UNIVERSIDAD AMAZÓNICA DE PANDO

VICERRECTORADO

CARRERA DE INGENIERÍA INFORMÁTICA - PET

LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE EN LAS ASIGNATURAS DE INGENIERÍA DE SOFTWARE DE LA CARRERA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD AMAZÓNICA DE PANDO

TESIS DE GRADO PARA OPTAR AL GRADO ACADÉMICO DE LICENCIATURA EN INGENIERÍA INFORMÁTICA.

Autor: Univ. Silvestre Antonio Bandeira Confessori

Tutor: MSc. Ing. Samuel Fuentes Chambi

Cobija – Pando – Bolivia

2022

UNIVERSIDAD AMAZÓNICA DE PANDO

VICERRECTORADO

CARRERA DE INGENIERÍA INFORMÁTICA - PET

METODOLOGÍAS DE DESARROLLO DE SOFTWARE EN LAS ASIGNATURAS DE INGENIERÍA DE SOFTWARE DE LA CARRERA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD AMAZÓNICA DE PANDO

Tesis de grado sometida a consideraciones de la Universidad Amazónica de Pando Programa Especial de Titulación.

Requisito para optar al grado de:

Licenciado en Ingeniería Informática

Por:

Silvestre Antonio Bandeira Confessori

Cobija – Pando – Bolivia

2022

Esta Tesis de Grado ha sido aceptado, en su presente forma, por la Universidad Amazónica de Pando, el Programa Especial de Titulación y aprobada por el Tribunal.

FIRMANTES:

_____
Lic. Aldo Orellana Lima
RESPONSABLE DEL PROGRAMA ESPECIAL DE TITULACIÓN

_____
M.Sc. Ing. Christian Miahuchi Nataly
TRIBUNAL

_____
Ing. María Aida Mireya Monje Ascarrunz
TRIBUNAL

_____
Ing. MSc. Samuel Fuentes Chambi
TUTOR

_____
Univ. Silvestre Antonio Bandeira Confessori
POSTULAN

DEDICATORIA

Dedicado a mi hija Mhía Nicole, mi inspiración de cada día para seguir adelante.

No pares de seguir tus sueños. Dentro de ti esta todo lo que necesitas para hacerlos realidad; confía siempre en ti misma.

Que nuestro Padre Celestial bendiga y guie cada paso de tu vida.

AGRADECIMIENTOS

A Dios por su infinito amor para conmigo, por ser la luz que guío mí camino para poder alcanzar este objetivo.

A mis padres por su sacrificio y sobre todo por su amor incondicional.

A mi tutor Msc. Ing. Samuel Fuentes Chambi por su guía, paciencia y apoyo durante el proceso de elaboración de este trabajo.

A mí mismo por seguir adelante sin rendirme hasta conseguir este objetivo.

RESUMEN EJECUTIVO

La problemática abordada en la presente investigación fue la baja articulación de las metodologías de desarrollo de software en las asignaturas de Ingeniería de software en la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, donde se adoptó un enfoque cuantitativo de carácter descriptivo considerando como unidades de análisis a docentes, estudiantes y documentación académica de la institución.

Que a partir de una revisión y análisis teórico y la fundamentación empírica se responde a este vacío de conocimiento con una propuesta de estrategia metodológica para la articulación de las metodologías de desarrollo de software en las asignaturas de ingeniería de software de la carrea de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

ABSTRACT

The problem addressed in the present research was the low articulation of software development methodologies in the subjects of Software Engineering in the Systems Engineering career of the Amazon University of Pando, where a quantitative approach of a descriptive nature was adopted, considering teachers, students and academic documentation of the institution as units of analysis.

That from a review and theoretical analysis and empirical foundation, this knowledge gap is responded to with a proposal for a methodological strategy for the articulation of software development methodologies in the software engineering subjects of the Systems Engineering career of the Amazon University of Pando.

ÍNDICE

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

Esta investigación está estructurada por capítulos. En el primer capítulo se plantea el diseño teórico y metodológico de la investigación que cubre la problemática, los objetivos, alcances y justificación de la Investigación.

En el segundo capítulo, se plantea el Diseño Metodológico de la Investigación, que contempla el enfoque, el tipo de investigación, la población – muestra y las técnicas e instrumentos de investigación.

En el tercer capítulo, se presenta todo el análisis de los referentes teóricos con relación al objeto de estudio "Las metodologías de Desarrollo de Software", categorizadas en metodologías convencionales, metodologías formales, metodologías ágiles y metodologías emergentes.

En el cuarto capítulo, se presenta el desarrollo del trabajo de campo, los resultados de las encuestas realizadas a docentes y estudiantes de la carrera de ingeniería de sistemas y también la información obtenida de la revisión documental de los proyectos formativos e informes. Termina este capítulo con la presentación de una propuesta metodológica de articulación de estrategias de desarrollo de software para las asignaturas de la Ingeniería de Software de la Carrera de Ingeniería de Sistemas en la Universidad Amazónica de Pando.

La investigación termina con el capítulo cinco de conclusiones y recomendaciones.

CAPÍTULO I

JUSTIFICACIÓN, PROBLEMA Y OBJETIVOS

1.1 Antecedentes

En la actualidad la rapidez y el dinamismo en la industria del software han hecho replantear los cimientos sobre los que se sustenta el desarrollo de software tradicional. Estudios recientes y el mismo mercado actual está marcando la tendencia en la ingeniería del software teniendo como características principales atender a las necesidades de rapidez, flexibilidad y variantes externas que hacen de nuestro entorno una ventaja más competitiva al aumentar la productividad y satisfacer las necesidades del cliente en el menor tiempo posible para proporcionar mayor valor al negocio.

Ante esta situación, el grado de adaptación de las metodologías tradicionales a estos entornos de trabajo no son del todo eficientes y no cubren las necesidades del mercado actual.

En la actualidad existen una gran cantidad de metodologías para el desarrollo de software, separadas en dos grandes grupos; las metodologías tradicionales o pesadas y las metodologías ágiles.

Las metodologías tradicionales se basan en las buenas prácticas dentro de la ingeniería del software, siguiendo un marco de disciplina estricto y un riguroso proceso de aplicación. Las metodologías ágiles, en cambio, representan una solución a los problemas que requieren una respuesta rápida en un ambiente flexible y con cambios constantes, haciendo caso omiso de la documentación rigurosa y los métodos formales.

La articulación de las metodologías en la enseñanza de las asignaturas de la Ingeniería de Software en carreras de educación superior no es una tarea fácil, esto se debe a que los niveles de pensamiento complejo son diferentes en cada nivel o semestre de la carrera; de ahí la importancia de encarar la metodología de desarrollo de software como objeto de estudio en las

asignaturas de Ingeniería de software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

1.2 Planteamiento del Problema

De acuerdo al plan de estudios 2018 de la carrera de Ingeniería de Sistema, la familia laboral de Sistemas/Tecnologías de Información y comunicación se encuentra la disciplina de la Ingeniería de Software, la misma que agrupa un conjunto de asignaturas con características similares, en la que deben, los estudiantes desarrollar habilidades y destrezas de análisis, diseño e implementación de sistema de software.

Por lo general el docente, al impartir las asignaturas de la disciplina de Ingeniería de software, realiza un esfuerzo por incluir una metodología para el desarrollo de los proyectos o sistemas de software, en su mayoría están basadas en el modelo de ciclo de vida en cascada que sigue una secuencia lineal de análisis, diseño, codificación, pruebas, despliegue y mantenimiento.

Al transitar de asignatura a asignatura en los diferentes niveles o semestres, los métodos y herramientas para el desarrollo de los proyectos de software aumentan en cantidad y complejidad. Esto ocasiona que la complejidad de articular una metodología de desarrollo lineal no se pueda alcanzar las competencias de aprendizaje formulados en el proyecto formativo del docente.

De acuerdo a lo descrito en párrafos anteriores, el problema principal identificado es la ausencia de metodologías de desarrollo de proyectos de software aplicables en cada nivel o ciclo formativo en el plan de estudios de la carrera de Ingeniería de Sistemas.

La Ingeniería de Software es una disciplina que se encuentra en constante cambio y permanente investigación, son tantos y variados los requerimientos por parte de los clientes por un producto acabado y con las mejores prestaciones de servicios para las demandas de la personas o instituciones. Sin embargo, la industria del desarrollo de software no es como la

industria de la manufactura o la industria de automóviles donde los procesos de producción son en serie, de alta calidad y completamente automatizados.

El desarrollo de software, aun no es industrial, el proceso que se sigue es semi automatizado, todavía el código fuente se escribe a mano, valiéndose de editores o entornos de desarrollo integrado profesional. Si bien existe una variedad de metodología de desarrollo desde las más antiguas como el ciclo de vida en cascada hasta las más modernas como son las metodologías ágiles, a la fecha no existe una sola metodología de desarrollo estándar ni una herramienta que soporte esta metodología.

El problema de las metodologías de desarrollo en las asignaturas de la disciplina de Ingeniería de Software, es que no existen los lineamientos, guías o directrices que permita articular mejor en el desarrollo de las asignaturas desde los primeros nivele hasta los niveles que abarcan estas asignaturas. Por lo que existe una baja articulación de las metodologías de desarrollo de software en las asignaturas de Ingeniería de software en la carrera de Ingeniería de Sistemas.

1.2.1 Formulación del Problema

¿Cómo se articulan las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, 2021?

Objeto de estudio. Metodologías de desarrollo de software
Campo de acción. Las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

1.3 Justificación

El surgimiento de las distintas metodologías de desarrollo de software desde las tradicionales hasta las modernas o llamadas metodologías ágiles, han hecho que sea complicado articular en

las diferentes asignaturas y semestres de la carrera de Ingeniería de Sistemas. Esto es un vacío de conocimiento que amerita una investigación para identificar las acciones estratégicas a considerar a la hora de implementar los procesos formativos por parte de los docentes.

Cada nivel o semestre debe contar con una clara y precisa metodología de desarrollo de software según las exigencias reflejadas en las competencias a alcanzar con la asignatura, de no realizar esta apreciación y propuesta de solución no se garantiza que se esté cumpliendo con los objetivos propuestos en el plan de estudios de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

1.4 Objetivos

1.4.1 Objetivo General

Diseñar una estrategia metodológica para la articulación de las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando

1.4.2 Objetivos Específicos

- Sistematizar los referentes teóricos relacionados a las metodologías de desarrollo de software

- Diagnosticar el estado actual de las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando

- Determinar los componentes que se van a integrar en el diseño de una estrategia metodológica de articulación de metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

1.5 Hipótesis

La articulación de las metodologías de desarrollo de software en las asignaturas de la Ingeniería de Software es significativa en la Carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, 2021.

1.5.1. Identificación de las variables

Variable Independiente: Metodologías de desarrollo de software

Variable dependiente: Asignaturas de la Ingeniería de Software

1.5.2. Conceptualización de las variables

Tabla 1

Conceptualización de variables

Variable Independiente(V1)	**Definición conceptual**
Metodologías de desarrollo de software	Una metodología de desarrollo de software es un marco de trabajo que se usa para estructurar, planificar y controlar el proceso de desarrollo de los sistemas de software.
Variable Dependiente(V2)	**Definición conceptual**
Asignaturas de la Ingeniería de Software	Temas, técnicas y herramientas que contribuyen de manera sustancia al desarrollo de la Ingeniería de software en la formación de los estudiantes en la Carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

Fuente: Elaboración propia, 2022

1.6 Operacionalización de variables

Tabla 2.

Operacionalización de variables

DIMENSIÓN(S)	INDICADORES	TÉCNICA	INSTRUMENTO	UNIDAD DE ANÁLISIS
Metodologías convencionales y ágiles del desarrollo de software.	Menciona en el proyecto formativo la metodología de desarrollo de software a ser utilizada	Revisión documental	Guía de Observación	UA2
	Menciona las etapas(fases) de la metodología de desarrollo de software en el proyecto formativo de la asignatura	Revisión documental	Guía de Observación	UA2
	Propone técnicas o lenguajes de modelamiento de software en el proyecto formativo de la asignatura	Revisión documental	Guía de Observación	UA2
	Emplea herramientas de software para el modelamiento(análisis, diseño e implementación) del software.	Encuesta	Cuestionario	UA0, UA1
	Emplea software para la gestión del proyecto y control de versiones	Encuesta	Cuestionario	UA0, UA1
	Organiza equipos de trabajo coherente con la metodología de desarrollo de software.	Encuesta	Cuestionario	UA0, UA1

Fuente: Elaboración propia, 2022

1.7 Matriz de Consistencia

Tabla 3.

Matriz de consistencia

PROBLEMA	OBJETIVO	HIPÓTESIS
baja articulación de las metodologías de desarrollo de software con las asignaturas de Ingeniería de software en la carrera de Ingeniería de Sistemas.	Diseñar una estrategia metodológica para la articulación de las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando	La articulación de las metodologías de desarrollo de software en las asignaturas de la Ingeniería de Software es significativa en la Carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, 2021.

1.8 Unidades de Análisis

Unidad de Análisis 0 (UA0): Estudiantes que cursan la carrera de Ingeniería de Sistemas (cantidad se toma una muestra), proporcionará información importante respecto a la metodología de desarrollo de software que emplean en la asignatura.

Unidad de Análisis 1 (UA1): Docentes que imparten al menos una asignatura de la ingeniería de software (cantidad 7, se asume censo), proporcionará información importante respecto a la metodología de desarrollo de software que emplean en su asignatura.

Unidad de Análisis 2 (UA2): Proyectos formativo (PF) de las asignaturas e Informes (cantidad 12, se asume censo). Esta entidad nos proporcionará información importante respecto a la planificación y ejecución de la asignatura respecto de las metodologías de desarrollo de software.

CAPÍTULO II
DISEÑO METODOLÓGICO

2.1 Enfoque de investigación

Por la característica de la investigación, se adopta el enfoque de investigación cuantitativo, que, con base en la información teórica y empírica del objeto de estudio, permitirá la medición numérica, la interpretación y análisis estadístico de aporte en la construcción de la propuesta de investigación.

2.2 Tipo de investigación

De acuerdo al alcance de la investigación, es de tipo descriptivo propositivo, que permitirá caracterizar el objeto de estudio especificando las propiedades, los elementos y las relaciones estructurales que ella contiene.

2.3 Población y muestra

2.3.1 Población

La población considerada, está compuesta por 24 docentes y 208 estudiantes de la carrera de Ingeniería de Sistemas, datos que fueron proporcionados del Sistema Académico Siringuero con parámetros de primer semestre y gestión 2021.

2.3.2 Muestra

De acuerdo a la información proporcionada por la Dirección de la Carrera de Ingeniería de Sistemas, la muestra está conformada por 7 docentes que imparten asignaturas de la disciplina de Ingeniería de Software, quienes son especialistas en la materia y tienen conocimiento pleno del objeto de estudio, esta técnica de selección es no probabilistica. Considerando la población

estudiantil de la carrera de Ingeniería de Sistemas, se ha tomado una muestra representativa de 136 estudiantes, con un nivel de confianza del 95% y un margen de error del 5%.

2.3.3 Tipo de muestreo

- La técnica o método aplicado a la población de docentes, es no probabilístico.
- La técnica o método aplicado a la población estudiantil fue el muestreo probabilístico aleatorio simple.

2.4 Fuentes de investigación

Las fuentes de información que se empleará en la investigación son la literatura científica en la disciplina de la Ingeniería de Software. Aquellas que se encuentran en la biblioteca central del Área de Ciencia y Tecnología. También se empleará para esta investigación información disponible en la nube en los repositorios digitales, revistas científicas, etc.; principalmente para la etapa de la Investigación que concierne al desarrollo del marco teórico.

Para la segunda etapa de investigación, el marco práctico, nuestras principales fuentes de información serán los encuestados docentes y estudiantes de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, y la documentación concerniente a los proyectos formativos e informes de ejecución de los procesos formativos llevados a cabo por los mismos docentes y estudiantes.

2.5 Técnicas e instrumentos de investigación

Las técnicas de investigación empleadas en la recopilación de datos e información son las encuestas, realizadas docentes y estudiantes de la carrera que imparten asignaturas de la disciplina. También se realizó la revisión y análisis documental de la información disponible con relación al tema de estudio, concretamente los proyectos formativos e informes de ejecución. Los instrumentos a emplear son los cuestionarios y la guía de observación.

CAPÍTULO III
MARCO TEORICO

3.1. Marco Teórico

Esta sección expone en breve la conceptualización del objeto de estudio

De acuerdo a Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015). Las metodologías de desarrollo de software son indispensables para crear o actualizar software de calidad que cumpla con los requisitos de los usuarios; son una parte fundamental de la Ingeniería de software la cual denomina metodología a un conjunto de métodos coherentes y relacionados por unos principios comunes (p.8).

El concepto de la metodología y el desarrollo según Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015) es el "conjunto de métodos coherentes y relacionados por unos principios comunes". El concepto de desarrollo, está vinculado a la acción de desarrollar o a las consecuencias de este accionar, por tanto, es necesario, rastrear el significado del verbo desarrollar: se trata de incrementar, agrandar, extender, ampliar o aumentar alguna característica de algo físico (concreto) o intelectual (abstracto). Por tato la metodología de desarrollo es: el estudio y determinación de cuál es el método más adecuado para dar incremento a algo en este caso al software. (p.69). Esto indica que el término desarrollo es el más utilizado para referirse a las actividades que involucran la creación, fabricación, actualización o modificación de software.

Según, Maida, E. G., & Pacienzia, J. (2015). "Una metodología es un conjunto integrado de técnicas y métodos que permite abordar de forma homogénea y abierta cada una de las actividades del ciclo de vida de un proyecto de desarrollo" (p.13). A lo largo de la historia, las metodologías de desarrollo han surgido y evolucionado desde el ciclo de vida o cascada, el Incremental, el Evolutivo, el modelo Espiral, los Prototipos y el Desarrollo basado en componentes, y actualmente la metodología de desarrollo como una respuesta a las demandas exigentes por cubrir situaciones requerimientos emergentes y cambiantes cada vez más con el

uso de las tecnologías de Información y Comunicación. A lo que actualmente se los denomina metodologías ágiles.

> En febrero de 2001, tras una reunión celebrada en Utah-EEUU, nace el término "ágil" aplicado al desarrollo de software. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos, desarrollar software rápidamente y responder a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas. (Rivas, C. I. et. al, 2015, p.2).

Aunque los creadores e impulsores de las metodologías ágiles más populares han suscrito el manifiesto ágil, cada metodología tiene características propias y hace hincapié en algunos aspectos más específicos. La mayoría de ellas ya están siendo utilizadas con éxito en proyectos reales, pero les faltaba una mayor difusión y reconocimiento: SCRUM, Crystal Methodologies, Dynamic Systems Development Method, Adaptive Software Development, Feature -Driven Development y Lean Development.

3.2. Estado del arte

Las metodologías de Desarrollo de Software (DS.) han experimentado un proceso histórico y evolutivo que inicia en los años 40 con la aparición de las primeras computadoras, entonces no se contaban con parámetros ni estándares, el DS. Era prácticamente empírico y artesanal lo que llevó a que una buena parte de los proyectos fallaran en cubrir las exceptivas de los usuarios, así como en entregas extemporáneas y presupuestos excedidos, sobreviniendo la "crisis del Software" la respuesta para superarla fue la adopción de modelos y metodologías clásicas que progresivamente fueron incorporando estándares, controles y formalidades al DS. La evolución no se detuvo, con la llegada del Internet surgen proyectos caracterizados por requerimientos cambiantes y tiempos de entregas breves para los que las metodologías existentes no se adaptaban idóneamente, surgen las metodologías ágiles, enfocadas en interacción equipo-usuarios, entregas tempranas y adaptación a los cambios; conviven con los esquemas tradicionales y agrupan a comunidades activas.

3.2.1 Metodologías de desarrollo de software convencional

Las metodologías de desarrollo de software convencionales o tradicionales se caracterizan por definir total y rígidamente los requisitos al inicio de los proyectos de ingeniería de software. Los ciclos de desarrollo son poco flexibles y no permiten realizar cambios. La organización del trabajo de las metodologías tradicionales es lineal, es decir, las etapas siguen en secuencia, no se puede empezar la siguiente sin terminar la anterior, ni se puede volver hacia atrás una vez se ha transitado a otra etapa. Estas metodologías, no se adaptan nada bien a los cambios, y el mundo actual que cambia constantemente. Las principales metodologías tradicionales o clásicas son:

- ✓ Cascada: metodología en la que las etapas se organizan de arriba a abajo, de ahí el nombre. Se desarrollan las diferentes funciones en etapas diferenciadas y obedeciendo un riguroso orden. Antes de cada etapa se debe revisar el producto para ver si está listo para pasar a la siguiente fase. Los requisitos y especificaciones iniciales no están predispuestos para cambiarse, por lo que no se pueden ver los resultados hasta que el proyecto ya esté bastante avanzado.

- ✓ Prototipado: se basa en la construcción de un prototipo de software que se construye rápidamente para que los usuarios puedan probarlo y retroalimentar. Así, se puede arreglar lo que está mal e incluir otros requerimientos que puedan surgir. Es un modelo iterativo que se basa en el método de prueba y error para comprender las especificidades del producto.

- ✓ Espiral: es una combinación de los dos modelos anteriores, que añade el concepto de análisis de riesgo. Se divide en cuatro etapas: planificación, análisis de riesgo, desarrollo de prototipo y evaluación del cliente. El nombre de esta metodología da nombre a su funcionamiento, ya que se van procesando las etapas en forma de espiral. Cuanto más cerca del centro se está, más avanzado está el proyecto.

- ✓ Incremental: en esta metodología de desarrollo de software se va construyendo el producto final de manera progresiva. En cada etapa incremental se agrega una nueva funcionalidad, lo que permite ver resultados de una forma más rápida en comparación con el modelo en cascada. El software se puede empezar a utilizar incluso antes de que

se complete totalmente y, en general, es mucho más flexible que las demás metodologías.

- ✓ Diseño rápido de aplicaciones (RAD): esta metodología permite desarrollar software de alta calidad en un corto periodo de tiempo. Los costes son mucho más altos y el desarrollo más flexible, aunque requiere una mayor intervención de los usuarios. Por otro lado, el código puede contener más errores, y sus funciones son limitadas debido al poco tiempo del que se dispone para desarrollarlas. El objetivo es iterar el menor número posible de veces para conseguir una aplicación completa de forma rápida.

3.2.2 Metodología de desarrollo de software ágiles

Las metodologías ágiles de desarrollo de software son las más utilizadas hoy en día debido a su alta flexibilidad y agilidad. Los equipos de trabajo que las utilizan son mucho más productivos y eficientes, ya que saben lo que tienen que hacer en cada momento. Además, la metodología permite adaptar el software a las necesidades que van surgiendo por el camino, lo que facilita construir aplicaciones más funcionales.

Las metodologías ágiles se basan en la metodología incremental, en la que en cada ciclo de desarrollo se van agregando nuevas funcionalidades a la aplicación final. Sin embargo, los ciclos son mucho más cortos y rápidos, por lo que se van agregando pequeñas funcionalidades en lugar de grandes cambios.

Este tipo de metodologías permite construir equipos de trabajo autosuficientes e independientes que se reúnen cada poco tiempo para poner en común las novedades. Poco a poco, se va construyendo y puliendo el producto final, a la vez que el cliente puede ir aportando nuevos requerimientos o correcciones, ya que puede comprobar cómo avanza el proyecto en tiempo real.

Las principales metodologías ágiles son:

- ✓ Kanban: metodología de trabajo inventada por la empresa de automóviles Toyota. Consiste en dividir las tareas en porciones mínimas y organizarlas en un tablero de trabajo dividido en tareas pendientes, en curso y finalizadas. De esta forma, se crea un flujo de trabajo muy visual basado en tareas prioritarias e incrementando el valor del producto.

- ✓ Scrum: es también una metodología incremental que divide los requisitos y tareas de forma similar a Kanban. Se itera sobre bloques de tiempos cortos y fijos (entre dos y cuatro semanas) para conseguir un resultado completo en cada iteración. Las etapas son: planificación de la iteración (planning sprint), ejecución (sprint), reunión diaria (daily meeting) y demostración de resultados (sprint review). Cada iteración por estas etapas se denomina también sprint.

- ✓ Lean: está configurado para que pequeños equipos de desarrollo muy capacitados elaboren cualquier tarea en poco tiempo. Los activos más importantes son las personas y su compromiso, relegando así a un segundo plano el tiempo y los costes. El aprendizaje, las reacciones rápidas y potenciar el equipo son fundamentales.

- ✓ Programación extrema (XP): es una metodología de desarrollo de software basada en las relaciones interpersonales, que se consideran la clave del éxito. Su principal objetivo es crear un buen ambiente de trabajo en equipo y que haya un feedback constante del cliente. El trabajo se basa en 12 conceptos: diseño sencillo, testing, refactorización y codificación con estándares, propiedad colectiva del código, programación en parejas, integración continua, entregas semanales e integridad con el cliente, cliente in situ, entregas frecuentes y planificación.

3.2.3 Metodología de desarrollo de software presente y futuro

El desarrollo de software se ha convertido en una de las actividades más importantes de la sociedad moderna. En un mundo donde la inteligencia artificial y la tecnología en general constituyen una parte fundamental de nuestras vidas, desarrollar software seguro de alta calidad será cada vez más importante. En el presente y también en el futuro del desarrollo de software se irán articulando otras disciplinas de la ciencia de la computación, tales como la Inteligencia Artificial(IA), la ciencia de datos(CD), la robótica, etc.

- La inteligencia artificial(IA)

La inteligencia artificial no es un campo nuevo, pero si ha tenido un enorme crecimiento en los últimos años. La tecnología que saltó de las películas de ciencia ficción ahora es una herramienta más para ser más eficiente en los trabajos e incluso, puede ser aplicada en casi todos los campos (financieros, bancarios, industriales, marketing). Los conocimientos requeridos para empezar a trabajar con IA van desde Python, álgebra lineal y otros campos disciplinares. La demanda por desarrollar aplicaciones de software que incorporen IA, es enorme, es el momento perfecto para comenzar a adquirir el aprendizaje necesario con muchas de posibilidades en el desarrollo de la industria del software.

- La Ciencia de Datos

Según estudios, la ciencia de datos tiene un crecimiento anual del 37% pero, ¿qué hace exactamente un científico de datos? Según José Antonio Guerrero, un científico de datos reconocido, esta es la definición:

"Es una persona con fundamentos en matemáticas, estadística y métodos de optimización, con conocimientos en lenguajes de programación y que además tiene una experiencia práctica en el análisis de datos reales y la elaboración de modelos predictivos".

Gran parte de la labor que desarrolla un científico de datos se basa en preparar y visualizar datos para luego analizarlos, crear modelos y presentar resultados.

- Robótica

La robótica está más presente que nunca en la vida de las personas alrededor del mundo. Se encuentran en sectores industriales, comerciales, de salud y hasta científicos. El sector tiene un crecimiento anual del 40% y está centrado en la automatización en los procesos. Par el desarrollo de la misma es necesario dominar y tener amplios conocimientos en electrónica, lenguajes como C++, Python y hasta Javascript.

3.3 Marco conceptual

La metodología de desarrollo de software en ingeniería de software es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Desde la aparición de las computadoras hasta la actualidad, gran cantidad de métodos emergieron algunas con mayores ventajas que otras.

Una metodología de desarrollo de software sigue la filosofía de desarrollo de programas de computación con el enfoque del proceso de desarrollo de software, herramientas, modelos y métodos para asistir al proceso de desarrollo de software. Estos son a diario vinculados a algún tipo de organización que desarrolla, apoya el uso y promueve la metodología.

Requerimientos del cliente. Tenemos un software útil y podemos incorporarlo en el producto.

3.3.1. Metodologías convencionales

1. Metodología Cascada

En Ingeniería de software el desarrollo en cascada, es secuencial o lineal por la posición de las fases, el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior. El modelo está diseñado para llevar a cabo una revisión final, que se encarga de determinar si el proyecto está listo para avanzar a la siguiente fase. Este modelo fue el primero en originarse y es la base de todos los demás modelos de ciclo de vida.

Las fases de la metodología en cascada son las siguientes:

- ✓ Análisis de requisitos.
- ✓ Diseño del sistema.
- ✓ Diseño del programa.
- ✓ Codificación.
- ✓ Pruebas.
- ✓ Despliegue del programa.
- ✓ Mantenimiento.

De esta forma, cualquier error de diseño detectado en la etapa de prueba conduce necesariamente al rediseño y nueva programación del código afectado, aumentando los costos del desarrollo.

2. Metodología del Prototipado

El Modelo de prototipos, en Ingeniería de software, pertenece a los modelos de desarrollo evolutivo. El prototipo debe ser construido en poco tiempo, usando los programas adecuados y no se debe utilizar muchos recursos. El diseño rápido se centra en una representación de aquellos aspectos del software que serán visibles para el cliente o el usuario final. Este diseño conduce a la construcción de un prototipo, el cual es evaluado por el cliente para una retroalimentación; gracias a esta se refinan los requisitos del software que se desarrollará. La interacción ocurre cuando el prototipo se ajusta para satisfacer las necesidades del cliente. Esto permite que al mismo tiempo el desarrollador entienda mejor lo que se debe hacer y el cliente vea resultados a corto plazo. Las etapas de la metodología son:

- ✓ Comunicación
- ✓ Analogía
- ✓ Plan rápido.
- ✓ Modelado, diseño rápido
- ✓ Construcción del Prototipo
- ✓ Desarrollo, entrega y retroalimentación
- ✓ Entrega del final

Ventajas: Este modelo es útil cuando el cliente conoce los objetivos generales para el software, pero no identifica los requisitos detallados de entrada, procesamiento o salida. También ofrece un mejor enfoque cuando el responsable del desarrollo del software está inseguro de la eficacia de un algoritmo, de la adaptabilidad de un sistema operativo o de la forma que debería tomar la interacción humano-máquina

El paradigma de construcción de prototipos ayuda al desarrollado de software y al cliente a entender de mejor manera cuál será el resultado de la construcción cuando los requisitos estén satisfechos. De esta manera, este ciclo de vida en particular, involucra al cliente más profundamente para adquirir el producto.

Inconvenientes: El usuario tiende a crearse expectativas cuando ve el prototipo de cara al sistema final. A causa de la intención de crear un prototipo de forma rápida, se suelen desatender aspectos importantes, tales como la calidad y el mantenimiento a largo plazo, lo que obliga en la mayor parte de los casos a reconstruirlo una vez que el prototipo ha cumplido su función. En aras de desarrollar rápidamente el prototipo, el desarrollador suele tomar algunas decisiones de implementación poco convenientes (por ejemplo, elegir un lenguaje de programación incorrecto porque proporcione un desarrollo más rápido). Con el paso del tiempo, el desarrollador puede olvidarse de la razón que le llevó a tomar tales decisiones, con lo que se corre el riesgo de que dichas elecciones pasen a formar parte del sistema final.

3. Método Incremental

El modelo incremental combina elementos del modelo en cascada con la filosofía interactiva de construcción de prototipos. Se basa en la filosofía de construir incrementando las funcionalidades del programa. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario. Cada secuencia lineal produce un incremento del software.

Cuando se utiliza un modelo incremental, el primer incremento es a menudo un producto esencial, sólo con los requisitos básicos. Este modelo se centra en la entrega de un producto

operativo con cada incremento. Los primeros incrementos son versiones incompletas del producto final, pero proporcionan al usuario la funcionalidad que precisa y también una plataforma para la evaluación.

Ventajas: Mediante este modelo se genera software operativo de forma rápida y en etapas tempranas del ciclo de vida del software. Es un modelo más flexible, por lo que se reduce el coste en el cambio de alcance y requisitos. Es más fácil probar y depurar en una iteración más pequeña. Es más fácil gestionar riesgos. Cada iteración es un hito gestionado fácilmente.

Desventajas: Cada fase de una iteración es rígida y no se superponen con otras. Pueden surgir problemas referidos a la arquitectura del sistema porque no todos los requisitos se han reunido, ya que se supone que todos ellos se han definido al inicio.

4. Metodología Espiral

El modelo de desarrollo en Espiral es una combinación entre el modelo en cascada y el modelo por iteraciones. El proceso pasa por distintas etapas, desde la conceptualización, siguiendo el desarrollo, luego una fase de mejoras, para finalizar con el mantenimiento.

Dentro de cada etapa, tendremos una serie de fases que transcurren desde la planificación, pasando por el análisis de riesgos, el desarrollo y finalizando en la evaluación de lo realizado. Se incorpora también una fase de enlace entre etapas, para facilitar la transición entre las mismas.

En definitiva, el equipo de desarrollo en este modelo de desarrollo en espiral comienza con un pequeño conjunto de requisitos y pasa por cada fase de desarrollo para ese conjunto de requisitos.

El equipo de desarrollo agrega la funcionalidad para el requerimiento adicional en espirales cada vez mayores, hasta que la aplicación está lista para la fase de producción.

Las fases del modelo espiral son:

- ✓ Incluye la estimación del coste, el calendario y los recursos para la Planificación iteración. Implica también la comprensión de los requisitos del sistema para la comunicación continua entre el analista de requerimientos y el cliente.
- ✓ Análisis del riesgo. La identificación de los riesgos potenciales se realiza mientras se planifica y finaliza la estrategia de mitigación de riesgos.
- ✓ Ingeniería. Incluye la codificación, pruebas y el despliegue del software.
- ✓ Evaluación. Evaluación del software por parte del cliente. Además, incluye la identificación y el seguimiento de riesgos tales como los retrasos en los plazos y los sobrecostes.

Ventajas: La funcionalidad adicional o los cambios se pueden hacer en una etapa posterior. La estimación del coste se hace fácil, ya que la construcción del prototipo se hace en pequeños fragmentos. El desarrollo continuo o repetido ayuda en la gestión de riesgos. El desarrollo es rápido y las características se añaden de forma sistemática. Siempre hay espacio para atender los comentarios de los clientes.

Desventajas: Riesgo de no cumplir con la planificación o el presupuesto. Funciona mejor para proyectos grandes, aunque en estos también requiera de una estricta evaluación de riesgos. Para su buen funcionamiento, el protocolo del modelo en espiral debe ser seguido estrictamente. Se genera más documentación al tener fases intermedias. No es aconsejable para proyectos pequeños, la ratio coste beneficio no es rentable.

3.3.2 Metodologías Formales

Los métodos formales surgieron como puntos de vista analíticos con los que es posible verificar el desarrollo de sistemas mediante la lógica y las matemáticas, lo que aporta grandes ventajas para mejorar la calidad de los programas y por tanto la Ingeniería de Software. En este campo del conocimiento, la especificación formal es una de las más importantes fases del ciclo de vida, labor que requiere mucho cuidado ya que su función es garantizar que tanto el funcionamiento como el desempeño del programa sean correctos, bajo cualquier situación. En

el futuro, los métodos formales deberían estar presentes como principios esenciales en el desarrollo de software, ya que se convierten en la base para aplicar las técnicas de prueba y, dado su principio matemático, es potencialmente automatizables.(Serna & Candidato, 2010)

1. Metodología RUP

La metodología de desarrollo RUP por sus siglas en inglés ó Proceso de Desarrollo Unificado es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

La metodología RUP puede utilizarse, desde el principio de un nuevo proyecto de software, y puede seguir utilizándolo en los ciclos de desarrollo subsiguientes tiempo después de que el proyecto inicial haya terminado. No obstante, la forma de utilizar RUP varía para ajustarse a sus necesidades.

Características del RUP

- ✓ Iterativo e Incremental: El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en flujos de trabajo: Análisis de requisitos, Diseño, Implementación y Prueba.

- ✓ Dirigido por los casos de uso: Los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

- ✓ Centrado en la arquitectura: El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

- ✓ Enfocado en los riesgos: El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

Las fases y sus flujos de trabajo o actividades del RUP

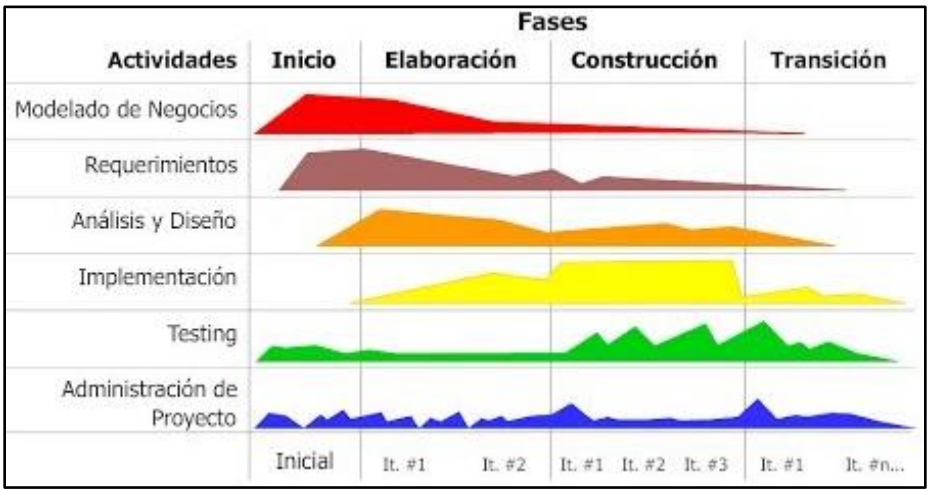

Ilustración 1. Fases y Flujos de Trabajo del RUP

Fuente: (Jacobson et al., 2006)

Fase: Inicio

La fase de inicio, es la fase más pequeña del proyecto e, idealmente, debe realizarse también en un periodo de tiempo pequeño (una única iteración).

Los objetivos que persigue esta fase son:

- ✓ Establecer una justificación para el proyecto.
- ✓ Establecer el ámbito del proyecto.
- ✓ Esbozar los casos de uso y los requisitos clave que dirigirán las decisiones de diseño.
- ✓ Esbozar las arquitecturas candidatas.
- ✓ Identificar riesgos.
- ✓ Preparar el plan del proyecto y la estimación de costes.

Fase: Elaboración

Durante esta fase se capturan la mayoría de los requisitos del sistema. Los objetivos principales de esta fase serán la identificación de riesgos y establecer y validar la arquitectura del sistema. La arquitectura se valida a través de la implementación de una Base de Arquitectura. Al final de la fase de elaboración la base de arquitectura ejecutable debe demostrar que soporta los aspectos clave de la funcionalidad del sistema y que muestra la conducta adecuada en términos de rendimiento, escalabilidad y coste. Al final de la fase se elabora un plan para la fase de construcción.

Fase: Construcción

- ✓ Es la fase más larga de proyecto.
- ✓ El sistema es construido en base a lo especificado en la fase de elaboración.
- ✓ Las características del sistema se implementan en una serie de iteraciones cortas y limitadas en el tiempo.
- ✓ El resultado de cada iteración es una versión ejecutable de software.
- ✓ El hito de capacidad operativa inicial marca el final de la fase.

Fase: Transición

- ✓ En esta fase el sistema es desplegado para los usuarios finales.
- ✓ La retroalimentación recibida permite incorporar refinamientos al sistema en las sucesivas iteraciones.
- ✓ Esta iteración también cubre el entrenamiento de los usuarios para la utilización del sistema.
- ✓ El hito de lanzamiento del producto marca el final de la fase.

2. Métrica V3

Métrica es una metodología para el desarrollo y mantenimiento de Sistemas de Información. Fue desarrolla por el Ministerio de Hacienda y Administraciones Públicas del Gobierno de España. Es de gran utilidad para dar soporte a grandes sistemas de información en el ciclo de vida del software además de un marco de gestión que asegura que los proyectos cumplen sus objetivos en términos de calidad, coste y plazos (V.3, n.d.).

Los objetivos a conseguir con esta metodología son:

- ✓ Dotar a la organización de productos software.
- ✓ Mejorar la productividad de los departamentos de sistemas y tecnologías de la información y las comunicaciones.
- ✓ Facilitar la operación, mantenimiento y uso de productos software.
- ✓ Definir sistemas de información que ayuden a conseguir los fines de la organización.

Las fases de la metodología son:

- ✓ Planificación de Sistemas de Información: Enfocado a partir del estudio de los últimos avances en este campo, alta competitividad y el cambio al que se someten las organizaciones.
- ✓ Desarrollo de Sistemas de Información: Este a su vez se divide en varios puntos.
- ✓ Estudio de viabilidad del sistema (EVS).
- ✓ Análisis del sistema de información (ASI).
- ✓ Diseño del sistema de información (DSI).
- ✓ Construcción del sistema de información (CSI).
- ✓ Implantación y aceptación del sistema (IAS).
- ✓ Mantenimiento de Sistemas de Información: Se realiza un registro de peticiones y se diagnostica el tipo de seguimiento, decidiendo si se le da respuesta o no en función del plan asociado al sistema de información.

Ventajas

- ✓ Se aplica a los procesos de las empresas.
- ✓ Permite anunciar el desempeño de futuras ejecuciones.
- ✓ Aumenta la capacidad de las empresas que desarrollan o mantienen software para ofrecer servicios de calidad y alcanzar niveles internacionales.
- ✓ Es muy práctico en su aplicación tanto a pequeñas empresas (PYMES) como a otros niveles.

Desventajas

- ✓ Se requiere modelos CMMI para asegurar la calidad de los procesos o productos.
- ✓ Es necesario que cada uno de los empleamos tenga el compromiso y criterio para seguir el plan.

3.3.3 Metodologías Ágiles

El 12 de febrero de 2001 diecisiete críticos de los modelos de mejora del desarrollo de software basados en procesos, convocados por Kent Beck, quien había publicado un par de años antes Extreme Programming Explained, libro en el que exponía una nueva metodología denominada Extreme Programming, se reunieron en Snowbird, Utah para tratar sobre técnicas y procesos para desarrollar software. En la reunión se acuñó el término "**Métodos Ágiles**" para definir a los métodos que estaban surgiendo como alternativa a las metodologías formales (como el RUP, el proceso unificado, por ejemplo) a las que consideraban excesivamente "pesadas" y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo(Beck et al., 2001).

1. Metodología SCRUM

La metodología SCRUM fue identificado y definido por Ikujiro Nonaka y Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica: Fuji-Xerox, Canon, Honda, NEC, Epson, Brother, 3M y Hewlett-Packard. En su estudio, Nonaka y Takeuchi compararon la nueva forma de trabajo en

equipo, con el avance en formación de melé (scrum en inglés) de los jugadores de Rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Scrum es una metodología de desarrollo ágil que, aunque surgió como modelo para el desarrollo de productos tecnológicos, es empleada en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad, como el desarrollo de software. Está centrada principalmente en la gestión del equipo de desarrollo, su enfoque es iterativo e incremental. Esta metodología ayuda a los equipos a desarrollar productos en periodos cortos, permitiendo una rápida retroalimentación(feedback) del cliente, adaptaciones y mejora continua.

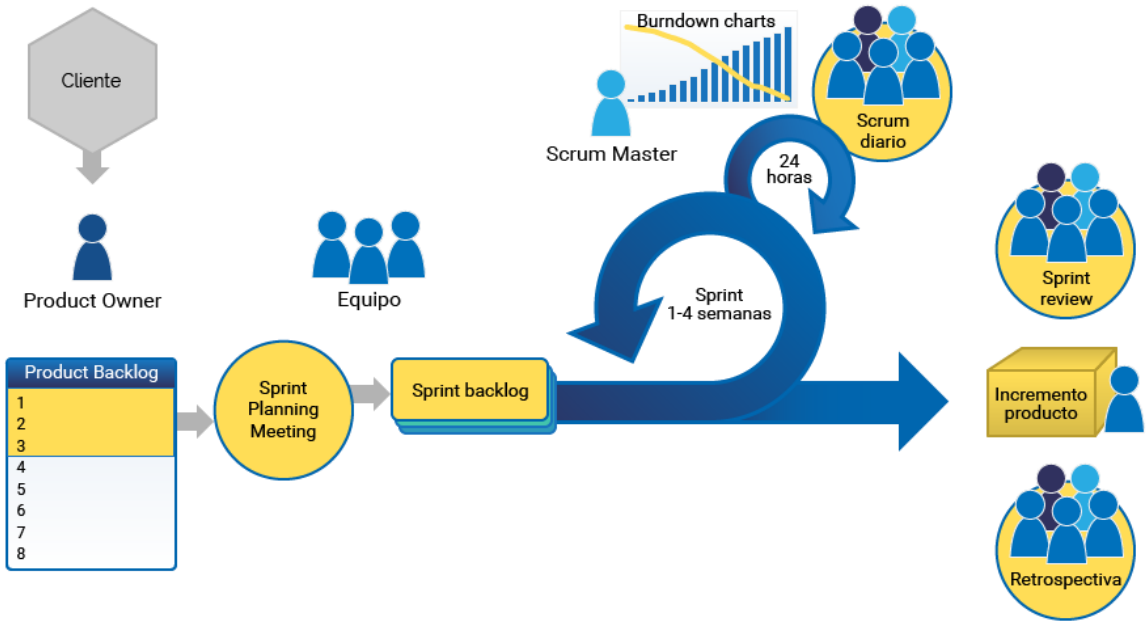

Ilustración 2. Esquema Metodología SCRUM
Fuente:Iglesias & Cuiñas, 2017

Las fases del Scrum se distribuyen en 16 procesos o tareas, que se resumen en 5 etapas de implementación:

1. Inicio
2. Planificación y estimación
3. Implementación
4. Revisión y retrospectiva
5. Lanzamiento

1. Inicio

La primera fase se encarga de estudiar y analizar el proyecto identificando las necesidades básicas del sprint. En el contexto de las metodologías ágiles, un sprint es un mini-proyecto con una duración no mayor a un mes que se interconecta con otros mini-proyectos. Para esta fase se debe hacer las siguientes preguntas: ¿Qué quiero? ¿Cómo lo quiero? y ¿Cuándo lo quiero? - Esta metodología da preferencia a la formación de equipos pequeños de mínimo 3 y máximo 5 personas, pues se facilita la fluidez de las ideas y se aporta creatividad al grupo. Los primeros pasos de Scrum son 6 procesos:

- ✓ Crear la visión del proyecto
- ✓ Identificar a los Master Scrum o ScrumMaster y a los stakeholders.
- ✓ Formar equipos Scrum
- ✓ Desarrollar épicas
- ✓ Crear backlogs o listas de requerimientos priorizando el producto
- ✓ Planificar el lanzamiento

2. Planificación y estimación

La segunda fase de Scrum incluye normalmente los siguientes pasos:

- ✓ Crear, estimar y comprometer historias de usuario.
- ✓ Identificar y estimar tareas.
- ✓ Crear el sprint backlog o iteración de tareas.

La clave para llevar una buena administración de los proyectos es hacer una planificación y estimación del sprint, lo que ayudará a establecer metas fijas y a cumplir con los plazos. Esta fase se considera como la más importante del proyecto, pues el Master Scrum tendrá que delegar las tareas correspondientes a cada grupo y hacer las estimaciones de tiempos de entrega, así como crear una lista ordenada para clasificar el trabajo según su prioridad.

3. Implementación

En esta etapa nos encontramos en la sala de reuniones donde se discute el sprint y se explora cómo optimizar el trabajo de cada grupo Scrum para darle forma definitiva al proyecto. En la implementación se cumple con los siguientes procesos:

- ✓ Crear entregables.
- ✓ Realizar daily stand-up.
- ✓ Refinanciamiento del backlog priorizado del producto.

En la fase de implementación o desarrollo no deberían hacerse cambios innecesarios de última hora (se supone que para evitarlo existe una fase de planificación). Aun así, si de ser necesario hacer un movimiento que será clave para el éxito del sprint, se debe proceder.

4. Revisión y retrospectiva

Una vez que ya todo está maquetado e implementado, deberás hacer la revisión del proceso, que no es más que la autocrítica o evaluación interna del grupo respecto a su propio trabajo. Es importante sumar opiniones constructivas y aportar soluciones viables. Entre los pasos más importantes para realizar en esta fase tenemos:

- ✓ Demostrar y validar el sprint.
- ✓ Retrospectiva del sprint.
- ✓ Lanzamiento
- ✓ La última de las fases del método Scrum es el lanzamiento.

Con esto nos referimos al desenlace del proyecto y entrega del producto, donde deberías cumplir con 2 únicas tareas que son:

- ✓ Enviar entregables.
- ✓ Enviar retrospectiva del proyecto.

5. Lanzamiento

La última de las fases del método Scrum es el lanzamiento. Que consiste en la entrega del producto, ello implica cumplir 2 únicas tareas:

- ✓ Enviar entregables.
- ✓ Enviar retrospectiva del proyecto.

2. Xtreme Programming (SXP)

En el año 2001, profesionales expertos de la ingeniería del software (17 en cantidad) creaban el Manifesto Ágil con el objetivo de establecer los principios para desarrollar software de forma rápida y adaptable a cambios, ofreciendo una alternativa a procesos de desarrollo de software tradicionales.

La metodología XP es un conjunto de técnicas que dan agilidad y flexibilidad en la gestión de proyectos. También es conocida como Programación Extrema (Extreme Programming) y se centra crear un producto según los requisitos exactos del cliente. De ahí, que le involucre al máximo durante el método de gestión del desarrollo del producto. La primera vez que escucho de esta metodología fue en 1999 en el libro Extreme Programming Explained, escrito por el ingeniero de software Kent Beck.

El uso de esta metodología supone, para muchos teóricos, una aproximación a la calidad óptima del producto, ya que durante el ciclo de vida del software ocurren cambios naturales. Se considera que cuantos más cambios, puede que más cerca estemos del mejor resultado que espera nuestro cliente. Por tanto, el cambio constante en el proyecto se considera como favorable.

Las características de la metodología XP son:

- ✓ Comunicación constante entre el cliente y el equipo de desarrollo.

- ✓ Respuesta rápida a los cambios constantes.
- ✓ La planificación es abierta con un cronograma de actividades flexible.
- ✓ El software que funciona está por encima de cualquier otra documentación.
- ✓ Los requisitos del cliente y el trabajo del equipo del proyecto son los principales factores de éxito del mismo.

Las fases de la metodología son:

1. Planificación
2. Diseño
3. Codificación
4. Pruebas
5. Lanzamiento

Se muestra en la figura a continuación sobre el esquema fases de la programación extrema:

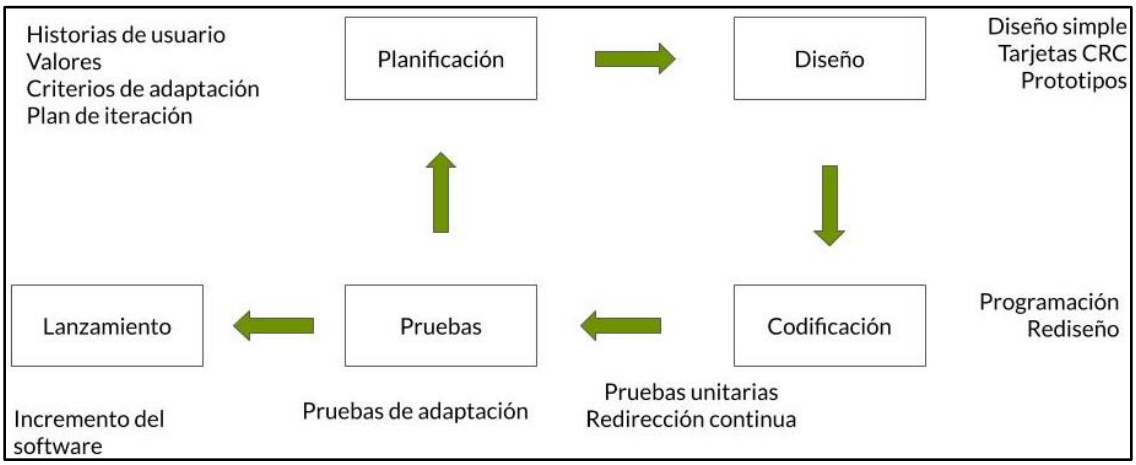

Ilustración 3. Esquema Fases de la Programación Extrema
Fuente: Iglesias & Cuiñas, 2017

Fase 1: Planificación

Esta fase consiste en la identificación de las historias de usuario, la priorización y descomposición de las mini-versiones del producto. Cada dos semanas aproximadamente de iteración se obtiene un software útil, funcional, listo para probar y lanzar.

Fase 2: Diseño

En este paso se trabaja un código sencillo, haciendo lo mínimo imprescindible para que funcione. Se obtendrá el prototipo. Además, para el diseño del software orientado a objetos, se crearán tarjetas CRC (Clase-Responsabilidad-Colaboración).

Fase 3: Codificación

En esta fase la programación se realiza en parejas en frente del mismo ordenador. Algunas veces se intercambian las parejas, esto para asegurar que se realice un solo código más universal y homogéneo con el que cualquier otro programador podría trabajar y entender. Y es que debe parecer que ha sido realizado por una única persona. Así se consigue una programación organizada y planificada.

Fase 4: Pruebas

En esta fase se realizan las pruebas automáticas continuamente. Al tratarse normalmente de proyectos a corto plazo, el testeo automatizado y constante es clave. Además, el propio cliente puede hacer pruebas, proponer nuevas pruebas e ir validando las mini-versiones.

Fase 5: Lanzamiento

En esta fase se ha probado todas las historias de usuario o mini-versiones con éxito, ajustándose a los requerimientos de los usuarios.

3. Metodología Kanban

El método Kanban surgió en el año 2005 gracias al ingeniero industrial japonés de Toyota David J. Anderson, y fue creado para conseguir una eficiente producción de automóviles. La empresa se ha convertido en uno de los referentes en su industria y ha experimentado un crecimiento notable gracia a ella.

El método Kanban está diseñada para soportar un control productivo y descentralizado por demanda. En el desarrollo de software ágil se utiliza para visualizar los proyectos, al colocar una serie de tarjetas en un panel o soporte denominada tareas Kanban. Por tanto, el método Kanban se basa en un sistema de señalización en el que se visualizan las tareas de producción por demanda mediante tarjetas.

El objetivo con el que se concibió la metodología Kanban, fue el de minimizar el esfuerzo(trabajo) en progreso y el stock entre procesos, esto quiere decir que el proceso que sigue a otro, solo se lleva a cabo si el inferior lo necesita, es decir, se realiza a demanda real.

Bajo esta metodología se considera que el peor gasto de una empresa es la sobreproducción, y para su prevención se debe controlar todos los procesos. Por ejemplo, si un trabajador de un nivel inferior necesita material de un nivel superior, se fabrica, pero el proceso superior nunca produce sin una previa instrucción o solicitud del inferior.

Características importantes de Kanban

El método Kanban se organiza con un tablon grande dividido en columnas. El número de columnas puede varia dependiendo de nivel de complejidad o fases del proceso. De acuerdo a (Rios, 2015) el flujo siempre avanza de izquierda a derecha asignando cada tarea en las columnas.

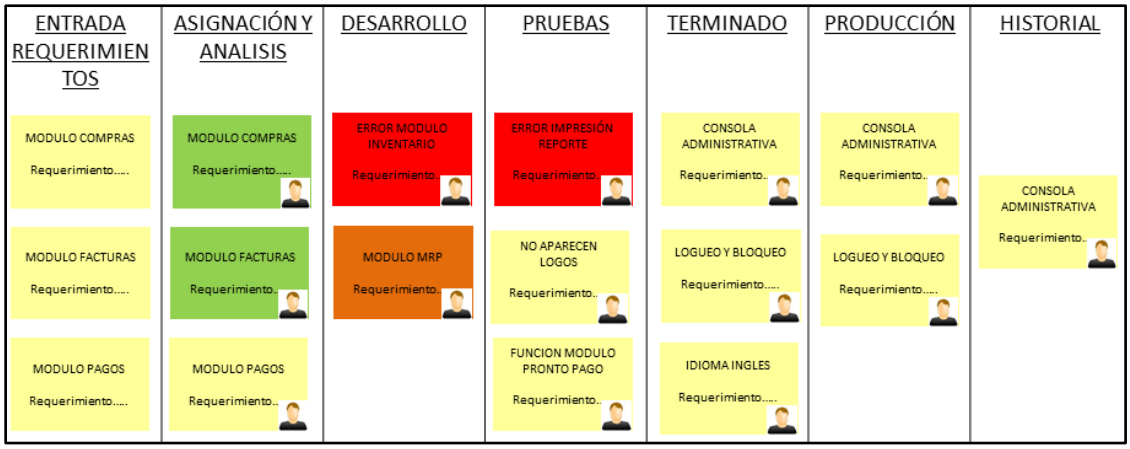

Ilustración 4. Caracterización de la Metodología XP
Fuente: Rios, 2015

Las características principales del método Kanban son las siguientes:

- Lista de tareas o <<Todo>>: aquí se engloban la lista de tareas que están pendientes y que se pueden afrontar inmediatamente. Para destacar la prioridad de las tareas, se colocan arriba las de mayor prioridad y orden descendente el resto, posteriormente, se puede pasar a otras columnas.
- El desarrollo o <<Doing>>: aquí se sitúa una tarea hasta que se complete. Si algo falla, la terea debe regresara a la columna previa.
- Pruebas: Se realizan las comprobaciones o test necesarios para ver si la tarea se ha realizado con éxito o no. Si esta todo correcto pasa a la siguiente fase, si no, volverá a la fase de desarrollo. Esta columna puede ir independiente o como una subcolumna dentro de <<Doing>>, aunque el resultado y el orden es el mismo.
- Despliegue: Una vez validado el código se añaden a esta columna para su subida a producción en el sistema. Si todo funciona correctamente se pasaría a la siguiente fase.
- Terminado o <<Done>>: Aquí se incluyen las tareas que están finalizadas por completo.

3.3.4 Metodologías Emergentes

Los desarrolladores de aplicaciones van cambiando por innovaciones tecnológicas, estrategias de mercado y otros avatares de la industria de la informática, esto lleva a los desarrolladores de aplicaciones a evolucionar para obtener aplicaciones en menor tiempo, más vistosas y de menor costo. Los usuarios exigen calidad frente a los requisitos y los desarrolladores de aplicaciones deben contar con técnicas y herramientas logrando satisfacer las necesidades de los usuarios y obteniendo sistemas fáciles de mantener, extender y modificar(De San Martín Oliva, 2009).

1. Metodología Mobile-D

Una metodología de desarrollo nueva, especialmente diseñada para el desarrollo de aplicaciones móviles, recibe el nombre de Mobile-D y es propuesta por Pekka Abrahamsson y su equipo del VTT (Valtion Teknillinen Tutkimuskeskus, en inglés Technical Research Centre of Finland) en Finlandia.

El ciclo del proyecto se divide en cinco fases:

1. Exploración,
2. Inicialización,
3. Productización,
4. Estabilización y
5. Prueba del sistema.

En general, todas las fases (con la excepción de la primera fase exploratoria) contienen tres días de desarrollo distintos: planificación, trabajo y liberación. Se añadirán días para acciones adicionales en casos particulares (se necesitarán días para la preparación del proyecto en la fase de inicialización).

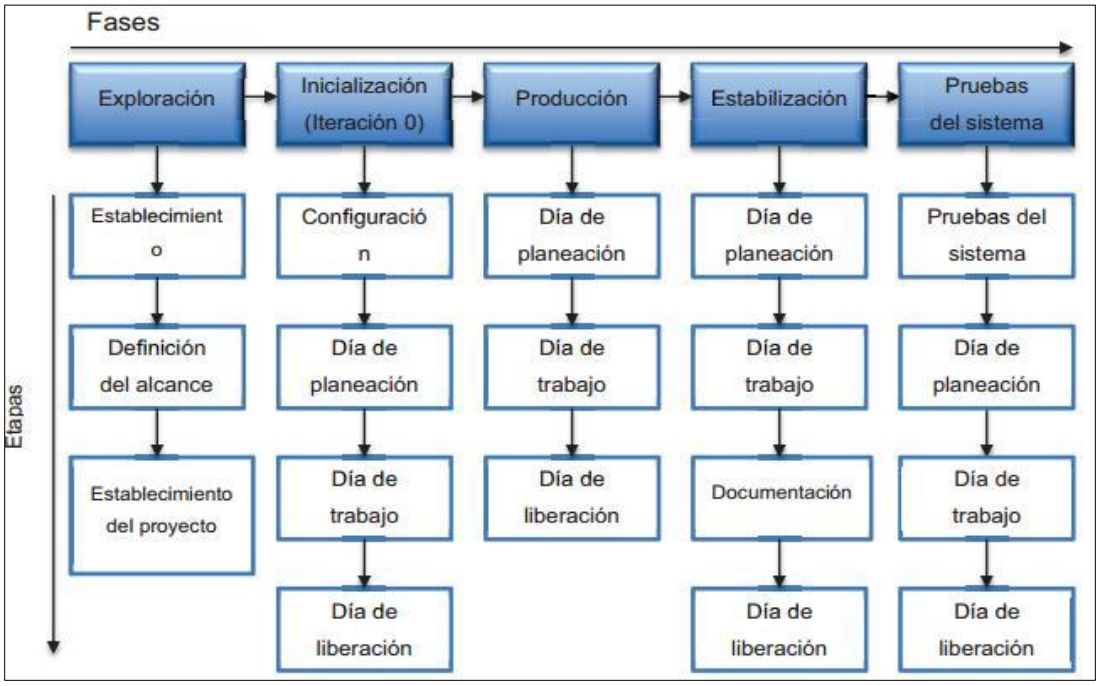

Ilustración 5. Fases Metodologia Mobile-D
Fuente: (Rios, 2015, p.95)

1. ONIXIC

El proceso ICONIX se define como un proceso de desarrollo de software práctico. Está entre la complejidad de RUP y la simplicidad y pragmatismo de XP, sin eliminar las tareas de análisis y diseño que XP no contempla. Es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto.

ICONIX presenta claramente las actividades de cada fase y exhibe una secuencia de pasos que deben ser seguidos. Además, está adaptado a patrones y ofrece el soporte UML, dirigido por Casos de Uso y es un proceso iterativo e incremental.

ICONIX deriva directamente del RUP y su fundamento es el hecho de que un 80% de los casos pueden ser resueltos tan solo con un uso del 20% del UML, con lo cual se simplifica muchísimo el proceso sin perder documentación al dejar solo aquello que es necesario. Esto implica un uso dinámico del UML de tal forma que siempre se pueden utilizar otros diagramas además de los ya estipulados si se cree conveniente(Universitat de Girona, 2013).

Iconix se estructura en cuatro fases:

- Análisis de Requisitos

En esta primera fase se realiza un Modelo de Dominio, que no es más que un Diagrama de Clases extremadamente simplificado. Este modelo contiene únicamente aquellos objetos de la vida real cuyo comportamiento o datos deban ser almacenados en el sistema. A partir de este pequeño modelo, se realiza un pequeño prototipo basándose en la storyboard de la interfaz gráfica obtenida previamente, el cual se mostrará al cliente y se refinará en sucesivas reuniones.

Normalmente este prototipo suele converger en dos o tres iteraciones. Una vez el prototipo ya es final y se han obtenido todos los requisitos del sistema por parte del cliente, se procede a realizar los casos de uso. Estos diagramas de casos de uso se agrupan en diagramas de paquetes

(es decir, utilizan referencias entre diagramas de casos de uso para simplificar su lectura) y se asocia cada requisito a un caso de uso para obtener la ya mencionada anteriormente trazabilidad.

- Análisis y Diseño Preliminar

A partir de cada caso de uso se obtienen sus correspondientes fichas de caso de uso. La ficha está formada por un nombre, que suele ser el del caso de uso, posee una breve descripción, una precondición que debe cumplir antes de iniciarse, una postcondición que debe cumplir al terminar si termina correctamente, un flujo normal que sigue el sistema en caso de que todo vaya correctamente y un flujo alternativo en caso de que haya cualquier problema. El resto de campos son opcionales. Después será necesario realizar lo que se conoce como Diagrama de Robustez, el cual pertenece al proceso Iconix que no forma parte del UML.

- Diseño

En esta fase se proceden a realizar los diagramas de secuencia, los cuales derivan directamente de las fichas de caso de uso. Obsérvese como, los diagramas de secuencia se relacionan con fichas de caso de uso que se relacionan con casos de uso que se relacionan con requisitos. Esto implica que, una vez finalizado el diseño, tras refinar nuevamente el diagrama de clases, podremos verificarlo directamente gracias a este factor de trazabilidad, y prepararnos para la siguiente fase. En caso de que no estemos satisfechos con el resultado, será necesario repasar todo el proceso hasta que éste sea correcto. Es vital que los requisitos se satisfagan correctamente para el éxito del proyecto.

- Implementación

De cara a poder distribuir el software correctamente, puede ser adecuado realizar un diagrama de componentes en algunos casos, pero no siempre es necesario. En cualquier caso, aquí es donde se escribe el código tal y como fue especificado en las fases anteriores y se planean las pruebas basándonos en los requisitos iniciales, al nivel que fuese necesario.

Aquí es donde hacemos uso real de la trazabilidad y donde realmente ponemos en práctica esa garantía de calidad que tanto hemos mencionado. Después de tener un buen diseño, es cuestión de crear un buen software a partir de ese diseño, y mediante los testeos y pruebas adecuados podemos garantizar que el sistema final cumple con los requisitos iniciales y por tanto proceder a su entrega.

CAPÍTULO IV

RESULTADOS DE LA INVESTIGACIÓN

4.1. Resultados de la Investigación

4.1.1. Resultados de Encuestas a Docentes y Estudiantes

En esta sección se realizó el trabajo de campo aplicando los instrumentos de recolección de datos a las unidades de análisis (docentes y estudiantes).

Los resultados se describen a continuación:

1. Asignaturas de Ingeniería de Software que Imparten los docentes de la Carrea de Ingeniería de Sistemas.

El 100% de los docentes que imparten las asignaturas de la disciplina de Ingeniería de software han encarado proyectos de desarrollo de software aplicando técnicas, prácticas metodológicas de la Ingeniería de software la misma se refleja tanto en la planificación de asignatura (proyecto formativo) y los informes.

La evaluación en relación al tema se hizo tomando en cuenta las siguientes asignaturas:

- PROGRAMACIÓN I
- PROGRAMACIÓN II
- PROGRAMACIÓN III
- INGENIERÍA DE SOFTWARE I
- INGENIERÍA DE SOFTWARE II
- ESTRUCTURA DE DATOS
- SISTEMAS DE INFORMACIÓN II
- TECNOLOGÍAS EMERGENTES
- SISTEMAS DE INFORMACIÓN I
- BASE DE DATOS I
- BASE DE DATOS II
- SISTEMAS DE INFORMACIÓN GEOGRÁFICA I

2. Metodología de desarrollo de software para encarar los proyectos de software (Docentes).

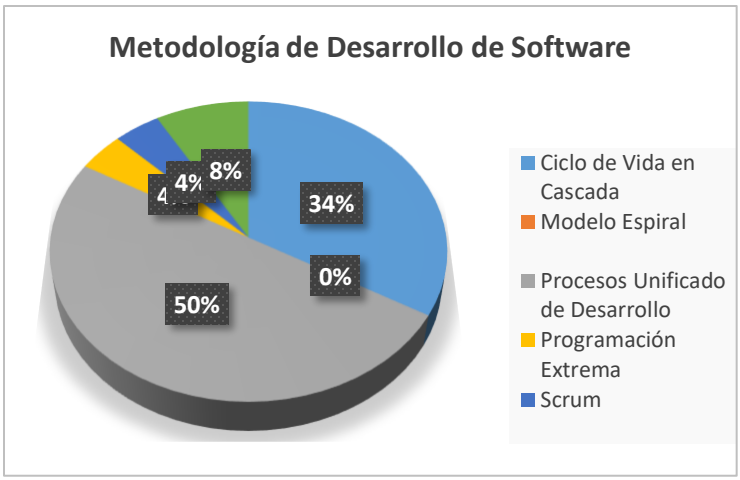

Ilustración 6. Metodologías de Desarrollo de Software
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 6. El 50% de los docentes que imparten asignaturas de Ingeniería de Software aplican en los proyectos de asignatura la metodología del Proceso Unificado de Desarrollo. El 34% de los docentes aplican el ciclo de vida en cascada. Y las demás metodologías de desarrollo en un porcentaje inferior al 10%.

3. Metodología de desarrollo de software (Estudiantes).

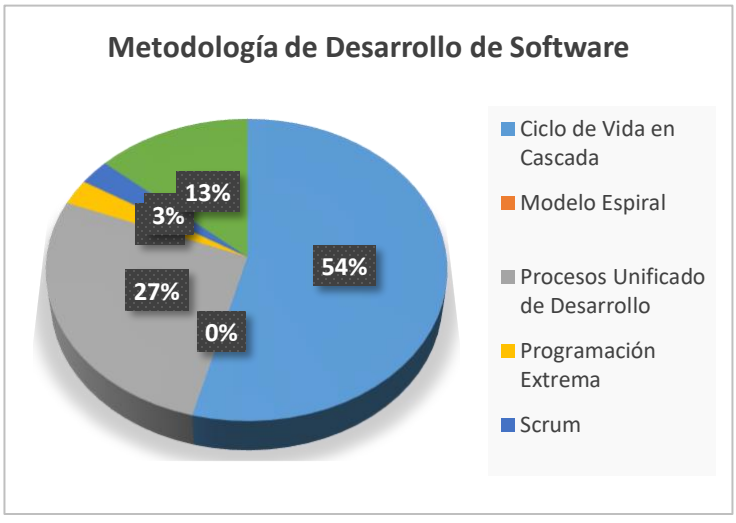

Ilustración 7. Metodología de Desarrollo de Software
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 7. El 54% de los estudiantes encuestados han indicado que en la elaboración de su proyecto aplican la metodología del Ciclo de vida en cascada. El 27% indicaron que aplican la metodología del Procesos Unificado de Desarrollo. El resto de las metodologías como el modelo espiral, el scrum y la programación extrema menor al 13%.

4. Criterios de selección de una metodología de desarrollo de software (Docentes).

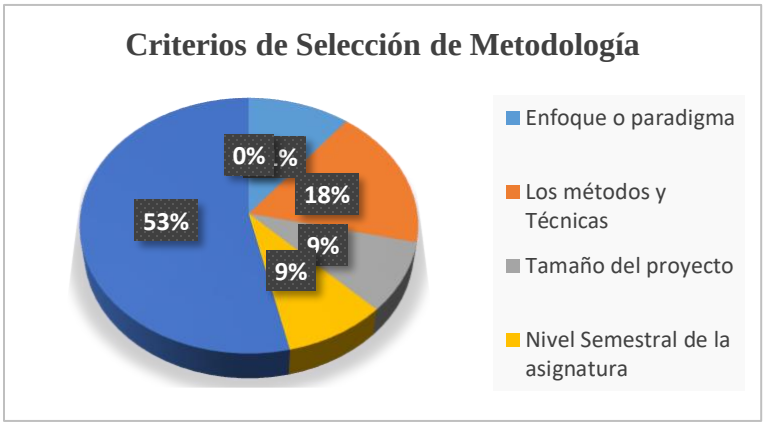

Ilustración 8. Criterios de Selección de Metodología
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 8. El 53% de los docentes encuestados indicaron que el criterio de selección de una metodología de desarrollo de software es el enfoque o paradigma. El 18% mencionaron que el criterio de selección de una metodología de desarrollo son los métodos y las Técnicas. El resto de los criterios en porcentaje mejor al 10%.

5. Criterios de selección de una metodología de desarrollo de software (Estudiantes).

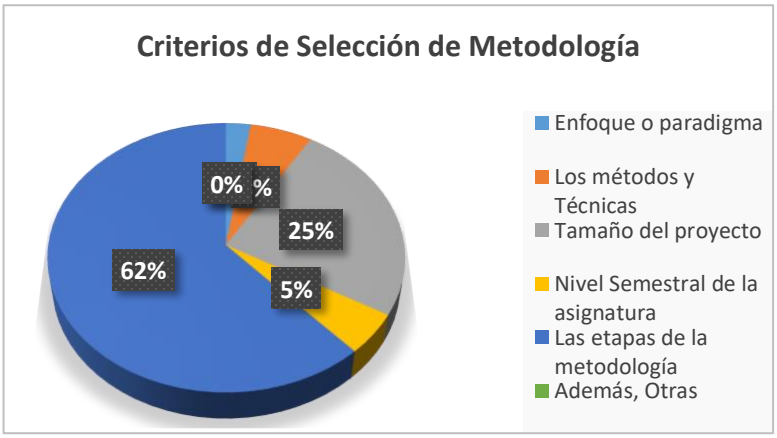

Ilustración 9. Criterios de Selección de Metodología
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 9. El 62% de los estudiantes encuestados indicaron que el criterio de selección de una metodología de desarrollo de software es el enfoque o paradigma. El 25% mencionaron que el criterio de selección de una metodología de desarrollo es el tamaño del proyecto. El resto de los criterios en porcentaje mejor al 10%.Técnicas de modelamiento de análisis y diseño de sistema.

5.Técnicas de modelamiento de análisis y diseño de sistemas (Docentes).

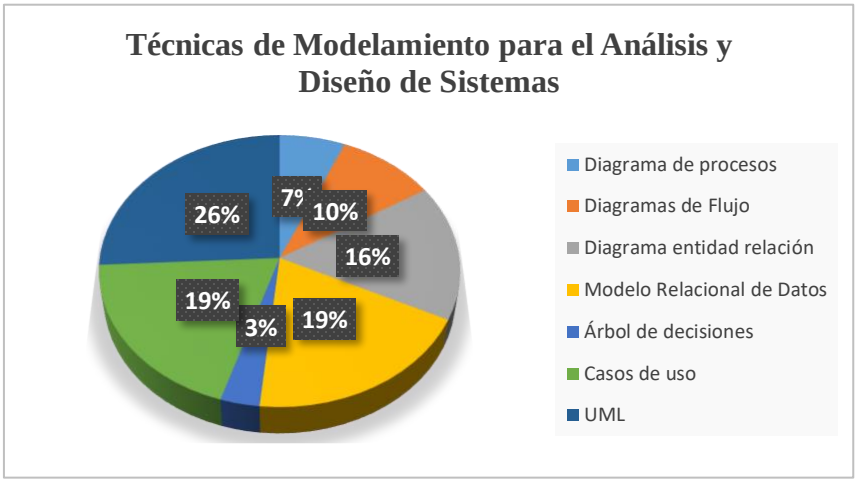

Ilustración 10. Técnicas de Modelamiento
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 10. El 25% de los docentes emplean como técnica de modelamiento de Análisis y Diseño de sistemas los diagramas de procesos. El 19% la técnica de los casos de uso. El 19% los diagramas de flujo. El 16% los diagramas de entidad relación. Y el resto de las técnicas en porcentaje menor al 10%.

6. Técnicas de modelamiento para análisis y diseño de sistema (Estudiantes).

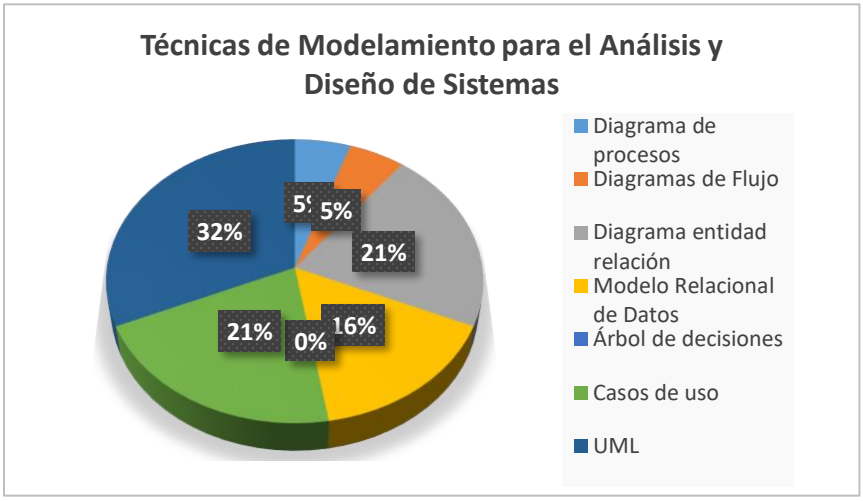

Ilustración 11. Técnicas de Modelamiento
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 11. El 32% de los estudiantes encuestados emplean como técnica de modelamiento de Análisis y Diseño el Árbol de Decisiones.

El 21% de los estudiantes emplean como técnica de modelamiento de Análisis y Diseño de sistemas la técnica de los casos de uso. El 21% la Diagramas Entidad Relación. El 16% el modelo relacional de datos. Y El resto de las técnicas en porcentaje menor al 10%.

7. Herramientas CASE en el Desarrollo de Software (Docentes).

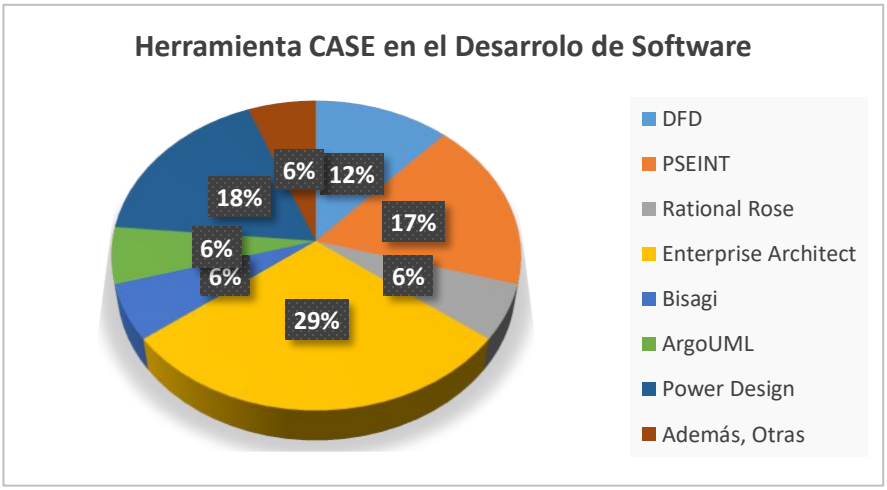

Ilustración 12. Herramientas CASE en el Desarrollo de Software
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 12. El 29% de los docentes emplean como herramienta CASE en el desarrollo de los proyectos de software el Enterprise Architect. El 18% el Bisagi. El 17% PSEINT. Y el resto en porcentaje menor al 15%.

8. Herramientas CASE en el Desarrollo de Software (Estudiantes).

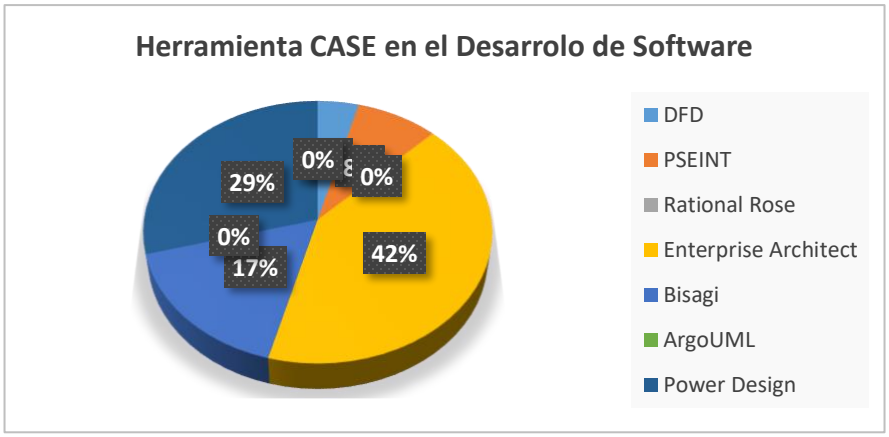

Ilustración 13. Herramientas CASE en el Desarrollo de Software
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 13. El 29% de los estudiantes encuestados utilizan el Enterprise Architect como herramienta de desarrollo. El 18% el programa de DFD para diseñar diagramas de flujo de datos. El 17% PSEINT. El resto de las herramientas en por porcentaje menor al 12%.

9. Herramientas de Software para el control de versiones (Docentes).

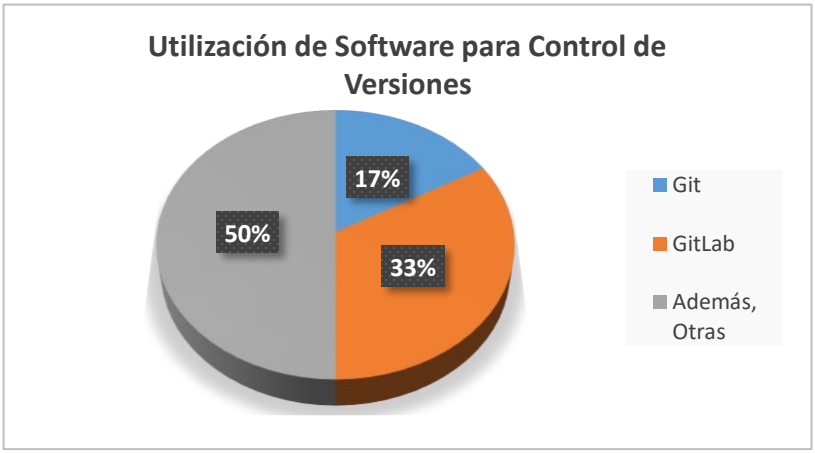

Ilustración 14. Software para el Control de Versiones
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 14. El 17% de los docentes emplean el **GIT** para el control de las versiones en el desarrollo del proyecto. El 33% utiliza el **GitLab**. El 50% utilizan otras herramientas para el control de versiones.

10. Utilización de software para control de versiones (Estudiantes).

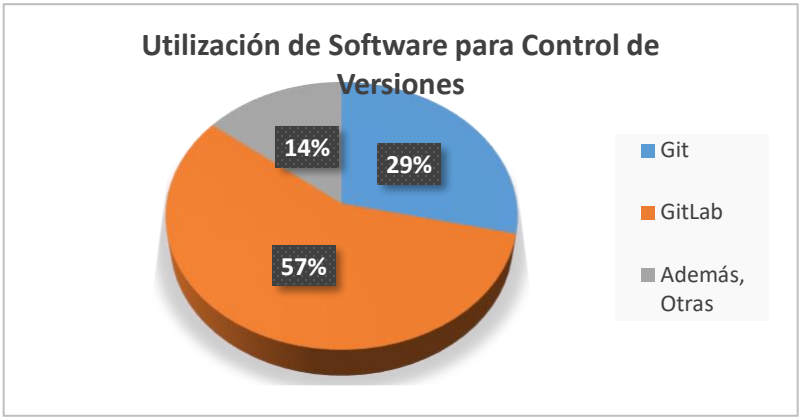

Ilustración 15. Software para el Control de Versiones
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 15. El 57% de los estudiantes encuestados utilizan el GitLab para el control de versiones durante el desarrollo del proyecto. El 29% de los estudiantes utilizan el GIT para el control de versiones del proyecto. El 14% utilizan otras herramientas para el control de versiones.

11. Organización de los equipos de trabajo para el desarrollo del proyecto de software (Docentes).

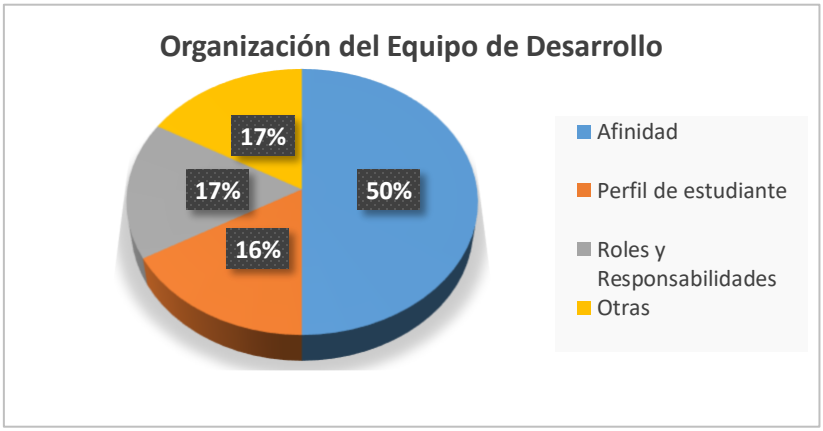

Ilustración 16. Organización de Equipos de Trabajo
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 16. El 50% de los docentes encuestados recomiendan que la organización de los equipos de desarrollo se realice por afinidad. El 17% de los docentes recomienda que se haga los grupos de trabajo de acuerdo a roles y responsabilidades. El 16% de acuerdo al perfil del estudiante. El 17% otras formas de organización.

12. Organización del equipo de desarrollo (Estudiantes).

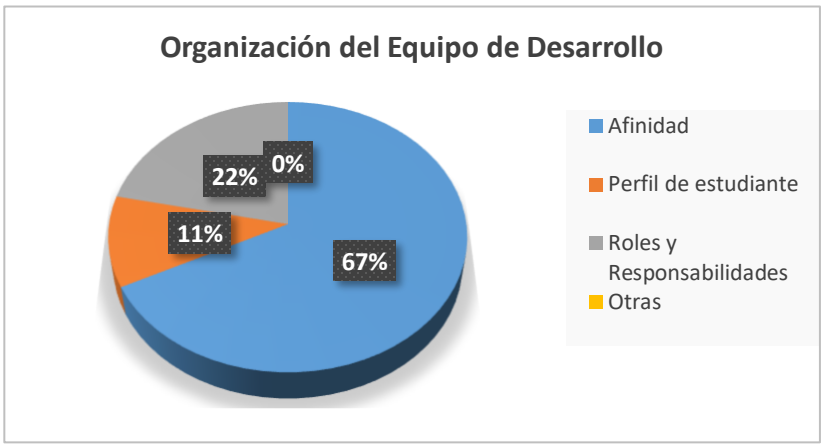

Ilustración 17. Organización de Equipos de Trabajo
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 17. El 67% de los estudiantes encuestados indicaron que el equipo de trabajo para el desarrollo del software se realiza por afinidad. El 22% indicaron que los equipos de trabajo son por roles y responsabilidades.

13. Grado de importancia de los Principios y valores del equipo de trabajo (Docentes).

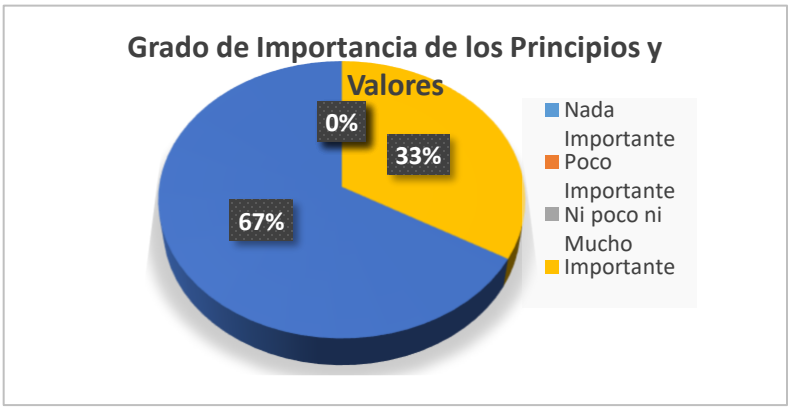

Ilustración 18. Grado de Importancia de los Principios y Valores
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 18. El 67% de los docentes encuestados mencionaron que es muy importancia los valores en los equipos de trabajo. El 33% de los mismos mencionar que es importante los valores en los equipos de trabajo.

14. Grado de importancia de los principios y valores (Estudiantes).

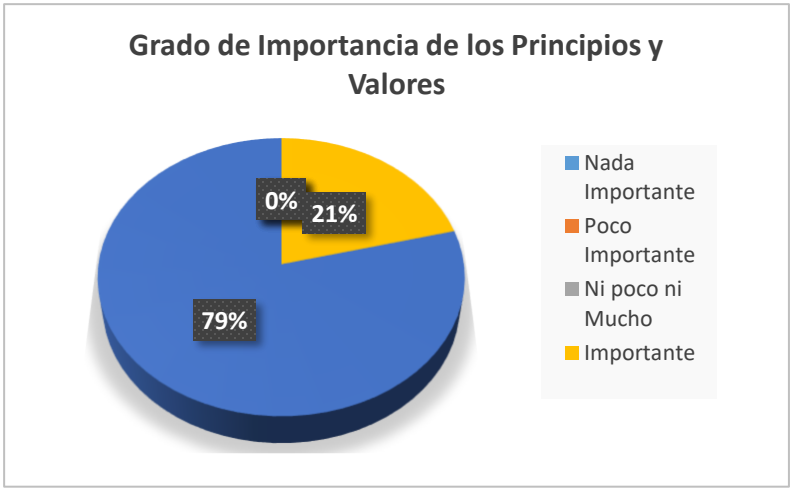

Ilustración 19. Grado de Importancia de los Principios y Valores
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 19. El 79% de los estudiantes encuestados mencionaron que es muy importancia los valores en los equipos de trabajo. El 21% indicaron que es importante los principios y valores en los equipos de trabajo.

4.2 Razones de la importancia de los principios y valores en los miembros del equipo de desarrollo.

Docentes:

a. Buscamos la formación de profesionales con responsabilidad y ética
b. Motivación y predisposición por cumplir objetivos planteados
c. Responsabilidad, ética, compromiso, etc
d. Por tener trabajo en equipo

Estudiantes:

a. Creación
b. Responsabilidad para trabajar juntos con la finalidad de tener un buen producto final
c. Son muchas
d. el interés y ganas de aprender algo nuevo Aprender
e. La responsabilidad y la lealtad al tener en cuenta que la idea de proyecto es de autora compartida, pero si alguien desea utilizar el sistema individualmente haga conocer para evitar plagios.
f. Hay estudiantes que saben poco y no comprenden bien a cuando se programa
   Cada uno tiene un punto de vista y manera de ver las cosas, con ellos se puede tomar varios puntos en cuenta y se puede mejorar detalles del proyecto

1. Lineamientos metodológicos para el desarrollo de proyectos de software en el plan de estudios (Docentes).

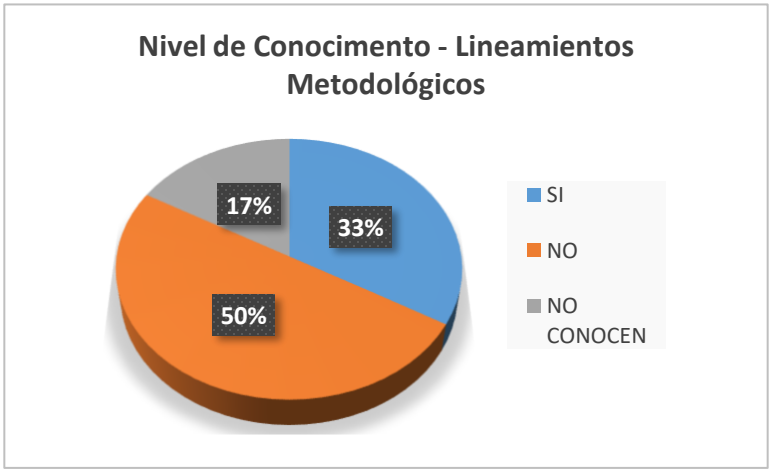

Ilustración 20. Nivel de Conocimiento-Lineamientos Metodológicos
Fuente: Encuesta-Docentes

De acuerdo a la Ilustración 20. El 17% de los docentes encuetados mencionaron no tener conocimiento de los lineamientos metodológicos en el plan de estudios. El 33% de los docentes

saben que existen los lineamientos metodológicos en el plan de estudios. El 50% indicaron que no existe los lineamientos metodológicos en el plan de estudios.

2. Nivel de Conocimiento-Lineamientos Metodológicos (estudiantes)

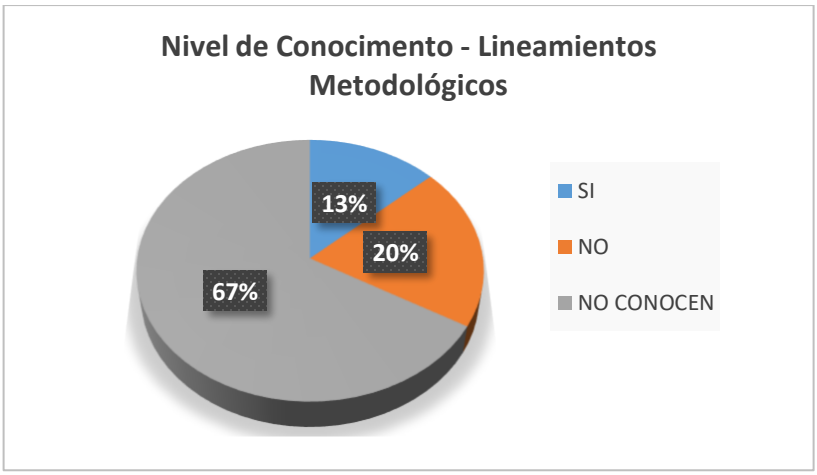

Ilustración 21. Nivel de Conocimiento-Lineamientos Metodológicos
Fuente: Encuesta-Estudiantes

De acuerdo a la Ilustración 21. El 67% de los estudiantes encuestados mencionaron no tener conocimiento de los lineamientos metodológicos en el plan de estudios. El 20% aseguran que no existen los lineamientos metodológicos en el plan de estudios. El 13% indicar que si conocen y saben que existe los lineamientos metodológicos en el plan de estudios.

4.3 Resultados de la Revisión documental

En esta sección se realizó la revisión documental a los proyectos formativos con el apoyo de una guía de observación como instrumento de recolección de datos.

Los resultados se describen a continuación:

1. Propone una metodología de desarrollo de software en el proyecto formativo:
   El 70% de los docentes proponen una metodología de desarrollo de software convencional como es el ciclo de vida en cascada. El 30% de los docentes proponen otras metodologías no convencionales como el RUP, Programación Extrema(XP), Scrum.

2. Describe las etapas de la metodología de desarrollo de software propuesto en la ruta formativa
   Las etapas de las metodologías de desarrollo están descritas parcialmente en los proyectos formativos algunos son condicen con el método de proceso formativo.

3. Describe los métodos o técnicas de análisis y diseño en el proyecto formativo
   Los métodos o técnicas para el análisis y diseño de sistemas están mencionados en los proyectos formativos, las mismas no se instrumentalizan mediante manuales o guías para su correcto uso por parte de los estudiantes.

4. Describe las herramientas de software, que se empleará para abordar el análisis, el diseño y la implementación del proyecto.
   Las herramientas son ampliamente descritas en los proyectos formativos, con referencias o enlaces a manuales y video tutoriales para los estudiantes.

5. En el proyecto formativo se puede evidenciar que existe coherencia entre las metodologías de desarrollo y la asignatura impartida en el semestre.
   Al respecto se pudo evidenciar que parcialmente está relacionado la metodología de desarrollo propuesto por el docente con la asignatura impartida.

6. En el proyecto formativo se puede evidenciar los plazos y los entregables en la planificación del proyecto formativo.
   Los plazos están correctamente descritos de acuerdo a la ruta formativa, sin embargo, existe ciertas ambigüedades en los entregables que deben desarrollar los estudiantes al encarar los proyectos de software.

7. Existe indicios de que el docente haga trabajo conjunto con otro docente de asignatura afín o similar.
   Los docentes proponen en los proyectos formativos acuerdos internos con otras asignaturas para encarar proyectos conjuntos, sin embargo, en la práctica no se concreta dichos acuerdos.

4.4. Análisis e Interpretación de resultados

1. Asignaturas de Ingeniería de Software que Imparten los docentes de la Carrea de Ingeniería de Sistemas.

El plan de estudios (2018) de la carrera de Ingeniería de Sistemas está estructurado por familias laborales y áreas de conocimiento de acuerdo a la sectorial del 2016. Las familias laborales son:

- Sistemas/Tecnologías de la Información y Comunicación

- Modelación y Optimización de Sistemas
- Dirección y Gestión Empresarial

Dentro de la familia labora de Sistemas/ Tecnologías de la Información y Comunicación, se encuentra la disciplina de la Ingeniería de Software y por ende las asignaturas que tributan, las cuales ha sido objeto de estudio, que se mencionan a continuación:

- PROGRAMACIÓN I
- PROGRAMACIÓN II
- PROGRAMACIÓN III
- INGENIERÍA DE SOFTWARE I
- INGENIERÍA DE SOFTWARE II
- ESTRUCTURA DE DATOS
- SISETMA DE INFORMACIÓN II
- TECNOLOGÍAS EMERGENTES
- SISETMA DE INFORMACIÓN I
- BASE DE DATOS I
- BASE DE DATOS II
- SISTEMAS DE INFORMACIÓN GEOGRÁFICA I

2. Metodología de desarrollo de software para encarar los proyectos de software

De la Figura 1 y Figura 2, los estudiantes indicaron aplicar en sus proyectos de software la metodología convencional con el 54% frente a lo que indicaron los docentes con el 34%.

El 50% de los docentes mencionaron encarar los proyectos de software con la metodología del proceso unificado en contra de lo que los estudiantes mencionaron con tan solo el 27%.

De acuerdo a las características de ambas metodologías la primera (ciclo de vida en cascada) en principio es lineal y con enfoque estructurado. Si bien el proceso unificado de desarrollo es orientada objetos, roles y responsabilidades con un procesos iterativo e incremental, la desventaja es que es muy burocrático y no aplica para proyectos pequeños como son los proyectos de asignatura.

De acuerdo a los resultados de las encuestas realizadas a docentes, estudiantes y la revisión documental a los proyectos formativos e informes de los docentes, se pudo evidenciar que efectivamente las metodologías tradicionales son las más empleadas,

seguida por las metodologías como el RUP. El resto de las metodologías son aplicadas en porcentajes menores.

3. Criterios de selección de una metodología de desarrollo de software.

   De las encuestas realizadas a docentes y estudiantes ambos grupos de estudio coincidieron en un porcentaje elevado con relación al criterio del Enfoque o paradigma, con un 53% y 62% respectivamente. Dándose menor importancia a los criterios de los métodos, técnicas y tamaño del proyecto y nivel semestral de asignatura, toda vez que estos aspectos son más importantes a considerar en el desarrollo de los proyectos de softwares.

4. Técnicas de modelamiento de análisis y diseño de sistema

   De las encuestas realizadas a docentes y estudiantes, ambos grupos de estudio coincidieron con relación a las técnicas de modelamiento de análisis y diseño de sistemas, con un 26% y 32% respectivamente. Con 19% y 21% en el uso y aplicación de los casos de uso respectivamente. Y en porcentajes menores el resto de las técnicas de modelamiento de análisis y diseño de sistemas.

   Estos resultados se pudieron corroborar con la revisión documento de los proyectos formativos del docente que imparten las asignaturas de la disciplina de Ingeniería de Software.

5. Herramientas CASE en el Desarrollo de Software

   Enterprise Architect es la herramienta CASE favorita tanto de docentes como de estudiantes con el 29% y 42% respectivamente. Este dato también se pudo corroborar en la revisión documento de los proyectos formativos. Enterprise Architect actualmente es una de las herramientas de software más completas para el análisis y diseño de sistemas soporta UML y otras notaciones o especificaciones de modelamiento.
   Le siguen otras herramientas de software como el bisagi, Power Design para el modelamiento de las

6. Herramientas de Software para el control de versiones

   De acuerdo a la Figura 9 y Figura 10. Los docentes y estudiantes utilizan git y gitlab para el control de versiones, además de otras herramientas de software. Sin embargo revisado los proyectos formativos e informes no se puedo evidenciar esta información.

7. Organización de los equipos de trabajo para el desarrollo del proyecto de software

   Tradicionalmente los equipos de trabajo se han organizado por afinidad, sin tomar en cuenta el perfil del estudiante referido a sus competencias (habilidades y destrezas). Otro

aspecto muy importante también es considerar los roles y las responsabilidades que van a asumir en el desarrollo del proyecto.

8. Grado de importancia de los Principios y valores del equipo de trabajo

   Metodologías traiciónales e incluso metodologías actuales no consideran para nada los principios y los valores que deben tener los desarrolladores. A propósito, fue la pregunta ya que metodologías ágiles como e Scrum y XP consideran este aspecto muy importante.

9. Razones de la importancia de los principios y valores en los miembros del equipo de desarrollo.

   Tanto los encuestado de docentes como estudiantes consideran que existe suficientes razones de la importancia de los principios y valores en los miembros del equipo de desarrollo.

10. Lineamientos metodológicos para el desarrollo de proyectos de software en el plan de estudios

    Aunque ambos grupos de estudio mencionaron tener conocimiento de la existencia de los lineamientos metodológicos en el plan de estudios. Sin embargo, en la revisión documental se observó que no cuentan con los lineamientos metodológicos en el plan de estudios (2018).

4.5. Propuesta

Estrategia Metodológica Para La Articulación De Las Metodologías De Desarrollo De Software En La Asignaturas De Ingeniería De Software De La Carrea De Ingeniería De Sistemas De La Universidad Amazónica De Pando.

4.5.1. Introducción

El concepto de estrategia con paso del tiempo ha tenido muchas usos y aplicaciones diversas, desde el campo militar, en el cual sostiene su origen, pasando por el ámbito político, administrativo, económico, educativo, religioso, cultural y social; en cada uno de ellos se ubica como un referente por la forma en que ha sido utilizada.

En el ámbito educativo la estrategia metodológica se refiere a aquellos principios y criterios, que mediante métodos, técnicas y procedimientos que constituyen una secuencia ordenada y planificada de actividades educativas, que permiten la construcción de conocimientos durante el proceso de enseñanza y aprendizaje.

Elaborar esta propuesta de estrategia metodológica, implicó en principio indagar los referentes teóricos con relación al objeto de estudio (las metodologías de desarrollo de software) y posteriormente un trabajo de campo que se hizo de manera práctica y concreta en el contexto de la Carrera de Ingeniería de Sistemas de la UAP, para responder al vacío de conocimiento planteado en el diseño teórico de la Investigación.

La propuesta presentada aquí, son las directrices o lineamientos, que permitirá al docente de la carrera de Ingeniería de Sistemas en la disciplina de la Ingeniería de Software en la familia laboral de Sistemas/Tecnologías de Información y Comunicación, contar con una guía o estrategia metodológica para el abordaje de su asignatura principalmente en la etapa de planificación y ejecución del proceso formativo.

4.5.2. Diagnóstico

Los resultados recabados en el trabajo de campo nos arrojan evidencias contundentes respecto a las metodologías de desarrollo de software utilizadas, por docentes y estudiantes en el proceso de enseñanza y aprendizaje.

Las metodologías utilizadas en cada semestre y asignatura no guardan una relación y coherencia correcta. Se considera que estas deben estar respaldadas y fundamentadas, por las técnicas de análisis/diseño y sus herramientas de programación.

Los referentes teóricos con relación a las metodologías de desarrollo de sistemas y sus inherentes técnicas de análisis y diseño de sistemas, están en permanente progreso, y una actualización constante de versión a versión año tras año de las herramientas de software para

la implementación, con prestaciones cada vez más versátiles, donde el esfuerzo de programación (codificación manual) se reduce sustancialmente.

4.5.3. Objetivo General

El objetivo general de la estrategia metodológica es:

Contribuir con una guía de acción de articulación metodológica de las metodologías de desarrollo de software en las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas.

4.5.3.1 Objetivos Específicos

- Caracterizar las metodologías de desarrollo de software, que le sirva de base o punto de partida para el docente que impartirá alguna asignatura relacionada a la Ingeniería de Software.

- Establecer en orden y prioridad en las metodologías de desarrollo de software, sus etapas, los métodos para el proceso formativo, las técnicas y herramientas de modelamiento y programación de sistemas.

- Describir los componentes de la propuesta de estrategia metodológica

4.6 Planeación Estratégica

Tabla 4.

Planeación estratégica de la Propuesta Metodológica

Asignatura	Semestre	Metodología de Desarrollo de Software	Etapas/Fases	Métodos para el Proceso Formativo	Técnicas y Herramientas de Modelado y Programación
PROGRAMACIÓN I	Primero	Metodología de la Programación Orientada a Objetos con Análisis Descendente y Ascendente: Fundamentos	Análisis Diseño Codificación(Implementación) Ejecución, verificación y depuración Mantenimiento Documentación	Aprendizaje Basado en Problemas	Diagramas de Flujo Pseudocódigo PSEINT Java
PROGRAMACIÓN II	Segundo	Metodología de la Programación Orientada a Objetos	Análisis Orientada a Objetos Diseño Orientada a Objetos Implementación OO Pruebas Orientadas a Objetos Mantenimiento Documentación	Aprendizaje Basado en Problemas	Lenguaje Unificado de Modelado Enterprise Architect Java
PROGRAMACIÓN III	Tercero	Metodología de la Programación Cliente / Servidor	Análisis Orientada a Objetos Diseño Orientada a Objetos Implementación OO Pruebas Orientadas a Objetos Mantenimiento Documentación	Aprendizaje Basado en Problemas	Lenguaje Unificado de Modelado Modelo Vista Controlador(MVC) Enterprise Architect J2EE
ESTRUCTURA DE DATOS	Cuarto	Metodología de la Programación Orientada a Objetos	Análisis Orientada a Objetos Diseño Orientada a Objetos Implementación OO Pruebas Orientadas a Objetos Mantenimiento Documentación	Aprendizaje Basado en Problemas	Lenguaje Unificado de Modelado Java
TECNOLOGÍAS EMERGENTES	Cuarto	Metodología para el Desarrollo de Aplicaciones Móviles(Mobile-D)	Exploración Iniciación Producto	Aprendizaje Basado en Problemas	Experiencia de usuarios Android

			Estabilización Pruebas		jQuery Mobile
▪ **BASE DE DATOS I** ▪ **SISTEMA DE INFORMACIÓN I**	Cuarto Quinto	Programación Extrema(XP)	Planificación Diseño Codificación Pruebas Lanzamiento	Aprendizaje Basado en Proyectos	Modelamiento de Procesos de Negocio(BPM) Visagi Enterprise Java, PHP o C#
▪ **BASE DE DATOS II** ▪ **SISTEMA DE INFORMACIÓN II**	Quinto Sexto	Metodología SCRUM	Inicio Planificación y estimación Implementación Revisión y Retrospectiva Lanzamiento	Aprendizaje Basado en Proyectos	Modelamiento de Procesos de Negocio(BPM) Visagi Enterprise Java, PHP o C#
SISTEMAS DE INFORMACIÓN GEOGRÁFICA	Sexto	SCRUM-Xtreme Programming (SXP)	Planificación Definición Desarrollo Mantenimiento	Aprendizaje Basado en Proyectos	ArGIS QGIs
INGENIERÍA DE SOFTWARE I	Sexto	Metodología RUP	Inicio Elaboración Construcción Transición	Aprendizaje Basado en Proyectos	Desarrollo basado en componentes Flujos de Trabajo JUnit
INGENIERÍA DE SOFTWARE II	Séptimo	Metodología RUP	Inicio Elaboración Construcción Transición	Aprendizaje Basado en Proyectos	Desarrollo basado en componentes Flujos de Trabajo JUnit

Fuente: Elaboración propia, 2022.

La tabla 4, describe en orden y prioridad de cómo se deben desarrollar las asignaturas de la disciplina de Ingeniería de Software en los diferentes semestres. Para cada una de ellas se ha propuesto la metodología de desarrollo de software, las etapas o fases a seguir, el método para el proceso formativo y las técnicas y herramientas de modelamiento.

Caracterización de las metodologías de desarrollo:

A continuación, se describen las metodologías de desarrollo de software como parte de la propuesta metodológica.

4.7 Metodología para el Desarrollo de Aplicaciones Móviles(Mobile-D)

Una metodología de desarrollo nueva, especialmente diseñada para el desarrollo de aplicaciones móviles, recibe el nombre de Mobile-D y es propuesta por Pekka Abrahamsson y su equipo del VTT (Valtion Teknillinen Tutkimuskeskus, en inglés Technical Research Centre of Finland) en Finlandia.

El ciclo del proyecto se divide en cinco fases:

1. Exploración,
2. Inicialización,
3. Productización,
4. Estabilización y
5. Prueba del sistema.

En general, todas las fases (con la excepción de la primera fase exploratoria) contienen tres días de desarrollo distintos: planificación, trabajo y liberación. Se añadirán días para acciones adicionales en casos particulares (se necesitarán días para la preparación del proyecto en la fase de inicialización).

4.7.1 Metodología SCRUM

Scrum (melé en español) es un modelo de desarrollo ágil que, aunque surgió como modelo para el desarrollo de productos tecnológicos, es empleada en entornos que trabajan con requisitos inestables y que requieren rapidez y flexibilidad, como el desarrollo de software. Está centrada principalmente en la gestión del equipo de desarrollo, su enfoque es iterativo e incremental. Esta metodología ayuda a los equipos a desarrollar productos en periodos cortos, permitiendo una rápida retroalimentación(feedback) del cliente, adaptaciones y mejora continua.

Las fases del Scrum se distribuyen en 16 procesos o tareas, que se resumen en 5 etapas de implementación:

1. Inicio
2. Planificación y estimación
3. Implementación
4. Revisión y retrospectiva
5. Lanzamiento

1. Inicio

La primera fase se encarga de estudiar y analizar el proyecto identificando las necesidades básicas del sprint. En el contexto de las metodologías ágiles, un sprint es un mini-proyecto con una duración no mayor a un mes que se interconecta con otros mini-proyectos. Para esta fase se debe hacer las siguientes preguntas: ¿Qué quiero? ¿Cómo lo quiero? y ¿Cuándo lo quiero? - Esta metodología da preferencia a la formación de equipos pequeños de mínimo 3 y máximo 5 personas, pues se facilita la fluidez de las ideas y se aporta creatividad al grupo. Los primeros pasos de Scrum son 6 procesos:

- ✓ Crear la visión del proyecto
- ✓ Identificar a los Master Scrum o ScrumMaster y a los stakeholders.
- ✓ Formar equipos Scrum
- ✓ Desarrollar épicas

- ✓ Crear backlogs o listas de requerimientos priorizando el producto
- ✓ Planificar el lanzamiento

2. Planificación y estimación:

La segunda fase de Scrum incluye normalmente los siguientes pasos:

- ✓ Crear, estimar y comprometer historias de usuario.
- ✓ Identificar y estimar tareas.
- ✓ Crear el sprint backlog o iteración de tareas.

La clave para llevar una buena administración de los proyectos es hacer una planificación y estimación del sprint, lo que ayudará a establecer metas fijas y a cumplir con los plazos. Esta fase se considera como la más importante del proyecto, pues el Master Scrum tendrá que delegar las tareas correspondientes a cada grupo y hacer las estimaciones de tiempos de entrega, así como crear una lista ordenada para clasificar el trabajo según su prioridad.

3. Implementación

En esta etapa nos encontramos en la sala de reuniones donde se discute el sprint y se explora cómo optimizar el trabajo de cada grupo Scrum para darle forma definitiva al proyecto. En la implementación se cumple con los siguientes procesos:

- ✓ Crear entregables.
- ✓ Realizar daily stand-up.
- ✓ Refinanciamiento del backlog priorizado del producto.

En la fase de implementación o desarrollo no deberían hacerse cambios innecesarios de última hora (se supone que para evitarlo existe una fase de planificación). Aun así, si de ser necesario hacer un movimiento que será clave para el éxito del sprint, se debe proceder.

4. Revisión y retrospectiva

Una vez que ya todo está maquetado e implementado, deberás hacer la revisión del proceso, que no es más que la autocrítica o evaluación interna del grupo respecto a su propio trabajo. Es importante sumar opiniones constructivas y aportar soluciones viables. Entre los pasos más importantes para realizar en esta fase tenemos:

- ✓ Demostrar y validar el sprint.
- ✓ Retrospectiva del sprint.
- ✓ Lanzamiento
- ✓ La última de las fases del método Scrum es el lanzamiento.

Con esto nos referimos al desenlace del proyecto y entrega del producto, donde deberías cumplir con 2 únicas tareas que son:

- ✓ Enviar entregables.
- ✓ Enviar retrospectiva del proyecto.

5. Lanzamient**o**

La última de las fases del método Scrum es el lanzamiento. Que consiste en la entrega del producto, ello implica cumplir 2 únicas tareas:

- ✓ Enviar entregables.
- ✓ Enviar retrospectiva del proyecto.

4.7.2 SCRUM-Xtreme Programming (SXP)

En 2001, 17 profesionales de la ingeniería del software creaban el Manifesto Ágil con el objetivo de establecer los principios para desarrollar software de forma rápida y adaptable a cambios, ofreciendo una alternativa a procesos de desarrollo de software tradicionales.

La metodología XP es un conjunto de técnicas que dan agilidad y flexibilidad en la gestión de proyectos. También es conocida como Programación Extrema (Extreme Programming) y se centra crear un producto según los requisitos exactos del cliente. De ahí, que le involucre al máximo durante el método de gestión del desarrollo del producto. La primera vez que oímos este tipo de metodología fue a través del libro Extreme Programming Explained: Embrace Change (1999), escrito por el ingeniero de software Kent Beck.

El uso de esta metodología supone, para muchos teóricos, una aproximación a la calidad óptima del producto. Pues durante el ciclo de vida del software, ocurren cambios naturales. Se dice cuantos más cambios, puede que más cerca estemos del mejor resultado que espera nuestro cliente. Por lo tanto, el cambio constante en el proyecto se considera como favorable.

Las características de la metodología XP son:

- ✓ Comunicación constante entre el cliente y el equipo de desarrollo.
- ✓ Respuesta rápida a los cambios constantes.
- ✓ La planificación es abierta con un cronograma de actividades flexible.
- ✓ El software que funciona está por encima de cualquier otra documentación.
- ✓ Los requisitos del cliente y el trabajo del equipo del proyecto son los principales factores de éxito del mismo.

Las fases de la metodología son:

1. Planificación
2. Diseño

3. Codificación
4. Pruebas
5. Lanzamiento

Fase 1: Planificación

En esta fase, según la identificación de las historias de usuario, se priorizan y se descomponen en mini-versiones. La planificación se va a ir revisando. Cada dos semanas aproximadamente de iteración, se debe obtener un software útil, funcional, listo para probar y lanzar.

Fase 2: Diseño

En este paso se intentará trabajar con un código sencillo, haciendo lo mínimo imprescindible para que funcione. Se obtendrá el prototipo. Además, para el diseño del software orientado a objetos, se crearán tarjetas CRC (Clase-Responsabilidad-Colaboración).

Fase 3: Codificación

La programación aquí se hace "a dos manos", en parejas en frente del mismo ordenador. Incluso, a veces se intercambian las parejas. De esta forma, nos aseguramos que se realice un código más universal, con el que cualquier otro programador podría trabajar y entender. Y es que debe parecer que ha sido realizado por una única persona. Así se conseguirá una programación organizada y planificada.

Fase 4: Pruebas

Se deben realizar pruebas automáticas continuamente. Al tratarse normalmente de proyectos a corto plazo, este testeo automatizado y constante es clave. Además, el propio cliente puede hacer pruebas, proponer nuevas pruebas e ir validando las mini-versiones.

Fase 5: Lanzamiento

Para esta fase se ha probado todas las historias de usuario o mini-versiones con éxito, ajustándonos a los requerimientos del cliente. Tenemos un software útil y podemos incorporarlo en el producto.

4.7.3 Metodología RUP

La metodología de desarrollo RUP por sus siglas en inglés ó Proceso de Desarrollo Unificado es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

La metodología RUP puede utilizarse, desde el principio de un nuevo proyecto de software, y puede seguir utilizándolo en los ciclos de desarrollo subsiguientes tiempo después de que el proyecto inicial haya terminado. No obstante, la forma de utilizar RUP varía para ajustarse a sus necesidades.

Características del RUP

- ✓ Iterativo e Incremental, El Proceso Unificado es un marco de desarrollo iterativo e incremental compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en flujos de trabajo: Análisis de requisitos, Diseño, Implementación y Prueba.

- ✓ Dirigido por los casos de uso, Los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración

tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas: diseño, implementación, prueba, etc.

- ✓ Centrado en la arquitectura, El Proceso Unificado asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de software de un sistema. La analogía con la construcción es clara, cuando construyes un edificio existen diversos planos que incluyen los distintos servicios del mismo: electricidad, fontanería, etc.

- ✓ Enfocado en los riesgos, El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

Fase: Inicio

La fase de inicio, es la fase más pequeña del proyecto e, idealmente, debe realizarse también en un periodo de tiempo pequeño (una única iteración).

Los objetivos que persigue esta fase son:

- ✓ Establecer una justificación para el proyecto.
- ✓ Establecer el ámbito del proyecto.
- ✓ Esbozar los casos de uso y los requisitos clave que dirigirán las decisiones de diseño.
- ✓ Esbozar las arquitecturas candidatas.
- ✓ Identificar riesgos.
- ✓ Preparar el plan del proyecto y la estimación de costes.

Fase: Elaboración

Durante esta fase se capturan la mayoría de los requisitos del sistema. Los objetivos principales de esta fase serán la identificación de riesgos y establecer y validar la arquitectura del sistema. La arquitectura se valida a través de la implementación de una Base de Arquitectura. Al final de la fase de elaboración la base de arquitectura ejecutable debe demostrar que soporta los aspectos clave de la funcionalidad del sistema y que muestra la conducta adecuada en términos de rendimiento, escalabilidad y coste. Al final de la fase se elabora un plan para la fase de construcción.

Fase: Construcción

- ✓ Es la fase más larga de proyecto.
- ✓ El sistema es construido en base a lo especificado en la fase de elaboración.
- ✓ Las características del sistema se implementan en una serie de iteraciones cortas y limitadas en el tiempo.
- ✓ El resultado de cada iteración es una versión ejecutable de software.
- ✓ El hito de capacidad operativa inicial marca el final de la fase.

Fase: Transición

- ✓ En esta fase el sistema es desplegado para los usuarios finales.
- ✓ La retroalimentación recibida permite incorporar refinamientos al sistema en las sucesivas iteraciones.
- ✓ Esta iteración también cubre el entrenamiento de los usuarios para la utilización del sistema.
- ✓ El hito de lanzamiento del producto marca el final de la fase.

CAPÍTULO V

CONCLUSIONES Y RECOMENDACIONES

CONCLUSIONES

Al culminar este proceso investigativo, cabe mencionar que se ha cumplido con el objetivo general de Diseñar una estrategia metodológica para la articulación de las metodologías de desarrollo de software en las asignaturas de la Ingeniería de Software para la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando, puntualizando las siguientes conclusiones específicos en respuesta al cumplimiento de los objetivos específicos:

- Se ha logrado sistematizar los referentes teóricos de diez metodologías de desarrollo de software convencionales y ágiles, que sirven de base y fundamento para la propuesta de la estrategia metodológica.

- Se hizo el análisis del objeto de estudio en el contexto de la carrera de Ingeniería de Sistemas, tomando en cuenta las unidades de análisis de docentes, estudiantes y proyectos formativos e informes con relación a las asignaturas de la Ingeniería de Software. Encontrándose debilidades en su aplicación y poca coherencia en los diferentes semestres académicos.

- Se ha Determinado los componentes y los elementos esenciales e imprescindibles que están integrados en el diseño de la estrategia metodológica propuesta, para las asignaturas de Ingeniería de Software de la carrera de Ingeniería de Sistemas de la Universidad Amazónica de Pando.

RECOMENDACIONES

A partir de esta investigación y la propuesta se recomienda profundizar en las especificidades que involucra las metodologías de desarrollo de software, sus procesos, sus técnicas, prácticas y herramientas tecnológicas inherentes de la disciplina de la Ingeniería de Software.

REFERENCIAS BIBLIOGRÁFICAS

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R. C., Mellor, S., Schwaber, K., Sutherland, J., & Thomas, D. (2001). *Manifiesto para el desarrollo de software ágil*. https://agilemanifesto.org/

De San Martín Oliva, C. R. P. (2009). Metodología ICONIX. *Portal Huerpa*, 1–7. http://www.portalhuarpe.com.ar/Seminario09/archivos/MetodologiaICONIX.pdf

Iglesias, M., & Cuiñas, I. (2017). *Scrum – DESIRE*. https://desire.webs.uvigo.es/contenidos/scrum/

Jacobson, I., Booch, G., & Rumbaug, J. (2006). *El Lenguaje Unificado de Modelado - Guia de Usuario*.

Rios, J. S. (2015). *Implementación del método Kanban en las empresas constructoras pequeñas y medianas en la ejecución de un proyecto en Colombia*. https://riunet.upv.es/bitstream/handle/10251/51733/MEMORIA_TFM Pinto de los Rios Juan Sebastian.pdf?sequence=1

Serna, E., & Candidato, M. (2010). *Métodos formales e Ingeniería de Software Formal Methods and Software Engineering Méthodes formelles et génie logiciel*. *30*, 158–184. http://revistavirtual.ucn.edu.co/index.php/RevistaUCN/article/viewFile/62/129

Universitat de Girona. (2013). *Manual Introductorio de ICONIX*. 5. http://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesICONIX.pdf%0Ahttp://ima.udg.edu/~sellares/EINF-ES2/%0Ahttp://ima.udg.edu/~sellares/EINF-ES2/Present1011/MetodoPesadesICONIX.pdf%0Ahttp://ima.udg.edu/~sellares/EINF-ES2/

V.3, Pa.-M. (n.d.). *Portal de Administración Pública del Gobierno de España*. Retrieved December 7, 2021, from https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html

Canós, J. H., & Letelier, M. C. P. P. (2012). Metodologías ágiles en el desarrollo de software.

Rivas, C. I., Corona, V. P., Gutiérrez, J. F., & Hernández, L. (2015). Metodologías actuales de desarrollo de software. *Revista de Tecnología e Innovación*, *2*(5), 980-986.

Maida, E. G., & Pacienzia, J. (2015). Metodologías de desarrollo de software.

Calero, C., & Velthuis, M. G. P. (2010). *Calidad del producto y proceso software*. Editorial Ra-Ma.

Dimes, T. (2015). *Conceptos Básicos de Scrum: Desarrollo de software Agile y manejo de proyectos Agile*. Babelcube Inc..

Grupo, I. S. S. I. (2003). Metodologías ágiles en el desarrollo de software. *Taller Metodologías Ágiles en el Desarrollo de Software. Obtenido de http://metasoso-2017. blogspot. com/2017/02/ensayo-1. html.*

Jacobson, I., Booch, G., & Rumbaug, J. (2006). *El Lenguaje Unificado de Modelado - Guia de Usuario*.

Jacobson, I., Booch, G., & Rumbaug, J. (2000). *El Lenguaje Unificado de Modelado Manual de Referencia.pdf*. PEARSON EDUCACIÓN.

Maida, E. G., & Pacienzia, J. (2015). *Metodologías de desarrollo de software*. https://repositorio.uca.edu.ar/handle/123456789/522

Weitzenfeld, A. (2005). *Ingeniería de software orientada a objetos con UML, Java e Internet*. http://books.google.com/books?id=MOviEp0ApQcC&pgis=1

Jacobson, I., Booch, G., & Rumbaug, J. (2000). *El Proceso Unificado de Desarroll(Ivar, Jacobson, Rumbaug).pdf* (p. 464). PEARSON EDUCACIÓN.

Kendall, K., & Kendall, J. (2011). *Análisis y Diseños de Sistemas* (Octava Edi). Pearson Educación. http://cotana.informatica.edu.bo/downloads/ld-Analisis y Diseno de Sistemas_Kendall-8va.pdf

Sommerville, I. (2011). *Ingeniería de Software* (Novena Edi). PEARSON EDUCACIÓN.

James, A. S. (1991). *Análisis y Diseño de Sistemas de Información* (Segunda Ed). Mc Graw Hill.

Kendall, K. (2005). *Análisis Y Diseño De Sistemas* (Sexta Edic). Pearson Educación.

Pressman, R. S. (2013). *Ingeniería de Software un enfoque práctico* (Séptima Ed, Vol. 53, Issue 9). Mc Graw Hill Educación.

ANEXOS

ANEXO 1

Lista de docentes de la carrera de Ingeniería de Sistemas

#	NOMBRE COMPLETO
1.	MSC. ING. LUDWING ARCIENEGA BAPTISTA
2.	ING. MAYKO ANTEZANA SOSA
3.	ING. ALEX ARGANDOÑA CRISPIN
4.	ING. MSC. SAMUEL FUENTES CHAMBI
5.	LIC. MSC. JUAN CARLOS HUANCA GUANCA
6.	LIC. EDUARDO ALBERTO ZUBIETA COPETICON
7.	ING. MSC. NELSON CHOCLO RUBIN DE CELIS
8.	ING. MIREYA MONJE ASCARRUNZ
9.	ING. JUAN CARLOS GALLARDO JIMENEZ
10.	LIC. JAVIER PATTY MAGNE
11.	ING. ABEL HUAYGUA CHALCO
12.	ING. JOSE BALDERRAMA MENDEZ
13.	ING. MSC. CHRISTIAN MIAHUCHI NATALY
14.	ING. MANUEL LOPEZ RENGIFO
15.	ING. MSC. FREDDY MORALES BLANCO
16.	LIC. MSC. HUMBERTO FERNANDEZ CALLE
17.	ING. PEDRO VÁSQUEZ PEREZ
18.	ING. JOSÉ AILTON SUÁREZ REBOSO
19.	ING. OMAR CALIZAYA QUIÑONEZ
20.	ING. ALEX YANAHUAYA ARCE

21.	ING. RENE EMIGDIO YANA CHOQUE
22.	ING. EDWIN MAMANI HUANCA
23.	ING. JOAQUIN ESTEBAN PLATA GUTIERREZ
24.	LIC. VICTOR MANUEL TRUJILLO SUÁREZ

Fuente: Elaboración propia, en base a documentos normativos designación docente.

ANEXO 2.

Docentes que imparten asignaturas de la disciplina de Ingeniería de Software

#	ASIGNATURAS	DOCENTE
1.	PROGRAMACIÓN I	ING. FREDDY MORALES BLANCO
2.	PROGRAMACIÓN II	LIC EDUARDO ZUBIETA COPETICON
3.	PROGRAMACIÓN III INGENIERÍA DE SOFTWARE I INGENIERÍA DE SOFTWARE II	LIC. JUAN CARLOS HUANCA GUANCA
4.	ESTRUCTURA DE DATOS SISETMA DE INFORMACIÓN II	LIC. JAVIER PATTY MAGNE
5.	TECNOLOGÍAS EMERGENTES SISETMA DE INFORMACIÓN I	ING. SAMUEL FUENTES CHAMBI
6.	BASE DE DATOS I BASE DE DATOS II	ING. JOSE BALDERRAMA MENDEZ
7.	SISTEMAS DE INFORMACIÓN GEOGRÁFICA I	ING. RENÉ YANA CHOQUE

Fuente: Elaboración propia, en base a documentos normativos y plan de estudios 2018.

ANEXO 3.

Guía de Observación

Tema/Objeto de Estudio: Metodologías del Desarrollo de Software en los PF
Docente:
Asignatura:
Semestre:

6. Propone una metodología de desarrollo de software en el proyecto formativo:

7. Describe las etapas de la metodología de desarrollo de software propuesto en la ruta formativa

8. Describe los métodos o técnicas de análisis y diseño en el proyecto formativo

9. Describe las herramientas de software, que se empleará para abordar el análisis, el diseño y la implementación del proyecto.

10. En el proyecto formativo se puede evidenciar que existe coherencia entre las metodologías de desarrollo y la asignatura impartida en el semestre.

11. En el proyecto formativo se puede evidenciar los plazos y los entregables en la planificación del proyecto formativo.

12. Existe indicios de que el docente haga trabajo conjunto con otro docente de asignatura afín o similar.

ANEXO 4.

Encuesta a Docentes

Estimado señor docente le solicito muy encarecidamente pueda colaborarme con la siguiente encuesta. Datos que serán utilizados para realizar una investigación para la carrera de Ingeniería de Sistemas bajo la modalidad de graduación. Muchas Gracias.

1. ¿Indique qué asignatura de la disciplina de la Ingeniería de software, ha impartido en los últimos 5 años en sus funciones de docente universitario en la carrera de Ingeniería de Sistemas de la UAP? Solo una en caso de haber impartido varios.

R.

2. De acuerdo a la asignatura mencionada en el Ítem anterior. ¿Qué metodología de desarrollo de software Ud., a empleado para encarar los proyectos de software con sus estudiantes?

a) Ciclo de Vida en Cascada
b) Modelo Espiral
c) Procesos Unificado de Desarrollo
d) Programación Extrema
e) Scrum
f) Otras……

3. ¿Cuáles son los criterios que Ud. Utiliza para seleccionar una metodología de desarrollo de software al encarar los proyectos de software al impartir su asignatura?

a) Enfoque o paradigma
b) Los métodos y Técnicas
c) Tamaño del proyecto
d) Nivel Semestral de la asignatura
e) Las etapas de la metodología
f) Otra……

4. De acuerdo a la metodología seleccionada. ¿Cuáles son las técnicas de modelamiento de análisis y diseño de sistemas, es la que sugiere a los estudiantes en el desarrollo del proyecto de software?

a) Diagrama de procesos
b) Diagramas de Flujo
c) Diagrama entidad relación
d) Modelo Relacional de Datos
e) Árbol de decisiones
f) Casos de uso
g) Lenguaje Unificado de Modelado(UML)
h) Además, Otros……

5. De acuerdo a las técnicas de modelamiento de análisis y diseño de sistemas. ¿Qué herramientas CASE o de software, es la que sugiere que los estudiantes utilicen en el desarrollo del proyecto de software?

a) DFD
b) PSEINT
c) Rational Rose
d) Enterprise Architect
e) Bisagi
f) ArgoUML
g) Power Design
h) Además Otras……

6. ¿Cuáles son las herramientas de software que Ud. Sugiere para el control de versiones en la implementación de proyectos de software?

a) Git
b) GibLab
c) Otro……

7. ¿Cómo organiza los equipos de trabajo para el desarrollo de los proyectos de software en su asignatura?

a) Afinidad
b) Perfil del Estudiante (Habilidades y Destrezas)
c) Roles y Responsabilidades
d) Otras…….

8. Considera que los principios y valores de los miembros del equipo de desarrollo son:

a) Nada Importante
b) Poco Importante
c) Ni poco ni Mucho
d) Importante
e) Muy Importante

9. Con relación a la respuesta del Ítem anterior. ¿Cuáles son las razones?

R.

10. ¿Existen lineamientos en el plan de estudios de la carrera que oriente la aplicación de las metodologías de desarrollo de software en coherencia con el nivel semestral de las asignaturas de la disciplina de Ingeniería de software?

a) Si
b) No
c) No Conoce

ANEXO 5.

Encuesta a Estudiantes

Estimado señor estudiante le solicito muy encarecidamente pueda colaborar con la siguiente encuesta. Datos que serán utilizados para realizar una investigación para la carrera de Ingeniería de Sistemas bajo la modalidad de graduación. Muchas Gracias.

1. ¿Indique qué asignatura de la disciplina de Ingeniería de software, ha cursado en sus últimos años de su formación en la carrera de Ingeniería de Sistemas de la UAP? Solo una en caso de haber tomado varias asignaturas.

R.

2. De acuerdo a la asignatura mencionada en el Ítem anterior. ¿Qué metodología de desarrollo de software Ud., a empleado para encarar su proyecto de software con su equipo de trabajo?

a) Ciclo de Vida en Cascada
b) Modelo Espiral
c) Procesos Unificado de Desarrollo
d) Programación Extrema
e) Scrum
f) Otra, metodología…

3. ¿Cuáles son las orientaciones que el docente realiza para seleccionar una metodología de desarrollo de software al encarar los proyectos de software?

a) Enfoque o paradigma de la metodología
b) Métodos y Técnicas
c) Tamaño del proyecto
d) Nivel Semestral de la asignatura
e) Las etapas de la metodología
f) Otras……

4. De acuerdo a la metodología seleccionada. ¿Cuáles son las técnicas de modelamiento de análisis y diseño de sistemas, es la que utiliza en el desarrollo de su proyecto de software?

a) Diagrama de procesos
b) Diagramas de Flujo
c) Diagrama entidad relación
d) Modelo Relacional de Datos
e) Árbol de decisiones
f) Casos de uso
g) Lenguaje Unificado de Modelado(UML)
h) Otros……

5. De acuerdo a las técnicas de modelamiento de análisis y diseño de sistemas. ¿Qué herramientas CASE o de software, es la que utiliza en el desarrollo su proyecto de software?

a) DFD
b) PSEINT
c) Rational Rose
d) Enterprise Architect
e) Bisagi
f) ArgoUML
g) Power Design
h) Otro……

6. ¿Cuáles son las herramientas de software que utiliza para el control de versiones en la implementación de su proyecto de software?

a) Git
b) GitHub
c) Otro……

7. ¿Cómo se organiza su equipo de trabajo para el desarrollo de los proyectos de software en su asignatura?

a) Afinidad
b) Habilidades y Destrezas de los compañeros
c) Responsabilidades y compromiso de los compañeros
d) Otras…….

8. ¿Considera que los principios y valores de los miembros de su equipo de desarrollo son:

a) Nada Importante
b) Poco Importante
c) Ni poco ni Mucho
d) Importante
e) Muy Importante

9. Con relación a la respuesta del Ítem anterior. ¿Cuáles son las razones?

R.

10. ¿Existen lineamientos en el plan de estudios de la carrera que oriente la aplicación de las metodologías de desarrollo de software en coherencia con el nivel semestral de las asignaturas de la disciplina de Ingeniería de software?

d) Si
e) No
f) No conoce