiene reuniones con el cliente y personal experto en el negocio, para definir facetas, analizar prioridades, se definen también los artefactos que vienen a ser la arquitectura del sistema, un bosquejo es decir que se debe realizar un prototipo.

Iteración de modelo funcional. Cada que se realiza una iteración se planifica contenidos y estrategias, luego se analizan los resultados. Los test se realizan las veces que sean necesarias, los artefactos que sobresalen a partir de esta fase son: lista de las funciones más importantes del sistema, documentos que consten la revisión del bosquejo, lista de requerimientos y análisis de riesgos.

Iteraciones de diseño y construcción. Debe complementarse con las reglas MoSCoW ya que trabaja con iteraciones y deben ser revisados por los usuarios

Implementación. Cuando el sistema ya está en producción, luego viene la capacitación, de aquí sales los siguientes artefactos: manuales de usuario y reportes.

Roles en la metodología DSDM

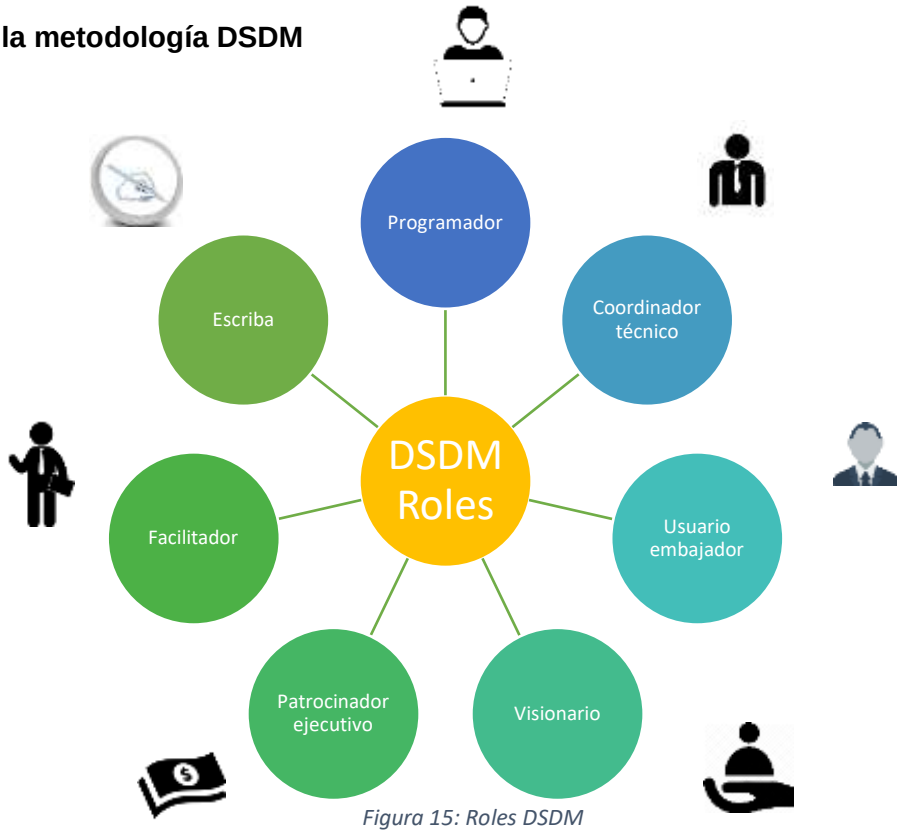


Figura 15: Roles DSDM
Fuente: Propia

2.4.3.5 Metodología FDD (Feature Driven Development)

Es un proceso que maneja iteraciones cortas, que permiten obtener incrementos funcionales en el sistema, para que los clientes tengan la posibilidad de ver, analizar y evaluar.

La metodología está definida en cinco fases que se muestra en la **Figura. 16**:

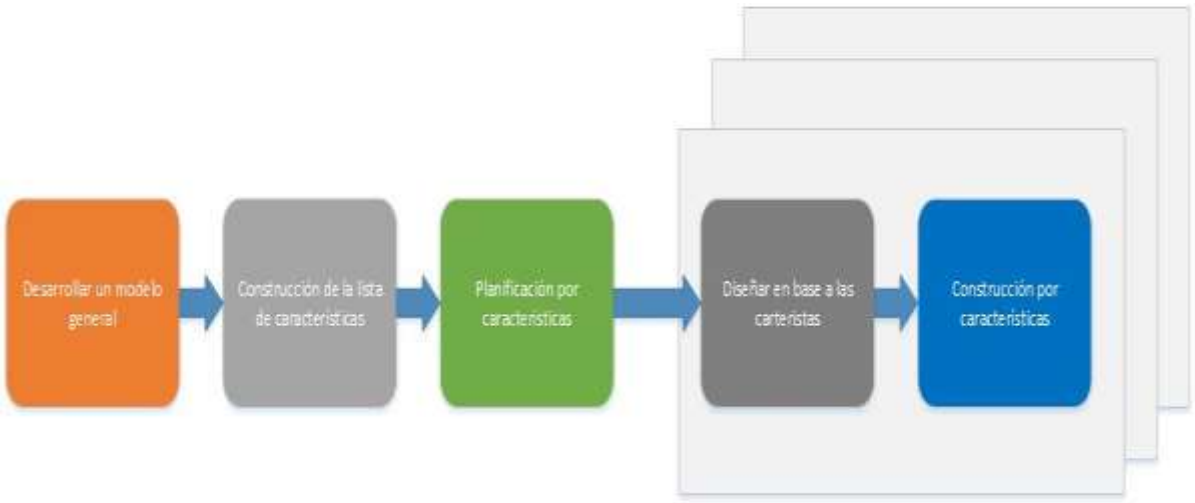


Figura 16: Diagrama Fases FDD
Fuente: (Humbert, 2014)

Descripción de cada fase:

Desarrollo de modelo general. En el momento que se da inicio a esta etapa los expertos del dominio deben ya tener una idea de los requerimientos que debe tener el sistema, probablemente este documento donde se especifican cada uno de los requisitos ya esté listo, pero la metodología FDD no se basa exactamente en la recolección y administración requerimientos.

Los encargados presentan un informe llamado Walkthrough para ser entregado a cada área, luego que finaliza cada Walkthrough, se reúnen el equipo desarrollador para decidir el modelo de objetos que se va a utilizar en cada área de dominio.

Construcción de la lista de características. Los Walkthrough, modelos de objetos y la lista de requerimientos son los elementos fundamentales para dar paso al feature list es decir la lista de características que deben explicar en forma resumida el funcionamiento del sistema, para que luego el equipo desarrollador presente al cliente cada una de las funcionalidades ya evaluadas y aprobadas por el mismo.

Planificación por características. Aquí se debe incluir un plan con el feature list donde deben ser ordenados según la prioridad entre cada una de ellas, en el momento que se ya se identifican a las clases se las asignan a cada programador.

Diseñar en base a las características. El jefe de programación analiza la siguiente lista de características a ser diseñadas, en la cual debe identificar las clases que están inmersas aquí para luego contactar al class Owner, luego el equipo encargado del desarrollo empieza a trabajar en el diagrama de secuencia haciendo una descripción de clases y métodos.

Construcción por características. Los class owner realizan el desarrollo de las clases que conforman cada feature o característica propuesta, luego pasan a ser evaluadas cada una de ellas, haciendo una revisión del código fuente y finalmente se realiza el testing de integración.

Las tres primeras fases son las que lleva tiempo cuando se realizan las primeras iteraciones, pero cuando se pasa a la dos últimas, toma mucho más tiempo mientras siga avanzando el proyecto, pero sin dejar atrás las primeras, ya que se necesita hacer un refinamiento.

Roles de la metodología FDD

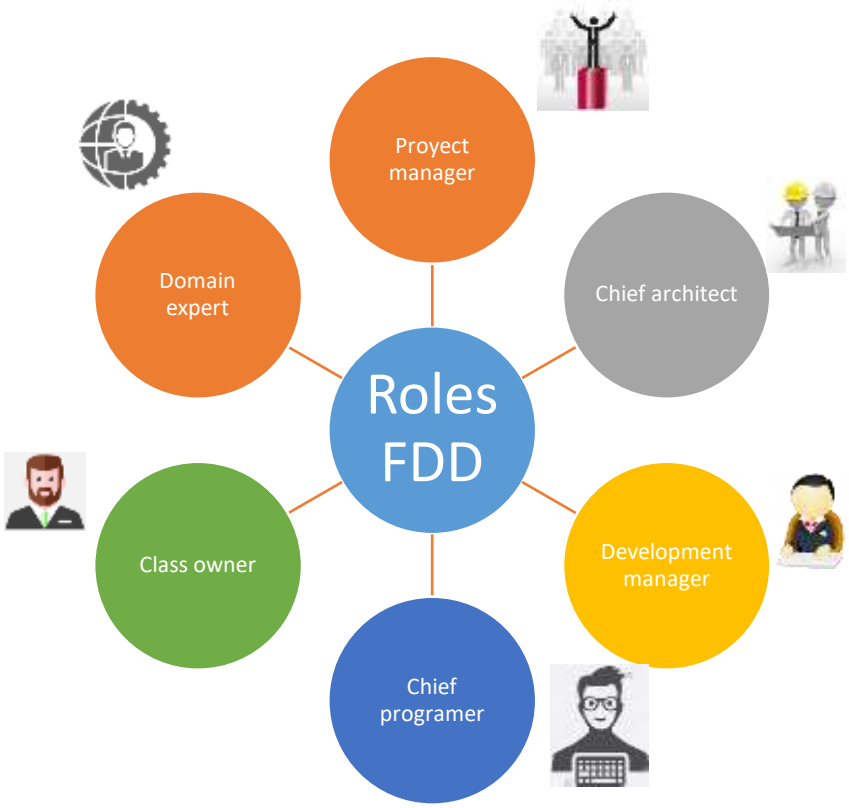


Figura 17: Roles FDD
Fuente: Propia

2.4.3.6 Metodología ASD (Adaptive Software Development)

Su significado es Desarrollo Adaptable de software, este modelo no trabaja con muchas reglas a seguir es una metodología abierta, se adapta a cambios que se pueden producir en las iteraciones a medida que se vayan realizando, igual que las otras esta propensa a errores, pero se debe trabajar conjuntamente para dar soluciones a los problemas que se presenten.

Características de la metodología ASD

Las características que identifican a este tipo de metodología son:

- Iterativo
- Orientado a componentes de software
- Tolerante a los cambios
- Guiado por los riesgos

La revisión de cada componente es de mucha ayuda para de esa manera aprender de los errores y si es necesario volver a iniciar el ciclo de desarrollo.

La metodología ASD antepone que las necesidades del cliente pueden ser cambiantes, en el momento que se dé inicio proyecto se identifican la misión, las características y por último fechas para entregas de avances, cada paso a desarrollarse dentro de la metodología puede comprenderse en un tiempo determinado entre cuatro y ocho semanas.

Ciclos de Vida ASD

Figura 18: Ciclos de la metodología ASD
Fuente: (Alex, 2013)

Consta de lo siguiente:

Especulación

Es una fase donde se planifican los objetivos, el tipo de características del software, el tiempo, el número de iteraciones. Es necesario volver a repetir los pasos hasta que el cliente se encuentre satisfecho con los resultados.

Colaboración

Empiezan a desarrollarse las características planteadas en la anterior etapa, manteniendo una excelente comunicación entre el equipo encargado del proyecto, para transmitir lo aprendido a los demás integrantes.

Aprendizaje

Es una etapa donde se debe analizar lo bueno y lo malo que salió de todo el trabajo realizado. Es la parte final de la metodología que pone énfasis en la revisión de la calidad y en el caso de que ya no existan errores se hace la entrega al cliente.

El creador de la metodología identifica tres tipos de aprendizaje:

- La calidad del producto desde el punto de vista del cliente
- La calidad del producto desde la perspectiva del equipo desarrollador
- La gestión de rendimiento
- Situación del proyecto

2.4.3.7 Metodología AUP (Proceso Unificado Ágil)

AUP todavía maneja conceptos validos de la metodología RUP en el desarrollo de aplicaciones de negocio, por otro lado, también maneja técnicas agiles que incluyen al desarrollo dirigido por test, modelados, gestión y refactorización pero en las bases de datos.

Fases de AUP

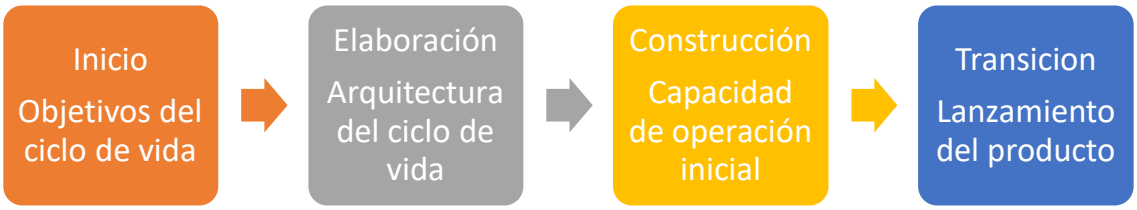


Figura 19: Ciclo de vida de AUP
Fuente Propia

La metodología consta de 4 fases:

Inicio

En la etapa de inicialización se comienza identificando el alcance, la viabilidad, los riesgos del proyecto, se realiza el levantamiento de los requerimientos funcionales y no funcionales y se obtiene un financiamiento luego de haber realizado los costes de programación.

Elaboración

Es en esta fase se define y valida la arquitectura para el proyecto, de igual manera se establece el equipo de trabajo que se encargara del desarrollo del sistema.

Esta fase llega a su finalización cuando llega a un hito llamado Life Cycle Architecture, el equipo de desarrollo debe mostrar una estrategia para dar inicio a la siguiente fase de construcción.

Construcción

Se empieza a trabajar en el modelado, diseño, construcción y test del sistema de igual manera se irá desarrollando la documentación necesaria del proyecto.

Transición

En esta etapa se realizan las pruebas necesarias, validando cada una de sus funciones principales, y finalmente se realiza las pruebas de integración ya en un entorno de producción.

A medida que se vaya avanzando en cada fase se va desarrollando las siguientes disciplinas que a diferencia de RUP aquí solo se toma en cuenta 7 de una manera iterativa.

1. **Modelado.** Se identifican los problemas que hayan surgido en el proyecto, seguidamente se deben analizar las posibles soluciones para su corrección.

2. **Puesta en práctica.** Los modelos que ya estén en funcionamiento es decir con código fuente ya ejecutable se deben realizar las pruebas para la detección de errores.

3. **Prueba.** Se asegura la calidad del producto, encontrando fallas existentes, realizando validaciones y hacer control de que se cumpla con los requisitos planteados.

4. **Despliegue.** Se realiza todo lo necesario para hacer la entrega del producto, luego ponerlo en ejecución en un ambiente de producción para los usuarios finales.

5. **Gerencia de la configuración.** Se realiza la gestión de los artefactos o herramientas que están disponibles para el desarrollo del proyecto y se debe hacer un control de seguimientos de los avances y los cambios que se llegue a producir.

6. **Gestión de proyectos.** Control de actividades durante el ciclo de vida del sistema, con asignación de tareas, gestionando los posibles riesgos que hayan, haciendo un control del progreso y verificando que se esté cumpliendo con el alcance antes planificado y cumpliendo con la entrega a tiempo del producto.

7. **Ambiente.** Entorno de trabajo amigable para los miembros del equipo, teniendo una comunicación para ayudar a solucionar cualquier duda existente, y así cumplir satisfactoriamente con el desarrollo del proyecto.

Roles de la metodología AUP

Figura 20: Roles AUP
Fuente: Roles AUP

2.4.4 Metodologías Híbridas

Las metodologías hibridas en el mercado actual están tomando una nueva perspectiva para el desarrollo de software, son la combinación de metodologías tradicionales con las ágiles o también una mezcla entre lo tradicional con lo tradicional y lo ágil con lo ágil haciendo una forma más eficaz en el desarrollo, que se adapten a cambios durante el proceso de desarrollo, haciendo que las iteraciones se manejen de una manera no tan compleja, ya que se toma las ventajas de las metodologías antes mencionadas para obtener un producto de calidad.

Las metodologías de estudio que se va tomar para la comparativa en este trabajo de grado son las siguientes:

- Metodología EssUP(Essential Unified Process)
- Metodología Híbrida SCRUM/XP

2.4.4.1 Metodología EssUP (Essential Unified Process)

Es un proceso unificado esencial hace la mezcla de la metodología tradicional RUP (Rational Unified Process), con la metodología ágil Scrum, haciendo así una integración de buenas prácticas para mejorar el proceso de desarrollo de software. “Incursiona en tres campos de procesos principales: el campo de proceso unificado, el campo de métodos ágiles y el campo de madurez del proceso. Cada uno de ellos aporta diferentes capacidades: estructura, agilidad y mejora de procesos”.(Jacobson, 2016c)

El proceso de la metodología tiene 8 prácticas esenciales que se las divide en:

Que consiste en el desarrollo y Prácticas Técnicas e ingeniería social, procesos y otras prácticas de Apoyo como se puede observar en la **Figura. 21**

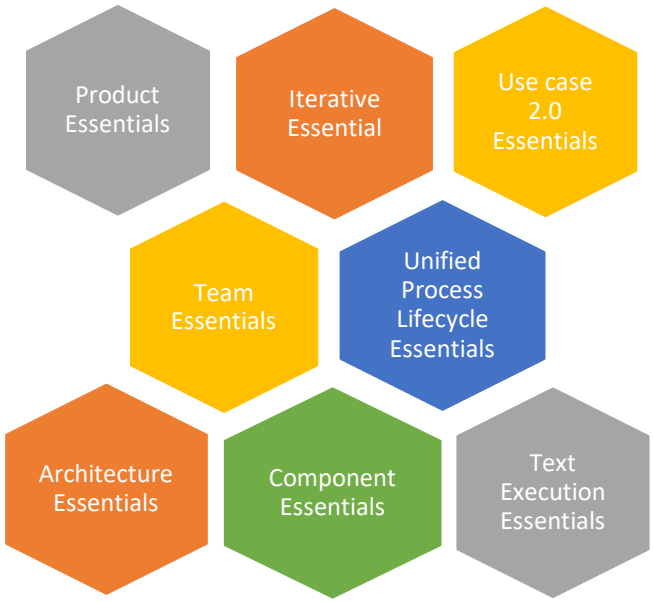


Figura 21: Prácticas EssUP
Fuente: Propia

Seguidamente se comenzará explicando cada una de las prácticas que propone EssUP.

Arquitectura Esencial

Ayuda a definir una buena arquitectura que vaya acorde al proyecto en ejecución.

Obligaciones del equipo de trabajo

- Se debe constituir una base consistente para el desarrollo incremental de una solución efectiva.
- Enfoque práctico de cada uno riesgos técnicos que se contraponen en el proyecto
- Reunirse los miembros del equipo para compartir las decisiones más importantes sobre la estructura y organización de sistema.
- Evaluar y verificar que el producto cumpla con las características claves esperadas por el cliente.
- Realizar pruebas objetivamente que el enfoque seleccionado sea el correcto para aplicarlo en el propósito.

Cosas con que trabajar

- La práctica de la arquitectura se la utilizar para entrar a producción con la lista de elementos a incluirse en la implementación y las pruebas.
- La arquitectura debe ser documentada por medio de la descripción detallada de la arquitectura, siendo un conjunto de pruebas, casos de usos y vistas arquitectónicas del modelo de diseño es lo que propone Ivar Jacobson.
- La arquitectura debe ser implementada en el inicio de la ejecución del proyecto cuando ya está en construcción realizando una prueba de la primera versión del proyecto. La arquitectura del sistema se desarrolla constantemente como parte esencial del proyecto.

Competencias Claves

Se necesitan habilidades, creatividad para darle un formato a los sistemas, para el desarrollo de prototipos arquitectónicos, pruebas de la arquitectura y por ende la participación de los clientes. Se necesita estar en constante entrenamiento del equipo como se explica en la **Figura. 22**

Figura 22: Competencia claves de la Arquitectura
Fuente: (Jacobson, 2016a)

- **Customer o cliente.** Es el representante de las partes interesadas
- **Solution o solución.** Representa el análisis, desarrollo y test del proyecto
- **Endeavor o esfuerzo.** Representa el liderazgo y entrenamiento que se necesita dentro del equipo de trabajo

Cosas que se deben hacer

- Inicia identificando y aclarando los requisitos que son de importancia arquitectónica, luego se establecen los objetivos sobre la arquitectura.
- Determina la arquitectura a ser implementada y posterior desarrolla una arquitectura para producir un sistema base que cumpla con los requisitos planteados.
- El equipo está debidamente entrenado en el uso de la arquitectura para asegurar que el producto cumpla con satisfacer las necesidades del cliente o la empresa que lo esté promoviendo.
- La arquitectura se desarrolla continuamente mediante la refactorización y la evolución de la implementación arquitectónica en respuesta a los resultados de validación sobre la arquitectura y requisitos emergentes. Cada una de las versiones arquitectónicas son evaluadas continuadamente para luego iniciar con la implementación.
- En definitiva, la práctica de la arquitectura esencial se fundamenta en actividades de test de sistema que validará la arquitectura mediante pruebas. Como se muestra en la siguiente **Figura. 23**

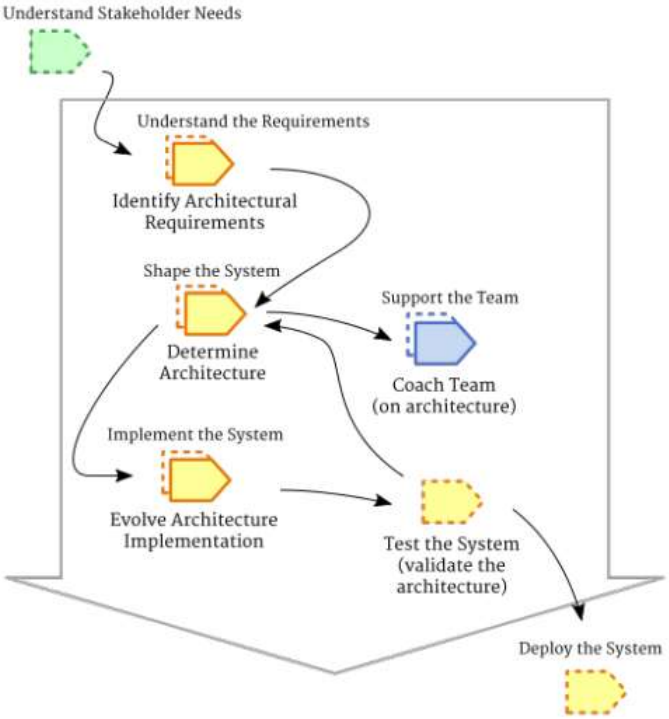


Figura 23: Arquitectura Esencial
Fuente: (Jacobson, 2016a)

Componentes Esenciales

- Desarrollo simple, escalable y basado en componentes.
- La utilización de esta práctica en el desarrollo de sistemas complejos como conjuntos de componentes más pequeños y simples.

Permite a los equipos:

- Gestionar la complejidad asociada con el desarrollo de software
- Desarrollar sistemas complejos de una manera extensible y mantenible
- Desarrollar y verificar las partes separadas de un sistema de forma independiente y en paralelo.
- Identificar oportunidades para la reutilización y explotación de los componentes reutilizables.
- Utilizar Frameworks y bibliotecas de componentes.

Cosas con las que trabajar

Implica en una serie de elementos de implementación y pruebas:

- Un modelo de diseño de sistema, identificando el conjunto de componentes requeridos
- Una descripción de cada componente, incluyendo su comportamiento requerido e interfaces
- Código fuente y pruebas unitarias para cada componente
- Construcción integrada del sistema de componentes

Competencias Claves

Se necesita que el equipo de trabajo sea experto en diseño e implementación del software, pruebas unitarias y pruebas de integración como se muestra en la siguiente **Figura. 24**

Figura 24: Principales Competencia de los Componentes Esenciales
Fuente: (Jacobson, 2016b)

- **Solution o solución.** Encargado del desarrollo y pruebas.
- **Endeavor o esfuerzo.** Encargado de tener un buen liderazgo.

Cosas por hacer

La práctica inicia identificando el conjunto de componentes necesarios para satisfacer los requerimientos del sistema, entender los requisitos.

Constantemente se encuentran definiendo los componentes incluyendo las interfaces y pruebas unitarias, dando inicio al desarrollo de los componentes para la implementación de las interfaces y pasar los test. Concluye integrando el sistema de software y soporta el despliegue cuando se liberan o se ponen en funcionamiento los componentes, como se observa en **Figura. 25**.

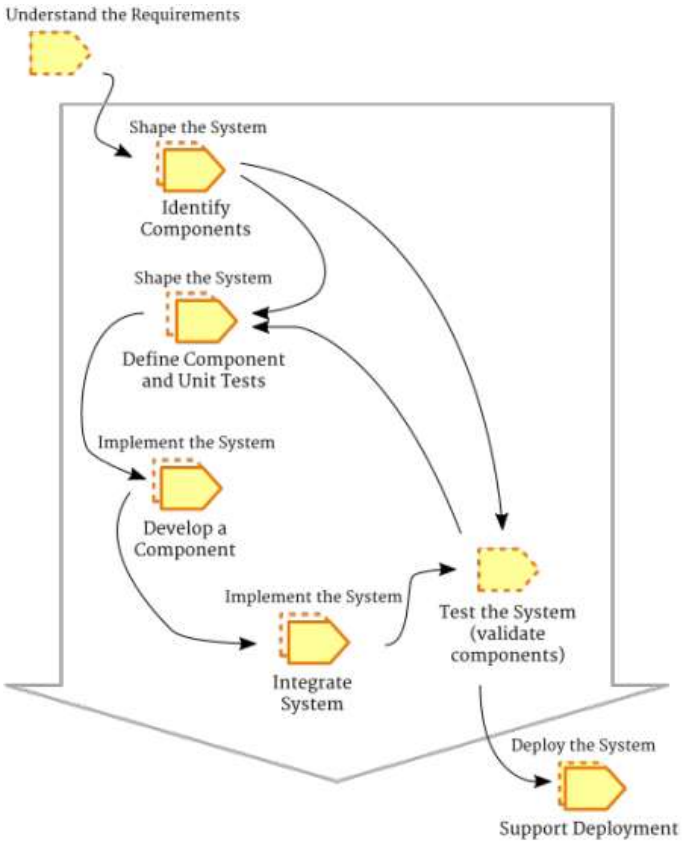


Figura 25: Componentes Esenciales
Fuente: (Jacobson, 2016b)

Esencial Iterativo

Esta es una forma de ayudar al equipo de trabajo a reducir riesgos y costos, administrando el cambio, mejorando la productividad y ofreciendo soluciones más eficaces y oportunas.

Permite al equipo:

- Colaborar objetivamente, planificar, ejecutar y rastrear el esfuerzo.
- Gestionar adecuadamente el tiempo, la calidad, las expectativas de costes y deben presentar respuestas ágiles al cambio.
- Realizar una exposición temprana del proyecto con los clientes y usuarios, para seguidamente hacer una retroalimentación con nuevas ideas.
- Proporcionar soluciones de mayor calidad, más apropiadas, y con más frecuencia.
- Montar un sistema operacional al inicio del proyecto que avance gradualmente hasta obtener un sistema completo.

Cosas con las que se debe realizar

- Implica la producción de una serie de elementos relacionados:

- El Backlog del producto se rellena con cambios, bugs y otras tareas que representan el trabajo a realizarse.
- El Plan Proyecto identifica el número de iteraciones a realizarse con un amplio esquema de lo que cada integrante deberá hacerlo.
- Los planes e iteraciones están impulsados por los riesgos del proyecto.
- Se elaboran planes y evaluaciones de la iteración se producen para atrapar la intención y el resultado de cada iteración.

Competencias Claves

Es necesario que el equipo sea experto en liderazgo, planificación e iteración, incluso si estas prácticas tienen como objetivo liderar el esfuerzo de desarrollo, todavía requiere que le equipo tengan las habilidades necesarias, para crear software de alta calidad dentro de los plazos establecidos por la iteraciones como se muestra a continuación en la **Figura. 26**.

Figura 26: Competencias Claves Iterativo Esencial
Fuente: (Jacobson, 2016d)

- **Customer o cliente.** Representante de las partes interesadas
- **Endeavor o esfuerzo**. Representa el liderazgo y la gestión

Cosas por hacer

Inicia acondicionado los planes de proyectos existentes para formar iteraciones en el enfoque de desarrollo.

Se plantean los objetivos, criterios de evaluación y trabajo para dar inicio a la primera iteración. La práctica sirve de guía al equipo a través de la iteración donde aplican otras prácticas para lograr cumplir con los objetivos para la cual fue creada la iteración. Al final de cada iteración, los resultados son evaluados y utilizados para adaptar los planes a la realidad del desarrollo y acordar la siguiente iteración. Esta secuencia se debe continuar para cada iteración adicional hasta que después de evaluar los resultados de la iteración final y se dé por concluida en el desarrollo a continuación se muestra el proceso en la siguiente **Figura. 27**.

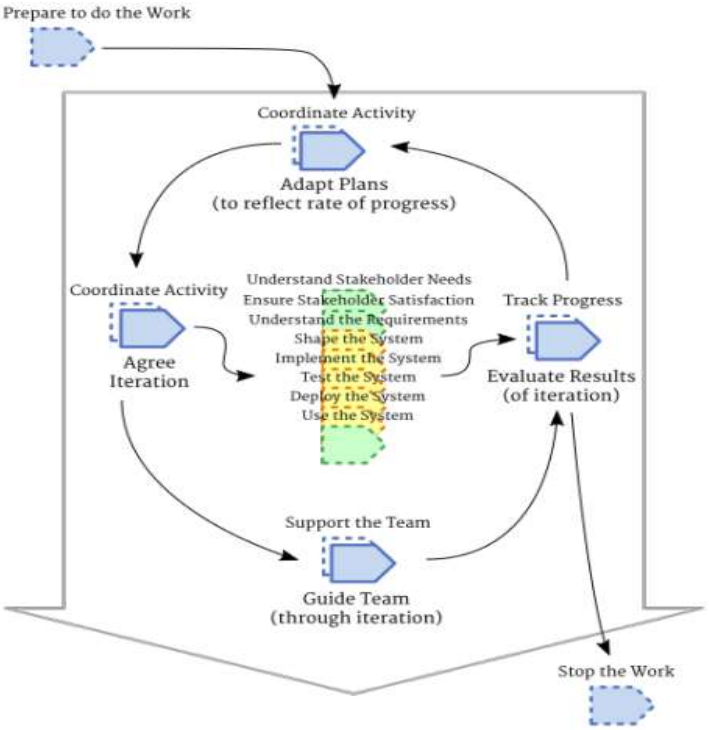


Figura 27: Iterativo Esencial
Fuente:(Jacobson, 2016d)

Patrones

Proporciona una lista de patrones que servirán de ayuda al equipo para la medición efectiva del progreso del proyecto.

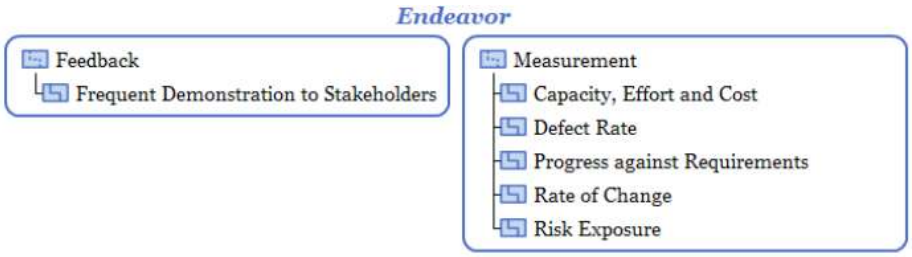


Figura 28: Patrones
Fuente: Jacobson, 2016c)

Productos Esenciales

- Enfoque que ayuda alcanzar y ofrecer evoluciones de productos de software fundamentadas en el valor del negocio.
- La utilización de esta práctica se basa en la gestión del desarrollo de evoluciones sucesivas de un sistema de software como una serie de lanzamientos de productos.

Permite a los equipos

- Desarrollar un sólido caso de negocios para el producto.

- Planificar el proyecto como una serie de lanzamientos de productos principales, cada uno de los cuales debe ofrecer beneficios reales para el negocio.
- Involucrar a las partes interesadas en el proceso de toma de decisiones.
- Asegurarse de que le producto desarrollado satisface las necesidades reales de las partes interesadas.
- Administración de la evolución del software de una manera controlada y centrada en el negocio.

Cosas con las que trabajar

- Implica la producción de una serie completa de productos de trabajo, negocio, planificación y requisitos:
- En el caso de los negocios se establece el valor del producto
- Análisis de las partes interesadas en este caso hace la referencia a los clientes y usuarios para que entiendan y que participación es muy importante en el mismo.
- Lista de características, un glosario y la descripción de la liberación, posterior definir el producto y su lanzamiento en el tiempo establecido.
- Un plan de proyecto con la descripción de cómo se trabajará en la serie de lanzamientos.
- Como se puede observar en la **Figura. 29**

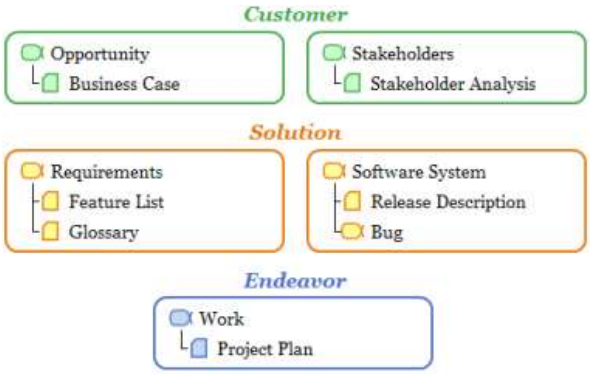


Figura 29: Cosas con las que trabajar
Fuente: (Jacobson, 2016e)

Competencias Claves

Requiere de un equipo altamente capacitado y calificado en la obtención de requerimientos, con la gestión de las partes interesadas, definición de los productos y planificación de emisiones.

De esto requiere las siguientes competencias que se indican en la **Figura. 30** las más importantes son las habilidades del representante del cliente y del analista.

Figura 30: Competencias Claves de Productos Esenciales
Fuente: (Jacobson, 2016e)

Cosas por hacer

La práctica inicia lanzando el proyecto y estableciendo el caso de negocio del producto.

Continúa con la especificación, planificación de los lanzamientos del producto y concluye con el lanzamiento del proyecto y su aceptación por parte del cliente. A continuación, se muestra el proceso en la **Figura. 31**

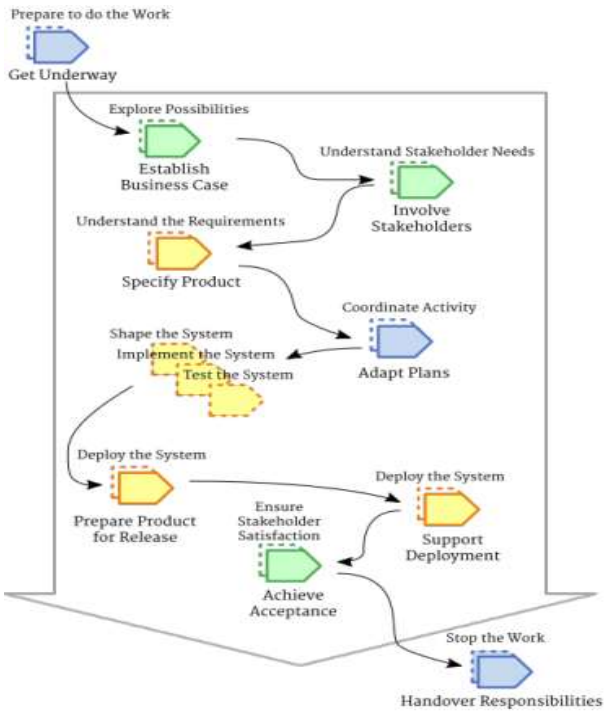


Figura 31: Productos Esenciales
Fuente: (Jacobson, 2016e)

Equipo Esencial

- Crea un ambiente de trabajo adecuado para que el equipo pueda sobresalir en cualquier circunstancia.
- La práctica se la utiliza para reunir a un equipo de desarrollo del proyecto y establecer un ambiente de trabajo efectivo.

Permite a los equipos

- Adoptar el liderazgo apropiado y los patrones organizacionales
- Establecer y obtener las competencias necesarias para tener éxito
- Desarrollar maneras efectivas de colaborar y organizar su trabajo.
- Permitir que todos los miembros del equipo contribuyan al máximo de su capacidad

Cosas con las que trabajar

- Implica el establecimiento de un equipo eficaz y adecuado para la producción de una serie de productos:
- En el momento que el equipo se encuentra establecido debe conocer la estructura y responsabilidades que deben ser capturadas por cada uno de los miembros.
- Se desarrollan formas efectivas de colaborar.

Competencias Claves

Requiere las siguientes competencias que se muestran en la **Figura. 32**

Figura 32: Competencias Claves del Equipo Esencial
Fuente: (Jacobson, 2016f)

El liderazgo y habilidades de gestión son necesarias para reunir al equipo para trabajar de una manera eficaz. Las habilidades de representación de las partes interesadas son necesarias para asegurar que el equipo permanezca vinculado con las necesidades y expectativas del cliente.

Se requiere la colaboración de todos los miembros del equipo en diferentes competencias relevantes tales como análisis, desarrollo y prueba, cada miembro debe unirse dependiendo el nivel específico en el que se desenvuelven.

Cosas para hacer

Se inicia definiendo al equipo de trabajo, planteándose una misión y un establecimiento de fundamentos en el entorno de trabajo.

El equipo es guiado mientras hacen su trabajo para mejorar la colaboración y eliminar cualquier obstáculo que impida el trabajo eficaz en equipo. Y los resultados se evalúen regularmente y exista una retroalimentación generada para que en el siguiente ciclo se tenga una base sólida

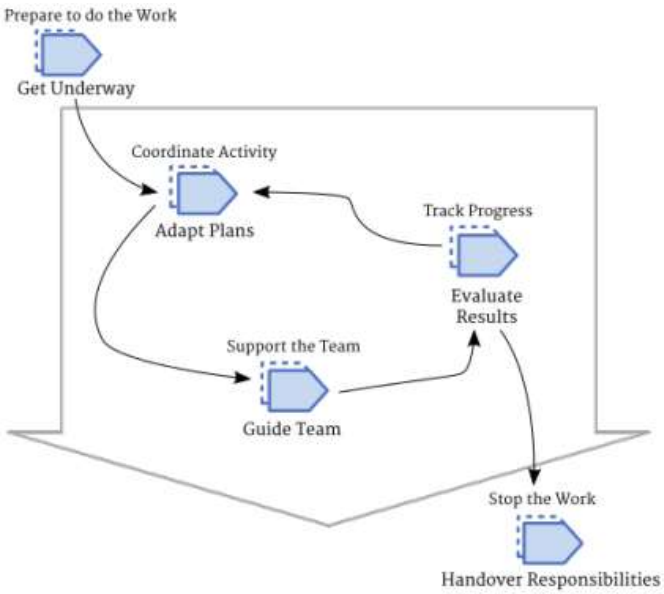


Figura 33: Equipo Esencial
Fuente: (Jacobson, 2016f)

Patrones

La práctica incluye una selección de patrones de trabajo ya demostrados para que el equipo de trabajo pueda apoyarse mutuamente, estableciendo relaciones laborales y ambientes correctos. Como se muestra en la siguiente **Figura. 34**

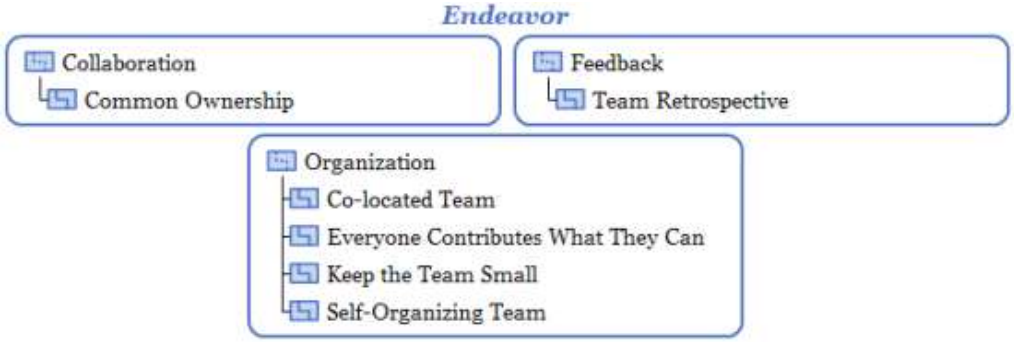


Figura 34: Patrones
Fuente: (Jacobson, 2016f)

Ejecución de pruebas

Utilización en un enfoque estructurado de las iniciativas de prueba y garantía de calidad de un proyecto para mejorar la estimación con el monitoreo y control de actividades en un ambiente de pruebas.

Permite a los equipos

- Realizar una identificación de pruebas, de acuerdo con los objetivos, que guiaran el desarrollo del software y verificaran que el software funcione como se debe tener especificado.
- Elaboración de pruebas que proporcionen los procedimientos e instrucciones de prueba necesarios.
- Ejecución de pruebas que verificaran que el software cumpla con los criterios de evaluación.

Cosas con las que trabajar

- Consiste en la elaboración de una serie de productos de prueba
- Especificadores de prueba que deban referirse a casos de pruebas.
- Resultados de las pruebas

Competencia Claves

Requiere un equipo de trabajo experto en pruebas de desarrollo, además de algunas habilidades de prueba y desarrollo, elementos importantes que son necesarios en el alcance y la cobertura del test. Como se muestra en la siguiente **Figura. 35**

Figura 35: Competencias claves de la ejecución de pruebas esenciales
Fuente: (Jacobson, 2016g)

Cosas para hacer

La práctica inicia identificando pruebas para llegar a uno o más objetivos, continuadamente se construye y ejecuta las pruebas, entregando resultados para el análisis. Los resultados contribuyen a la medida de calidad de software.

Un resultado de análisis adicional es la necesidad potencial de refinar las pruebas existentes o identificar nuevas pruebas. A continuación, se muestra el proceso de la pruebas de ejecución en la **Figura. 36**.

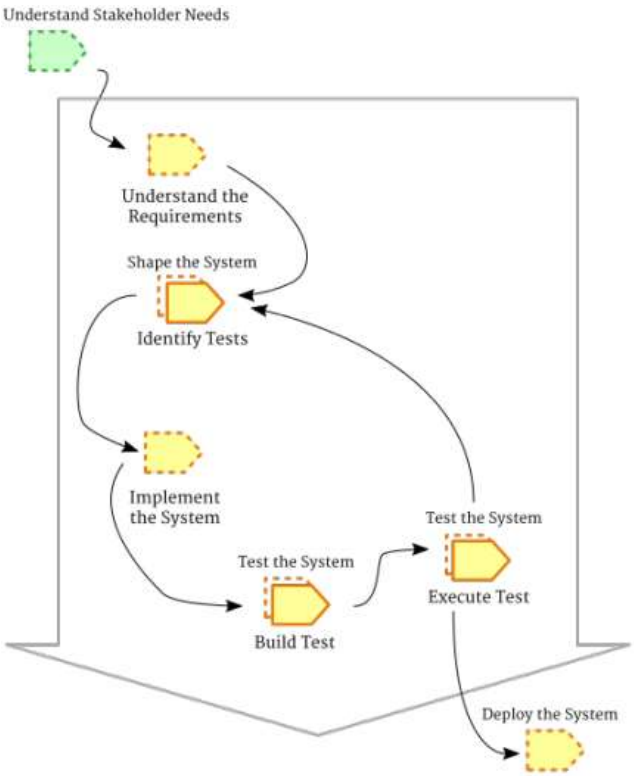


Figura 36: Prueba de Ejecución Esencial
Fuente: (Jacobson, 2016g)

Proceso Unificado práctica ciclo de vida

- Una forma ágil y escalable de controlar, planificar y rastrear proyectos de desarrollo de software.
- Utilizada para establecer controles sobre el ciclo de vida para el proyecto de desarrollo iterativo

Permite a los equipos:

- Establecer un ciclo de vida para el proyecto y planificar eficazmente de acuerdo con las circunstancias del proyecto.
- Compartir un conjunto de hitos comunes con otros proyectos y equipos
- Identificar los objetivos a corto plazo para reducir los niveles de riesgo que enfrentan
- Estructura los planes en una secuencia de fases bien entendidas
- Aprovecha al máximo los beneficios del desarrollo iterativo

Patrones de planificación de ciclo de vida

La práctica contiene un conjunto de patrones de planificación del ciclo de vida que aportan al equipo a:

- Entender en que parte se encuentra el proyecto y en qué circunstancias están para enfrentar los riesgos.
- Adoptar un marco de control estándar y establecer objetivos e hitos apropiados
- Planificar e iterar de manera controlada.
- Equilibrar la evolución de la arquitectura y los requisitos junto con el desarrollo de solución de software de alta calidad.

Como se muestra en la **Figura. 37**

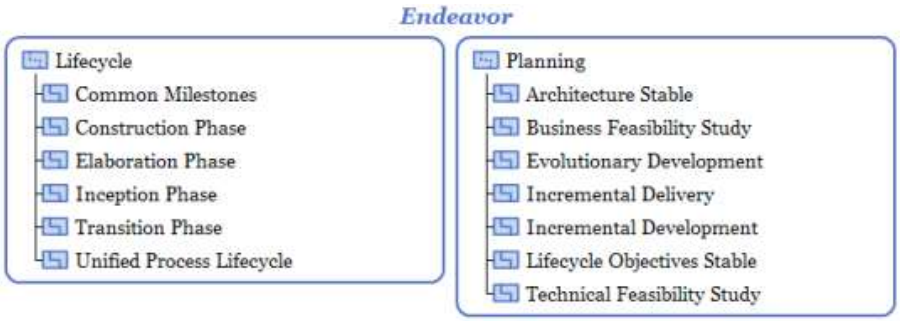


Figura 37: Patrones de planificación del ciclo de vida
Fuente: (Jacobson, 2016h)

Hitos Comunes

Este patrón determina el conjunto de hitos, que están adecuados para obtener una planificación y seguimiento de todos los estilos del proyecto.

Dicho patrón describe 3 hitos para cada uno de los ciclos de liberación de producto:

Objetivos del ciclo de vida (LCO). Toma decisiones sobre el lanzamiento del producto, se deben tomar los requerimientos principales operativos para el funcionamiento del producto.

Arquitectura del ciclo de vida (LCA). Se establece la arquitectura y se determina riesgos más importantes que tiene el software.

Capacidad operativa inicial (COI). En este hito el software está en total funcionamiento, y se lo debe preparar para pasar al proceso de transición a un ambiente de producción.

Como se observa en la siguiente **Figura. 38**

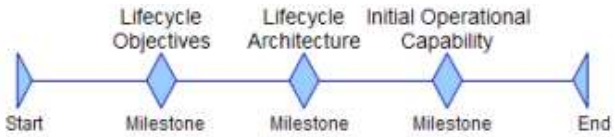


Figura 38: Hitos Comunes
Fuente: (Jacobson, 2016h)

Fases del ciclo de vida

Hace un refinamiento al patrón de hitos comunes, dentro de la cuales definen 4 fases del proyecto como se muestra en la **Figura. 39** siguiendo los 3 hitos comunes para llegar al éxito del proyecto

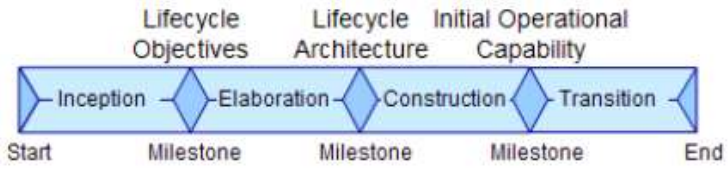


Figura 39: Fases del Proyecto
Fuente: (Jacobson, 2016h)

Las fases para el lanzamiento del producto se describen a continuación:

Incepción. Análisis de los objetivos, alcance ya planteados y los riesgos que surgirían

Elaboración. Plan de proyecto establecido, control de riesgos arquitectónicos y técnicos

Construcción. Desarrollo de producto bajo las normas de control de riesgos e inicio de la ejecución del proyecto.

Transición. Pruebas y entrega de producto, controlando los riesgos del despliegue

Control de la Evolución

Con la ayuda de estos patrones de fase e hitos se encuentra controlado la evolución del proyecto. A continuación, se muestra en la **Figura. 40**

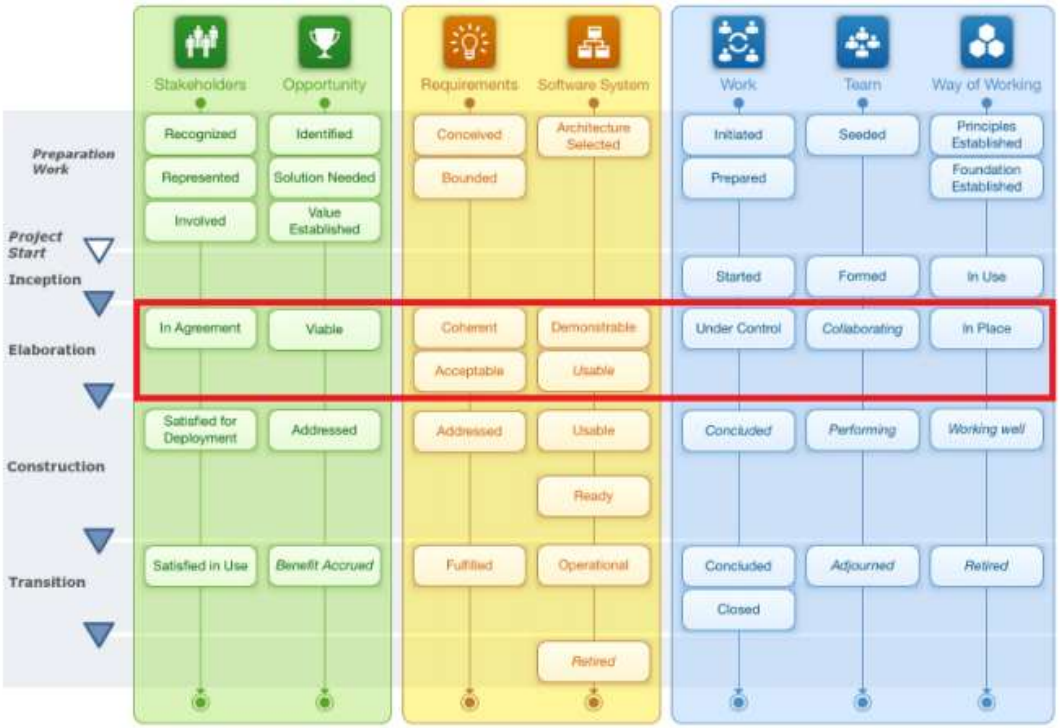


Figura 40: Control de Evolución
Fuente: (Jacobson, 2016h)

Fundamentos de casos de uso 2.0

Utiliza esta práctica para encerrar los requerimientos más importantes para promover el desarrollo del proyecto.

Permite a los equipos:

- Describir exactamente lo que debe hacer el sistema
- Reunir todos los requerimientos
- Tener la capacidad de adaptarse a los cambios que surjan durante el transcurso del proyecto según las necesidades del cliente

- Obtener un modelo sencillo de los requerimientos importantes, y deban ser entendibles para el equipo desarrollador y por ende para el cliente.
- Aprovechar al máximo el desarrollo realizado por iteraciones.

Cosas con las que trabajar

- Incluye obtener una serie de requerimientos, elementos de diseño y pruebas
- Especificaciones que suelen estar basadas en casos de usos de los requisitos de historias
- La elaboración de los casos de usos que promuevan el desarrollo del proyecto.

Competencias Claves

Se necesita que el equipo de trabajo esté capacitado en el levantamiento de los requerimientos de software, diseño codificación, integración y pruebas, las habilidades que sobresalen son la representación y análisis de las partes interesadas, ya que sin la comunicación necesaria que ayude a determinar el alcance se estaría desarrollando incorrectamente y los casos de usos no estarían cumpliendo con las necesidades del cliente.

Como se puede observar en la **Figura. 41**

Figura 41: Competencias Claves de Casos de Uso 2.0 Esenciales
Fuente: (Jacobson, 2016i)

Cosas para hacer

Inicia con la búsqueda de actores y casos de usos, haciendo una lista con lo más relevante de los casos de uso que se los va a elaborar.

Seguidamente se detalla los casos de usos y los casos de prueba que sean más necesarios para evaluar cada segmento desarrollado

La práctica concluye con el seguimiento del progreso del desarrollo del proyecto, para verificar, analizar los resultados que ayuden a manejar cambios y según lo obtenido el equipo de trabajo tenga una visualización de todos los resultados positivos negativos y vaya en busca de mejoras.

Como se muestra en la **Figura. 42**

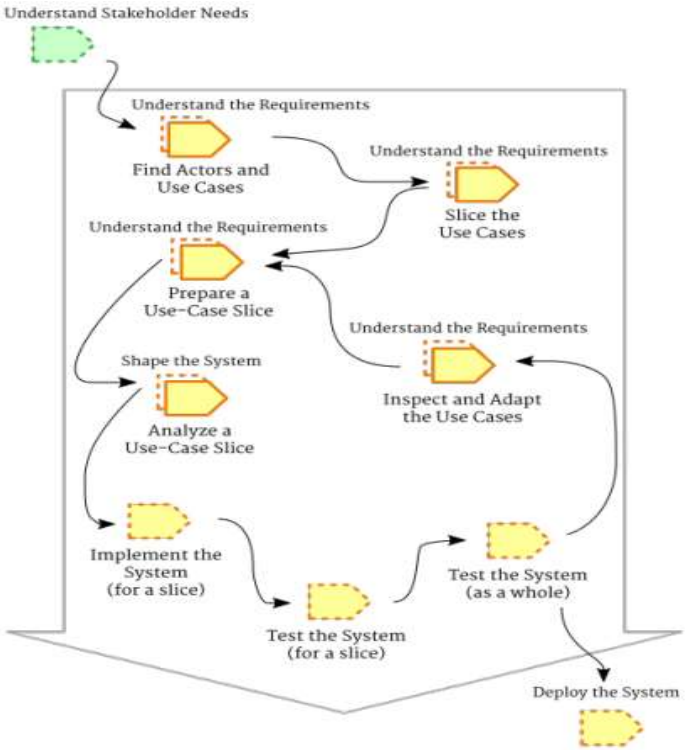


Figura 42: Use - Case 2.0 Essentials
Fuente: (Jacobson, 2016i)

2.4.4.2 Metodología Híbrida Ágil Scrum/XP

La mezcla de las metodologías Scrum con XP, una opción factible para el desarrollo de software, Scrum es una de las metodologías agiles más utilizadas como lo indica la encuesta realizada por **VersionOne** que la mantiene con un porcentaje de utilización de un 58% en el mercado de desarrollo en las que se puede citar empresas a nivel mundial que se identifican con este tipo de metodología por su adaptación a los cambios, la construcción rápida y eficaz, las empresas más reconocidas como Spotify, Adove , Google, Apple que representan grandes potenciales de la industria del software, plantean innovaciones de nuevos servicios y soluciones, con esto se da a conocer que Scrum se mantiene en un nivel alto por su forma de adaptarse al trabajo. Según el reporte de VersionOne. (VERSIONONE, 2016)

El dato también revela que en el mercado se está utilizando la metodología hibrida Scrum/XP en un 10%, es decir que las empresas poco a poco irán adaptándose a la metodología hibrida en un futuro. Como se muestra en la **Figura. 43**

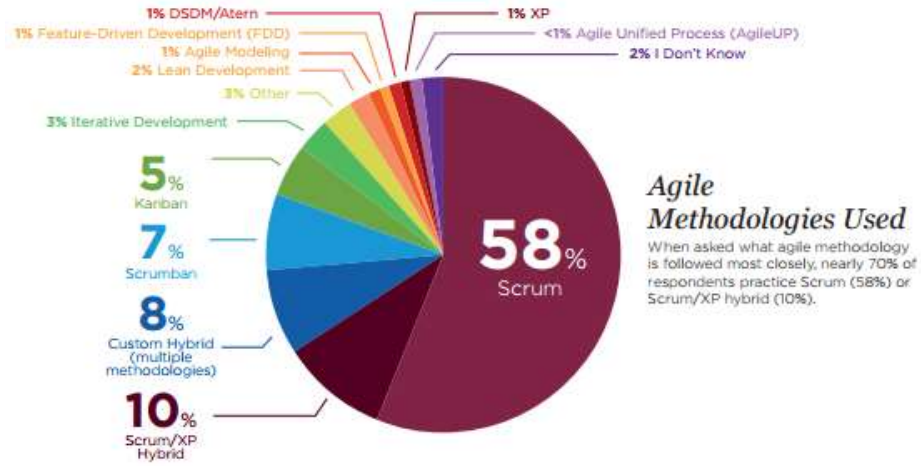


Figura 43: Metodologías Agiles
Fuente: (VERSIONONE, 2016)

La programación extrema o XP es una metodología ligera para el desarrollo de software que propicia un conjunto de valores y prácticas para aumentar la productividad, manteniendo una buena comunicación o una retroalimentación entre el cliente y el equipo de desarrollo, enfrentando cambios que surgen durante el trascurso del desarrollo del proyecto.

Al combinar Scrum que es un marco de trabajo centrado en prácticas de gestión y organización con XP que está centrada en las prácticas de programación, hacen una muy buena complementación entre ellas y manejan diferentes áreas.

Las prácticas que utiliza la metodología son las siguientes:

Programación por parejas

La programación en pareja está compuesta por un conductor que es la persona encargada de escribir el código y un observador que es el encargado de supervisar.

Identificándose con la siguiente frase:

"Unirse es un buen comienzo, mantener la unión es un progreso y trabajar juntos es la victoria". (Henry Ford)

Es una buena forma la de trabajar en parejas, ya que existe menos distracción, se comparte el código entre los integrantes del equipo de trabajo, es así como todos tienen conocimientos de lo que se está desarrollando, de esta manera el equipo está en capacidad de resolver cualquier problema, en determinado momento, cuando un integrante se encuentre ausente.

El trabajo en pareja aumenta la calidad del producto en desarrollo, aunque muchas personas piensen lo contrario, pero se puede entregar un mejor trabajo, ya que se requiere una sola computadora, con esto los integrantes están garantizando grandes beneficios, por lo que se pueden reducir los errores y costos en las pruebas.

Desarrollo guiado por pruebas (TDD)

Depende de otras dos prácticas, la primera es que debe escribir primero las pruebas, y la segunda la Refactorización es decir la restructuración del código, que la variación sea internamente sin afectar el comportamiento externo.

El desarrollo guiado por pruebas indica escribir un test automático, y a continuación escribir el código suficiente para pasar dicho test, luego realizar una refactorización al código en plan de mejorar la legibilidad y eliminar duplicaciones.(Henrik, 2015)

El TDD es muy complejo entenderlo para los programadores, ya que en muchas ocasiones se debe buscar un compañero que sea experto en TDD con el único afán de ajustarlo a las necesidades del proyecto, tiene un efecto muy significativo en el diseño del proyecto.

Algunas de las herramientas utilizadas para realizar pruebas son: JUnit, httpUnit, jWebUnit, HSQLDB base de datos embebida, Jetty como un contenedor web embebido. En el TDD se realizan las pruebas de aceptación tipo “Caja Negra” que significa evaluar los componentes como una caja negra, estas pruebas se inician con todo el sistema en memoria, las bases de datos y servidores web y acceder al sistema utilizando únicamente interfaces públicas como HTTP.(Henrik, 2015)

El TDD hace que los ciclos de desarrollo compilación y pruebas sean mucho más rápidos, proporciona al equipo de trabajo suficiente confianza para hacer la refactorización de código en cualquier momento, manteniendo un desarrollo limpio y simple mientras el sistema va creciendo.

TDD (Test Driven Development)

El TDD se lo realiza a todos los nuevos proyectos, porque representan beneficios grandes en el desarrollo.

TDD en código antiguo

TDD es complejo cuando no se lo realiza desde el inicio del proyecto

Diseño incremental

Es mantener el diseño simple desde el inicio del proyecto luego ir mejorándolo continuamente, invirtiendo el tiempo necesario para realizar la refactorización de código en mejora de los diseños existentes.

Integración continúa

El objetivo principal de la integración continua es que las actualizaciones de código no deben retrasar el desarrollo ni generar anomalías en el trascurso del proyecto. Es una de las prácticas que ofrece la programación extrema que consiste en verificar el funcionamiento del aplicativo de manera automática con la ayuda de una plataforma que detecta fallas en el menor tiempo posible.

Para el desarrollador es un poco complejo analizar la actualización del código, por el número de funciones que tenga la aplicación, pero a la vez es mejor tener una visualización de todo el desarrollo, porque las pruebas que se las realicen se las hacen en un entorno de producción.

Terminología de la integración continúa

Build. Está relacionada con la compilación y la elaboración de los entregables, en el momento que se realizan las pruebas.

Commit. Es la operación que se realiza para las validaciones de código fuente cuando se producen actualizaciones, se encuentra alojado en el directorio de la máquina en la que se esté desarrollando.

Update. Esta operación se la realiza a partir de la referencia del directorio local de la maquina

Checkout. Es la extracción de una versión del proyecto en desarrollo desde el directorio local de la máquina.

Para la realización de la integración continua se trabaja con servidores como se muestra a continuación.

- Cruise Control
- Hudson
- Continuum
- Bamboo

Propiedad colectiva del código

La propiedad colectiva del código o Collective Code Ownership toma el código fuente como propiedad del proyecto mas no como propiedad de la persona que desarrollo, porque en cualquier momento ese código puede utilizar otra persona para modificarlo, y todas aquellas personas que estén en el mismo ambiente de trabajo son responsables de la buena o mala calidad del producto. Con esto se demuestra que no solo se debe sentir orgulloso del código que programó, sino de todo el equipo que aporto para sacar adelante el proyecto.

Espacio informativo

Es un espacio donde todos pueden hacer uso de pizarras o espacios vacíos con información relativa al proyecto en ocasiones se observa que las paredes se encuentran llenas de papeles referentes al proyecto.

Estandarización de código

Cada programador tiene su forma de escribir su código fuente detalles como el estilo, los comentarios, en algunos casos esto no importa, pero en otros puede conducir a una severa inconsistencia del diseño del sistema y a un código difícil del leer. (Henrik, 2015)

Algunos ejemplos que proponen Henrik Kniberg Scrum y XP desde las Trincheras:

- Romper las reglas, pero asegurarse de que exista una buena razón para ello y documentarla
- Utilizar las convenciones que vienen por defecto en Java
- Evitar utilizar demasiadas abreviaturas

Utilizar técnicas de codificación sólidas y realizar buenas prácticas de programación con una visión de obtener código de muy buena calidad que ofrezca un buen rendimiento, utilizando técnicas de programación apropiadas que luego se someta a pruebas de rutina de evaluación

de código para que se convierta en un sistema de fácil entendimiento y mantenimiento.(Microsoft, 2016)

Ritmo sostenible/trabajo energético

Trabajar el tiempo que sea necesario sin excederse del límite, porque no se obtendría un trabajo bien hecho sino con errores que pueden afectar a la calidad del producto, no abusar de las horas de trabajo, ya que el equipo necesita estar en buen estado para solventar cualquier tipo de problema que surja, pero si el equipo no se encuentra en buenas condiciones puede afectar el aseguramiento del desarrollo del proyecto que en un futuro producirían retrasos en la entrega y por consiguiente problemas con el cliente.

CAPÍTULO II

3.1 Introducción al desarrollo del estudio comparativo

En este estudio comparativo se establecerán parámetros para determinar cuál metodología será la mejor opción para el desarrollo de software, donde las metodologías a tratar son las siguientes metodología EssUP (Essentials Unified Process) como de la metodología híbrida SCRUM/XP (eXtreme Programming), para lo cual se va ajustar a la norma ISO/IEC 12207 que se refiere a los procesos del ciclo de vida del software, donde se encuentran los parámetro de comparación, luego de realizar la comparativa los resultados obtenidos se verán reflejados en el aplicativo Sistema didáctico para enseñanza de Inglés para niños.

Se iniciará con la definición acerca de la norma a ser utilizada para realizar el benchmarking de las metodologías en estudio.

3.2 Fase de planeación

Es una fase muy impórtate ya que aquí se analizarán los parámetros de comparación acerca de las metodologías a comparar.

3.3 Norma ISO/IEC 12207

Proporciona un conjunto de procesos sobre el ciclo de vida del software, su primera publicación se la realiza el 1 de agosto de 1995, dicha norma consta de procesos, actividades y tareas, en los años 2003 y 2004 se realizaron reformas como las de añadir propósitos y resultados de la norma, conjuntamente con la norma ISO/15288 que trata acerca del proceso de vida del sistema creado por el hombre, complemento que mejorara el diseño de los sistemas. La norma puede ser utilizada de las siguientes maneras:

3.3.1 En una organización y asesores

Ayudar a mejorar el proceso organizacional estableciendo un entorno de trabajo, mediante evaluaciones. Los procesos pueden ser apoyados por una infraestructura de métodos, procedimientos, técnicas, herramientas y personal capacitado. En este punto la norma es utilizada para la evaluación de la conformidad del conjunto de procesos del ciclo de vida.(ISO/IEC, 2008)

3.3.2 En un proyecto

Ayuda a seleccionar, estructurar y emplear los elementos de un conjunto establecido de procesos de ciclo de vida para proporcionar productos y servicios. En este modo esta Norma Internacional se utiliza en la evaluación de la conformidad del proyecto con el entorno declarado y establecido.(ISO/IEC, 2008)

3.3.3 En un adquiriente y un proveedor

Se utiliza para ayudar a desarrollar un acuerdo sobre procesos y actividades, que se encuentran establecidos en un acuerdo, el que consta de seleccionar, negociar, realizar los procesos y actividades de la Norma Internacional. Es así como la Norma se la utiliza como una guía para elaborar acuerdos.(ISO/IEC, 2008)

3.4 Estándar

Conjunto de reglas y procedimientos muy bien documentados que ayudan a cumplir con los objetivos establecidos.

3.5 ISO

Que significa International Organization for Standardization que en español significa Organización Internacional de la Estandarización, organismo que desarrolla las normas sobre productos, comercio, comunicación, servicios, para lo cual su objetivo principal es el de estandarizar todos los productos a nivel internacional, en el que cada país es miembro de esta organización, donde su principal responsabilidad es la de respetar las reglas que son establecidas por la ISO.(Gallegos & Ortiz, 2011)

3.6 IEC

International Electronic Comission es la parte encargada de la normalización en el área de lo eléctrico, electrónico y tecnologías afines está relacionada con la Norma ISO, de aquí el nombre de las ISO/IEC.(Gallegos & Ortiz, 2011)

3.7 Norma ISO/IEC 12207:2008

Establece un marco para los procesos de ciclo de vida del software, los cuales están clasificados de la siguiente manera:

- Cinco procesos principales
- Ocho procesos de apoyo
- Cuatro procesos organizativos

La norma es utilizada para definir, controlar y mejorar los procesos del ciclo de vida del software.

Como se muestra en la **Figura. 44**

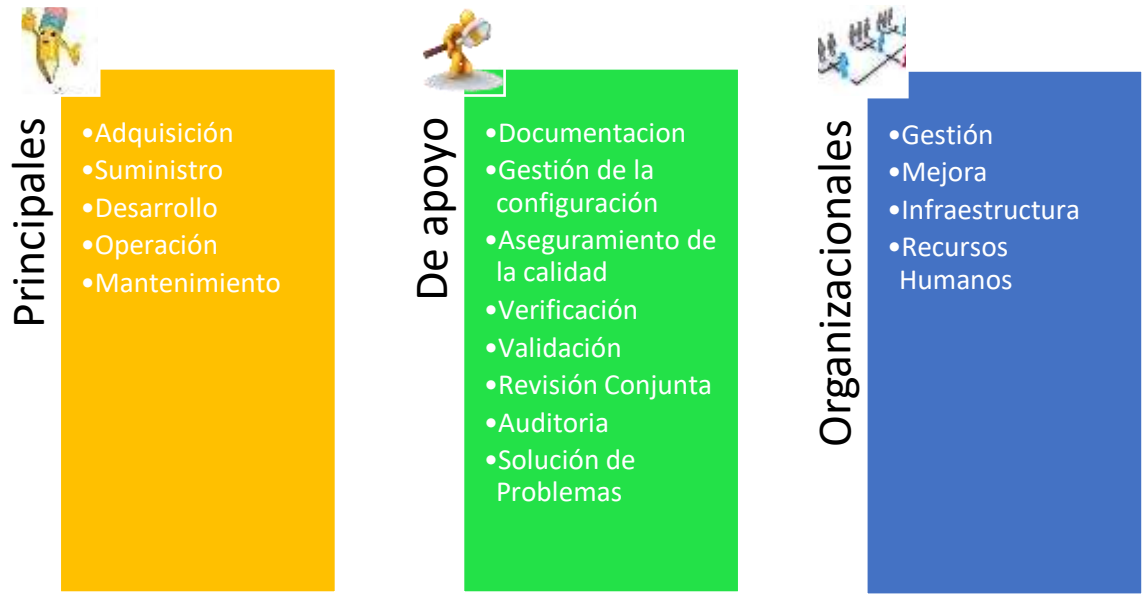


Figura 44: Procesos Ciclo de Vida Software
Fuente: ISO/IEC 12207

3.7.1 Procesos Principales

Conforman la parte principal para mejorar las funciones del ciclo de vida, como se vio anteriormente, consta de los siguientes procesos:

3.7.1.1 Adquisición

Inicia cuando se tiene identificado el producto a ser desarrollado. Continuando con la preparación y publicación de propuestas, incluyendo la selección de un proveedor y la gestación del proceso de adquisición hasta la aceptación del producto.(Huancho Arroyo, 2011). Consiste en las siguientes actividades:

- Inicio
- Preparación de la solicitud de propuestas
- Preparación y actualización del contrato
- Seguimiento del proveedor
- Aceptación y finalización

3.7.1.2 Suministro

Contiene las actividades, y tareas del proveedor, se inicia cuando se llega a firmar un contrato por la adquisición de un sistema, el proceso sigue con la determinación de procedimientos necesarios para realizar el aseguramiento del producto.(Huancho Arroyo, 2011). Consiste de las siguientes actividades

- Inicio
- Preparación de la respuesta
- Contrato
- Planificación
- Ejecución y control
- Revisión y evaluación
- Entrega y finalización

3.7.1.3 Desarrollo

Contiene todas las actividades que debe realizar el desarrollador, actividades relacionadas con el análisis de los requerimientos, diseño, codificación, pruebas e instalación y aceptación.(Huancho Arroyo, 2011). Consiste de las siguientes actividades

- Implementación del proceso
- Análisis de los requerimientos del sistema
- Diseño de la arquitectura del sistema
- Análisis de los requerimientos software
- Diseño de la arquitectura del software
- Diseño detallado del software
- Codificación y pruebas del software
- Integración del software
- Pruebas de calificación del software
- Integración del sistema
- Pruebas de calificación del sistema
- Instalación del software
- Apoyo a la aceptación del software

3.7.1.4 Operación

Cubre la operación del producto y el apoyo a la operación de los usuarios, como lo explica (Huancho Arroyo, 2011) con las siguientes actividades:

- Implementación del proceso
- Pruebas de operación
- Operación del sistema
- Soporte de Usuario

3.7.1.5 Mantenimiento

Actividades del responsable del encargado del mantenimiento. Este proceso da inicio cuando se ha realizado modificaciones al producto y a la documentación, el objetivo es mantener la integridad del producto a pesar de haberlo modificado con el fin de satisfacer necesidades. Consta de las siguientes actividades

- Implementación del proceso
- Análisis de problemas y modificaciones
- Implementación de las modificaciones
- Revisión/aceptación del mantenimiento
- Migración
- Retirada del software

3.7.2 Procesos de Apoyo

Es una responsabilidad por parte de la organización para llevar a cabo los procesos. Consta de los siguientes procesos.(Huancho Arroyo, 2011)

3.7.2.1 Documentación

Proceso en el cual se elabora la documentación necesaria de cada actividad desarrollada del ciclo de vida, con las siguientes actividades planificar, diseñar, desarrollar, producir, editar, distribuir y mantener con el afán de conservar en orden todos los documentos que necesiten la gerencia, ingenieros y usuarios del sistema.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación
- Diseño y desarrollo
- Producción

3.7.2.2 Gestión de la configuración

Proceso de aplicar procedimientos técnicos y administrativos a lo largo del ciclo de vida para identificar, definir y establecer la línea base de los elementos del sistema, controlar modificaciones y releases de los elementos, registrar e informar el estado de los elementos y

peticiones de modificaciones; asegurar la completitud, consistencia y corrección de los elementos y controlar el almacenamiento, manipulación y entrega de los elementos.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación del proceso
- Identificación de la configuración
- Control de la configuración
- Determinación del estado de la configuración
- Evaluación de la configuración
- Gestión de releases y entrega

3.7.2.3 Aseguramiento de la calidad

La calidad consiste en libertad organizativa y autoridad con respecto al personal responsable del desarrollo del proyecto.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación proceso
- Aseguramiento del producto
- Aseguramiento del proceso
- Aseguramiento del sistema de calidad

3.7.2.4 Verificación

Se hace la verificación para ver si cada actividad realizada cumple con los requerimientos planteados. Consta de las siguientes actividades

- Implementación del proceso
- Verificación

3.7.2.5 Validación

Proceso que determina si se cumplen cada uno de los requerimientos anteriormente planteados. En el cual se puede ejecutar en diversos grados de independencia. El grado varía dependiendo la persona u organización, cuando se trata de la validación por la persona es que se la debe realizar dentro de la misma organización que significa un grado de separación variable. En el caso de la organización el proceso de validación toma por nombre validación independiente.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación del proceso
- Validación

3.7.2.6 Revisión conjunta

Evalúa los estados y el producto, conjuntamente a nivel de gestión del proyecto y técnico que se mantiene a lo largo de la vida del contrato. La revisión puede ser empleada por la revisora o la revisada.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación del proceso
- Revisiones de la gestión del proyecto
- Revisiones técnicas

3.7.2.7 Proceso de Auditoria

Proceso para determinar el cumplimiento de los requerimientos que lo realiza el auditor o la parte auditada.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación del proceso
- Auditoria

3.7.2.8 Proceso de solución de problemas

Proceso para analizar y resolver problemas cualesquiera fuese su naturaleza u origen que se hallen durante la ejecución de los procesos de desarrollo, operación, mantenimiento. Siendo el objetivo principal llegar a buscar la solución de los problemas en el menor tiempo.(Huancho Arroyo, 2011). Consta de las siguientes actividades

- Implementación del proceso
- Solución de problemas

3.7.3 Procesos Organizativos

Establece una infraestructura constituida por procesos y personal asociado al ciclo de vida, mejorando la infraestructura los cuales son 4:

3.7.3.1 Proceso de la gestión

Contiene actividades genéricas, en el cual el gerente es directamente el responsable de la gestión de producto, del proyecto, y de las tareas de los procesos aplicables como: la adquisición, suministro, desarrollo, operación, mantenimiento o soporte.(Huancho Arroyo, 2011). Consta de las siguientes actividades:

- Inicio y definición del alcance
- Planificación
- Ejecución y control

- Revisión y evaluación
- Finalización

3.7.3.2 Proceso de Infraestructura

Proceso que establece y mantiene la infraestructura las cuales incluye a: software, hardware, herramientas, técnicas, normas e instalaciones para el desarrollo, operación y mantenimiento.(Huancho Arroyo, 2011). Consta de las siguientes actividades:

- Implementación del proceso
- Establecimiento de la infraestructura
- Mantenimiento de la infraestructura

3.7.3.3 Proceso de mejora de proceso

Proceso que evalúa, medir, controlar y mejorar un proceso del ciclo de vida. (Huancho Arroyo, 2011). Consta de las siguientes actividades:

- Establecimiento del proceso
- Evaluación del proceso
- Mejora del proceso de mejora

3.7.3.4 Proceso de Recursos Humanos

Proceso que mantiene al personal capacitado. La adquisición, suministros, desarrollo, operación o mantenimiento del producto, el personal que se quede encargado del sistema debe tener conocimientos en ingeniería y gestión de software.(Huancho Arroyo, 2011). Consta de las siguientes actividades:

- Implementación del proceso
- Desarrollo del material de formación
- Implementación del plan de formación

3.8 Fase de análisis comparativo

El proceso de comparación entre las metodologías EssUP y Scrum/XP inicia presentando un cuadro amplio de ventajas y desventajas entre las dos, las mismas que serán medidas en la escala de Likert, siendo establecidas por ítems.

3.8.1 Escala Tipo Likert

Es un tipo de instrumento de medición o de recolección de datos que se dispone en la investigación social para medir actitudes. Consiste en un conjunto de ítems bajo la forma de afirmaciones o juicios ante los cuales se solicita la reacción favorable o desfavorable, positiva o negativa de los individuos.(Malave, 2007)

Alternativas o Puntos Tipo Likert

Tabla 2: Valoración Escala Likert

Alternativas	Símbolo	Valor
Si	S	1
No	N	0

Fuente: (Malave, 2007)

Las alternativas presentadas de la escala tipo Likert en la tabla, se tomarán como referencia para la realización del análisis de la información acerca de las metodologías en estudio.

Donde:

S: significa si cumple con el parámetro

N: significa que no cumple con el parámetro

Se da el inicio a la comparativa con los siguientes parámetros referentes al ciclo de vida del software. Se inicia con lo siguiente:

3.8.2 Procesos Principales

Son procesos que brindan servicios a las partes principales durante el ciclo de vida del software.(Huancho Arroyo, 2011)

3.8.2.1 Proceso de Adquisición

En este punto se identifica y evalúa las actividades del adquiriente y el producto a ser desarrollado para las dos metodologías en los siguientes parámetros:

Tabla 3: Análisis del Proceso de Adquisición

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Inicio	S		S	
• Preparación de la solicitud de propuestas	S		S	
• Preparación y actualización del contrato	S		S	
• Seguimiento del proveedor		N		N
• Aceptación y finalización		N		N
Valoración	**3**	**2**	**3**	**2**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de adquisición relacionado con los procesos principales de cada una de las metodologías en lo que concierne al parámetro de inicio las dos metodologías cumplen con actividades relacionadas al análisis de requerimientos.

La preparación de solicitud de propuestas si cumplen con actividades relacionadas a requerimientos, alcance, restricciones, términos y condiciones que se plantean al inicio del proyecto.

Preparación y actualización de contrato donde se tratan asuntos relacionados a la adquisición que incluye costos, plazos y derechos de marco de uso, propiedad, garantía y licencias.

Seguimiento al proveedor las dos metodologías no cumplen con este parámetro porque cada metodología tiene la forma de cumplir con su parte en el contrato establecido.

Aceptación y finalización no cumple con el parámetro porque cada una tiene su forma de interactuar con el cliente.

3.8.2.2 Proceso de Suministro

En este punto se identifica y evalúa las actividades con respecto al proveedor, que tienen que ver con la firma del contrato y la adquisición del software

Tabla 4: Análisis del proceso de Suministro

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Inicio	S		S	
• Preparación de la respuesta	S		S	
• Contrato	S		S	
• Planificación	S		S	
• Ejecución y control	S		S	
• Revisión y evaluación	S		S	
• Entrega y finalización	S		S	
Valoración	**7**		**7**	

Fuente: Propia

Descripción de resultados

El resultado obtenido en el análisis del proceso de suministro según cada parámetro en las dos metodologías nos da como resultado que en su mayoría de parámetros el resultado es positivo cumple con cada especificación.

Proceso de Desarrollo

En este punto se identifican y evalúan las actividades del desarrollador que consiste en: el análisis de requerimientos, diseño, codificación, pruebas e instalación y aceptación.

Tabla 5: Análisis del Proceso de Desarrollo

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S			N
• Análisis de los requerimientos del sistema	S		S	
• Diseño de la arquitectura del sistema	S		S	
• Análisis de los requerimientos del software	S		S	
• Diseño de la Arquitectura del software	S		S	
• Diseño detallado del software	S		S	
• Codificación y pruebas de software	S			N
• Integración del software	S		S	
• Pruebas de calificación del software		N		N
• Integración de sistema	S		S	
• Pruebas de calificación del sistema		N		N
• Instalación del software	S		S	
• Apoyo a la aceptación del software	S		S	
Valoración	**11**	**2**	**9**	**4**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de documentación cumplen en su mayoría las dos metodologías, en los casos en los cuales no cumplen es porque cada metodología tiene su forma de documentar como es el caso de Scrum/XP, que no necesariamente necesita de la documentación de todos los artefactos, pero si se detalla en el caso de cada sprint realizado.

3.8.2.3 Proceso de Operación

En este punto se identifica y evalúan las tareas del operador y a nivel de organización que tiene que ver con el proceso de la gestión e infraestructura y a nivel de organización con el proceso de mejora de procesos y de recursos humanos, con los siguientes parámetros.

Tabla 6: Análisis del Proceso de Operación

Metodologías / **Parámetros**	**EssUP**		**Scrum/XP**	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Pruebas de operación	S		S	
• Operación del sistema	S		S	
• Soporte de Usuario	S		S	
Valoración	**4**		**4**	

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el proceso de operación cumplen con todos los parámetros en su totalidad, dando como un resultado la implementación del sistema de acuerdo al contrato establecido previamente.

3.8.2.4 Proceso de Mantenimiento

En este punto se identifica y evalúa actividades del responsable de mantenimiento, o entidad encargada de las modificaciones.

Tabla 7: Análisis del Proceso de Mantenimiento

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Análisis de problemas de las modificaciones	S		S	
• Implementación de las modificaciones	S		S	
• Revisión/aceptación del mantenimiento		N		N
• Migración		N		N
• Retirada del software		N		N
Valoración	**3**	**3**	**3**	**3**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el proceso de mantenimiento no cumplen en su totalidad con los parámetros de parte de las dos metodologías porque no cuentan con procesos de mantenimientos ya que son la mezcla de metodologías agiles y es evidente que en el proceso de desarrollo pueden efectuarse cambios que representarían a la mejora del sistema para que en su momento de puesta en producción sea exactamente lo que el cliente necesita.

3.8.3 Proceso de Apoyo

Ahora se inicia evaluando los procesos de apoyo de la siguiente manera

3.8.3.1 Proceso de Documentación

En este punto se evalúan la documentación que se produce después de cada proceso, lo cual consiste en planificar, diseñar, desarrollar, producir, editar, distribuir y mantener la documentación necesaria que sea de validez para los usuarios.

Tabla 8: Análisis del Proceso de Documentación

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Diseño y desarrollo	S			N
• Producción	S		S	
Valoración	**3**		**2**	**1**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos dentro de los procesos de apoyo lo que concierne al proceso de documentación cumplen con todos los parámetros comprendidos acerca de la documentación que se genera a partir del ciclo de vida del software.

3.8.3.2 Proceso de Gestión de la configuración

En este punto se evalúan procedimientos técnicos y administrativos que son la línea base de los elementos del software.

Tabla 9: Análisis del Proceso de Gestión de la Configuración

Metodologías / **Parámetros**	**EssUP**		**Scrum/XP**	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso		N		N
• Identificación de la configuración	S			N
• Control de la configuración	S		S	
• Determinación del estado de la configuración		N		N
• Evaluación de la configuración	S		S	
• Gestión de releases y entrega	S		S	
Valoración	4	2	4	2

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de gestión de la configuración cumplen con los parámetros de la mitad más uno, relacionados a los releases, modificaciones que se producen a lo largo del ciclo de vida del software.

3.8.3.3 Proceso de Aseguramiento de la Calidad

En este punto se evalúa la seguridad cumpliendo con los requisitos previamente establecidos en la planificación.

Tabla 10: Análisis del Proceso de Aseguramiento de la Calidad

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S			N
• Aseguramiento del producto	S		S	
• Aseguramiento del proceso	S		S	
• Aseguramiento del sistema de calidad		N		N
Valoración	**3**	**1**	**2**	**2**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso del aseguramiento de la calidad, se puede observar que las dos metodologías cumplen mitad por mitad ya que cada metodología tiene su forma de manejar los contratos según las políticas de la empresa y cumplir con las determinadas actividades a las que estén orientadas.

3.8.3.4 Proceso de Verificación

En este punto se evalúa el cumplimiento con los requerimientos que fueron impuestos

Tabla 11: Análisis del Proceso de Verificación

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S			N
• Verificación	S		S	
Valoración	**2**		**1**	**1**

Fuente: Propia

Descripción de resultados

El resultado obtenido en el análisis del proceso de verificación cumple con uno de los dos parámetros impuestos en la tabla esto quiere decir que el único parámetro que logra cumplir es el de verificación en el cual especifica que se debe cumplir con todo lo establecido por el proveedor para cumplir con las necesidades del cliente.

3.8.3.5 Proceso de Validación

En este punto se evalúa la validación y construcción del producto a partir de su construcción.

Tabla 12: Análisis del Proceso de Validación

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Validación	S		S	
Valoración	**2**		**2**	

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de validación cumplen con los parámetros de las dos metodologías, proceso en el cual se encarga de cumplir con todas las validaciones correspondientes al sistema.

3.8.3.6 Proceso de Revisión Conjunta

En este punto se evalúa el estado y productos de las actividades del proyecto

Tabla 13: Análisis del Proceso de Revisión Conjunta

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Revisión de la Gestión del proyecto		N		N
• Revisiones técnicas	S		S	
Valoración	**2**	**1**	**2**	**1**

Fuente: Propia

Descripción de resultados

Los resultados del análisis del proceso de revisión conjunta arrojan que el único parámetro en el cual no se ajustan ninguna de las dos metodologías, es en la revisión de la gestión del proyecto, ya que no cumple con todas las actividades que se encuentran en el proceso las cuales son mantenimiento, cambio y la evaluación.

3.8.3.7 Proceso de Auditoria

En este punto se evalúa el cumplimiento de los requerimientos, planes y contrato.

Tabla 14: Análisis del Proceso de Auditoria

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso		N		N
• Auditoria		N		N
Valoración		**2**		**2**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el proceso de auditoria no cumplen con los parámetros en las dos metodologías, ya que depende de la organización en la que se encuentre, para que pueda habilitar el proceso de auditoria

3.8.3.8 Proceso de Solución de Problemas

En este punto se evalúa la resolución de problemas cualquiera fuese su naturaleza.

Tabla 15: Análisis del Proceso de Solución de Problemas

Metodologías Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Solución de problemas	S		S	
Valoración	**2**		**2**	

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de solución de problemas cumplen con los parámetros especificados en las metodologías, cumpliendo así con la detección y posterior solución para cualquier problema que surja en el trayecto del desarrollo del software.

3.8.4 Procesos Organizativos

Ahora se inicia evaluando los procesos de apoyo de la siguiente manera

3.8.4.1 Proceso de Gestión

En este punto se evalúa la gestión de los procesos durante el ciclo de vida,

Tabla 16: Análisis del Proceso de Gestión

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Inicio y definición del alcance	S		S	
• Planificación	S		S	
• Ejecución y control	S		S	
• Revisión y evaluación	S		S	
• Finalización	S		S	
Valoración	**5**		**5**	

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de gestión arrojan que se cumplen todos los parámetros estipulados con respecto a la gestión del cumplimiento de los requerimientos plateados en el contrato, cuando se dé inicio al proyecto.

3.8.4.2 Proceso de Infraestructura

En este se punto se evalúa la infraestructura en hardware, software.

Tabla 17: Análisis del Proceso de Infraestructura

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso		N		N
• Establecimiento de la infraestructura		N		N
• Mantenimiento de la infraestructura		N		N
Valoración		**3**		**3**

Fuente: Propia

Descripción de resultados

El resultado del análisis del proceso de infraestructura no cumple con los parámetros establecidos porque la infraestructura le compete al cliente al cual se le va a implementar el sistema y debe brindar un buen espacio físico, los equipos necesarios para que el software se integre en un entorno de producción a la medida del sistema.

3.8.4.3 Proceso de Mejora de Procesos

En este punto se evalúa las actividades de la organización con respecto a establecer, medir, controlar y mejora de procesos.

Tabla 18: Análisis del Proceso de Mejora de Procesos

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Establecimiento del proceso	S		S	
• Evaluación del proceso		N		N
• Mejora del proceso de mejora	S		S	
Valoración	2	1	2	1

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de mejora de procesos cumplen con dos de los tres parámetros establecidos de parte de las dos metodologías en los cuales se hace referencia a recopilar datos acerca de los procesos implantados en el desarrollo del proyecto para realizar mejoras en dichos procesos.

3.8.4.4 Proceso de Recursos Humanos

En este punto se evalúa las actividades concernientes a la búsqueda del personal capacitado para el desarrollo de los proyectos.

Tabla 19: Análisis del Proceso de Recursos Humanos

Metodologías / Parámetros	EssUP		Scrum/XP	
	SI	**NO**	**SI**	**NO**
• Implementación del proceso	S		S	
• Desarrollo del material de formación		N		N
• Implementación del plan de formación	S		S	
Valoración	**2**	**1**	**2**	**1**

Fuente: Propia

Descripción de resultados

Los resultados obtenidos en el análisis del proceso de recursos humanos cumplen con los parámetros establecidos para obtener un personal capacitado para trabajar a tiempo completo y bajo responsabilidades en los proyectos.

3.9 Análisis Comparativo

En esta fase se realiza un análisis comparativo entre las dos metodologías híbridas.

3.10 Análisis Final

El análisis realizado anteriormente se puede observar que en la mayoría de los resultados las dos metodologías EssUP Y Scrum/XP cumplen con los parámetros establecidos seguidos por la norma ISO/IEC 12207 acerca del ciclo de vida del software a partir del análisis se realizará la aplicación sistema didáctico para enseñanza de inglés.

Se concluye como la mejor opción para el desarrollo del aplicativo a la metodología Scrum/XP por la combinación de las dos metodologías sacando buenas prácticas para desarrollo de software, que en la actualidad esta metodología híbrida está siendo utilizada, más que las metodologías tradicionales y agiles, será evaluado en la escala de tipo Likert, con esto se llegará a la contabilización para la obtención de los valores de la comparación.

Tabla 20: Análisis Final Comparativa

Metodologías	**EssUP**		**Scrum/XP**	
	SI	**NO**	**SI**	**NO**
Procesos Principales				
Proceso de Adquisición				
• Inicio	S		S	
• Preparación de la solicitud de propuestas	S		S	
• Preparación y actualización del contrato	S		S	
• Seguimiento del proveedor		N		N
• Aceptación y finalización		N		N
Valoración	**3**		**3**	
Proceso de Suministro				
• Inicio	S		S	
• Preparación de la respuesta	S		S	
• Contrato	S		S	
• Planificación	S		S	
• Ejecución y control	S		S	
• Revisión y evaluación	S		S	
• Entrega y finalización	S		S	
Valoración	**7**		**7**	
Proceso de Desarrollo				
• Implementación del proceso	S			N
• Análisis de los requerimientos del sistema	S		S	
• Diseño de la arquitectura del sistema	S		S	
• Análisis de los requerimientos del software	S		S	
• Diseño de la Arquitectura del software	S		S	
• Diseño detallado del software	S		S	

• Codificación y pruebas de software	S			N
• Integración del software	S		S	
• Pruebas de calificación del software		N		N
• Integración de sistema	S		S	
• Pruebas de calificación del sistema		N		N
• Instalación del software	S		S	
• Apoyo a la aceptación del software	S		S	
Valoración	**11**		**9**	
Proceso de Operación				
• Implementación del proceso	S		S	
• Pruebas de operación	S		S	
• Operación del sistema	S		S	
• Soporte de Usuario	S		S	
Valoración	**4**		**4**	
Proceso de Mantenimiento				
• Implementación del proceso	S		S	
• Análisis de problemas de las modificaciones	S		S	
• Implementación de las modificaciones	S		S	
• Revisión/aceptación del mantenimiento		N		N
• Migración		N		N
• Retirada del software		N		N
Valoración	**3**		**3**	
Procesos de Apoyo				
Proceso de Documentación				
• Implementación del proceso	S		S	
• Diseño y desarrollo	S			N
• Producción	S		S	
Valoración	**3**		**2**	
Proceso de Gestión de Configuración				
• Implementación del proceso		N		N
• Identificación de la configuración	S			N
• Control de la configuración	S		S	
• Determinación del estado de la configuración		N		N
• Evaluación de la configuración	S		S	
• Gestión de releases y entrega	S		S	
Valoración	**4**		**3**	
Proceso de Aseguramiento de la calidad				
• Implementación del proceso	S			N
• Aseguramiento del producto	S		S	
• Aseguramiento del proceso	S		S	
• Aseguramiento del sistema de calidad		N		N
Valoración	**3**		**2**	
Proceso de Verificación				
• Implementación del proceso	S			N
• Verificación	S		S	
Valoración	**2**		**1**	
Proceso de Validación				
• Implementación del proceso	S		S	

• Validación	S		S	
Valoración	**2**		**2**	
Proceso de Revisión Conjunta				
• Implementación del proceso	S		S	
• Revisión de la Gestión del proyecto		N		N
• Revisiones técnicas	S		S	
Valoración	**2**		**2**	
Proceso de Auditoria				
• Implementación del proceso		N		N
• Auditoria		N		N
Valoración		**2**		**2**
Proceso de Solución de Problemas				
• Implementación del proceso	S		S	
• Solución de problemas	S		S	
Valoración	**2**		**2**	
Procesos Organizativos				
Proceso de Gestión				
• Inicio y definición del alcance	S		S	
• Planificación	S		S	
• Ejecución y control	S		S	
• Revisión y evaluación	S		S	
• Finalización	S		S	
Valoración	**5**		**5**	
Proceso de Infraestructura				
• Implementación del proceso		N		N
• Establecimiento de la infraestructura		N		N
• Mantenimiento de la infraestructura		N		N
Valoración		**3**		**3**
Proceso de Mejora de Procesos				
• Establecimiento del proceso	S		S	
• Evaluación del proceso		N		N
• Mejora del proceso de mejora	S		S	
Valoración	**2**	**1**	**2**	**1**
Proceso de Recursos Humanos				
• Implementación del proceso	S		S	
• Desarrollo del material de formación		N		N
• Implementación del plan de formación	S		S	
Valoración	**2**		**2**	
Total	**55/73**		**49/73**	
Porcentaje de 100%	**75,34**		**67,12**	

Fuente: Propia

Como se puede observar en la tabla, nos indica que la metodología que cumple con la mayoría de parámetros establecidos es Essential Unified Process, porque cumple con la norma ISO/IEC 12207 sobre el ciclo de vida del software, por otro lado se tiene la metodología Scrum/XP que no cumple con algunos de los requisitos de la norma, porque al representar una metodología hibrida ágil se ajusta a cambios producidos a lo largo del desarrollo del

proyecto con el único fin de satisfacer las necesidades del cliente, es por esto que la metodología escogida para el desarrollo de aplicativo sistema didáctico para enseñanza de Inglés es Scrum/XP.

Para sacar el porcentaje se realizó una regla de tres con los datos obtenidos en el análisis. Según el porcentaje obtenido en el análisis es el siguiente por cada metodología como se muestra en la **Figura. 45**.

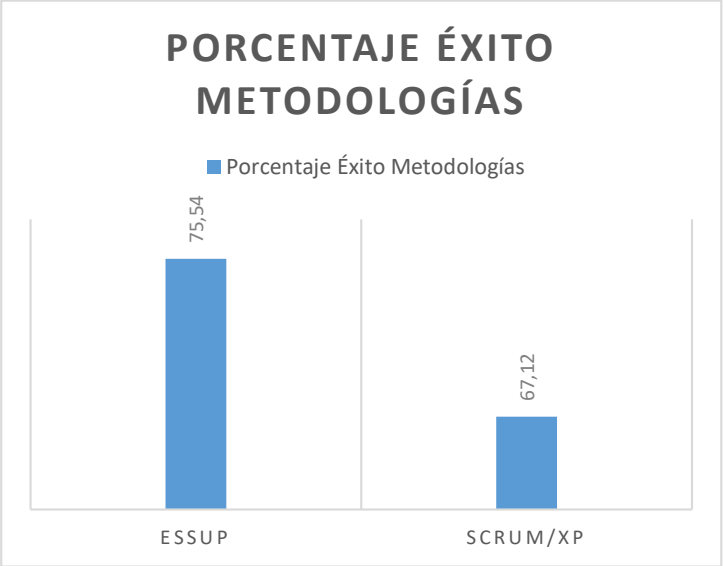


Figura 45: Porcentaje Análisis Metodologías
Fuente: Propia

El porcentaje obtenido de la metodología EssUP es superior al obtenido con Scrum/XP la diferencia está en que EssUP no ha perdido la esencia de la metodología RUP en lo que se refiere a la documentación que se la realiza cuando se la utiliza, EssUP se centra en prácticas que no son necesarias utilizarlas todas ellas, sino aquellas que se ajusten a las necesidades que tenga que cubrir, dicha metodología es utilizada en proyectos grandes en los cuales se tenga equipos de más de 50 personas encargadas de la realización del proyecto.

Por otro lado tenemos a Scrum/XP metodología que no se centra mucho en la documentación de los artefactos, solo en la necesaria por ejemplo en los Sprint es donde se necesita tener respaldo de las actividades que se realizaron, en el caso que existiera algún cambio que se debería realizar, de ahí lo demás es trabajado en comunicación con todo el equipo de desarrollo, esto lo hacen en pizarras donde es más clara la explicación y aclaración de dudas que existan por los miembros del equipo, una forma más práctica de trabajar. Scrum tienen más que ver con la organización de proyecto y XP en la parte de desarrollo, al unir estas dos

metodologías se obtienen las prácticas necesarias para el desarrollo de proyectos con el objetivo de hacer más rápido el trabajo disminuyendo tiempo.

3.11 Conclusión Scrum/XP

En el estudio realizado con respecto a las metodologías híbridas para el desarrollo de software, Scrum/XP metodología que se ajusta más en proyectos que van a duran en un lapso de tiempo de entre 2 a 6 meses, es una opción factible para trabajar en el ambiente que propicia esta metodología, al estar en constante comunicación con el cliente que es una pieza importante dentro del equipo de trabajo, la cual produce una retroalimentación entre el cliente y el equipo.

Cuando se produce algún problema en el transcurso del desarrollo, el equipo está en condiciones de resolverlo de la mejor manera sin afectar al trabajo que ya está realizado, es así como el personal debe estar bien capacitado en el caso de que surgiera algún inconveniente con algún miembro del equipo.

Conforme ha ido avanzando el tiempo este tipo de metodologías híbridas están siendo utilizadas en el campo de desarrollo de software en grandes empresas como es el caso de Scrum metodología más utilizada por empresas reconocidas como Google, Adobe, entre otras.

En conclusión, la metodología que se utilizará para el diseño del aplicativo es Scrum/XP por la realización de un estudio aplicando parámetros de comparación acerca de las metodologías hibridas y las prácticas que se deben combinar para su realización sin perder la esencia que tiene las dos metodologías por separado.

CAPÍTULO III

4.1 Especificación de la metodología a utilizar para el desarrollo del sistema didáctico para enseñanza de inglés.

El siguiente es el Marco de trabajo Scrum con XP, con el que se va a desarrollar el aplicativo, a continuación, se detallar cada proceso que entra a partir de la combinación de las dos metodologías. El marco que se presenta a continuación propone (León, 2015)

Tabla 21: Marco de Trabajo Scrum/XP

Marco de Trabajo Scrum/XP				
Roles	Product Owner			
	Scrum Master			
	Scrum Team(de 5 a 7 miembros)			
	Encargado de Pruebas			
Actividades o Procesos	Sprint Planning	Planeación		
		Análisis		
		Diseño	Metáfora	
			Diseño simple	
	Sprint	Codificación		
		Pruebas	Pruebas Unitarias	
			Pruebas de Aceptación	
			Pruebas de Integración	
			Uso de Bancos de Datos Automatizados	
	Scrum diario			
	Sprint Review Meeting			
	Retrospectiva			
	Replanificación del proyecto Release Planning Meeting			
Artefactos	Product Backlog	Historias de Usuario		
	Sprint Backlog	Tarjetas CRC (Clase Responsabilidad Colaboración)		
		Task cards		
	Burn down chart			
	Burn up chart			
Valores	Comunicación			
	Retroalimentación			
	Responsabilidad			
Prácticas técnicas	Iteraciones de 2 a 4 semanas			
	Refactoring			
	Programación en parejas			
	Liberaciones cortas			
	Código estándar			
	Integración continua			
	Propiedad colectiva			

Fuente:(León, 2015)

4.1.1 Roles

Roles que propone el marco de trabajo con la metodología híbrida Scrum/XP

4.1.1.1 Product Owner

Es el dueño del producto representa al cliente, y es el encargado de identificar las funcionalidades del producto.

4.1.1.2 Scrum Master

Encargado de aclarar dudas que tengan los miembros del equipo, para resolver los problemas.

4.1.1.3 Scrum Team

Representa al equipo de trabajo que deben ser integrantes de entre 3 a 5 personas

4.1.1.4 Encargado de Pruebas

Encargado de generar entornos de pruebas en el cual hace participación el cliente para la ejecución de las mismas.

4.1.2 Actividades/Procesos

Actividades y procesos que deben ejecutarse luego de la definición de los roles

4.1.2.1 Sprint Planning

Planificación de las tareas a ser realizadas en las iteraciones que son divididas en dos componentes:

1. Reunión con el cliente y el equipo de trabajo con el fin de presentar los requerimientos y planificar una meta para la iteración, luego de esto el equipo examinara toda la lista de requerimientos, si surgen preguntas al cliente estas deben ser expuestas a todo el equipo para mantener un panorama más claro de lo que se vaya a desarrollar.(León, 2015)

2. El equipo de trabajo debe realizar una planificación de cada iteración determinando estrategias para conseguir el resultado esperado. Definiendo tareas necesarias en el desarrollo de las iteraciones que son creadas en el Sprint Backlog.(León, 2015)

4.1.2.2 Sprint

Este es un punto muy importante, porque su principal propósito es tener suficiente conocimiento y brindar información al equipo de trabajo, para poder iniciar con el trabajo en las iteraciones que están planteadas.(León, 2015)

Para lograr todo esto se necesita de lo siguiente (León, 2015)

- Una meta bien definida

  Tener una meta bien definida es lo que hace que el sprint se concentre en lo que se tiene que cumplir para llegar a la finalización y entrega del producto

- La lista de miembros del equipo y su compromiso

  El equipo es muy bien definido por profesionales, los cuales se encargan de las historias de usuarios y tareas para finalizar las iteraciones. Las mismas tareas en que el equipo debe comprometerse para el cumplimiento en la entrega de funcionalidad de los avances del proyecto.

- La fecha y lugar para la demostración del avance

El desarrollo de cada una de las tareas deben ser aplicadas al TDD, desarrollo basado por pruebas con la ayuda del Tester se definen las pruebas y cuando estén listas se procede al desarrollo del código para analizar la funcionalidad de la pruebas.(León, 2015)

4.1.2.3 Scrum Diario

Son reuniones que se realizan a diario de una forma rápida de 15 minutos, para revisar los avances realizados por equipo, exponiendo lo que se hizo antes de la reunión, y lo que se va a realizar en el siguiente día de trabajo. Realizando el grafico del Burn Down con todas las tareas actualizadas de cada miembro del equipo, este tipo de reuniones se las realiza en tableros o pizarras una forma más rápida de tratar asuntos.

4.1.2.4 Sprint Review Meeting (Demo)

Es una reunión de duración de no más de 4 horas en la cual se trata asuntos relacionados con el avance de las funcionalidades planificadas en el Sprint detalladamente desde que punto hasta qué punto se va a realizar la revisión, con una demostración de los avances solo lo que este detallado en la agenda de presentación, esto se presenta al Product Owner y Stakeholders.

Siguiendo las siguientes reglas(León, 2015)

- Mostrar
- Ilustrar
- Aplicar
- Sección de preguntas

4.1.2.5 Retrospectiva

Es una reunión en la cual se trata asuntos acerca de todo lo que se realizó en las iteraciones que están finalizando, y evaluar si se está cumpliendo con los objetivos planteados.

4.1.2.6 Planificación de siguiente Iteración/Release Planning Meeting

Revisión de prioridades más importantes del proyecto junto con el cliente.

4.1.3 Artefactos

Procesos que deben ser usados durante el desarrollo de un proyecto los cuales se especifican a continuación.

4.1.3.1 Product Backlog

Representa una lista priorizada de la visión y expectativas del cliente con respecto a los objetivos y entregas del producto.

Donde el cliente tiene las siguientes responsabilidades(León, 2015):

- Registrar y actualizar los requerimientos necesarios para el sistema.

Responsabilidades del Scrum master:

- Supervisar la pila de requerimientos
- Mantener reuniones con el cliente en el caso de exista alguna duda que aclarar.

Responsabilidades del equipo de trabajo:

- Tener conocimiento y una muy buena comprensión de los requerimientos
- Resolución de dudas en el caso de que las existieran

4.1.3.2 Historias de Usuario

Este artefacto se lo utiliza para que el cliente se responsabilice y gestione la lista de requerimientos con la ayuda del Scrum Master.(León, 2015)

4.1.3.3 Sprint Backlog

Es un documento en el cual se registra el detalle de todos los requerimientos que se desarrollaran en las iteraciones, en el cual el cliente debe estar presente para la realización de la panificación del Sprint para las posibles soluciones de dudas en las historias de usuarios.(León, 2015)

4.1.3.4 Tarjetas CRC (Clase Responsabilidad Colaboración)

La creación de estas tarjetas es partir de las clases que se presentan, donde consta en la parte izquierda de la tarjeta representa las responsabilidades y en la parte derecha la lista de los colaboradores de la clase, es una forma de ayudar a entender al equipo que tipo de tareas tienen asignadas.(León, 2015)

4.1.3.5 Tarjetas de Tarea (Task Card)

Es muy recomendable usar este tipo de tarjetas que sirven para determinar las tareas de cada miembro del equipo y se trabaja de manera gráfica en pizarras donde se puede observar que actividades van a ser desarrolladas o están siendo trabajadas, esto lo realizan en reuniones diarias.(León, 2015)

4.1.3.6 Burn down chart

Es un gráfico que indica la velocidad de las tareas del equipo en las iteraciones, para comprobar si existe algún retraso en las actividades.

El equipo es el responsable de mantener actualizado este grafico dependiendo de la fecha en la que se haya planificado terminar cada tarea

4.1.3.7 Burn up chart

Es un gráfico en el cual indica la evolución del producto, y este sea visualizado por el cliente.

El equipo de trabajo como el Scrum Master debe tener conocimiento del gráfico para poder resolver cualquier duda que surja durante el desarrollo.(León, 2015)

4.1.4 Valores

Valores que el equipo debe tener para una mejor comunicación entre cada uno de los miembros.

4.1.4.1 Comunicación

Este es un valor muy importante desde el momento en el cual se realiza el levantamiento de los requerimientos con el cliente, así como también en las reuniones sobre re planificación que se realizaran posterior al inicio del proyecto, haciendo de este tipo de reuniones frente a frente para que exista mucha más comunicación entre todos los integrantes.

4.1.4.2 Retroalimentación

Es necesario tener este tipo de retroalimentación ya sea con el cliente, y también con el equipo de trabajo para poder aclarar dudas y proseguir con el desarrollo, este va de la mano con el valor de la comunicación.

4.1.4.3 Responsabilidad

Este valor se debe aplicar para todo el equipo, porque cada miembro debe ser responsable de las tareas que se le asignan y el tiempo que se le estima en el desarrollo del proyecto.

4.1.5 Prácticas Técnicas

Prácticas que el equipo mantiene, para hacer el desarrollo del proyecto, haciendo que el equipo de desarrollo tenga una visión amplia de lo que se desarrolló.

4.1.5.1 Iteraciones

La práctica se la toma desde Scrum con el fin de establecer tareas que duren no más de 2 semanas por Sprint, para ir en junta de las historias de usuario para no generar complicaciones que requieran mayores esfuerzos y luego se debe pasar a las revisiones.

4.1.5.2 Refactoring

Permite tener un código mucho más limpio y fácil de entender, removiendo el código que sea innecesario, realizando comentarios haciendo que las personas que vayan hacer uso del mismo puedan entender de lo que se trata.

4.1.5.3 Programación en parejas

Es una forma de trabajar en parejas es mucho más fácil la detección de errores en código, ya que mientras el uno hace el otro va revisando.

4.1.5.4 Liberaciones cortas

Son pequeñas versiones en las que el equipo de desarrollo trabaja, para mostrar al cliente cómo evoluciona el proyecto.

4.1.5.5 Código estándar

Cada programador tiene su forma de escribir su código fuente detalles como el estilo, comentarios, en algunos casos no importa, pero en otros puede conducir a una severa inconsistencia del diseño del sistema y un código complejo de entender.(Henrik, 2015)

4.1.5.6 Integración continúa

La integración continua hace referencia a las actualizaciones que se realizan en el código, en el cual no deben retrasar el desarrollo ni generar anomalías y para la verificación del código utilizan plataformas especializadas en la integración continua del código.

4.1.5.7 Propiedad Colectiva

Al hablar de propiedad colectiva se refiere a que el código no pertenece a la persona que lo desarrollo sino propiedad del proyecto, porque en cualquier momento puede ser utilizado por otra persona para ser modificado.

4.2 Desarrollo del aplicativo aplicando la metodología Scrum/XP

Se da inicio al desarrollo del aplicativo denominado sistema para enseñanza de inglés, aplicando la metodología Scrum/XP.

4.2.1 Introducción

En este capítulo se planifico trabajar con la metodología híbrida Scrum/XP que salió a partir de la realización de la comparativa entre dos metodologías híbridas, seguidamente se procede a la aplicación del marco de trabajo propuesto por Scrum/XP.

4.2.2 Definición de las políticas de calidad

Estas políticas son estipuladas por la empresa en la cual vaya a ser aplicada la metodología, debe ser dirigida por los altos ejecutivos, quienes deben tener en claro el mejoramiento continuo de la misma, teniendo en cuenta que el objetivo es satisfacer la necesidad del cliente y obtener productos de calidad. Todos los miembros de la empresa deben tener conocimiento acerca de las políticas, para trabajar conjuntamente, en beneficio de la misma.

4.2.3 Conformación de los equipos de trabajo

La conformación de los equipos de trabajo tiene que ver mucho con el ambiente en el cual esté trabajando, con esto el cliente sabe que el equipo de trabajo puede adaptarse a nuevos procesos y a cambios inesperados. Esperando tener alta flexibilidad, disciplina y

responsabilidad por parte de cada uno de sus miembros. Para lo cual se conformará de la siguiente manera:

- Product Owner
- Scrum Master
- Encargado de Pruebas

Es así como queda conformado el nuevo equipo de trabajo con la metodología Scrum/XP, ya que se necesita que cada miembro se ayude mutuamente, tanto en la programación como en la comunicación con los compañeros de trabajo y por ende con los usuarios.

4.2.4 Integración de los equipos

Cuando un equipo está bien fundamentado está en la capacidad de cumplir una meta en común, compartiendo valores como el trabajo grupal, la solidaridad y a coparticipación entre todas las personas que conforman la empresa. Dentro de la misma se pueden ejecutar jornadas de capacitaciones relacionadas al proyecto que se encuentre en ejecución.

4.2.5 Definición de equipo

La siguiente tabla está definida por (León, 2015)

Tabla 22: Descripción Roles

Cargo	Descripción	Responsable
Product Owner	Encargado del levantamiento de los requerimientos	
Scrum Master	Encargado de la planificación, ejecución y control de proyecto	
Analista de negocio	Encargado de requerimientos de los usuarios y programación	
Programador Senior	Diseña la arquitectura de la aplicación	
Programador Junior	Encargado del desarrollo de la aplicación	
Tester	Encargado de realizar las pruebas de aceptación junto con el cliente	

Fuente:(León, 2015)

4.2.6 Gráfico de Burn Up

Este gráfico está conformado por los dos ejes en el plano cartesiano, donde el eje de las abscisas o eje X corresponde al tiempo en Sprints, en el eje de las ordenadas o eje Y se coloca el esfuerzo estimado para construir las historias de usuarios.

4.2.7 Fase 1. Definición del proyecto

Se realiza una reunión con en Product Owner y el Scrum Master para definir lo que se quiere desarrollar, en este caso se necesita elaborar un sistema para enseñanza de inglés para niños de 6 a 7 años, entonces empieza la actividad del Scrum Master con la toma de nota de los requerimientos del cliente y el Product Owner. Luego esto se le transforma en historias de usuarios.

4.2.8 Fase 2. Conformación del equipo

Para la realización del proyecto se constituyó el siguiente equipo:

Tabla 23: Cargos y Responsabilidades

Cargo	Descripción	Responsable
Product Owner (PO)	Encargado del levantamiento de los requerimientos	Mery E. Mesa A.
Scrum Master (SM)	Encargado de la planificación, ejecución y control de proyecto	Ing. Daisy Imbaquingo
Analista de Negocio (AN)	Encargado de requerimientos de los usuarios y programación	Mery E. Mesa A.
Programador Senior (PS)	Diseña la arquitectura de la aplicación	Mery E. Mesa A.
Programador Junior (PJ)	Encargado del desarrollo de la aplicación	Mery E. Mesa A.
Tester (T)	Encargado de realizar las pruebas de aceptación junto con el cliente	Mery E. Mesa A.
Stakeholders (SH)	Son los altos directivos que no están directamente, asociados en el desarrollo del proyecto	Directivos

Fuente: Propia

Tabla 24: Matriz RACI Asignación de responsabilidades según la matriz RACI

Matriz RACI: Roles / Responsabilidades: R: Responsable, A: Aprobador, C: Consultado, I: Informado. **Elaborado por: Mery Mesa**							
Actividad		**Roles/Responsabilidades**					
ID Actividad	**Actividad**	**PO**	**SM**	**SH**	**AN**	**PS**	**PJ**
1	Product Backlog	A, R	I	C, I	I	I	I
2	Sprint Backlog	C	C, I	C, I	C, I	C, I	C, I
3	Burn down charts	C, I	A, R	C, I	C, I	C, I	C, I
4	Sprint Planning	C, I	A, R	C, I	C, I	C, I	C, I
5	Sprint Review	C, I	A, R	C, I	C, I	C, I	C, I
6	Sprint Retrospective	C, I	A, R	C	C, I	C, I	C, I
7	Daily Scrum Meeting	C, I	A, R	C, I	C, I	I	I
8	Sprint	I	A, R	I	I	I	I
9	Sprint Planning	C, I	A, R	I	I	I	I
10	Analizar y evaluar el Product Backlog	C, I	A, R	I	I	I	I
11	Seleccionar el objetivo del Sprint	C	A, R	I	I	I	I
12	Definir las Historias de Usuario	C	A, R	I	I	I	I
13	Definir las tareas	C	A, R	I	I	I	I
14	Estimar duración de las Tareas	C	A, R	I	I	I	I
15	Investigación	C, I	I	I	I	R	I

16	Desarrollo	C, I	I	I	I	R	I
17	Verificación y Corrección de bugs	C, I	I	I	I	R	I
18	Pruebas Unitarias	C, I	I	I	I	R	I

Fuente: (PMOinformática.com, 2013)

Tabla 25: Descripción Roles y Responsabilidades

Roles y Responsabilidades	
Rol/ Responsabilidad	**Descripción**
R	**Responsable:** Este rol es el que realiza (ejecuta) el trabajo asociado con la actividad, lo habitual es que cada actividad tenga un solo "R", si existe más de uno es recomendable subdividir la actividad.
A	**Aprobador:** Es el encargado de aprobar (firmar), el trabajo realizado, a partir de esa aprobación, este se vuelve responsable por la actividad. Como regla general debe existir un solo "A" por actividad. Este rol es quien asegura que se ejecutan las tareas, por ejemplo, Líderes de área técnica, área de gestión de proyecto, entre otros.
C	**Consultado:** Posee alguna información o capacidad que se necesita para mantener el trabajo. Se le informa y consulta información, de manera bidireccional con el responsable y/o aprobador.
I	**Informado:** Rol que debe ser informado sobre el progreso y los resultados del trabajo. En este caso la comunicación es unidireccional (se le da información, pero no se recibe información).

Fuente: (PMOinformática.com, 2013)

4.2.9 Fase 3 kick Off

En esta fase se procede a presentación del equipo de trabajo, junto con el Cliente y el Product Owner en la cual se procede a la definición de objetivos que tienen que ver con los compromisos que el equipo de desarrollo debe asumir.

El ambiente de trabajo en el cual van a estar sometidos es aquel que genere el Scrum Master donde involucra al Cliente con el Product Owner.

También se realiza la conformación de roles y responsabilidades que los integrantes del equipo de trabajo van a tener.

Se realiza una revisión de los riesgos del proyecto y a forma en la cual se va comunicar los avances del proyecto.

4.2.10 Fase 4 Sprint 0

Se prepara los ambientes de trabajo en el cual se va a dar inicio al proyecto como servidores, equipos locales y la arquitectura del proyecto, se trabaja en el Product Backlog, y poniendo más énfasis en las historias de usuarios.

4.2.11 Fase 5 Inicio de los procesos

El desarrollo del proyecto se da inicio con los siguientes pasos, que se encuentran muy bien detallados.

4.2.11.1 Paso 1 Sprint Planning Iteración o Sprint 1

Se da inicio con la priorización de las historias de usuario, en la cual se describe detalladamente la planificación de cada iteración de la siguiente manera:

Tabla 26: Historia de usuario global del aplicativo

Historia de Usuario	
Iteración: 1	
Número: 1	Nombre Historia de Usuario: Diseño de un aplicativo para la enseñanza de inglés
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Descripción: Como usuario de la aplicación quiero diseñar un aplicativo que ayude en la enseñanza de inglés para los niños de edad comprendida de 7 años. Al ingresar a la aplicación se debe visualizar una pantalla de inicio en la que muestre el Menú en el cual se va a poder acceder a cada uno de las lecciones. Deben constar las siguientes lecciones: Lección de Vocales Lección de Abecedario Lección de Números Lección de Vocabulario Lección de Colores Lección de Animales En las cuales también va a tener incluido la parte de test Test Abecedario Test Números Test Vocabulario Test Colores Test Animales Al realizar cada una de las lecciones debe incluir los contenidos en español e inglés con un audio en donde se escuchará la pronunciación en las lecciones de vocales, abecedario, números, colores y animales. Al final de cada lección se debe realizar el test. Por cada lección una historia de usuario junto con el test una sola historia de usuario.	
En la pantalla de Inicio se podrá visualizar los siguientes Menús • Vocales • Abecedario • Números • Vocabulario • Colores • Animales	

Fuente: Propia

Tabla 27: Historia de usuario lección vocales

Historia de Usuario	
Iteración: 1	
Número: 2	Nombre Historia de Usuario: Diseño de la Lección Vocales.
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada vocales, donde los niños puedan identificar cada una de las vocales con una escritura en español, como también en inglés y la reproducción de un audio donde se hace referencia a la pronunciación de cada una de ellas, para así tener un aprendizaje más fácil y entendible en los niños	
En la pantalla de la lección vocales se podrá visualizar lo siguiente: • Una imagen especificando la vocal • Palabra en inglés referente a la vocal • Palabra en español referente a la vocal • Audio en inglés sobre la pronunciación de cada vocal	

Fuente: Propia

Tabla 28: Historia de usuario lección y test abecedario

Historia de Usuario	
Iteración: 2	
Número: 3	Nombre Historia de Usuario: Diseño de la lección Alfabeto
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada abecedario, donde los niños puedan identificar cada una de las letras del alfabeto, mostrando una imagen, contenido en español e inglés y un audio donde se escuche la pronunciación de cada una de ellas, haciendo un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés debe estar relacionado con el contenido de la lección abecedario, tomando como base los audios, en las cuales se debe realizar una comparación entre audios para poder obtener la respuesta correcta a la pregunta.	
En la pantalla de la lección llamada alfabeto debe contener lo siguiente: • Una imagen acerca de cada una de las letras del abecedario • Palabra en español de cada letra del abecedario • Palabra en inglés de cada letra del abecedario • Un audio sobre la pronunciación de cada letra En la pantalla del test debe contener lo siguiente: • Audios donde se pueda escuchar la pronunciación de las letras • Un cuadro de selección • Un mensaje en pantalla que presente el resultado de correcto o incorrecto	

Fuente: Propia

Tabla 29: Historia de usuario lección y test números

Historia de Usuario	
Iteración: 3	
Número: 4	Nombre Historia de Usuario: Diseño de la lección Números
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada números, en la cual los niños puedan identificar de una manera clara, concisa, mostrando contenido en español e inglés y la reproducción de un audio con la pronunciación de los mismos haciendo un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés está relacionado con el contenido de la lección números, realizando un crucigrama con preguntas sobre los números en inglés, donde como parámetro debe estar establecida una respuesta a una pregunta formulada que debe ir autocompletando el crucigrama de acuerdo a la respuesta.	
En la pantalla de la lección llamada números debe contener lo siguiente: • Una imagen que represente a cada uno de los números • Palabra en español de los números • Palabra en inglés sobre la escritura de los números • Un audio donde se escuche la reproducción de la pronunciación de los números en inglés En la pantalla del test debe contener lo siguiente: • Preguntas en inglés relacionadas a los números • Las respuestas deben ser colocadas en cada línea del crucigrama • Un mensaje que muestre en pantalla el resultado de correcto incorrecto que se visualizará en la pantalla • Debe visualizarse la puntuación una vez finalizado el test	

Fuente: Propia

Tabla 30: Historia de usuario lección y test vocabulario

Historia de Usuario	
Iteración: 4	
Número: 5	Nombre Historia de Usuario: Diseño de la lección Vocabulario
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada vocabulario, en la cual los niños puedan identificar de una manera clara, concisa, mostrando contenido en español e inglés haciendo de esto un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés está relacionado con el contenido de la lección vocabulario, realizando un banco de preguntas en español e inglés fácil de comprender, al nivel básico en el cual se encuentran los niños.	
En la pantalla de la lección llamada vocabulario debe contener lo siguiente: • Palabra en español sobre contenido básico • Palabra en inglés referente al vocabulario En la pantalla del test debe contener lo siguiente: • Preguntas en inglés y español • Respuestas deben ser contestadas en inglés • Un mensaje que se muestre en pantalla de correcto o incorrecto que debe visualizarse en la pantalla • Se debe visualizar la puntuación una vez finalizado el test	

Fuente: Propia

Tabla 31: Historia de usuario lección y test colores

Historia de Usuario	
Iteración: 5	
Número: 6	Nombre Historia de Usuario: Diseño de la lección Colores
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada colores, en la cual los niños puedan identificar de una manera clara, concisa, mostrando contenido en español e inglés con un audio incluido para su respectiva pronunciación haciendo un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés está relacionado con el contenido de la lección colores, se debe realizar un banco de preguntas las mismas que deben contar con una selección de respuestas de acuerdo a la pregunta establecida.	
En la pantalla de la lección llamada colores debe contener lo siguiente: • Imagen con respecto a cada color • Palabra en español de acuerdo al color • Palabra en inglés referente al color • Audio en inglés con la pronunciación de cada color En la pantalla del test debe contener lo siguiente: • Preguntas en inglés y español referente a los colores • Respuestas deben ser tratadas con imágenes de acuerdo a los colores • Un mensaje que debe presentar en pantalla de correcto o incorrecto que debe visualizarse en pantalla • Se visualizará la puntuación una vez finalizado el test	

Fuente: Propia

Tabla 32: Historia de usuario lección y test animales

Historia de Usuario	
Iteración: 6	
Número: 7	Nombre Historia de Usuario: Diseño de la lección Animales
Usuario: Todos	
Prioridad en Negocio: Alta	
Riesgo en Desarrollo: Alta	
Como usuario de la aplicación quiero realizar una lección llamada animales, en la cual los niños puedan identificar de una manera clara, concisa, mostrando contenido en español e inglés, con un audio en el cual se especifique la pronunciación de cada uno de los animales haciendo un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés está relacionado con el contenido de la lección animales, se debe realizar una evaluación donde se maneje imágenes y palabras, las cuales deben ser contestadas mediante la unión de los elementos según lo que corresponda.	
En la pantalla de la lección llamada animales debe contener lo siguiente: • Imagen con respecto a los animales • Palabra en español de acuerdo a cada animal descrito en la imagen • Palabra en inglés referente a los animales • Audio donde se escuche la pronunciación de los nombres de cada uno de los animales. En la pantalla del test debe contener lo siguiente: • Preguntas relacionadas al contenido de la lección • Respuestas con imágenes para poder realizar la unión según corresponda • Un mensaje que debe presentar en pantalla de correcto o incorrecto que se visualizará en pantalla • Se visualizará la puntuación una vez finalizado el test	

Fuente: Propia

Este paso se lo realiza en el Product Backlog con el levantamiento de requerimientos del sistema, en una reunión con el cliente, en la cual se hace hincapié en cada uno de los puntos a realizar durante el tiempo que dure el desarrollo de cada iteración del proyecto.

Las reuniones se las va planificando junto con el Scrum Master con el fin de tener en cuenta las iteraciones que se deben llevarse a cabo y en el menor tiempo posible.

Los puntos que se proporciona a cada historia de usuario, porque es un referente a qué nivel tan importante esta la historia, bien se la puede o no poner.

Product Backlog aplicado en el desarrollo del aplicativo

Desarrollo ágil hibrido: Pila de Producto

Proyecto: Aplicativo para enseñanza de Ingles

Elaborado por: Mery Mesa

Tabla 33: Product Backlog

Identificador ID de la Historia	Enunciado de la Historia	Alias	Estado	Dimensión/ Esfuerzo	Iteración (Sprint)	Prioridad	Comentarios
1	Como usuario de la aplicación tengo la necesidad de contar con un aplicativo que ayude a la mejorar la enseñanza del inglés.	Diseño de un aplicativo para la enseñanza de inglés.	Planificada	50	Sprint 1	1	
2	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada vocales con la finalidad de brindar enseñanza con la utilización de este aplicativo.	Diseño de la lección vocales	Planificada	30	Sprint 1	2	
3	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada alfabeto y a su vez un test con la finalidad de brindar enseñanza, a la vez evaluar lo aprendido en la misma.	Diseño de la lección alfabeto junto con el test de evaluación	Planificada	30	Sprint 1	3	
4	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada números y a su vez un test con la finalidad de brindar enseñanza, al mismo	Diseño de la lección números junto con el test de evaluación	Planificada	30	Sprint 1	4	

	tiempo evaluar lo aprendido en la misma.						
5	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada vocabulario y a su vez un test con la finalidad de brindar enseñanza, al mismo tiempo evaluar lo aprendido en la misma.	Diseño de la lección vocabulario junto con el test de evaluación	Planificada	30	Sprint 1	5	
6	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada colores y a su vez un test con la finalidad de brindar enseñanza, al mismo tiempo evaluar lo aprendido en la misma.	Diseño de la lección colores junto con el test de evaluación	Planificada	30	Sprint 1	6	
7	Como usuario de la aplicación tengo la necesidad de contar con una lección llamada animales y a su vez un test con la finalidad de brindar enseñanza, al mismo tiempo evaluar lo aprendido en la misma.	Diseño de la lección animales junto con el test de evaluación	Planificada	30	Sprint 1	7	

Fuente: Propia

Para lo cual también se debe elaboran las tareas a realizarse, metáforas que ayuden a un mejor entendimiento de lo que se quiere generar en el proyecto. Como se muestra en la siguiente tabla

Tabla 34: Metáfora

Identificador de la Metáfora	
Aplicativo para la enseñanza de Ingles	
Metáfora del sistema	
El aplicativo tendrá una funcionalidad como un libro de inglés para principiantes es decir está orientado exclusivamente para niños, donde se quiere llegar a una comprensión entendible y no confusa ya que el nivel en el que los niños se encuentran es muy básico. A partir de ello se avanzará con cada lección y luego de haber finalizado, se podrá realizar un test de cómo fue su aprendizaje, mismo que mostrará en pantalla un resultado de evaluación.	
Información de aprobación de la Metáfora	
Mery Mesa	MSc. Daisy Imbaquingo
Firma del Entrenador	**Firma del Cliente**

Fuente: Propia

Luego de obtener las metáforas se procedió a la realización de las tarjetas de clase responsabilidad y colaboración que son las clases modelo, vista y controlador. Como se muestra en la siguiente tabla

Tabla 35: Clase - Responsabilidad

Clases	
Clases controladoras	**Clase entidades**
Responsabilidad 1 Mantenimientos de las entidades	**Colaborador 1 Ingreso de los datos**
InglesBean	InglesImpl
InglesBeanAnimales	InglesImplAnimales
InglesBeanColores	InglesImplColores
InglesBeanNumeros	InglesImplNumeros
InglesBeanVocabulario	InglesImplVocales
InglesBeanVocales	VocabularioImpl
Comentarios	
Permite realizar el mantenimiento de las lecciones que se incorpora dependiendo al tipo de clase que se vaya implementando.	

Fuente: Propia

Y las responsabilidades de cada uno frente a cada tarea asignada como se muestra a continuación:

Tabla 36: Clase - Colaborador

Clases	
Test	
Responsabilidad 1 Mantenimiento de los test	**Colaborador 1**
Diseño test Abecedario	Colaborador 2
Diseño test Números	Colaborador 3
Diseño test Vocabulario	Colaborador 4
Diseño test Colores	Colaborador 5
Diseño test Animales	Colaborador 6
Responsabilidad 2 Se encargará de la obtención de la información que debe estar basada en la lección del abecedario Responsabilidad 3 Se encargará de la obtención de la información que debe estar basada en la lección de los números Responsabilidad 2 Se encargará de la obtención de la información que debe estar basada en la lección del vocabulario Responsabilidad 2 Se encargará de la obtención de la información que debe estar basada en la lección de los colores Responsabilidad 2 Se encargará de la obtención de la información que debe estar basada en la lección de los animales	
Comentarios	
Se realiza la creación de los test para evaluar el aprendizaje que fue luego de haber revisado las lecciones.	

Fuente: Propia

A continuación, se estimará los tiempos en horas disponibles por el equipo de desarrollo

Tabla 37: Tiempo disponible por persona

Longitud del Sprint			**4 semanas**
Días laborables durante el Sprint			**20 días**
Miembro del equipo	**Días disponibles durante el Sprint**	**Horas disponibles por día**	**Total de horas disponibles**
Mery Elizabeth Mesa	6	8	48

Fuente: (León, 2015)

Luego de haber establecido los días laborables de cada miembro del equipo junto con el Scrum Master se empieza a delegar responsabilidades, con cada historia de usuario, para dar inicio a los Sprints, después de haber finalizado cada iteración, se debe cambiar de estado "Planificado" a "En Proceso" como se muestra a continuación:

Tabla 38: Cambio de estado Planificado a En Proceso

Identificador (ID) de la historia de usuario	Enunciado de la Historia	Estado	Iteración (Sprint)	Prioridad
1	Como usuario de la aplicación quiero diseñar un aplicativo que ayude en la enseñanza de inglés para los niños de 7 años. Al ingresar a la aplicación se debe visualizar una pantalla de inicio en la que muestre el Menú en el cual se podrá acceder a cada una de las lecciones.	En proceso	Sprint	1
2	Como usuario de la aplicación quiero realizar una lección llamada vocales donde los niños puedan identificar cada una de las vocales con una escritura en español como también en inglés y la reproducción de un audio donde se hace referencia a la pronunciación de cada una de ellas para así tener un aprendizaje más fácil y entendible para los niños	En proceso	Sprint	1
3	Como usuario de la aplicación quiero realizar una lección llamada alfabeto, donde los niños puedan identificar cada una de ellas, mostrando contenido en español e inglés y un audio donde se escuche la pronunciación de las mismas haciendo un aprendizaje mucho más sencillo y fácil de aprender. El diseño del Test en inglés está relacionado con el contenido de la lección alfabeto, haciendo una elección de la respuesta correcta al escuchar los audios propuestos en el test y así evaluar los conocimientos.	En proceso	Sprint	1

Fuente: (León, 2015)

Y así si es como se va cambiado los estados de cada Sprint, para seguidamente proceder con las tareas a realizarse para completar con cada historia de usuario planificada en la Pila de Producto o Product Backlog para cada iteración.

Se procede a tomar referencia de la primera historia de usuario para la definición de las tareas, como se puede apreciar en la siguiente tabla cada tarea se le asigna a un miembro del equipo de acuerdo a sus habilidades, también se estima el tiempo que le llevara a cada uno de ellos en el desarrollo de la iteración, es así como se está cumpliendo el nuevo marco

de trabajo de responsabilidad. A continuación, se detalla las tareas a realizarse durante el Sprint:

Tabla 39: Tareas 1 Sprint 1

Tareas Sprint 1	
Número Tarea: 1	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección vocales	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible en la interfaz de usuario.	

Fuente: Propia

Tabla 40: Tarea 2 Sprint 1

Tareas Sprint 1	
Número Tarea: 2	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección abecedario	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible en la interfaz de usuario	

Fuente: Propia

Tabla 41: Tarea 3 Sprint 1

Tareas Sprint 1	
Número Tarea: 3	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración test abecedario	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible para que el usuario pueda tomar una evaluación acerca de la lección aprendida, haciendo que cada vez que se quiera realizar el test, se haga una selección el audio que corresponda a la pregunta.	

Fuente: Propia

Tabla 42: Tarea 4 Sprint 1

Tareas Sprint 1	
Número Tarea: 4	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección números	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible, con una interfaz amigable.	

Fuente: Propia

Tabla 43: Tarea 5 Sprint 1

Tareas Sprint 1	
Número Tarea: 5	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración test números	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible para que el usuario pueda tomar una evaluación acerca de la lección aprendida, realizando un crucigrama con preguntas relacionadas a los números en Inglés	

Fuente: Propia

Tabla 44: Tarea 6 Sprint 1

Tareas Sprint 1	
Número Tarea: 6	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección vocabulario	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible	

Fuente: Propia

Tabla 45: Tarea 7 Sprint 1

Tareas Sprint 1	
Número Tarea: 7	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración test vocabulario	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible para que el usuario pueda tomar una evaluación acerca de la lección aprendida, realizando preguntas que ayuden a identificar palabras en inglés.	

Fuente: Propia

Tabla 46: Tarea 8 Sprint 1

Tareas Sprint 1	
Número Tarea: 8	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección colores	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible, con una interfaz más amigable.	

Fuente: Propia

Tabla 47: Tarea 9 Sprint 1

Tareas Sprint 1	
Número Tarea: 9	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración test colores	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible para que el usuario pueda tomar una evaluación acerca de la lección aprendida, realizando la utilización de imágenes para el reconocimiento de los colores en inglés.	

Fuente: Propia

Tabla 48: Tarea 10 Sprint 1

Tareas Sprint 1	
Número Tarea: 10	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración lección animales	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible, con una interfaz amigable.	

Fuente: Propia

Tabla 49: Tarea 11 Sprint 1

Tareas Sprint 1	
Número Tarea: 11	Historia 1. Diseño del aplicativo para enseñanza de inglés
Nombre Tarea: Elaboración test animales	
Estado: En Proceso	
Tiempo Estimado:	6 (horas)
Tipo Tarea: Desarrollo	
Fecha Inicio:	Fecha Fin:
Programador Responsable: Mery Mesa	
Descripción: Se genera una estructura internamente que ayude en el proceso para que dicha información sea presentada al usuario final de una manera entendible para que el usuario pueda tomar una evaluación acerca de la lección aprendida, realizando el reconocimiento de los animales en inglés mediante la unión de imágenes con palabras.	

Fuente: Propia

Con esto se está culminando una, parte más del nuevo marco de trabajo Scrum/XP.

4.2.11.2 Paso 2 Sprint

En este paso se procede a la codificación de las tareas elaboradas anteriormente

- Codificación

Las tareas desde la 1 hasta la 11 son codificadas y se las realiza mediante el desarrollo basado en pruebas es decir el TDD (Test Driven Development), que es una forma de construir software.

- Pruebas unitarias

Se dio inicio al proceso con la creación de un Proyecto llamado Aprendiendo inglés el mismo que contiene código del modelo, vista y controlador, en el cual se realizó todo el procedimiento necesario para cumplir con el requisito del desarrollo basado en pruebas.

La programación se la realiza por parejas para hacerla mucho más interactiva, y se describe cada función por quien fue realizada en este caso lo realizó una sola persona, pero ya un ambiente de trabajo en el cual se utilice este tipo de metodología es esencial trabajar en parejas, porque no representa un riesgo para disminuir la calidad del producto.

- Refactorización de código

La refactorización del código inicia luego de haber finalizado el test y que este en perfecto funcionamiento, se elimina o se añade el código que no es útil para la aplicación, en el proyecto se eliminó código y se añadió en la parte de reproducción del audio.

En el diseño del sistema se obtuvo un error al momento de la reproducción del audio en el cual se tuvo que realizar las siguientes modificaciones en la clase InglesBean

Antes se lo tenía de la siguiente manera, código que el momento de la reproducción del audio se actualizaba la página, pero se duplicaban.

```
public List<Ingles> getListaInglesNumeros(){
    listaIngles=inglesImplNumeros.listaNumeros();
    return listaIngles;
}
```

Después se añadió el siguiente código para controlar que el audio sea reproducido sin que se dupliquen los datos.

```
public List<Ingles> getListaInglesNumeros(){
    listaIngles= new ArrayList<Ingles>();
    inglesImplNumeros = new InglesImplNumeros();
    listaIngles=inglesImplNumeros.listaNumeros();
    return listaIngles;
}
```

- Código Estándar

Se realizó la estandarización del código utilizando variables de la siguiente manera:

listaIngles variable que indica el tipo de dato que será devuelto en una lista

Se utilizan variables que sean entendibles en su naturaleza como, por ejemplo:

actionAudioNum() que denota que es una variable de un método donde vamos a extraer el audio para poder imprimirlo.

- Pruebas de Aceptación

Cuando se finaliza con la codificación y funcionalidad del sistema, se inicia con las pruebas de aceptación que se debe realizar con el personal experto en este tema, para esto se procede a revisar los casos de pruebas, en donde se explica el funcionamiento de debe cumplir con cada parámetro establecido en el caso, cuando se procedió a la ejecución de la prueba se obtuvo un error que fue corregido inmediatamente para que el Sprint sea finalizado y aprobado. A continuación, se muestra los casos de pruebas:

Casos de Pruebas

Proyecto Aplicativo para la enseñanza de Ingles

Elaborado Mery Mesa

Ciclo de Pruebas

Tabla 50: Caso de Pruebas

ID	Caso de Prueba	Descripción	Fecha	Área Funcional	Funcionalidad Característica	Resultado Esperado	Requerimientos de Ambiente de Pruebas
1	Diseñar un aplicativo para la enseñanza de inglés	Diseñar una estructura interna del sistema en su totalidad	09/02/2017	Área administrativa	Presentar al usuario todo el aplicativo	Diseñar el aplicativo para enseñanza de inglés con éxito	Ninguno
2	Diseñar lecciones que ayuden al aprendizaje	Diseñar lecciones de fácil entendimiento porque se está trabajando con niños	09/02/2017	Área administrativa	Presentar al usuario cada lección desarrollada con su respectivo contenido	Elaboración de cada una de las lecciones, vocales, alfabeto, números, vocabulario, colores y animales con éxito.	Ninguno
3	Diseño test que evalúen el conocimiento adquirido mediante las lecciones Test alfabeto	Diseña las evaluaciones muy comprensivas para los niños	09/02/2017	Área administrativa	Presentar al usuario contenido de las evaluaciones referente a las lecciones.	Elaboración de los test de evaluación Alfabeto	Ninguno
4	Diseñar test que evalúen el conocimiento adquirido mediante las lecciones	Diseña las evaluaciones muy	09/02/2017	Área administrativa	Presentar al usuario contenido de las evaluaciones referente a las lecciones.	Elaboración de los test de evaluación números	Se necesita como parámetro escribir la respuesta correcta

	Test números	comprensivas para los niños					
5	Diseñar test que evalúen el conocimiento adquirido mediante las lecciones Test Vocabulario	Diseña las evaluaciones muy comprensivas para los niños	09/02/2017	Área administrativa	Presentar al usuario contenido de las evaluaciones referente a las lecciones.	Elaboración de los test de evaluación vocabulario	Se necesita como parámetro escribir y seleccionar la respuesta correcta.
6	Diseñar test que evalúen el conocimiento adquirido mediante las lecciones Test Colores	Diseña las evaluaciones muy comprensivas para los niños	09/02/2017	Área administrativa	Presentar al usuario contenido de las evaluaciones referente a las lecciones.	Elaboración de los test de evaluación Colores	Se necesita como parámetro la identificación de los colores.
7	Diseñar test que evalúen el conocimiento adquirido mediante las lecciones Test Animales	Diseña las evaluaciones muy comprensivas para los niños	09/02/2017	Área administrativa	Presentar al usuario contenido de las evaluaciones referente a las lecciones.	Elaboración de los test de evaluación Animales	Se necesita como parámetro la identificación de los animales aprendidos en la lección.

Fuente: Propia

Continuación de la tabla caso de prueba

Tabla 51: Continuación Caso de Pruebas

		Información del Seguimiento de proyecto			
Procedimientos especiales requeridos	**Dependencias con otros casos de Pruebas**	**Resultado Obtenido**	**Estado**	**Ultima fecha de Estado**	**Observación**
Ninguno	Ninguna	Diseño del Aplicativo exitoso	Abierto	09/02/2017	Añadir videos
Ninguno	Ninguna	Diseño de las lecciones con éxito	Cerrado	01/02/2017	Ninguna
Ninguno	Ninguna	Diseño del test con éxito	Cerrado	02/02/2017	Ninguna
Ninguno	Ninguna	Diseño del test con éxito	Cerrado	03/02/2017	Ninguna
Ninguno	Ninguna	Diseño del test con éxito	Cerrado	09/02/2017	Ninguna
Ninguno	Ninguna	Diseño del test con éxito	Cerrado	06/02/2017	Ninguna
Ninguno	Ninguna	Diseño del test con éxito	Cerrado	07/02/2017	Ninguna

Fuente: Propia

- Pruebas de Integración

En este sistema quedó perfectamente en ejecución, luego de haber finalizado la ejecución unitaria.

Con un ejemplo se explica cómo se debe realizar las pruebas de integración:

Plan de pruebas de Integración

Proyecto: Aplicativo para enseñanza de inglés

Elaborado por: Mery Mesa

Tabla 52: Pruebas de Integración

Observación: de cada caso de prueba se debe registrar lo siguiente - Número del caso de Pruebas, seguido de un número de secuencia que identifique a cada caso de prueba descrito anteriormente. - Componentes que hace referencia al caso de prueba - Una descripción de lo que se va a realizar para cada caso de prueba - Datos de entrada - Datos de salida - Resultados solo se presentarán cuando se hayan ejecutado las pruebas					
Número del Caso de Prueba	**Componente**		**Descripción de lo que se probará**		**Prerrequisitos**
CA01	Elaboración de la aplicación de enseñanza de inglés		Revisión de las lecciones y test		Carga de todos los datos desde las entidades
CA02	Diseño de las lecciones		Revisión de cada una de las lecciones añadidas en el sistema.		Carga de los datos desde las entidades
CA03	Diseño de los test		Revisión y prueba del funcionamiento de cada uno de los test de evaluación implementados		Carga de datos a partir de la lección establecida en el sistema
CPI01					
Paso	**Descripción de pasos a seguir**	**Datos de entrada**	**Datos de salida**	**Resultado**	**Observaciones**
1	Revisar los datos que contiene el aplicativo	Para la parte de los test	El porcentaje que se obtiene después de haber realizado la evaluación	Ok	Ninguna
CPI02					
1	Revisión de cada una de las lecciones	Ninguno	Presentar información de fácil	Ok	Ninguna

			entendimiento al usuario		
CPI03					
1	Generar el test de Abecedario	Ingreso de la pronunciación por teclado	Realiza el check con mensaje de Correcto	Ok	Ninguna
2	Generar el test de Números	Se necesita como dato la respuesta a la pregunta formulada en Ingles	Realiza el check con mensaje de Correcto	Ok	Ninguna
3	Generar el test de Vocabulario	Ingreso de respuestas a partir de preguntas formuladas	Realiza el check con mensaje de Correcto	Ok	Ninguna
4	Generar el test de Colores	Ninguna	Realiza el check con mensaje de Correcto	Ok	Ninguna
5	Generar el test de Animales	Ninguna	Realiza el check con mensaje de Correcto	Ok	Ninguna

Fuente:(León, 2015)

- Propiedad Colectiva

El código que fue implementado en el sistema, es con el objetivo de cumplir con las expectativas del usuario brindándole calidad al producto, cabe mencionar que aquí en este desarrollo el código fuente no pertenece al desarrollador sino al equipo completo y que cada miembro tiene la responsabilidad cumplir con la práctica de integración continua del código.

4.2.11.3 Paso 3 Scrum

Se realiza una pequeña reunión diaria no más de 15 minutos a una hora en la cual el equipo se ponga de acuerdo, luego de que cada miembro del equipo se iguale con sus tareas asignadas, con la finalidad de establecer una planificación para continuar con trabajo en las siguientes horas, las preguntas que se deben cuestionar son las siguientes:

- ¿Qué se hizo ayer?
- ¿Qué se va hacer hoy?
- ¿Qué impedimentos se tiene?

Con estas preguntas se puede a llegar a tener el conocimiento de las actividades que realiza cada miembro del equipo, en el caso de tener inconvenientes se debe dar aviso al Scrum Master para resolverlos en el menor tiempo posible.

4.2.11.4 Paso 4 Sprint Review Meeting (Demo)

En este paso se realiza una reunión para dar por finalizada la iteración, luego de esto se realiza la demostración del sistema en funcionamiento al usuario o a los dichos interesados. Dicha convocatoria se realiza con anticipación, anotando en una agenda de la siguiente manera:

“Estimados, se solicita su presencia para el Demo N°1 del proyecto sistema para enseñanza de inglés, el cual se llevara a cabo el día Jueves 9 de Marzo a las 10:00”

Luego se expone la agenda a tratarse con puntos relacionados a los Sprints, la funcionalidad del sistema y por último preguntas que realizarán los asistentes.

Con el único objetivo de presentar a los Stakeholders y Product Owner la funcionalidad del sistema, con el fin de realizar comentarios, observaciones y críticas por parte de ellos, para que siendo el caso de añadir o eliminar algo se lo realiza inmediatamente.

La funcionalidad del sistema se realizó desde un servidor local en el cual se puede mostrar el contenido y funcionalidad de todo el sistema sin ningún problema.

Es aquí donde se aplican los valores de comunicación y retroalimentación que va a nivel de equipo y Stakeholders, al finalizar con esta reunión se obtiene código fuente listo y probado para ser utilizado, con esto se cumple con la practica en el nuevo marco de trabajo que son las liberaciones cortas de producto que se encuentra funcional y ya utilizable.

4.2.11.5 Paso 5 Retrospectiva

La reunión se la realiza al finalizar la demostración de los incrementos con el objetivo de mejorar de manera continua la productividad del trabajo en equipo, reunión en la cual se trata asuntos relacionados al cumplimiento de los objetivos del Sprint.

Las actividades que menciona (León, 2015) son las siguientes:

Respondiendo las siguientes cuestiones:

- ¿Qué se hizo bien en esta iteración?
- ¿Qué se hizo mal?

- ¿Qué cosas debemos mejorar para la próxima iteración?
- ¿Qué se ha aprendido?

De esta forma se mantiene comunicados a todos los involucrados de cómo está siguiendo el proyecto, el trabajo del Scrum Master también es el indicado para informar de la situación en la que se encuentra el software.

A continuación, se muestra el Resumen de la Retrospectiva contestando las preguntas antes mencionadas:

¿Qué se hizo bien en esta iteración?

- Finalizar con éxito las historias de usuario del sistema
- Funcionalidad del sistema
- Pruebas de funcionalidad

¿Qué se hizo mal?

Se detalla lo que no salió bien durante el tiempo del Sprint, como por ejemplo si existió alguna demora en las tareas asignadas.

¿Qué cosas debemos mejorar para la próxima iteración?

- Actualización de las tareas a diario
- Debe existir una comunicación interna y externa con el cliente
- Afinamientos en tareas de la programación en parejas

¿Qué se ha aprendido?

- Se detalla todas las lecciones que se aprendió durante el desarrollo de los Sprints
- Informe presentado por el Scrum Master
- N° de Sprints terminados: 1 al 11
- N° de Sprints restantes: Ninguno
- Velocidad de desarrollo actual: 80/100
- Fecha de terminación de proyecto: 07/02/2017

4.2.11.6 Paso 6 Planificación de la siguiente iteración

En este paso se realiza la planificación de la siguiente iteración que debe estar en estado planificada en la Pila del Producto o Product Backlog, luego de que se diera por terminada la iteración.

4.2.11.7 Paso 7 Lecciones aprendidas por iteración

Se destaca lo positivo y negativo que se vivió durante la realización del Sprint 1, detallando cada cosa, para que en un futuro se pueda solucionar problemas que tengan las mismas características como se muestra en la **tabla 53**:

Tabla 53: Lecciones aprendidas por Sprint

Proyecto:		Sistema para enseñanza de inglés					
Fecha Inicio:		09/01/2017	**Fecha Fin:**	10/02/2017			
Entidad Ejecutora		UTN					
Líder del Proyecto		Mery E. Mesa A.					
Financiador del Proyecto							
Miembros del Equipo		Mery E. Mesa A.					
Cliente Final		Ing. Daisy Imbaquingo					
N° Referencia	**Tema**	**Descripción**	**Fase del proyecto**	**Categoría**	**Acciones Implementadas**	**Resultados obtenidos**	**Recomendaciones**
1	Levantamiento de requerimientos	Detalle de los requerimientos y usuarios involucrados	Sprint 1	Gestión de requerimientos	Se realizó una reunión con el personal involucrado para explicar el funcionamiento del sistema en si	El levantamiento de requerimientos fue positivo y se obtuvo el bosquejo del sistema	Mantener comunicación constante con el usuario
2	Diseño de las lecciones	Diseño de la estructura interna de cada lección	Sprint 1	Gestión técnica	Se añadió código en la parte del audio para corregir el error	Se corrigió el problema con resultado positivo	Analizar bien lo que realmente se quiere mostrar al usuario

Fuente: Propia

Luego de haber finalizado con la elaboración del sistema para enseñanza de inglés se procede a mostrar lo siguiente:

- Pantalla Inicio

Aquí se muestra los contenidos sobre los Menús que contiene el sistema

Figura 46: Pantalla Inicio
Fuente: Propia

- Pantalla Lección Vocales

Muestra contenido con imágenes sobre las vocales en inglés, español y audio con la pronunciación de cada una.

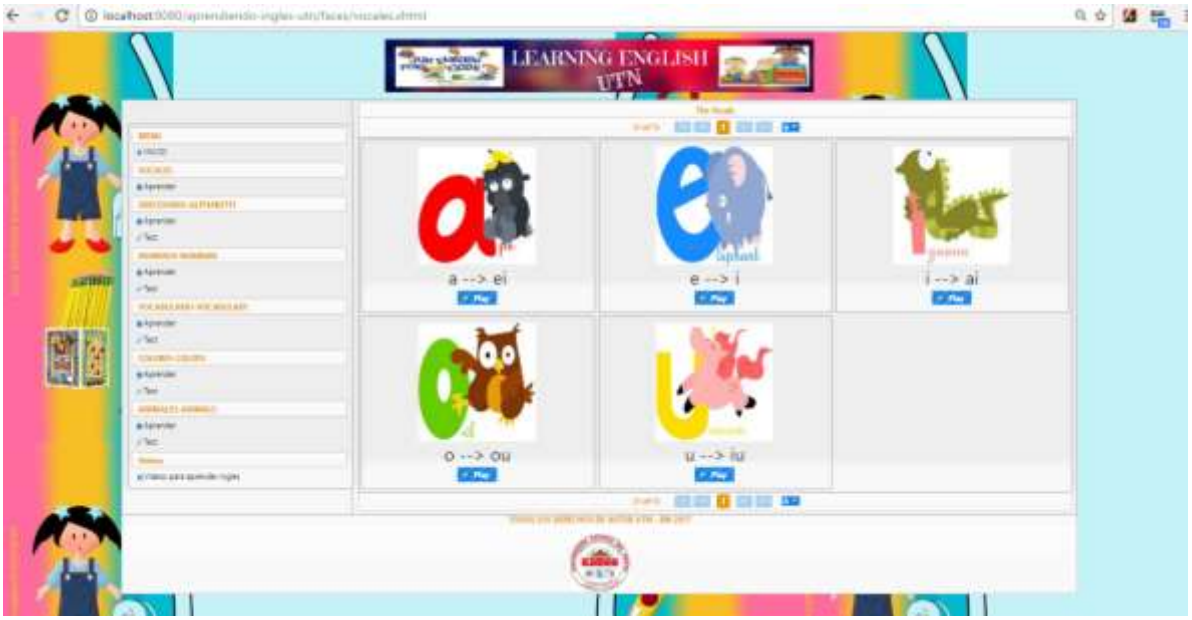


Figura 47: Pantalla Vocales
Fuente: Propia

- Pantalla Lección Abecedario

Muestra contenido con imágenes sobre el abecedario en inglés, español y audio con la pronunciación de cada una.

Figura 48: Pantalla Abecedario
Fuente: Propia

- Pantalla Test Abecedario

Muestra un test de evaluación con audios sobre el contenido del abecedario en inglés, pidiendo que se conteste a la pregunta formulada.

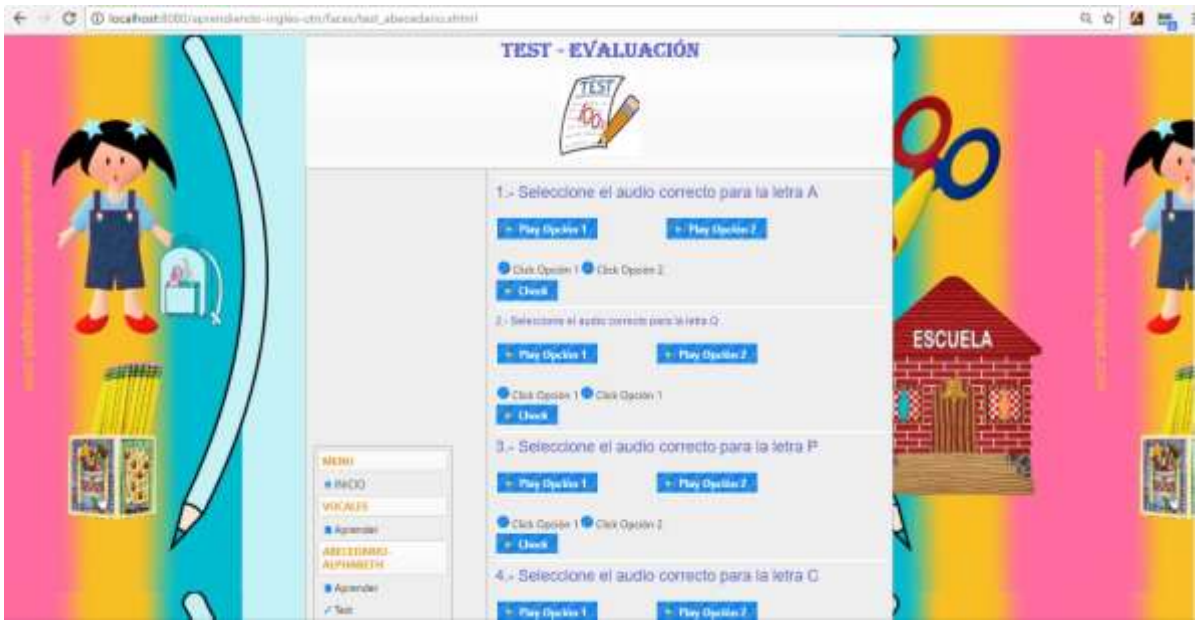


Figura 49: Pantalla Test Abecedario
Fuente: Propia

- Pantalla Lección Números

Muestra contenido con imágenes sobre los números en inglés, español y audio con la pronunciación de cada una.

Figura 50: Pantalla Números

Fuente: Propia

- Pantalla Test Números

Muestra contenido con un crucigrama con preguntas referentes a los números en inglés, que mientras se vaya contestando correctamente se irán ubicando en el lugar indicado.

Figura 51: Pantalla Test Números
Fuente: Propia

- Pantalla Vocabulario

Muestra contenido sobre vocabulario básico que se debe aprender, en inglés y español.

Figura 52: Pantalla Vocabulario
Fuente: Propia

- Pantalla Test Vocabulario

Muestra un test con preguntas variadas acerca de lo que se aprendió en la lección de vocabulario.

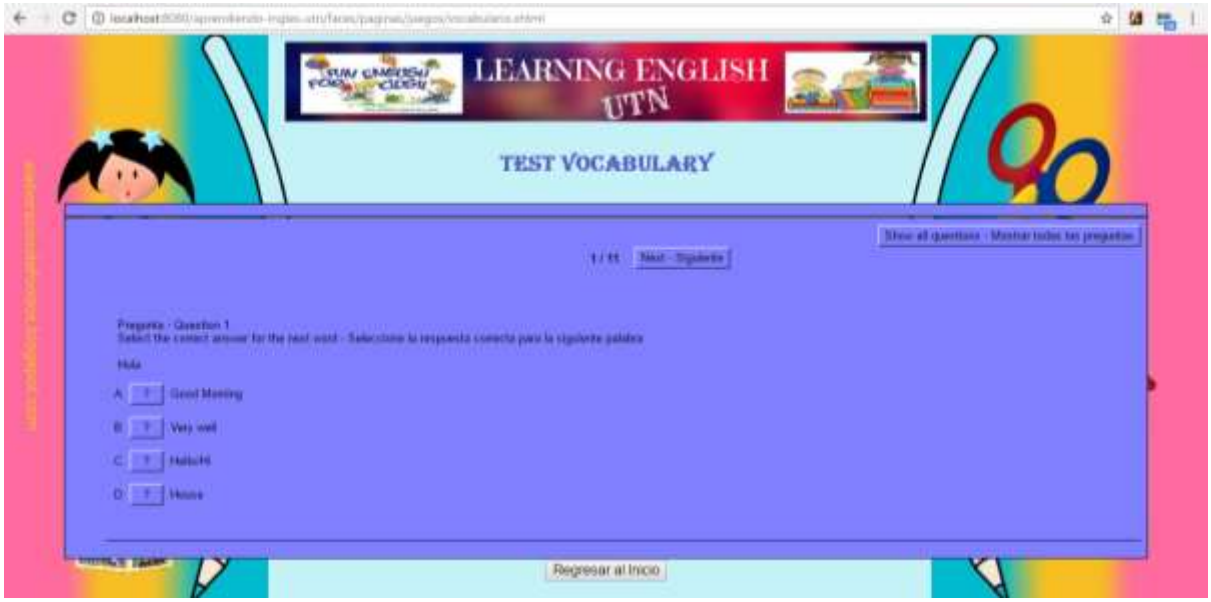


Figura 53: Pantalla Test Vocabulario
Fuente: Propia

- Pantalla Colores

Muestra contenido con imágenes sobre los colores en español, inglés y un audio que se lo escuchará con la pronunciación de cada uno de ellos.

Figura 54: Pantalla Colores

Fuente: Propia

- Pantalla Test Colores

Muestra un test con preguntas variadas, con el fin de hacer el reconocimiento de cada uno de los colores acerca de lo que se aprendió en la lección.

Figura 55: Pantalla Test Colores
Fuente: Propia

- Pantalla Animales

Muestra contenido con imágenes sobre los animales en español, inglés y un audio que se lo escuchará con la pronunciación de cada uno de ellos.

Figura 56: Pantalla Animales
Fuente: Propia

- Pantalla Test Animales

Muestra un test, donde se debe mover elementos para unirlos con otros referentes a lo que aprendió en la lección.

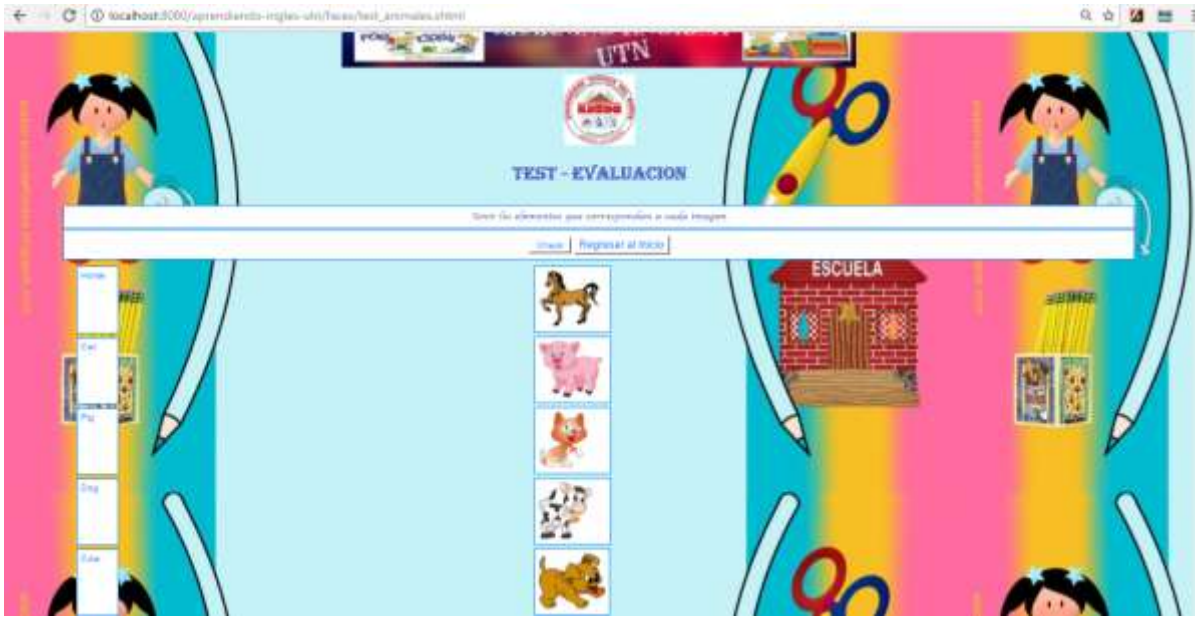


Figura 57: Pantalla Test Animales
Fuente: Propia

- Pantalla Videos

Muestra contenido de videos sobre inglés básico para niños.

Figura 58: Pantalla Videos
Fuente: Propia

4.3 Resultados de trabajo con el nuevo marco de trabajo "Scrum y XP"

- Un marco de trabajo muy fácil de aprender ordenado, partiendo de los principios que tienen tanto Scrum como XP, uno de los elementos principales de este marco de trabajo que propone XP, es el trabajo en parejas, con el único fin de establecer un ambiente de interactividad, comunicación entre los dos miembros del equipo, con esto se plasma con lo propuesto, cumpliendo con los valores que se establecen en Scrum/XP.

- El levantamiento de los requerimientos se lo realizó en base de historias de usuarios, es una manera de interactuar con el usuario mediante reuniones con los miembros del equipo Product Owner y Scrum Master, quienes son los encargados de tomar las decisiones pertinentes para transmitir al equipo de desarrollo de cualquier acontecimiento importante que el cliente haya sugerido.

- Scrum define su trabajo por medio de iteraciones, que luego de ser analizados en las historias de usuario mencionadas anteriormente, se procede al cumplimiento con cada tarea especificada en el Product Backlog, que una vez que está en un estado de planificación, al aplicar al marco de trabajo, pasa a un estado en Proceso delegando responsabilidades a cada miembro del equipo que se encargarán de cada una de las tareas asignadas.

- Las políticas que se establecen al inicio del diseño del producto, deben estar definidas por empresa y son las que deben ayudar a mantener la calidad en el desarrollo del producto, además permite a la empresa mantenerse en un buen nivel.

- La delegación de roles y responsabilidades, ayudan a mantener una buena comunicación con cada miembro del equipo, haciendo de esto que cada uno de ellos se vuelvan multidisciplinarios, auto organizados, con el fin de resolver problemas presentados en los Sprints.

- El lanzamiento de cada iteración con funcionalidad, la revisión y pruebas que son realizadas con el usuario, permiten que al cerrar cada iteración se lo haga ya con la satisfacción de los clientes finales.

- Cuando se realizan reuniones a diario con los miembros del equipo de trabajo y por ende los usuarios, se puede mantener una comunicación más cercana, para poder resolver cualquier inconveniente que surja en la realización de tareas, llegando a tener una visión más amplia de lo que se está desarrollando.

- El uso de las plantillas de lecciones aprendidas, en las cuales se registran los problemas y soluciones luego de haber finalizado los Sprints, que deben ser bien documentadas, son una ayuda para que otros equipos puedan solucionar problemas similares a los descritos en esta plantilla.

4.4 Las ventajas de utilizar Scrum/XP

La utilización de este marco de trabajo dio como resultado positivo lo siguiente:

- La constante comunicación que existe de parte del usuario con los miembros de equipo para resolver cualquier problema.

- La entrega de resultados de código ya en funcionamiento es una parte importante para cumplir con las exigencias del cliente.

- La utilización de Scrum hace que el trabajo sea más organizado y aumente productividad y calidad en el desarrollo del producto.

- Las prácticas de XP ofrecen un diseño de software de calidad, mediante la aplicación de pruebas que se realiza al finalizar la iteración, es así como se puede obtener un producto de calidad y probado por el cliente.

- La utilización de tarjetas CRC, metáforas y tareas hace que se tenga una visión general de lo que se quiere construir, según lo que se realizó en el levantamiento de los requerimientos.

- El manejo de código estándar es una forma de conducir el código fuente muy bien definido y entendible.

- La práctica de trabajo en parejas es muy eficiente, fomenta la comunicación entre los dos miembros del equipo y esto hace que sea un trabajo fructífero.

Análisis de Impactos

El análisis de impactos presentados acerca del tema, determina los ámbitos social, económico y ambiental en los cuales debe ser evaluado los positivo y negativo del presente proyecto educativo tecnológico.

Rango de niveles de impactos positivos y negativos:

Tabla 54: Rango de niveles de impactos

Valor	**Impacto**
0	No hay impacto
1	Impacto bajo
2	Impacto medio
3	Impacto alto

Fuente: Propia

Se determina una matriz en la cual se especifican los valore en los cuales serán evaluados los impactos, a cada indicador se le asigna un valor de impacto, y luego de haber finalizado la evaluación se procede a la suma de los valores y se divide por el número de indicadores y finalmente se obtiene el nivel de impacto en los diferentes ámbitos.

5.1 Impacto Socio Cultural

Tabla 55: Impacto socio cultural

Nivel de Impacto Indicador	0	1	2	3	Total
Transferencia de conocimiento			x		2
Formación en el área para el desarrollo			x		2
Comunicación				x	3
Total			4	3	7

Fuente: Propia

Total de impacto social = 7/3
Total de impacto social = 2.33
Nivel de impacto social = Impacto Medio

La evaluación del impacto socio cultural dio como resultado un impacto medio esto quiere decir que en un corto, mediano y largo plazo la transferencia del conocimiento sobre las metodologías híbridas beneficiará a la sociedad de una manera más ágil y con una mejor comunicación.

5.2 Impactos Económicos

Tabla 56: Impactos económicos

Nivel de Impacto / Indicador	0	1	2	3	Total
Incremento de productividad				x	3
Talento humano				x	3
Generación de nuevas estrategias metodológicas			x		2
Total			**2**	**6**	**8**

Fuente: Propia

Total de impacto económico = 8/3
Total de impacto económico = 2.66
Total de impacto económico = Impacto Medio

La avaluación del impacto económico refleja que la utilización de este tipo de metodologías beneficiará a las entidades en su incremento en la productividad, la obtención de talento humano que abarque en las diferentes áreas de desarrollo de software.

5.3 Impacto Ambientales

Tabla 57: Impacto ambientales

Nivel de Impacto / Indicador	0	1	2	3	Total
Reducción de la utilización de hojas de papel				x	3
Total				**3**	**3**

Fuente: Propia

Total de impacto económico = 3/1
Total de impacto económico = 3
Total de impacto económico = Impacto Alto

La evaluación del impacto ambiental refleja que con el sistema para enseñanza de inglés se disminuye la utilización de hojas de papel, un factor importante que contribuye al cuidado del medio ambiente.

Conclusiones

- La falta de información de las metodologías híbridas hizo compleja la investigación, ya que no existe suficiente documentación que ayude a profundizar en contenidos extensos referente a este tema.

- De acuerdo a las estadísticas realizadas en estos últimos años, las metodologías híbridas se están posicionando de a poco en el mercado de desarrollo de software, debido a su rápida adaptación a los cambios y así obtener marcos de trabajo mucho más rápidos y eficientes, cumpliendo con las exigencias del cliente.

- A través del estudio de las dos metodologías de desarrollo de software híbridas, se concluye que, EssUP es una metodología que sigue los lineamientos de la metodología tradicional RUP; es decir, que todavía tiene como punto principal la documentación de los artefactos y Scrum/XP sigue los lineamientos de la metodología Scrum, en la cual la documentación es un punto no tan importante, pero, ayuda a solucionar problemas mediante reuniones diarias con el equipo de trabajo.

- La aplicación de la metodología permitió desarrollar el sistema para enseñanza de inglés para niños, siguiendo el marco de trabajo propuesto por la metodología Scrum/XP, haciendo que el desarrollo sea más eficiente y documenta solo lo necesario.

Recomendaciones

- Se recomienda en especial a la biblioteca de la Universidad Técnica del Norte que adquiera documentación acerca de las metodologías híbridas, para que los estudiantes que necesiten profundizar en conocimientos sobre este tema, tengan acceso a información verídica.

- Se recomienda que en la Carrera de Ingeniería en Sistemas Computacionales de la Universidad Técnica del Norte, se inserte como un tema en el pensum de estudio a las metodologías híbridas, para que así los estudiantes adquieran conocimientos desde los primeros niveles y profundicen en la investigación sobre este tipo de temas.

- Se recomienda la utilización de la metodología de desarrollo de software Scrum/XP en el diseño de productos de software, ya que la mezcla de las prácticas de estas dos metodologías que forman una híbrida constituyen un marco de trabajo eficaz y rápido con valores que mantiene al equipo de desarrollo en un buen ambiente de trabajo.

- Se recomienda utilizar la Norma ISO/IEC 12207 cuando se necesite evaluar y cotejar trabajos comparativos en área de desarrollo de software, debido a que contienen parámetros concretos del ciclo de vida del software.

- Se recomienda la utilización de la metodología Scrum/XP para el desarrollo de software, una buena opción, porque la mejor comunicación que tiene el equipo es mediante el tipo de reuniones que propone esta metodología, en la cual se resuelven problemas que van surgiendo durante la realización de tareas asignadas por cada miembro del equipo.

Referencias bibliográficas

1. Alex, T. (2013). Presentacion ASD GSI. Retrieved from http://es.slideshare.net/alextorres50999/presentacion-asd-gsi

2. Apr.com. (2015). Modelos de desarrollo. Retrieved from http://www.aprenderaprogramar.com/foros/index.php?topic=2144.0

3. Barco-Camaronero. (2009). Modelo-Rup. Retrieved from http://bbarcocamaronero.blogspot.com/2009/12/modelo-rup.html

4. Benítez, E. (2014). *IMPLEMENTACION DELSERVICIO WEB CALIFICACIÓN DE JUGADORES EN LINEA PARA LA ASOCIACIÓN DE FÚTBOL AMATEUR DE PICHINCHA (AFAP) UTILIZANDO LA METODLOGÍA ICONIX*. Escuela Politécnica Nacional. Retrieved from http://bibdigital.epn.edu.ec/bitstream/15000/7283/1/CD-5411.pdf

5. Campos, T. (2016, March). Benchmarking. *Caracas Marzo Del 2016*, p. 22. Universidad Central de Venezuela. Retrieved from https://catedraalimentacioninstitucional2.files.wordpress.com/2015/03/benchmarking-2016-w.pdf

6. Chicaiza, A. (2007). *DESARROLLO DE SOFTWARE DE NOMINA DE EMPLEADOS UTILIZANDO LA METODOLOGIA CRYSTAL*. Escuela Politécnica del Ejercito.

7. Córazon, A., García, E., Luna, L., Padilla, G., & Abigail, R. (2011). Informática V - ICONIX. Retrieved from http://informatica-v-iconix.blogspot.com/2011/08/normal-0-21-false-false-false-es-x-none.html

8. Espinosa, I. (2015). Proceso para el desarrollo de software. Retrieved from http://es.slideshare.net/000sam000/proceso-para-el-desarrollo-de-software-ponencia-mcivet-espinosa-conde

9. Gallegos, A., & Ortiz, P. (2011). *Elaboración del Estandar de Aplicaciones de la Norma ISO/IEC 12207, Al Desarrollo de Aplicaciones de Software para la UTIC de la ESPE*. Escuela Politécnica del Ejército.

10. Grupo-Espiral-Php. (2009). Modelo Espiral. Retrieved from

http://modeloespiral.blogspot.com/

11. Henrik, K. (2015). *SCRUM AND XP FROM THE TRENCHES*. (C. Ana, Ed.) (2da Edició). United States: InfoQ.com. Retrieved from http://www.infoq.com/resource/minibooks/scrum-xp-from-the-trenches-2/en/pdf/Scrum-and-XP-from-the-Trenches-2nd-edition.pdf

12. Huancho Arroyo, V. (2011). ISO 12207 Ciclo de vida del software. Retrieved from http://unfviso12207.webcindario.com/

13. Humbert, R. (2014). METODOLOGÍAS ÁGILES EN TI. Retrieved from http://es.slideshare.net/huraja/metodologas-giles-en-ti

14. IDS. (2015). Métodos de Desarrollo de Sistemas Dinámicos(DSDM). Retrieved from https://ingenieriadelsoftwareuah2015.wordpress.com/2015/03/29/metodos-de-desarrollo-de-sistemas-dinamicos-dsdm/

15. ISO/IEC. (2008). *System and software engineering - Software life cycle processes*. Retrieved from https://webstore.iec.ch/preview/info_isoiec12207%7Bed2.0%7Den.pdf

16. Jacobson, I. (2016a). *Architecture Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/architecture_essentials.pdf

17. Jacobson, I. (2016b). *Component Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/component_essentials.pdf

18. Jacobson, I. (2016c). EssUP. Retrieved from http://classic-web.archive.org/web/20080331082508/http://www.ivarjacobson.com/products/essup.cfm

19. Jacobson, I. (2016d). *Iterative Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/iterative_essentials.pdf

20. Jacobson, I. (2016e). *Product Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/product_essentials.pdf

21. Jacobson, I. (2016f). *Team Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/team_essentials.pdf

22. Jacobson, I. (2016g). *Test Execution Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/test_execution_essentials.pdf

23. Jacobson, I. (2016h). *Unified Process Lifecycle Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/unified_process_essentials.pdf

24. Jacobson, I. (2016i). *Use - Case 2.0 Essentials*. Retrieved from https://www.ivarjacobson.com/sites/default/files/field_iji_file/article/use-case2.0_essentials.pdf

25. José, P. (2011). *DISEÑO DE UN MODELO PARA EVALUACION/PRUEBAS DEL SOFTWARE EN BASE A INGENIERIA DE PRUEBAS APLICANDO EL ESTANDAR ISO/IEC 29119 EN LA EMPRESA OMNISOFT DE LA CIUDAD DE QUITO*. ESCUELA POLITÉCNICA DEL EJÉRCITO. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/6385/1/T-ESPEL-CDT-1008.pdf

26. León, G. (2015). *Marco de Trabajo Ágil de Desarrollo de Software Combinando Scrum con XP. Aplicación a un Caso de Estudio*. Universidad de las Américas. Retrieved from http://dspace.udla.edu.ec/jspui/bitstream/33000/4364/1/UDLA-EC-TMGSTI-2015-24.pdf

27. Malave, N. (2007). Escala tipo Likert. Retrieved from http://webcache.googleusercontent.com/search?q=cache:http://190.202.16.27/documentos/F%25C3%25ADsico%2520de%2520Escala%2520Likert.pdf

28. Microsoft. (2016). Revisiones de código y estándares de codificación. Retrieved from https://msdn.microsoft.com/es-es/library/aa291591(v=vs.71).aspx

29. Morales de la Torre, J. (2013). *APLICACIÓN DISTRIBUIDA WEB-MÓVIL ADMINISTRABLE PARA LA GESTIÓN Y DIFUSIÓN GEO-LOCALIZADA DE ATRACTIVOS TURÍSTICOS Y HOTELES PARA LA CIUDAD DE IBARRA, CON TECNOLOGÍA GIS Y SOFTWARE LIBRE*. UNIVERSIDAD TÉCNICA DEL NORTE. Retrieved from http://repositorio.utn.edu.ec/bitstream/123456789/2615/1/04 ISC 283

TESIS .pdf

30. Páez, J. (2011). *DISEÑO DE UN MODELO PARA EVALUACION/PRUEBAS DEL SOFTWARE EN BASE A INGENIERIA DE PRUEBAS APLICANDO EL ESTANDAR ISO/IEC 29119 EN LA EMPRESA OMNISOFT DE LA CIUDAD DE QUITO*. Escuela Politecnica del Ejercito Ecuador.

31. Párraga, J. (2014). Gestión de proyectos de Software, metodología de desarrollo ágil: Scrum. Retrieved from http://www.vbote.com/vbote-solutions-academy-blog/86-gestion-de-proyectos-metodologia-de-desarrollo-agil-scrum.html

32. PMOinformática.com. (2013). Planilla de la matriz RACI. Retrieved from http://www.pmoinformatica.com/2013/07/plantilla-matriz-raci-asignacion.html

33. Ponce, S. (2011). *SISTEMA WEB PARA EL DEPARTAMENTO DE ASESORÍA JURÍDICA DE LA DIRECCIÓN PROVINCIAL DE EDUCACIÓN DE IMBABURA, MEDIANTE LA UTILIZACIÓN DEL FRAMEWORK SYMFONY.* Universidad Técnica del Norte.

34. Powered, H. (2016). Crystal. Retrieved from http://metodosdesarrolloagil.wikispaces.com/-+Crystal

35. Pressman, R. (2010). *Ingeniería del software*.

36. Toapanta, K. (2012). *MÉTODO ÁGIL SCRUM, APLICADO A LA IMPLANTACIÓN DE UN SISTEMA INFORMÁTICO PARA EL PROCESO DE RECOLECCIÓN MASIVA DE INFORMACIÓN CON TECNOLOGÍA MÓVIL*. ESCUELA POLITÉCNICA DEL EJÉRCITO. Retrieved from http://repositorio.espe.edu.ec/bitstream/21000/5893/1/T-ESPE-034427.pdf

37. Toborda, L., & Gonzales, J. (2015). Modelo en Espiral. Retrieved from https://www.emaze.com/@ALQLQCTT/MODELO-EN-ESPIRAL

38. Ulyla. (2010). Cliente/Servidor. Retrieved from https://ulyla003.files.wordpress.com/2010/02/imagen3.png

39. Urs.cod. (2007). *Ciclo de vida del Software*. Retrieved from https://procesosdesoftware.wikispaces.com/file/view/ciclosdevidadelsoftware.pdf/579330701/ciclosdevidadelsoftware.pdf

40. Vergara, M. (2014). *Metodologías Actuales*.

41. VERSIONONE. (2016). The 10 anual state of agile report. Retrieved from https://versionone.com/pdf/VersionOne-10th-Annual-State-of-Agile-Report.pdf

GLOSARIO

A

Agile. Grupo de metodologías que definen un marco de trabajo para el desarrollo de software

B

Benchmarking. Proceso que se utiliza para evaluar productos, que consiste en tomar comparadores con diferentes aspectos y así tomar las mejores prácticas.

Build. Significa construcción o elaboración, en este caso se hablaría de la construcción de software

C

Crystal. Metodología para el desarrollo de software

Commit. Finaliza un proceso que haya estado en curso.

Checkout. Método que hace un pare si necesita modificar alguna línea de código

D

DSDM. Dynamic Systems Development Method, metodología que brinda un framework de desarrollo ágil.

E

EssUP. Essentials Unified Process, metodología híbrida para el desarrollo de software

F

FDD. Feature Driven Development, metodología de desarrollo ágil que está basada en la calidad del software, incluyendo un monitoreo a menudo del proyecto.

H

Hito. Es una tarea de duración cero que consisten en conocer los avances del proyecto, que simboliza un logro.

I

Iconix. Metodología de desarrollo tradicional, que tiene el control muy estricto de ciclo de vida del software.

Iteración. Consiste en repetir un proceso, con el objetivo de cumplir con una meta planteada.

M

Metodología. Conjunto de procedimientos a seguir para cumplir con el objetivo

MSF. Microsoft Solutions Frameworks, metodología que permite entregar soluciones tecnológicas de forma rápida.

Metáfora del sistema. Es un resumen de todo lo que debe contener un producto de software.

N

Norma ISO/IEC 12207. Es un estándar para los procesos del ciclo de vida del software.

P

Product Owner. Es la persona encargada de transmitir al equipo de trabajo una visión del proyecto que se necesita elaborar.

R

RUP. Rational Unified Process, metodología tradicional más utilizada en el desarrollo de software caracterizado por su robusta manera de trabajar.

Retrospectiva. Reunión que se hace con el equipo de trabajo con el objetivo de mejorar la calidad y productividad del proyecto en desarrollo.

Refactoring. Acción de agregar o quitar código fuente en el desarrollo de software, con el fin de hacerlo más entendible.

S

Software. Conjunto de programas, que permiten el funcionamiento de diferentes tareas en un sistema.

Scrum. Marcos de desarrollo ágil, en la cual se aplican buenas prácticas para trabajar en equipo

Sprint. Es la accione de ejecutar una iteración que ya estuvo planificada anteriormente.

Stakeholders. Son personas que no forman parte directa en el diseño de software pero si deben ser tomados en cuenta.

T

Tarjetas CRC. Son las tarjetas Clase Responsable y Colaborador, que permite tener una estructura interna del sistema en desarrollo.

U

UML. Lenguaje de modelado que permite crear diferentes diagramas en el cual se puede identificar usuarios y clases del sistema.

V

VersionOne. Empresa encuestadora que define la posición de las metodologías en el mercado de desarrollo de software.

W

Win – Win. Metodología de desarrollo de software que está relacionada con el método en espiral, que define un conjunto de procesos a seguir para el diseño del proyecto.

X

XP. eXtream Programming metodología que se caracteriza por trabajar en parejas aplicando prácticas y reglas para el desarrollo.