UNIVERSIDAD ANDINA SIMÓN BOLÍVAR
SEDE CENTRAL
Sucre – Bolivia

MAESTRÍA EN SOFTWARE LIBRE

PROPUESTA DE UN MARCO DE TRABAJO ÁGIL DE DESARROLLO DE SOFTWARE, PARA PROYECTOS DE GRADO, BASADA EN SCRUM, KANBAN Y EXTREME PROGRAMMING

Tesis presentada para optar el Grado Académico de Magíster en Software Libre

MAESTRANTE: NINOSKA SOFÍA DELGADO VACA GUZMÁN

Sucre - Bolivia

2022

UNIVERSIDAD ANDINA SIMÓN BOLÍVAR
SEDE CENTRAL
Sucre – Bolivia

MAESTRÍA EN SOFTWARE LIBRE

PROPUESTA DE UN MARCO DE TRABAJO ÁGIL DE DESARROLLO DE SOFTWARE, PARA PROYECTOS DE GRADO, BASADA EN SCRUM, KANBAN Y EXTREME PROGRAMMING

Tesis presentada para optar el Grado Académico de Magíster en Software Libre

MAESTRANTE: NINOSKA SOFÍA DELGADO VACA GUZMÁN

TUTOR: MAURICIO CANSECO TORRES

Sucre - Bolivia

2022

DEDICATORIA

Con mucho amor para Ignacio Adrián

AGRADECIMIENTO

A mis padres por todo su amor

RESUMEN

A nivel nacional, diferentes programas de formación académica en Ingeniería de Sistemas o carreras relacionadas, exigen la realización de proyectos de grado en forma individual acorde a reglamentos establecidos y los estudiantes pueden elegir realizar como proyecto de grado el desarrollo de software. Actualmente, existen dos grandes enfoques de metodologías de desarrollo de software, las tradicionales y las ágiles, los estudiantes han utilizado ambas en la realización de su proyecto de grado, sin embargo, ninguna de estas metodologías está orientado específicamente al desarrollo de software en forma individual en un semestre académico de 20 semanas.

Considerando este contexto, esta investigación, propone un Marco de Trabajo Ágil, de Desarrollo de Software para proyectos de grado, llamada DSP "Desarrollo de Software para Proyectos de Grado", que permita a los estudiantes universitarios, tener una guía de trabajo como referencia. Este Marco de Trabajo Ágil está basado en Scrum, Kanban, XP porque los requisitos del cliente son muy cambiantes, en este sentido se decidió utilizar un enfoque ágil en vez de un enfoque tradicional, por las ventajas que puede aportar a esta investigación. La tesis está estructurada según el reglamento de la Universidad Andina Simón Bolívar por la Guía Metodológica para la Elaboración de Tesis de Grado Programas Académicos Postgrado de la siguiente forma:

- Una Introducción
- El primer Capítulo: Aspectos Generales
- El segundo Capítulo: Marco Teórico
- El tercer Capítulo: Metodología de la Investigación
- El cuarto Capítulo: Estudio Comparativo de Scrum, Kanban y Programación Extrema XP
- El quinto Capítulo: Resultados, Conclusiones y Recomendaciones de la Investigación
- El sexto Capítulo: Propuesta de Mejoramiento
- El séptimo Capítulo: Validación mediante la Ejemplificación de la Propuesta
- El octavo Capítulo: Validación mediante un Framework
- Y finalmente se consideran los Anexos y la Bibliografía

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ANEXOS

INTRODUCCIÓN

Tener una guía de referencia para el desarrollo de software en la elaboración de Proyectos de Titulación de grado, tiene una gran importancia para los estudiantes universitarios.

La existencia de una reglamentación para la realización de proyectos de grado a nivel del Sistema de Universidades y a nivel Facultativo, limita a varias metodologías ágiles o tradicionales a ser utilizadas adecuadamente por los estudiantes, por lo que una guía o marco de trabajo considerando esta normativa es de gran ayuda como referencia de apoyo durante el proceso de realización del Proyecto de grado.

La reglamentación existente proviene del Sistema de Universidades de Bolivia, que es reflejada en todas las Universidades Bolivianas, indicando que los trabajos de proyecto de grado se deben realizar en forma individual y en un periodo académico correspondiente a un semestre.

En este sentido, se plantea un Marco de Trabajo Ágil de desarrollo de software para proyectos de grado basada en Scrum, Kanban y XP por las ventajas que tiene el enfoque ágil en relación del enfoque tradicional sobre requisitos cambiantes.

El contexto para realizar esta investigación, es la Carrera de Ingeniería de Sistemas, de la Facultad de Ciencias y Tecnología de la Universidad Mayor Real y Pontificia de San Francisco Xavier de Chuquisaca.

CAPÍTULO I

1 ASPECTOS GENERALES

En el Capítulo I: Aspectos Generales, como su nombre indica se menciona las generalidades de toda la investigación, desde los antecedentes que preceden a esta tesis, se especificó el planteamiento del problema, los objetivos de la tesis, la idea que se quiere defender, la operacionalización de los objetivos de estudio, los alcances y delimitaciones de la investigación y una revisión del estado de arte; toda esta estructura se encuentra según la Guía Metodológica para la Elaboración de Tesis de Grado Programas Académicos Postgrado de la Universidad Andina Simón Bolívar.

1.1 Antecedentes

Según reglamentación existente, emanada por el Sistema de la Universidad Boliviana, en el Reglamento General de Tipos y Modalidades de Graduación, está claramente establecido que la realización de un Proyecto de Grado, deberá ser de forma individual, en un semestre de la gestión académica. Esta normativa, se refleja internamente, en el Reglamento General de Graduación de las Universidades en toda Bolivia. El Reglamento General de Graduación para carreras de Licenciatura y Técnico Superior de la Facultad de Ciencias y Tecnología, también establece la normativa de realización de Proyectos de Grado de forma personal, en un semestre académico (ver anexo 3). Al mismo tiempo el Sistema Informático de esta Universidad, claramente establece que el tiempo, que corresponde a un semestre académico es de 20 semanas académicas. En este contexto, la realización de un Proyecto de Grado, implica el desarrollo de software, en forma personal, en un semestre académico.

El conjunto de metodologías para el desarrollo de software se cataloga en dos grandes enfoques: las tradicionales que participan de un ciclo de vida de software, que responden a un plan detallado, mientras que las metodologías ágiles, se caracterizan por la flexibilidad en el proceso de desarrollo iterativo e incremental y requisitos cambiantes. Para el desarrollo de software en Proyectos de Grado se puede utilizar cualquier metodología existente ya sea ágil o tradicional, sin embargo, ninguno está enfocado específicamente al desarrollo de software en forma individual en un tiempo de 20 semanas académicas.

Existen antecedentes de metodologías como el PSP Personal Software Process para desarrollar software en forma personal y algunos Proyectos de Grado y/o Tesis de Grado que han realizado estudios relacionados a la creación de una Metodología Ágil de Desarrollo de Software, para pequeños grupos de personas o de forma personal en contextos diferentes, pero ninguno está enfocado específicamente para desarrollar software de forma individual en 20 semanas académicas.

1.2 Planteamiento del Problema

El Reglamento General de Graduación para carreras de Licenciatura y Técnico Superior de la Facultad de Tecnología (ver anexo 3), que es el caso de estudio considerado en esta investigación, claramente indica: en el Artículo 3, que los trabajos de graduación deberán ser elaborados por el estudiante en forma personal; en el Artículo 31, afirma que el alcance y contenido del tema a ser estudiado será tal que permita su conclusión en un semestre académico. Y según el Sistema Informático de la Universidad San Francisco Xavier de Chuquisaca, un semestre corresponde a 20 semanas académicas.

El Reglamento General de Tipos y Modalidades de Graduación del Sistema de la Universidad Boliviana mediante el Congreso Nacional de Universidades (ver anexo 1), explícitamente indica en el Artículo 7, que el tiempo de los tipos y modalidades de graduación deben estar dentro de las 400 a 600 horas académicas programadas en los dos últimos semestres de la gestión académica, también afirma en el Artículo 31, que se debe proponer el trabajo de graduación en forma individual.

Según el Plan de Estudios de la carrera de Ingeniería de Sistemas del caso de estudio (ver anexo 2), los dos últimos semestres son designados para realizar la Modalidad de Graduación, en el penúltimo semestre se realiza solo el Perfil, aprobando este, se pasa al último semestre donde se elabora el proyecto de grado hasta su defensa ante el tribunal.

En este contexto, se puede percibir que los estudiantes universitarios para la realización de su proyecto de grado, solo cuentan con un tiempo de realización de 20 semanas académicas y con un equipo de desarrollo de software unipersonal; restricciones que deben cumplir según reglamentos mencionados anteriormente. También se debe considerar que actualmente algunas Metodologías y/o Marcos de Trabajo de Desarrollo de Software, están orientadas con preferencia a equipos de personas que cumplen diferentes roles para optimizar el tiempo de desarrollo y/o no consideran el desarrollo de

software exactamente en 20 semanas académicas. Considerando este argumento, se percibe la necesidad de una guía o marco de trabajo que sirva de referencia a los estudiantes universitarios durante la realización de su proyecto de grado que contemple las normativas de graduación actualmente vigentes.

Actualmente si bien los estudiantes han estado utilizando diferentes metodologías o marcos de trabajo existentes en sus proyectos de grado para el desarrollo de software, por las características de las mismas, estas herramientas han tenido que ser modificadas por el estudiante para ser adecuadas a la reglamentación vigente y así poder cumplir con el proyecto de grado, incluso existe el caso de algunos estudiantes que les tomó más de un semestre terminar todo su proyecto.

En este sentido, se propone un Marco de Trabajo Ágil de desarrollo de software para Proyectos de Grado, que sirva como guía de referencia; basada en Scrum, Kanban y XP, por las ventajas en relación a la existencia de requisitos cambiantes que brinda las Metodologías Ágiles de Desarrollo de Software en comparación a las Tradicionales. La propuesta al considerar los reglamentos vigentes, se utilizará de guía sin la necesidad de adecuarla como las metodologías y/o marcos de trabajo ya existentes, facilitando el desarrollo de software y su respectiva documentación.

A. Formulación del problema:

¿De qué manera se puede efectivizar el desarrollo del proyecto de grado asociado al desarrollo de software permitiendo guiar al estudiante universitario de la carrera de Ingeniería de Sistemas o ramas afines, en el cumplimiento óptimo de los plazos y características estipuladas en la reglamentación vigente?

1.3 Objetivos

1.3.1 Objetivo General

Plantear un Marco de Trabajo Ágil de desarrollo de software de Proyectos de Grado para que sirva como guía de referencia a los estudiantes universitarios de la carrera de Ingeniería de Sistemas durante la realización de esta modalidad de graduación.

1.3.2 Objetivos Específicos

- Estructurar un Marco Teórico como soporte de la propuesta investigativa relacionado con el Marco de Trabajo Ágil de Desarrollo de Software

- Definir la Metodología de la Investigación como apoyo para la elaboración de esta investigación
- Elaborar un Estudio Comparativo de Scrum, Kanban y Programación Extrema utilizando el estudio realizado por Abrahamson, Salo, Ronkkainen y Warsta publicado en su libro "Agile Software Development Methods Review and Análisis" para identificar elementos que se puedan utilizar en la propuesta.
- Diseñar la Propuesta de Mejoramiento en base a un Marco de Trabajo Ágil de Desarrollo de Software de Proyectos de Grado para que sirva como guía de referencia a los estudiantes universitarios de la carrera de Ingeniería de Sistemas
- Realizar una pequeña ejemplificación resumida aplicando la propuesta como validación de la misma
- Utilizar los identificadores del Framework para Evaluación de Metodologías Ágiles de Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani como validación de la propuesta.

1.4 Idea a Defender

La idea a defender es un postulado teórico una suposición que sostiene el investigador y que está sustentada por relaciones esenciales que permiten dar respuesta al problema planteado.

La Idea a Defender en esta investigación es "La propuesta de un Marco de Trabajo Ágil de Desarrollo de Software para proyectos de grado, dirigido a estudiantes del décimo semestre de la Carrera de Ingeniería de Sistemas es una guía de referencia que permitirá orientar la realización de esta modalidad de graduación elegida por el estudiante".

1.5 Operacionalización de los Objetivos de Estudio

Según la Guía Metodológica para la Elaboración de Tesis de Grado Programas Académicos Postgrado de la Universidad Andina, en el caso de Tesis de Grado se debe considerar los elementos conceptuales necesarios para desarrollar la investigación; en este caso estos están detallados en el Capítulo Marco Teórico. Así mismo considera los instrumentos que se utilizaron para la investigación, en este caso, estos se encuentran en el Capítulo Metodología de la Investigación.

1.6 Alcances y Delimitaciones de la Investigación

Geográficas

- El Marco de Trabajo Ágil de Desarrollo de Software propuesto, está enfocado solo a la modalidad de graduación de Proyecto de Grado, donde el estudiante decida como trabajo de titulación realizar el desarrollo de software, de la Carrera de Ingeniería de Sistemas de la Facultad de Ciencia y Tecnología de la Universidad de San Francisco Xavier de Chuquisaca. Si el estudiante decide realizar otro tipo de trabajo en su Proyecto de Grado, no puede utilizar este Marco de Trabajo propuesto.
- Para la Carrera de Ingeniería de Sistemas; se debe especificar el uso de esta propuesta en el Perfil del Proyecto de Grado, en el noveno semestre; en este sentido, el Marco de Trabajo Ágil de Desarrollo de Software propuesto, considera que ya se tiene el perfil aprobado y se cuenta con la asignación de un tutor/a según establece los reglamentos y parte desde este punto para ser utilizado como guía de referencia durante el décimo semestre.

Temporales

- El Marco de Trabajo Ágil propuesto, debe utilizarse por el estudiante durante el décimo semestre de la carrera de Ingeniería de Sistemas.
- Esta investigación se ha estado desarrollando desde el 2016 hasta la fecha, ya que la Maestría de Software Libre versión I es de la gestión 2015.

Conceptuales

- La línea de investigación donde se inserta esta tesis de Maestría son las Metodologías Ágiles de Desarrollo de Software: XP Programación Extrema, Kanban, el Marco de Trabajo Ágil Scrum, y el Manifiesto Agile. Correspondientes al Módulo 4 Desarrollo de Software Libre de la Estructura Curricular del Programa de Maestría en Software Libre; versión I – gestión 2015.
- No se realiza un estudio de la complejidad y/o tamaño de proyecto de grado porque el alcance y contenido del tema propuesto por el estudiante ya está reglamentado en el Artículo 31 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología, en este sentido esta investigación se basa en este reglamento.
- Al ser un Marco de Trabajo Ágil de desarrollo de software, considera el Manifiesto

ágil, donde uno de los cuatro valores fundamentales es “Software funcionando sobre documentación extensiva” en el mismo Manifiesto indican los autores “aunque valoramos los elementos de la derecha, valoramos más los de la izquierda”, en este sentido, se aclara que el manifiesto ágil si considera que debe existir documentación del software desarrollado ya que menciona que también la valoran, aunque no en la misma proporción que el software. En este sentido, para esta guía de referencia, el documento como el software son importantes para ser presentados ante el tribunal en la defensa pública.

Operativas

- El Marco de Trabajo Ágil de Desarrollo de Software propuesto, no se podrá aplicar en el desarrollo de software que implique la realización de más de 20 semanas académicas y que involucre un equipo de desarrollo de varias personas con varios roles multidisciplinarios.
- El Marco de Trabajo Ágil de Desarrollo de Software propuesto, puede ser modificado para adecuarse al desarrollo de software según el contexto del estudiante interesado ya que se considera solo como una guía de referencia. Existiendo la posibilidad de utilizar otras herramientas complementarias durante el desarrollo del software si fuese pertinente.
- Se considera como marco legal utilizar: el Reglamento General de Graduación de la Universidad de San Francisco Xavier de Chuquisaca de la Facultad de Tecnología de la Carrera de Ingeniería de Sistemas de la ciudad de Sucre, Bolivia. Y el Reglamento General de Tipos y Modalidades de Graduación del Sistema de la Universidad Boliviana, porque actualmente están vigentes.
- En el presente, si existiese un reglamento de Graduación propio de la Carrera de Sistemas, el mismo no se considera en esta investigación porque de forma verbal desde dirección de esa carrera se facilitó el Reglamento General de Graduación de la Universidad de San Francisco Xavier de Chuquisaca de la Facultad de Tecnología como documento de uso oficial actualmente vigente.
- Se define Marco de Trabajo del inglés Framework, a un conjunto de conceptos y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

1.7 Revisión del Estado del Arte

Actualmente, existen algunas investigaciones relacionadas al tema propuesto, pero con la diferencia de que no están enfocados específicamente a proyectos de titulación de grado, entre estas tenemos:

En Perú:

El Informe del Trabajo Práctico de Suficiencia de Ingeniería de Sistemas e Informática, cuyo Título es "Metodología en el Desarrollo de Software" realizado por el estudiante Wilfredo Marcial García Pérez, en la Universidad Nacional de la Amazonía Peruana en Iquitos, Perú. En ella pretende determinar una metodología ágil de desarrollo de software para la aplicación en proyectos de pequeña escala, donde se intenta minimizar la burocracia que según el autor subyace en la utilización de procesos determinísticos que son utilizados en proyectos con gran cantidad de recursos. Para ello empieza mencionando las metodologías y notaciones actualmente empleadas las tradicionales y las ágiles, indicando los roles, fases, procesos, artefactos y herramientas utilizadas en cada una de ellas. Finalmente define la metodología propuesta definiendo roles para un equipo de trabajo, las fases e hitos, disciplinas, artefactos y actividades que se utilizan para el desarrollo y aplicación de la metodología propuesta. (García, 2016)

La Tesis de Grado de Ingeniería Industrial y de Sistemas, cuyo Título es "Manual para elegir una metodología de desarrollo de software dentro de un proyecto informático" realizado por el estudiante Arnaldo Andrés Espinoza Meza, en la Universidad de Piura en Piura, Perú. En ella propone un manual, para la selección de la alternativa de una metodología idónea, para un proyecto de sistemas informáticos. Para ello recopiló información que permitió elaborar un estudio comparativo de un grupo de metodologías previamente escogidas. A partir de este estudio definió los elementos que serían incluidos en el contenido del manual. Afirma que no garantiza un proyecto exitoso con su manual, pero permite una buena elección que se acerca más a las características del proyecto, creando los elementos y conexiones necesarias para cumplir con los requisitos del producto solicitado por los clientes. (Espinoza, 2013)

En Chile:

La Tesis de Grado de Maestría, cuyo Título es "Propuesta de Metodología Ágil de Desarrollo de Software con integración de TAM – Z-Ágil" realizado por el estudiante

Juan Francisco García Toledo, en la Pontificia Universidad Católica de Valparaíso en Valparaíso en Chile. En ella propone la creación de una Metodología Ágil con integración del Modelo de Aceptación de la Tecnología TAM, con la inclusión de este modelo se espera obtener información sobre la aceptación, las percepciones de utilidad y usabilidad, datos que servirán como indicadores para realizar mejoras y correcciones a los sistemas durante su construcción. En su estado de arte, menciona las definiciones de TAM, de Metodologías ágiles de Desarrollo y mencionó algunos casos de estudio de empresas en su país. (García, 2016)

En Argentina:

La Tesis de Grado de Ingeniería en Informática, cuyo Título es "Diseño de una Metodología Ágil de Desarrollo de Software" realizado por el estudiante universitario Schenone Marcelo Hernán, en la Universidad de Buenos Aires en Buenos Aires, Argentina. Esta tesis, empieza explicando las desventajas de las metodologías tradicionales existentes, como ser el hecho de no tomar en cuenta el factor humano (el desarrollador y el cliente). Posteriormente el autor explica la metodología diseñada para su tesis de grado usando UML (Unified Modeling Language) como notación y dos casos de uso. La metodología en cuestión la denomina AgEnD (Agile Enhanced Development) la cual toma muchas de las bases del manifiesto del desarrollo ágil de software y que pretende aplicar en proyectos de pequeña escala y riesgo limitado. También está basada y con varias similitudes a la metodología Scrum, pero que según el autor tiene un diseño propio y más adecuado para grupos de desarrollo pequeños. (Schenone, 2017)

La Tesis de Grado de Licenciatura en Sistemas y Computación, cuyo Título es "Metodologías de Desarrollo de Software" realizado por el estudiante universitario Maida, Esteban Gabriel, Paciencia Julián, en la Pontificia Universidad Católica Argentina Santa María de los Buenos Aires, en Buenos Aires, Argentina. Esta tesis presenta una introducción sobre las existentes metodologías para el desarrollo de software y los paradigmas que marcan la diferencia entre un método estructurado y un método ágil para así poder identificar cuál se adapta de manera más eficiente a un proyecto determinado. Realiza el estudio y análisis de casos reales de éxito y de fracaso enfocándose en el uso de metodologías de desarrollo de software.

En Colombia:

La Monografía de Ingeniería de Sistemas, cuyo Título es “Diseño e Implantación de una tecnología para el desarrollo de aplicaciones enfocadas en la utilización de las metodologías ágiles” realizado por el estudiante universitario Marlon Alexander Pineda Vásquez, en la Universidad EAFIT en Medellín, Colombia. Esta monografía, empieza explicando las metodologías ágiles de desarrollo de software existentes mencionando las características y fases de cada una de ellas, menciona tres casos de uso exitoso usando algunas de estas metodologías ágiles. Posteriormente propone una metodología transversal basada en las necesidades locales y la aplicación de técnicas ágiles, en su propuesta indica las características, fases y actividades a realizar, además de los diferentes roles del equipo de trabajo, los entregables, las técnicas y las herramientas propuestas. (Pineda,2015)

En Bolivia:

El Proyecto de Grado de Ingeniería de Sistemas, cuyo Título es “Propuesta de metodología ágil para proyectos individuales basada en Scrum y Kanban” realizado por la estudiante universitaria Torrico Gabriela Nathaly, en la Universidad Mayor de San Simón, Cochabamba, Bolivia. Este Proyecto de Grado realiza un análisis de los fundamentos y prácticas de las metodologías ágiles Scrum y Kanban, para utilizar las herramientas adecuadas de estas metodologías para el uso de una sola persona.

CAPÍTULO II

2 MARCO TEÓRICO

Para el Marco Teórico se han identificado fuentes primarias y secundarias sobre las cuales se pudo investigar; la lectura de bibliografía especializada permitió estructurar este capítulo en: Marco Conceptual, Marco Referencial, Marco Jurídico o Legal y Marco Histórico. El Marco Conceptual es un referente teórico donde se realizó una recopilación de información de varios autores, los cuales están mencionados en bibliografía. En el Marco Referencial se tiene una breve descripción de la información relacionada con la investigación de la Carrera de Ingeniería de Sistemas. En el Marco Jurídico o Legal se tiene los reglamentos que avalan esta investigación y que sirvieron de base para proponer el Marco de Trabajo Ágil de Desarrollo de Software. En el Marco Histórico se tiene un breve resumen de la evolución histórica de las metodologías de desarrollo de software.

2.1 Marco Conceptual

2.1.1 Ingeniería de Software

“Es una disciplina de la ingeniería, que comprende todos los aspectos de la producción del software. Es la aplicación práctica del conocimiento científico al diseño y construcción de programas de computadora y a la documentación asociada requerida para desarrollar, operar y mantenerlos. Se conoce también como desarrollo de software o producción de software”. (Hunt y Thomas, 1999, p.34)

2.1.2 Software

“Es un conjunto de instrucciones lógicas, que le permite al usuario interactuar con el computador o dispositivo electrónico pertinente, a través de una interfaz, es lo que comúnmente se conoce como los programas informáticos. Son programas encargados de definir y administrar el comportamiento de los dispositivos físicos y lógicos de una computadora o de un dispositivo inteligente. La producción de software implica 4Ps: Personas, Producto, Proceso, Proyecto”. (Weitzenfeld A, 2004, p.38)

2.1.3 Proceso de desarrollo de software

Es el conjunto estructurado de las actividades, acciones y tareas requeridas para construir teóricamente un software. Se utiliza para mejorar la comprensión del problema a resolver, la comunicación entre los participantes del proyecto y el mantenimiento de un sistema.

Una estructura general del proceso de desarrollo de software puede tener las siguientes actividades o etapas: Planificación, Análisis de Requisitos, Diseño, Desarrollo, Validación y Verificación, Instalación y Mantenimiento. (Hunt y Thomas, 1999, p.42)

2.1.4 Código Fuente

"Es el conjunto de líneas de texto escrito por el ser humano y redactado en un lenguaje de programación para crear instrucciones claras para el ordenador o dispositivo inteligente y que este sea capaz de traducirlas a su propio lenguaje. Todo este texto forma un programa o aplicación que es un software". (Weitzenfeld A, 2004, p.48)

2.1.5 Método

"Del "methodos" método, el camino que conduce a un lugar. Conjunto de pasos, actividades o acciones que hay que seguir para conseguir un objetivo". (Knuth, 1997, p.12)

2.1.6 Modelo

"Del italiano "modello" modelo. Es un arquetipo o punto de referencia que se toma como referencia para ser imitada, reproducida o copiada". (Knuth, 1997, p.18)

2.1.7 Paradigma

"Del griego "para" junto y "digma" ejemplo o modelo. Se refiere a un determinado modelo de pensamiento o de interpretación de las cosas que promueve una forma de pensar en particular por sobre otros pensamientos. Teoría o conjunto de teorías cuyo núcleo central se acepta sin cuestionar y que suministra la base y modelo para resolver problemas y avanzar en el conocimiento". (Knuth, 1997, p.24)

2.1.8 AUP Proceso Unificado Ágil

Creada por Scott Ambler, es una versión simplificada del Proceso Unificado de Rational RUP. Se trata de un método de desarrollo de aplicaciones de negocios, que usan las técnicas ágiles conocidas como TDD (test Driven Development), MDD (Model Driven Development) y la gestión del software. El método AUP se divide en 4 fases: Lanzamiento: identificación del alcance del proyecto y definición de la arquitectura potencial del sistema, implicación de las partes interesadas del proyecto y elaboración del presupuesto. Diseño: definición de la arquitectura del sistema y demostración si fuera

necesario. Realización: desarrollo del software o producto, durante un proceso gradual priorizando según las funciones. Entrega: validación e instalación del sistema. (Vernon, 2013, p.17).

2.1.9 Open UP Proceso Unificado Abierto

Fue creada por IBM, pero pasó a manos de la empresa Eclipse quien en 2006 fue lanzada bajo una licencia gratuita. Open UP es un método y un proceso de desarrollo de software, dirigido a gestión de proyectos de software basados en desarrollo iterativo, ágil e incremental apropiado para proyectos pequeños y de bajos recursos; y es aplicable a un conjunto amplio de plataformas y aplicaciones de desarrollo. El Open UP es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario está incluido. Por lo tanto, no provee lineamientos para todos los elementos que se manejan en un proyecto, pero tiene los componentes básicos que pueden servir de base a procesos específicos. (Vernon, 2013, p.35)

2.1.10 PSP Personal Software Process

Fue propuesto por Watts S. Humphrey en 1993 del Software Engineering Institute en la Carnegie Mellon University. Es un conjunto de prácticas disciplinadas para la gestión del tiempo y mejora de la productividad personal de los programadores o ingenieros de software, en tareas de desarrollo y mantenimiento de sistemas. PSP ofrece una serie de niveles en los cuales vas avanzando en la aplicación de buenas prácticas, hasta llegar a tener un proceso de desarrollo de software totalmente controlado y predecible. (Vernon, 2013, p.41)

2.1.11 UML Lenguaje Unificado de Modelado

Nació en 1994, los padres de UML son Grady Booch, James Rumbaugh e Ivar Jacobson. Es un lenguaje de modelado de sistemas de software, para la representación de procesos o esquemas de software, fue creado para forjar un lenguaje gráfico de modelado visual para especificar, construir y documentar un sistema. UML ofrece un estándar para describir un “plano o modelo” del sistema, incluyendo aspectos conceptuales tales como procesos, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados. UML guarda una relación directa con el análisis y el diseño orientados a objetos. (Vernon, 2013, p.57)

2.1.12 Marco de Trabajo

“Un Marco de Trabajo o Framework, es un conjunto de conceptos y criterios para enfocar un tipo de problemática particular, que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar”. (Knuth, 1997, p.32)

2.1.13 Metodología

El término tiene su génesis en el griego meta, el cual significa ir más allá, camino y logos, lo cual significa estudio, razón o análisis. Una serie de métodos y técnicas de rigor científico que se aplican sistemáticamente durante un proceso de investigación para alcanzar un resultado teóricamente válido. En este sentido, la metodología funciona como el soporte conceptual que rige la manera en que aplicamos los procedimientos en una investigación. Este término se encuentra vinculado directamente con la ciencia, sin embargo, la metodología puede presentarse en otras áreas. (Knuth, 1997, p.45)

2.1.14 Metodologías de desarrollo de software

“Una metodología de desarrollo de software es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una Metodología está formada por fases, cada una de las cuales se puede dividir en sub fases. Estas pueden ser: Metodología Tradicional, Metodología Ágil, Metodología Híbrida” (Knuth, 1997, p.59)

2.1.15 Metodología tradicional

Basadas en el detalle de la planificación y de la estructuración del proyecto, enfocado en los procesos asociados a la planeación y control del proyecto. No se puede planificar un proyecto hasta no tener el alcance definido todo aquello que se debe hacer. Dependen de los procesos. Plan de trabajo a larga duración. Las fases del ciclo de vida del software son secuenciales van encadenadas ocurren justo uno después de otro, no puedes realizar la siguiente fase sin haber terminado la anterior fase. El autor identifica como ventajas: Se tiene una estructura definida que se puede seguir para ir paso a paso construyendo el proyecto cuando se tiene una meta, alcance y solución. Realiza énfasis en la definición del proceso, roles, actividades y artefactos. Mientras que las desventajas: Produce excesiva documentación innecesaria. Incapacidad de adaptarse al cambio. Se debe esperar la finalización de una etapa para poder iniciar la siguiente. Si algo está mal en una etapa

y no se identifica afecta a las etapas posteriores. Al no ser el cliente parte del equipo de desarrollo, debe esperar hasta el final para conocer el producto terminado. (Cockburn, 2006, p.33)

2.1.16 Metodología ágil

Basadas en los principios y valores de la filosofía del Manifiesto Ágil, se enfoca en procesos iterativos e incrementales con entregas funcionales y retroalimentación con el cliente. Aborda los detalles del alcance de forma sucesiva a lo largo del proyecto. Dependen de las personas. Considera un plan de trabajo a corta duración. Las fases del ciclo de vida del software ocurren mediante iteraciones, cada iteración tiene su cierre para aceptar entregables específicos. El autor identifica como ventajas: Énfasis en los aspectos humanos, el individuo y el trabajo en equipo. Es flexible a cambios durante el proyecto, es decir se adaptan a las necesidades del cliente y a las circunstancias. El cliente interactúa y es parte del equipo de desarrollo. Reduce la cantidad de documentación realizada. Mientras que las desventajas: La o las personas que integran el o los equipos de desarrollo de software deben trabajar preferentemente en el mismo sitio. (Cockburn, 2006, p.43)

2.1.17 Metodología Híbrida

Es una combinación de metodologías ágiles y tradicionales. El autor identifica como ventaja que se encuentra basada en las mejores prácticas de la combinación de metodologías ya existentes y como desventajas: Una mala combinación de metodologías, puede llegar a confundir y complicar el proceso de desarrollo de software. (Cockburn, 2006, p.57)

2.1.18 Metodologías Ágiles vs Metodologías Tradicionales

Las metodologías tradicionales se caracterizan por un ciclo de vida secuencial, orientarse en una planeación exhaustiva, el control de proceso apoyado en abundante documentación. Se puede generalizar en cuatro etapas: Identificación de requerimientos; Diseño y planeación del trabajo; Desarrollo de productos y Etapa de pruebas con la intervención del usuario al finalizar el proyecto. Las metodologías ágiles se caracterizan por el desarrollo del software en forma incremental e iterativa, equipos de trabajo autoorganizados, flexibilidad en los procesos y manejo de tiempo corto en la toma de decisiones y entrega de productos. Para atender a frecuentes cambios en los requerimientos del software. (Cockburn, 2006, p.61)

Tabla 1: Diferencias entre metodologías ágiles y tradicionales

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Tienen cierta resistencia a los cambios
Proceso menos controlado con pocos principios	Proceso mucho más controlado con numerosas políticas y normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software.	La arquitectura del software es esencial y se expresa mediante modelos.
Fuente: Cockburn, 2006, p.62	

2.1.19 Lenguaje de programación

Es cualquier lenguaje artificial, que puede utilizarse para definir una secuencia de un conjunto de instrucciones, para desarrollar un software y su procesamiento por un ordenador o computadora o dispositivo inteligente. Está formado de un conjunto de símbolos, reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones; mediante las cuales se construye el código fuente de una programa informático o software. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación. (Weitzenfeld A, 2004)

2.1.20 Cliente y Usuario

“Cliente o comprador es la persona que realiza una transacción comercial donde a cambio de un pago recibe el producto o servicio que adquiere. Usuario es la persona que utiliza el producto o servicio que ha sido adquirido”. (Hunt y Thomas, 1999, p.57)

2.1.21 Manifiesto Ágil para el desarrollo de software

The Agile Alliance organización sin ánimo de lucro redactó el Manifiesto Ágil que contiene la Filosofía Ágil con el término de Métodos ágiles para definir los modelos de trabajo que estaban surgiendo en base a su filosofía. Su objetivo es esbozar los valores y principios que deberían permitir a los desarrolladores de software responder rápidamente a los cambios que puedan surgir a lo largo de un proyecto. Redactado por 17 expertos en programación, es extensible al desarrollo de cualquier otro producto diferente de un software. (Cockburn, 2006, p.55)

Tabla 2: Valores del Manifiesto Ágil

N	Valores	Descripción
1	Individuos e Interacciones sobre Procesos y Herramientas	Las personas son el factor más importante en el desarrollo de software, porque se necesita creatividad, innovación y talento para realizar las tareas.
2	Software Funcionando sobre Documentación Extensiva	Los prototipos previos al software final, ofrecen una retroalimentación que visualiza el avance del proyecto. Los sistemas ágiles se basan en la entrega a intervalos regulares al cliente de partes pequeñas de software que funciona parcialmente. Mientras que la documentación es solo un soporte de transferencia del conocimiento, registra información histórica legal o normativa.
3	Colaboración con el Cliente sobre Negociación Contractual	Proporcionar las especificaciones del software al inicio y durante su ciclo de vida, con una retroalimentación continua con el cliente. La agilidad implica la entrega del software de manera interactiva e incremental
4	Respuesta Ante el Cambio sobre Seguir un Plan	En el desarrollo de software de requisitos inestables se valora la anticipación, adaptación y capacidad de respuesta a los cambios.
Los 17 autores reconocen la importancia de los elementos de la derecha, pero valoran más los de la izquierda.		
Fuente: https://agilemanifesto.org		

Tabla 3: Principios del Manifiesto Ágil

N	Principios
1	Nuestra mayor prioridad es satisfacer al cliente, mediante la entrega temprana y continua de software con valor.
2	Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio, para proporcionar ventaja competitiva al cliente.
3	Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible
4	Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5	Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan y ser confiables en la ejecución del trabajo.
6	El método más eficiente y efectivo de comunicar información al equipo de desarrollo, y entre sus miembros, es la conversación cara a cara.
7	El software funcionando es la medida principal de progreso.
8	Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9	La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10	La simplicidad o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11	Las mejores arquitecturas, requisitos y diseños emergen de equipos auto organizados.
12	A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.
Fuente: https://agilemanifesto.org	

2.1.22 Técnica o Método de Pomodoro

Es un método de gestión del tiempo laboral que sirve para mejorar la productividad, fue inventado a principios de los años 90 's por el desarrollador, empresario Francesco Cirilo. Quien llamó a su sistema "Pomodoro" por el temporizador con forma de tomate que usaba para seguir su trabajo como estudiante universitario. Pomodoro proviene de la palabra tomate en italiano. Esta técnica puede ayudar a controlar las distracciones y hacer las cosas en ráfagas cortas, mientras tomas descansos frecuentes para respirar y relajarse. La idea sobre la que se fundamenta el método Pomodoro es que las pausas frecuentes serían capaces de mejorar la agilidad mental. Se aplica para realizar tareas o actividades de estudio, lectura, otros tipos de trabajos que requieran concentración. Básicamente se trata de utilizar un temporizador para dividir cualquier actividad en intervalos regulares de 25 minutos, seguida, cada una, de intervalos de descanso de 5 minutos. Los intervalos de trabajo se llaman Pomodoros. Una sesión completa de Pomodoro dura cuatro Pomodoros, entonces se debe realizar descansos de 15 minutos como mínimo entre cada sesión. (Scholz & Tie Tie, 2001, p.17)

Tabla 4: Técnica de Pomodoro

1 sesión completa de Pomodoro equivale a:							
25 min	5 min	25 min	5 min	25 min	5 min	25 min	5 min
Actividad de Concentración Pomodoro 1	Pausa Descanso	Actividad de Concentración Pomodoro 2	Pausa Descanso	Actividad de Concentración Pomodoro 3	Pausa Descanso	Actividad de Concentración Pomodoro 4	Pausa Descanso
Totales: 25+25+25+25 = 100 minutos de Trabajo 5+5+5+5 = 20 minutos de Descanso 120 minutos dura una sesión completa de Pomodoro							
Fuente: https://pomofocus.io/							

2.1.23 Empirismo

El empirismo es una corriente filosófica y epistemológica que alude que todo conocimiento que posee o adquiere el hombre es producto de la experiencia, bien sea interna o externa, y por ello es visto como una consecuencia de los sentidos. Considera la experiencia y la percepción sensorial como el mejor camino hacia la verdad de las cosas. Afirma que la vía para alcanzar el conocimiento es la experiencia. (Vernon, 2013, p.67)

2.1.24 Pirámide de Maslow

La Pirámide de Maslow, o jerarquía de las necesidades humanas, es una teoría psicológica propuesta por Abraham Maslow en su obra: Una teoría sobre la motivación humana de 1943, que posteriormente amplió. Maslow formula en su teoría una jerarquía de necesidades humanas y defiende que conforme se satisfacen las necesidades más básicas (parte inferior de la pirámide), los seres humanos desarrollan necesidades y deseos más elevados (parte superior de la pirámide). La escala de las necesidades de Maslow se describe como una pirámide que consta de cinco niveles de la base del triángulo a la punta son: Necesidad Fisiológica, Necesidad de Seguridad, Necesidad de Afiliación, Necesidad de Reconocimiento, Necesidad de Autorrealización. Necesidades Fisiológicas básicas para mantener la homeostasis (referente a la salud) las más evidentes son: Necesidad de respirar, beber agua, y alimentarse. Necesidad de mantener el equilibrio del pH y la temperatura corporal. Necesidad de dormir, descansar y eliminar los desechos. (Scholz & Tie Tie, 2001, p.25)

2.1.25 Comportamiento del cerebro con varias tareas o roles realizados al mismo tiempo

Marcel Just, catedrático de Psicología y director adjunto del Centro de Imágenes Cognitivas del Cerebro de la Universidad Carnegie Mellon de Pittsburg en Pensilvania, realizó un estudio de la actividad cerebral que emplea imágenes de resonancia magnética para comparar lo que sucede en la cabeza de las personas cuando se realizan tareas complejas y se pretende hacer dos cosas al mismo tiempo, este estudio revela un hecho inquietante: parece ser que el cerebro tiene una cantidad limitada de espacio para dedicar a las tareas que requieren atención. Cuando una persona intenta realizar dos actividades al mismo tiempo o cumplir dos roles diferentes, su actividad cerebral no se multiplica por dos, sino que la cantidad de actividad cerebral que se dedica a cada una de las tareas

disminuye realmente. En consecuencia, quienes realizan de manera simultánea dos tareas que exigen atención no hacen verdaderamente ninguna de las dos igual de bien, que como lo harían por separado. (Scholz & Tie Tie, 2001, p.72)

2.1.26 Kanban.

Kanban es una palabra japonesa que proviene de "Kan= visual" y "Ban=Tarjeta o Tablero". Creada por Taiichi Ohno Ingeniero Industrial, utilizada en el Sistema de Producción de Toyota, donde implementó en su producción ajustada, la fabricación JIT Just In Time (Justo a Tiempo) para controlar su inventario. Solo se producen los productos que son requeridos para satisfacer la demanda de los clientes "Sistema Pull

- Tirar" y solo se reemplaza los productos cuando estos se hayan vendido. Es una estrategia de producción que se utiliza para reducir el desperdicio en el proceso de fabricación, eliminando la práctica tradicional de fabricar productos e intentar venderlos en el mercado. Su objetivo es minimizar el WIP (Work in progress), o el stock entre los procesos. David J Anderson, neurobiólogo estadounidense publicó su libro "Kanban Esencial Condensado" donde utiliza Kanban en el desarrollo del software. A partir de ahí fue adaptado a la industria del software, Kanban no es iterativo, pero si es incremental divide el flujo de trabajo en partes o tareas que se visualizan en un Tablero. La idea es dividir un proyecto de software en las tareas más pequeñas y terminarlas rápidamente. (Morales R, 2015, p. 6)

PRINCIPIOS (Morales R, 2015, p. 12)

- Visualizar el flujo de trabajo (workflow) y las tareas del ciclo de producción mediante fichas o tarjetas colocadas en un tablero
- Limitar la cantidad de trabajo en progreso WIP (work in progress) Limita que se realizan muchas tareas a la vez de acuerdo con la capacidad de trabajo personal.
- Gestionar el flujo de trabajo mediante el uso de Métricas: Medir el tiempo en completar una tarea "Lead Time" empieza desde que se hace una petición creando la tarjeta de la tarea, hasta que se hace la entrega de la tarea. Se mide lo que ven los clientes, lo que esperan. "Cycle Time" empieza desde que el trabajo sobre una tarea comienza hasta que termina su desarrollo. Se mide el rendimiento del proceso.
- Hacer que las políticas de procesamiento sean explícitas: la calidad debe incorporarse al proceso y no ser inspeccionada posteriormente.

- Reconocer las oportunidades de mejora continua: es un proceso evolutivo que toma cada iteración como una nueva oportunidad de mejora.

ROLES (Morales R, 2015, p. 14)

No tiene definido específicamente los roles. Porque asume que la ausencia de papeles es una ventaja para el equipo.

ARTEFACTOS (Morales R, 2015, p. 15)

No tiene definido específicamente Artefactos.

REUNIONES (Morales R, 2015, p. 16)

Establece reuniones rápidas de todo el equipo alrededor del tablero para que cada miembro, cuente en que está trabajando y en qué etapa se encuentra. Se realizan diariamente o dependiendo de las necesidades durante máximo 15 minutos.

EVENTOS

Tablero de Kanban (Morales R, 2015, p. 18)

Kanban utiliza técnicas visuales para ver la situación de cada tarea durante todo el flujo de trabajo, la prioridad de las tareas, en qué etapa está cada tarea, quien es la persona encargada de cada tarea. Se representa en un tablero físico (pizarra con papeles de colores) o digital (Trello, Jira) utilizado para mapear, visualizar y gestionar cómo se van completando las tareas y avanzando el trabajo a través del movimiento de tarjetas.

El tablero está compuesto por: (Morales R, 2015, p. 19)

- Fichas o Tarjetas, cada una de ellas representará una tarea de trabajo.
- Columnas, es una secuencia de pasos o estados específicos sucesivos por los que debe transitar una tarea de trabajo desde que se solicita hasta que está realizada. Cada columna visualiza una fase o etapa del proceso de realización de la tarea El nombre que se les da está en función a los objetivos del desarrollo del producto, que sean funcionales y claros.
- Filas o carriles, representan diferentes tipos de actividades específicas, se las utiliza para agrupar tareas según el tipo de trabajo (desarrollo web, desarrollo móvil), según su prioridad (alta, media, baja), o tareas que se basan en el desarrollo del producto.

Tabla 5: Tablero Kanban

Reserva: Requested: Pedido: Pendientes: *lista de tareas pendientes, sin priorizar, en espera*	To Do: Por hacer: Ready: Listo: *lista de tareas seleccionadas que tenemos que hacer*	Doing: Haciendo: In Progress: En Proceso: *lista de tareas en ejecución*	Review: Revisión: *lista de tareas en testeo*	Done: Terminado: *lista de tareas hechas*
Tarjetas de las tareas que se deben realizar en algún momento.	Tarjetas de tareas que se deben realizar	Tarjetas de las tareas que se encuentran en proceso de ejecución	Tarjetas de tareas en prueba	Tarjetas de las tareas terminadas
Fuente: Morales R, 2015, p.20				

Tarjeta Kanban

Cada tarjeta visualiza el progreso de las tareas de trabajo en un flujo de trabajo desde el momento en que se solicitan, hasta el momento en que se consideran hechas, debe ser capaz de comunicar su contenido de forma clara y concisa. Una tarjeta Kanban contiene información significativa sobre la tarea como ser: su objetivo, el solicitante, su descripción, tipo de trabajo, prioridad, estimación de duración, fecha límite, fecha de inicio, tiempo del ciclo de trabajo, título, a quién se le ha asignado, prioridad y sub tareas. Se colocan en las columnas que sean pertinentes según el estado en el que se encuentran. Se mueven de izquierda a derecha indicando su progreso a través de las columnas a medida que el trabajo avanza. Cuando una tarea no puede ser terminada por falta de información debe ser marcada como impedida hasta que pueda ser completada. La

cantidad de tarjetas Kanban que están en progreso en el tablero debe ser limitada. (Morales R, 2015, p. 22)

2.1.27 Scrum

Es un Framework o Marco de Trabajo que sigue los fundamentos establecidos en el manifiesto ágil. Viene de la palabra en inglés "Scrum=Melé" (Melé es una jugada de Rugby). Toma sus principios de los estudios realizados en los años 80 por los japoneses Hirotaka Takeuchi e Ikujiro Nonaka, quienes investigaron nuevas prácticas de producción, inicialmente para productos tecnológicos, en el artículo "El nuevo juego para el desarrollo de productos (The New Product Development Game)". Ken Schwaber y Jeff Sutherland adaptaron Scrum para el desarrollo de software para la empresa Easel Corporation y escribieron la primera Guía de Scrum; su última actualización fue en noviembre de 2017. (Schwaber K. y Beedle M,2001, p.8)

PILARES O PRINCIPIOS (Schwaber K. y Beedle M,2001, p.10)

Scrum se sostiene sobre tres principios obtenidos de la teoría del control empírico de procesos:

A. Transparencia: Todos los integrantes de la administración del proceso deben conocer los aspectos y resultados que inciden sobre aquél. Estos aspectos deben tener estados definidos, con términos concretos y no ambiguos.
B. Inspección: La frecuencia de inspección debe ser suficiente como para identificar variaciones que podrían afectar de manera negativa un proyecto. El éxito de las actividades dependerá de la habilidad, cautela, eficiencia y experiencia de los inspectores.
C. Adaptación: La adaptación rápida permite ajustar el proceso cuando, a través de la inspección, se hallan aspectos fuera de los límites y que producen desviaciones al objetivo principal del proceso. Para inspeccionar y adaptar, el proceso se vale de reuniones diarias, revisión de iteraciones y reuniones de planificación.

ROLES (Schwaber K. y Beedle M,2001, p.16)

El equipo Scrum (Scrum Team) es autoorganizado y tiene capacidad de adaptación. Se subdivide en: Personas que están comprometidas con el desarrollo del software que son

el Product Owner, el Scrum Master y el Development Team. Y personas que están involucradas con el desarrollo del software que son Stakeholders.

A. Product Owner - Propietario del producto:

Es responsable de transmitir e identificar las prioridades de implementación de los requerimientos del software identificados por el cliente, que se encuentran en una lista denominada Lista del Producto (Product Backlog). El Propietario conoce a plenitud el entorno del negocio del cliente y de la visión del producto (las características que se demandan y que se espera que posea). Debe difundir claramente el contenido del Product Backlog y las prioridades de sus elementos entre todo el equipo Scrum. Además, en base a su conocimiento del proyecto, se encargará de su financiamiento. Él es el único acreditado para tomar decisiones con respecto al Product Backlog y sus prioridades ya que es el nexo entre los interesados y el equipo de trabajo.

B. Development Team - Equipo de desarrollo

Grupo de personas que desarrollan el producto. Se encargan de la conversión del Product Backlog a un incremento ejecutable. Son grupos auto-organizados y autogestionados, porque ellos deciden la forma cómo llevarán a cabo esta conversión. Está formado por personas donde cada una de ellas tiene habilidades especializadas, para completar los incrementos. Esto implica compartir conocimientos entre miembros y un aprendizaje de nuevas habilidades. Para que el trabajo del equipo sea óptimo, y evitar limitaciones interacción (en el caso de pocos integrantes) o aumento en la complejidad y coordinaciones (cuando hay muchos integrantes, problemas que afectan la productividad, los autores proponen que debe ser un equipo suficientemente pequeño como para permanecer ágil y suficientemente grande como para completar una cantidad de trabajo significativa, formado por 5 a 9 personas responsables de desarrollar y entregar software con valor, mediante incrementos de productos terminados en cada Sprint. Es el responsable de proporcionar todas las estimaciones del tiempo.

C. Scrum Master - Maestro Scrum

Encargado de entrenar al equipo y garantizar que éste trabaje según el modelo Scrum. El Scrum Manager instruirá al equipo con respecto a los valores, prácticas y normas Scrum. Guía al equipo para crear productos de alto valor, contribuye a que sean auto organizados, elimina impedimentos para el progreso del equipo de desarrollo, facilita los eventos, guía

al dueño del producto a crear y ordenar la lista de Producto para maximizar el valor. Su rol de líder permitirá que el equipo mejore paulatinamente la calidad de su desempeño (productividad) y del software desarrollado.

D. Stakeholders – Parte Interesada

Personas que tienen interés directo en el software ya que les producirá el beneficio que justifica su desarrollo. Definen los requerimientos del software que transmiten al Dueño del Producto; recibe el producto al final de cada Sprint y proporciona el Feedback (retroalimentación).

ARTEFACTOS (Schwaber K. y Beedle M,2001, p.29)

Representan trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades para la inspección y adaptación.

A. Product Backlog - Lista del Producto

Lista ordenada de los requisitos necesarios del sistema, representados por características, funciones, mejoras que debe contener el software para ser exitoso. Cada elemento debe constar de una descripción, una prioridad, y una estimación de esfuerzo (magnitud de tiempo que tomará realizar una tarea determinada). Esta lista no está completa, evoluciona a medida que el entorno en el que se usará también lo hacen. Esta lista es dinámica, cambia constantemente para identificar lo que el producto necesita para ser adecuado, competitivo y útil. No se debe pretender tener todos los requisitos desde el inicio. Se irá actualizando a través del tiempo. Se trabajará con el Product Backlog teniendo como actividades inmediatas las correspondientes a obtener los requisitos con mayor prioridad y, por consiguiente, detalladas con mayor nivel. El responsable del Product Backlog es el propietario del producto. Los miembros del equipo tienen acceso a él y participan en su elaboración. Utiliza el formato de Historias de Usuario de XP (Como: quien se beneficia, Quiero: acción a realizar, Para: objetivo de la acción), de la metodología Programación Extrema.

B. Sprint Backlog - Lista de Pendientes de Sprint

Es una lista de tareas seleccionadas por el equipo de desarrollo que se realizan durante el Sprint para alcanzar un incremento. Las tareas son agregadas al Sprint Backlog tomando como referencia el Product Backlog en la reunión de Planificación del Sprint. Luego estos

elementos son descompuestos en tareas más pequeñas hasta lograr un entendimiento mejor de cada una de ellas. Cada tarea debe ser realizada por una persona del equipo de desarrollo, con un tiempo y los recursos para completarla. El Sprint Backlog puede ser cambiado exclusivamente por el equipo, ya sea modificando tareas y sus características (estimaciones de tiempo), agregando tareas, o eliminando las que no aportan valor.

C. Product Increment – Incremento del Producto

Es la suma de todos los elementos de la Lista de Producto terminados durante un Sprint. Este incremento de producto se integra en los incrementos de todos los Sprints anteriores formando el producto final. Es el Ejecutable resultado de un Sprint. La principal característica de un incremento es que debe estar "hecho", esto es: Que los elementos del Product Backlog y el incremento en su totalidad haya completado el análisis, diseño, refactorización, programación, documentación y pruebas (testeo unitario, de sistema, de usuario y de regresión, pruebas no funcionales, pruebas de rendimiento, de estabilidad, de seguridad y de integración). El incremento debe estar en condiciones de utilizarse sin importar si el dueño del producto decide liberarlo o no.

EVENTOS (Schwaber K. y Beedle M,2001, p.47)

Son bloques o cajas de tiempo (Time Boxes) con una duración máxima, terminan cuando se alcanza el objetivo del evento, constituyen una oportunidad para habilitar los pilares de transparencia, inspección, adaptación.

A. Sprint

Es un periodo de tiempo que contiene al resto de los eventos, para desarrollar un incremento de producto terminado (Product Increment) utilizable. Cada nuevo sprint debe empezar después de finalizar el anterior. La duración será constante para todos los Sprints del proyecto. Durante el Sprint no se realizan cambios que afecten al objetivo del Sprint (Sprint Goal), sólo el Dueño del Producto puede cancelar un Sprint. Durante el Sprint se realiza: Planificación del Sprint, Scrums Diarios, Trabajo de Desarrollo de Software, Revisión del Sprint, Retrospectiva del Sprint. El Trabajo de Desarrollo de Software o Development Work es donde se realiza la iteración de programación que debe durar de 2 a 4 semanas.

B. Sprint Planning - Planificación de Sprint

Jornada de trabajo realizada al inicio de cada Sprint, durante 8 h (para un sprint de un mes), en ella se reúnen el Scrum Master, Equipo de Desarrollo, Propietario del Producto, donde se tomará decisiones acerca del trabajo y objetivos que se engloban en la iteración.

En concreto se debe fijar:

- El objetivo del Sprint (Sprint Goal)
- Los elementos del Sprint Backlog considerando el Producto Backlog y la capacidad del equipo de desarrollo.
- Divide en tareas el Sprint Backlog que el equipo va a desarrollar durante un Sprint y el plan para terminarlos.
- El tiempo de desarrollo de las tareas que se realizarán durante el Sprint, basados en herramientas adicionales como el Planning Poker entre otros.

C. Daily Scrum – Reuniones Diarias

Reuniones diarias de 15 min (para un sprint de un mes), realizadas al inicio de cada jornada, en el mismo lugar y a la misma hora, donde el equipo de desarrollo, con el Scrum Master coordinan acerca de:

- Se establece un plan para las próximas 24h
- ¿Qué hicieron ayer que ayudó al Equipo de Desarrollo a lograr el Objetivo del Sprint?
- ¿Qué harán hoy para ayudar al Equipo de Desarrollo a lograr el Objetivo del Sprint?
- ¿Vieron algún impedimento que evite que el Equipo de Desarrollo o yo logremos el Objetivo del Sprint?

D. Sprint Review - Revisión de Sprint

Esta reunión se realiza al final de cada Sprint, durante 4 h (para un sprint de un mes) consiste en la presentación, efectuada por el equipo de desarrollo, del incremento funcional generado en la iteración al propietario del producto e interesados Stakeholders, con la supervisión del Scrum Master, donde se analiza y revisa los siguientes puntos:

- Los pros y contra del incremento del producto (Product Increment).
- Avance y lo que aún falta por completar.

- Dificultades halladas y soluciones aplicadas.
- Preguntas y dudas acerca del trabajo completado presentado.
- Estado actual del Product Backlog y posibles nuevas historias de usuarios
- Lo que posiblemente se realizaría en el siguiente Sprint.

E. Sprint Retrospective - Retrospectiva de Sprint

Realizada después de la Sprint Review y antes del siguiente Sprint Planning. Es una reunión que dura 3 h (para un sprint de un mes) en la cual, el Scrum Master y el Equipo de desarrollo, analizan, en base al marco de proceso y prácticas de Scrum, cómo se ha llevado el trabajo, las relaciones de personas, el manejo de herramientas y el mismo proceso en el último Sprint. Se hallarán las fortalezas y los aspectos a mejorar o modificar a fin de incrementar la productividad en la siguiente iteración.

HERRAMIENTAS DE APOYO (Schwaber K. y Beedle M,2001, p.56)

Existen herramientas que sirven como soporte y guía en algunos aspectos del proceso. Entre ellos tenemos los siguientes:

A. Tablero Kanban

Es una herramienta de la Metodología Kanban, que permite una mejor visualización del avance de la iteración. Posibilita la comunicación e interacción en las reuniones de seguimiento de Sprint.

B. Gráfico Burn-up

Es un gráfico que permite al Propietario del Producto hacer un seguimiento del avance alcanzado por el Equipo de desarrollo hasta el último Sprint, comparado con la estimación de esfuerzo total necesario previsto en el Product Backlog para culminar el proyecto.

C. Gráfico Burn-down

Gráfico de ayuda para el Equipo de desarrollo, quien realiza un rastreo del avance diario en el Sprint, y además revisando el esfuerzo faltante para completar las tareas de la iteración. Las desviaciones significativas del avance real con el estimado, permiten identificar si ha habido subvaloración o sobrevaloración del esfuerzo estimado para el Sprint, pudiéndose tomar la decisión si se aumenta o disminuye tareas en el Sprint Backlog.

D. Planning Poker - Estimación de Póquer

El método fue descrito por primera vez por James Grenning en 2002 y más tarde se volvería más popular y comercial por Mike Cohn con el libro Agile Estimating and Planning. Es una herramienta de XP. Es una técnica para calcular una estimación de esfuerzo a tareas de un Sprint Backlog, basada en el consenso del equipo de desarrollo. Utiliza cartas que tienen como números la Serie de Fibonacci. En resumen, cada participante, de acuerdo a determinados criterios de trabajo del equipo, escoge de la cantidad de cartas que tiene en su poder, una cuyo número (de la serie de Fibonacci) que, a su juicio, corresponde con la estimación de esfuerzo que él asigna a una tarea. Los resultados son consensuados y finalmente se obtiene una estimación final. En caso se decida que el esfuerzo de una tarea es mayor al límite de tamaño de una tarea, se procede a replantear o desglosarla.

2.1.28 Programación extrema XP

Del inglés Extreme Programming es una metodología ágil de desarrollo de software planteada por Kent Beck en el año 1999, para él la mejor metodología era un proceso que enfatiza la comunicación dentro del equipo, que la implementación fuera sencilla, que el usuario tenía que estar muy informado e implicado y que la toma de decisiones tenía que ser muy rápida y efectiva. Está enfocada en el modelado de datos orientada a objetos y solo se trabaja 40 horas por semana. XP unifica prácticas conocidas desde los inicios del desarrollo de software, que reunidas buscan satisfacer al cliente con software de desarrollo sencillo, facilitar la respuesta a los cambios que pueden experimentar las necesidades del cliente en el tiempo, maximizar la productividad del grupo de trabajo y ofrece el software que necesita el cliente a medida que lo necesite. (Beck K. y Andrés C, 2004, p.10)

VALORES (Beck K. y Andrés C, 2004, p.14)

A. Comunicación: Los integrantes del grupo de proyecto deben mantener una comunicación constante, puesto, que varias de las prácticas empleadas en XP, requieren de comunicación entre clientes, gerentes y desarrolladores.
B. Simplicidad: Se debe desarrollar la funcionalidad planificada para hoy y esperar para el futuro, que intentar hacer más funcionalidad, y luego tener que deshacer porque un requerimiento no se necesitó, con la demanda de costo, tiempo y

esfuerzo que ha implicado en total.

C. Retroalimentación: Dentro del proyecto, debe existir un flujo de retroalimentación permanente. La relación de la retroalimentación con la comunicación y con la simplicidad es directa con respecto al beneficio, pues mientras mayor sea la retroalimentación, aumentan los otros dos valores y viceversa.
D. Valentía: En XP se requiere valentía para afrontar los problemas que se susciten, tomando decisiones incluso radicales, que a la larga resultaría más beneficioso para el proyecto. Se necesita valentía para asumir retos y afrontarlos, introducir cambios cuando las cosas no funcionan.
E. Respeto: Se respeta el trabajo de todos los integrantes del equipo XP

ROLES (Beck K. y Andrés C, 2004, p.19)

El equipo debe ser como máximo de 12 personas.

A. Programador - Programmer: Desarrolla el diseño, código y las pruebas de unidad e integración del software, considerando simplicidad y refactorización. Debe tener capacidad de comunicación para programar en pareja. Su rol en XP, implica, además comunicación permanente con el equipo.
B. Cliente - Customer: Debe estar presente ya que es miembro del equipo de XP, responsable de relatar y escribir las Historias del Usuario, fijándose prioridades, indicando el orden de inclusión en las entregas, cambiarlas o modificarlas y de definir las pruebas funcionales para las versiones ejecutables, decide cuando un requisito es satisfecho o no. Puede personificar a un grupo de personas interesadas en el sistema, sin embargo, es su obligación representar fielmente sus necesidades.
C. Encargado de Pruebas - Tester: Responsable de crear y realizar las pruebas funcionales con el cliente, las pruebas unitarias, de publicar los resultados y asegurar el funcionamiento correcto de las herramientas de prueba.
D. Encargado de seguimiento - Tracker: Brinda retroalimentación al equipo de proyecto respecto comparaciones, estadísticas, desviaciones entre los tiempos y recursos estimados y reales. Recomienda soluciones o cambios para corregir defectos, optimizar la productividad o adaptarse a los cambios de objetivo, requerimientos o entorno en las siguientes iteraciones. Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado. Valora si los objetivos se pueden

alcanzar con los recursos disponibles (tiempo, personal) o si es necesario hacer algún cambio.

E. Entrenador o Tutor - Coach: Con amplio conocimiento y experiencia sobre valores, principios, prácticas y proceso de Extreme Programming, responsable de capacitar, guiar al equipo, detectar desviaciones, ofrecer soluciones dentro de este marco. El entrenador tratará de orientar buscando la madurez del grupo en XP, interviniendo lo menos que se pueda en el desarrollo y de la manera más cauta posible. Se pretende que el equipo, a partir de ideas, trabaje en soluciones, en vez que el entrenador lo resuelva todo, evitando de este modo el riesgo de dependencia.

F. Consultor

   Es externo al equipo XP y experto en un tema en específico, requerido por el equipo de proyecto, cuando se necesite información especializada para resolver un problema. Cabe aclarar que, al igual que con el entrenador, el consultor da pautas acerca de cómo se podría llegar una solución, sin llegar a depender de este último para alcanzar la respuesta.

G. Gestor - Big Boss

   El gestor es el gerente del proyecto, nexo entre el cliente y el equipo de desarrollo. Garantiza que las condiciones de trabajo del equipo sean las óptimas para su mejor desempeño. En virtud del cumplimiento de esta función, mantiene contacto con el equipo recogiendo sus posiciones con respecto a cómo afectaría al proyecto un cambio de escenario, estudiados y finalmente tomando una decisión. Se mantiene al tanto de los avances y resultados. Identifica la relación entre éstos con las prácticas y formas de trabajo adoptadas por el equipo de desarrollo.

HERRAMIENTAS (Beck K. y Andrés C, 2004, p.33)

XP considera las siguientes herramientas:

A. Historias de Usuarios: Especifica los requisitos del cliente funcionales o no funcionales que requiere el sistema. Para ser escritos no necesitan un nivel de especificación compleja al inicio del proyecto. Los detalles serán divulgados con el equipo de desarrollo en la misma etapa de implementación. Las historias de usuario se descomponen en tareas de programación y se asignan a los programadores para ser implementadas durante una iteración. Responden a tres preguntas: Quién se beneficia

ósea los *actores* (Como). Que se quiere hacer indica la *funcionalidad* (Quiero). Cuál es el *beneficio* el objetivo para el negocio (Para).

B. Tarjetas CRC Clase Responsabilidad Colaboración

Las reuniones de programadores utilizan Tarjetas Clase Responsabilidad y Colaboración. Cada tarjeta corresponde a un objeto, en la cual se agrega su responsabilidad y las clases con las que interactúa para cumplir con dicha responsabilidad. Su utilidad reside en que todo el equipo de desarrollo puede participar del diseño, sin necesidad de profundizar (al menos de manera previa) en análisis más complejos.

C. Planning Poker - Estimación de Póquer

El método fue descrito por primera vez por James Grenning en 2002 y más tarde se volvería más popular y comercial por Mike Cohn con el libro Agile Estimating and Planning. Es una técnica para calcular una estimación de esfuerzo a tareas, basada en el consenso del equipo de desarrollo. Utiliza cartas que tienen como números la Serie de Fibonacci. En resumen, cada participante, de acuerdo a determinados criterios de trabajo del equipo, escoge de la cantidad de cartas que tiene en su poder, una cuyo número (de la serie de Fibonacci) que, a su juicio, corresponde con la estimación de esfuerzo que él asigna a una tarea. Los resultados son consensuados y finalmente se obtiene una estimación final. En caso se decida que el esfuerzo de una tarea es mayor al límite de tamaño de una tarea, se procede a replantear o desglosarla.

FASES (Beck K. y Andrés C, 2004, p.67)

A. Fase de Exploración

En esta fase inicial, cliente y equipo de desarrollo realizan actividades de preparación, estrechamente interrelacionadas (los clientes y programadores mantendrán constante comunicación, recibiendo retroalimentación) para la producción del sistema. El Cliente realiza la elaboración de historias de usuario primarias, en las que el cliente mejorará su habilidad y conocimiento para redactarlas. Para mejorar la composición de las historias de usuario, el cliente se retroalimenta de las necesidades de los desarrolladores. Al término de la fase, las historias de usuario estarán listas para las reuniones de planificación.

El Equipo de desarrollo realiza la experimentación con diferentes tipos de tecnologías, técnicas para ejecutar tareas y arquitectura, evaluando y comparando características, desempeño, límites y riesgos y escogiendo las mejores para desarrollar el software. Con esta información y la presentada en las historias de usuario, podrán incrementar su capacidad para estimar el tiempo que tomarán las tareas de programación de este proyecto en particular. El equipo de desarrollo estimará el tiempo de implementación de las historias de usuario en un mínimo de una semana y en un máximo de tres. Fuera de este intervalo, se deberá replantear, juntar con otras historias, o dividir la historia en unas más pequeñas. La duración de esta fase es de aproximadamente 2 a 4 semanas, obedeciendo al conocimiento previo de los programadores de la tecnología a aplicar.

B. Fase de Planificación

En esta fase se realizan reuniones permanentes. Plan de entregas: Es la reunión para la creación del plan de entregas, los clientes asignan prioridades a las historias de usuario. Luego, en conjunto con las estimaciones de tiempos realizados por los programadores, se confecciona un cronograma de entregas, base para el plan de iteraciones. Posterior a algunas iteraciones, se pueden observar desviaciones en los objetivos o en los avances del plan de entregas, que precisarán de ajustes.

Plan de iteraciones: Antes de una iteración, se convoca a una reunión para planificar el trabajo a realizar. Las historias que han sido incluidas para esta entrega y las historias que no pasaron las pruebas de aceptación de la última iteración, son analizadas por los desarrolladores y divididas en tareas y luego estimadas en días de trabajo (idealmente entre 1 y 3 días). De acuerdo a la velocidad de desarrollo recolectada de iteraciones anteriores, se puede renegociar el plan de entregas.

Reuniones diarias de seguimiento: Son de corta duración, donde los participantes comunican sus avances y las dificultades que van hallando. Se analizan las posibles soluciones. Para entender mejor los problemas se pueden valer de una computadora y una pantalla. Para agilizar la dinámica y acortar su duración, el modo en que se lleva a cabo es teniendo todos de pie.

C. Iteraciones

Aquí las historias de usuario son implementadas en versiones ejecutables. El cliente, como miembro del equipo de desarrollo, profundiza en el detalle de las historias que se

trabajarán según el plan de entregas. Antes de culminar la iteración, se ejecutan las pruebas funcionales por parte del cliente, concluyendo cuáles historias son completadas con éxito, o requieren ser enviadas a la siguiente iteración para su corrección.

Diseño: El diseño debe ser sencillo, se diseña solo lo que realmente sea necesario en el presente. Guía la implementación de cada Historia del Usuario. Cuando sea pertinente se debe refactorizar el código (estructurar el código ordenando y organizando las líneas de código fuente, sin alterar su funcionalidad). Utiliza tarjetas CRC (clase, responsabilidades, colaboradores) que identifican y organizan las clases orientadas a objetos que son relevantes para el incremento actual de software. Se diseña un prototipo de la interfaz de usuario y de la base de datos.

Codificación: Se realiza mediante Programación en Parejas. Programan según Estándares de Codificación. Cada pareja desarrolla una tarea de la Historia del Usuario y diseña pruebas unitarias para cada historia del usuario. Implementa e integra el código de dicha tarea. Cuando se termina la codificación de una Historia de Usuario completa se integra con el trabajo de los demás programadores, realizando una Integración Continua. Nadie es dueño del código, cualquier programador puede cambiar el código en cualquier momento porque existe Propiedad Colectiva del Código. Se realiza refactorización al código, para mantener limpio y legible su estructura

D. Producción

Al terminar toda la fase de iteraciones, el software se deriva a producción. En iteraciones planificadas, de generalmente una semana, se realizan pruebas que certifiquen que las funciones se ejecuten adecuadamente y los ajustes que demanden los cambios que aparezcan. El riesgo es mayor con respecto a las mejoras, adiciones y modificaciones. Entonces, se debe evaluar con cautela si los complementos agregan valor o no al producto.

Prueba de Aceptación: Se realizan Pruebas Unitarias Continuas para verificar la funcionalidad de las tareas de las Historias del Usuario y detectar los errores o desviaciones. Las pruebas unitarias se automatizan para que puedan ejecutarse repetidas veces. Se desarrollan las aplicaciones independientes de realizar las pruebas funcionales y las pruebas no funcionales. Las pruebas se realizan antes y después de la Integración Continua de código al sistema. Se realizan pruebas de integración, pruebas de validación,

pruebas de aceptación del cliente. Si se encuentra un bug (error) se crea un test para comprobar que no vuelve a aparecer.

E. Mantenimiento

Se exploran nuevas tecnologías, se idean más recodificaciones, por medio de la resolución de solicitudes de soporte al cliente (incluso elaborando nuevas historias de usuario), para enfrentar los cambios que aparecen mientras la primera versión del software se encuentra en funcionamiento.

F. Muerte

Cuando el cliente está satisfecho con el producto y no tiene más características por agregar o no tiene más historias de usuario que entregar. En ese caso se debe redactar un documento guía de todo el sistema para proyectos futuros sobre el programa. Razones negativas para finalizar el proyecto son que el sistema tenga muchos errores y sea imposible hacer mejoras o cuando el costo de mantenimiento del software es insostenible para la empresa porque no devuelve beneficios económicos por su uso a dicha organización.

2.2 Marco Referencial

2.2.1 Carrera de Ingeniería de Sistemas

Como caso de estudio para esta investigación, se tomó como referencia el contexto de la Carrera de Ingeniería de Sistemas de la Facultad de Ciencias y Tecnología de la Universidad Mayor Real y Pontificia de San Francisco Xavier de Chuquisaca. La carrera cuenta con un plan de estudios dividido en diez semestres. En el noveno semestre se cursa la materia de Taller de Grado SHC 180, en la cual se elabora el Perfil del Proyecto de Grado hasta su aprobación por un tribunal conformado por docentes de la misma carrera. En el décimo semestre se cursa la materia de Trabajo de Titulación SHC190, en el cual se elabora el Proyecto de Grado hasta su aprobación por un tribunal conformado por docentes de la misma carrera y/o profesionales externos según reglamento.

La Facultad a la que pertenece esta carrera, tiene actualmente de forma oficial y vigente un Reglamento de Graduación para carreras de Licenciatura y Técnico Superior de la Facultad de Tecnología. En el que establece que los trabajos de titulación de proyecto de grado deben realizarse de forma individual y en un semestre académico. El semestre

académico corresponde a 20 semanas académicas, según el Sistema Informático de esta universidad.

En la gestión 2021, el contexto académico fue virtual por la Pandemia mundial del COVID, en este sentido se percibe que ha disminuido el número de estudiantes de todos los semestres, por diversos factores, económicos, salud entre otros.

2.3 Marco Jurídico o Legal

El Marco Jurídico o Legal de esta investigación está en base al Reglamento General de Tipos y Modalidades de Graduación del Sistema de la Universidad Boliviana y en el Reglamento General de Graduación para Carreras de Licenciatura y Técnico Superior de la Facultad de Tecnología. Ambos actualmente vigentes y oficiales.

2.3.1 Reglamento General de Tipos y Modalidades de Graduación del Sistema de la Universidad Boliviana (ver anexo 1)

"Título I - Disposiciones Generales

Capítulo Único: Objetivos y Ámbito de Aplicación

Artículo 2: "Todas las Carreras y Programas de la Universidad Boliviana deberán incorporar los correspondientes tipos y modalidades de graduación de acuerdo a su correspondencia contenida dentro del presente reglamento en sus planes de estudios a partir de la gestión académica 2000"

Artículo 4: "Los tipos y modalidades de graduación deben contemplar aspectos relacionados a la solución de problemas sociales reales y demandas actuales y futuras de la sociedad, de acuerdo a las características de cada carrera y programa""

"Título II - Tipos y Modalidades de Graduación Capítulo I: De los Tipos y las Modalidades

Artículo 5: "Los tipos y modalidades de graduación que adopta la Universidad Boliviana para los diferentes niveles académicos se describen en:

- Licenciatura: Tesis de grado, Proyecto de grado, Examen de grado, Examen de Contenidos, Examen de Expedientes, Examen Clínico, Internado rotatorio, Trabajo dirigido, Externo, Interno (Adscripción), Por excelencia, Rendimiento Académico, Reconocimiento a la calidad.

- Bachiller Universitario en Ciencias o Artes Directa (Conclusión satisfactoria del plan de estudios)
- Técnico Universitario Superior: Pasantía, Monografía, Proyecto de grado técnico,
- Tesina
- Técnico Universitario Medio (Programa) Directa. (Conclusión satisfactoria del plan de estudios)""

"Capítulo II: Del Tiempo y Ubicación de los tipos y modalidad de graduación *Artículo 7*: "El tiempo de los tipos y modalidades debe estar al menos dentro de las 400 a 600 horas académicas, para optar a los grados de Técnico Superior y Licenciatura respectivamente, estas horas deberán ser programadas en los dos últimos semestres de la gestión académica""

"Capítulo III: De las Definiciones Operativas

Artículo 9: "Proyecto de Grado para Licenciatura, Es el trabajo de investigación, programación y diseño de objetos de uso social y que cumple con exigencias de metodología científica con profundidad similar al de una tesis""

"Título III - Del Proceso de Graduación

Capítulo II: De los Componentes Derechos y Obligaciones,

Artículo 27: "Componentes que intervienen en la modalidad de graduación son: Los postulantes, Los tutores, El tribunal"

Artículo 28: "De los postulantes, Son postulantes todos los estudiantes que, cumpliendo con todos los requisitos estipulados en cada carrera y en el nivel correspondiente, se inscriben a una modalidad de graduación"

Artículo 29: "Los Tutores, En los tipos y modalidades de graduación podrán ser tutores los: Docentes del sistema universitario; Podrán ser designados tutores externos profesionales con grado académico universitario igual o superior al que aspira el estudiante previa aceptación del mismo y designación del consejo de carrera y/o su equivalente. Los tutores deberán asumir la tarea de orientar y guiar a un postulante en el cumplimiento de las exigencias teórico metodológicas del trabajo que implique la modalidad de graduación"

Artículo 31: “Son derechos de los Postulantes: c. Proponer el trabajo de graduación en forma individual, de acuerdo a reglamento específico”

Artículo 33: “Son obligaciones de los tutores:

a) Orientar y guiar en la elaboración de los trabajos de graduación del postulante
b) Realizar las revisiones previas a la defensa y/o trabajo de graduación para evaluar y determinar el nivel de suficiencia que permita a los postulantes a ingresar en la siguiente etapa.
c) Participar en la defensa oral del trabajo de graduación, para absolver dudas que se plantean a propósito del tema y su desarrollo.
d) Solicitar el cambio de uno o más miembros del tribunal, por causales justificadas.
e) Formar equipo de asesores para el seguimiento y evaluación de etapas intermedias de la modalidad de graduación elegida, si el caso así lo requiere
f) Exigir al postulante el cumplimiento de todos los requisitos por la modalidad de graduación elegida
g) Todos los docentes de la universidad tienen la obligación de ser tutores de los estudiantes que así lo soliciten de acuerdo a su especialidad, competencia y disponibilidad de tiempo en cumplimiento de las resoluciones del IX congreso””

“Título IV Disposiciones complementarias y transitorias Capítulo II Disposiciones Transitorias

Artículo 48: “Los estudiantes que aprobaron la totalidad de las asignaturas del plan de estudio, podrán inscribirse en uno de los Tipos o Modalidades Vigentes en su carrera, previo cumplimiento de los requisitos previstos en el presente reglamento””

Interpretación del Reglamento

En función a este contexto jurídico, se percibe una normativa general que se debe aplicar a todas las Universidades de Bolivia, emanada por el Sistema de Universidades de Bolivia, para los Reglamentos de Graduación de las diferentes universidades a nivel nacional. Donde una de las Modalidades de Graduación para el Grado Académico de Licenciatura es el Proyecto de Grado, en el que uno de los tipos de trabajos es la programación, indica que debe ser realizado en forma individual dentro de un semestre académico; también menciona los componentes que intervienen en esta modalidad.

2.3.2 Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología (ver anexo 3)

"Capítulo I Generalidades

"El XII Congreso Nacional de Universidades ha aprobado y emitido el Reglamento General de Tipos y Modalidades de Graduación. Este reglamento en los artículos 1 al 4, mencionan que las modalidades de graduación constituyen parte integrante del Plan de Estudios de cada Carrera, es decir, representan parte del Proceso de Enseñanza Aprendizaje en cada Unidad Académica. En cumplimiento de este reglamento, las diferentes carreras de la Facultad de Tecnología han incluido sus modalidades de graduación en los respectivos planes de estudio""

"Capítulo II Modalidades de Graduación: Definición, Objetivos y Propósitos

Artículo 1: "Las modalidades de Graduación en la Facultad de Tecnología, son las siguientes: Nivel de Licenciatura: Excelencia académica, Tesis, Proyecto de Grado, Trabajo Dirigido Externo, Trabajo Dirigido Interno"

Artículo 2: "Definiciones: El Proyecto de Grado Es el trabajo de investigación, programación y diseño de objetos de uso social y que cumple con exigencias de metodología científica con profundidad similar al de una tesis"

Artículo 3: "Los trabajos de graduación propuestos en cualquiera de las modalidades de graduación deben estar relacionados con las asignaturas propias de cada carrera, además del perfil profesional y deberán ser elaborados por el estudiante en forma personal y original""

"Capítulo III El Sistema de Graduación - Docentes

Artículo 8: "Los docentes de la asignatura de trabajo de titulación son docentes con conocimientos en las disciplinas de especialidad de la facultad.

Artículo 9: "Son funciones de los docentes de Trabajo de Titulación: Informar a los estudiantes sobre la metodología de trabajo en la asignatura y sobre formatos y estructura de elaboración del informe de trabajo de titulación"

"Capítulo III El Sistema de Graduación - Tutores

Artículo 10: "El Tutor es un profesional que posee grado académico igual o superior al que postula el estudiante"

Artículo 12: "Sus funciones son:

- Asesorar de forma individualizada de uno hasta tres trabajos de graduación, desde la elaboración del perfil hasta la defensa pública del trabajo y su aprobación. Excepcionalmente podrá tutorar más trabajos en función de las características y necesidades de cada carrera.
- Coordinar con otros profesionales para efectuar alguna interconsulta, cuando el tema lo requiere
- Participar al docente de la asignatura de trabajo de titulación sobre el cumplimiento
- cualquier irregularidad en la elaboración del trabajo de graduación.
- Establecer reuniones periódicas con sus asesorados para efectuar el seguimiento de los trabajos de graduación, según cronograma elaborado en cada semestre.
- Participar en la defensa oral del trabajo de graduación, pudiendo realizar las aclaraciones que sean necesarias sobre el alcance del trabajo presentado. Su participación en el acto tendrá carácter informativo, no pudiendo participar en la calificación final.""

"Capítulo IV Estructura y Características del Trabajo de Graduación

Artículo 23: "El documento de trabajo de graduación tendrá la siguiente estructura general: Resumen, Introducción, Cuerpo del trabajo o texto, Conclusiones y recomendaciones, Bibliografía, Referencias Bibliográficas, Anexos""

"Capítulo V Procedimientos y Plazos

Artículo 25: "El procedimiento que seguirá un trabajo de graduación será realizado en las siguientes fases: Elaboración y aprobación del perfil, Seguimiento de trabajo de graduación, Defensa privada, Defensa pública"

Artículo 26: "Elaboración y aprobación del perfil: El perfil de trabajo de graduación deberá ser presentado al inicio del semestre al director de Carrera en cuatro ejemplares avalado por el docente de trabajo de titulación y deberá ser defendido ante el tribunal en un plazo no mayor a 15 de días"

Artículo 27: “Seguimiento de trabajo de graduación:

27.10 Los estudiantes que no hubiesen concluido su trabajo de graduación en la fecha señalada, deberán nuevamente inscribirse el siguiente semestre en la asignatura de trabajo de titulación y proseguir con su trabajo hasta su finalización.

27.11 La vigencia del tema de un trabajo de graduación para el postulante, será de un año a partir de su aprobación pudiendo ampliarse un año más previa evaluación.””

“Capítulo VI Selección de Temas y Evaluación del Trabajo de Graduación

Artículo 31: “El alcance y contenido del tema a ser estudiado será tal que permita su conclusión en un semestre académico.””

“Capítulo VII Presentación Escrita del Trabajo de Graduación

Artículo 40: “El postulante deberá presentar tres ejemplares empastados del trabajo de graduación en limpio y en formato digital para la presentación oral, debiendo para el efecto cumplir las normas señaladas en el presente documento.”

Artículo 42: “El documento del trabajo de graduación será elaborado en un procesador de texto y tendrá el formato señalado en Anexo II.””

Interpretación del Reglamento

En funciona a este contexto jurídico, se percibe que en el reglamento establecido por la Facultad de Ciencias y Tecnología se establece como una de las Modalidades de Graduación para el Grado Académico de Licenciatura, al Proyecto de Grado, que debe ser realizado en forma individual dentro de un semestre académico; también identifica al tutor y sus funciones, el documento del trabajo de graduación y características relacionadas. Además, establece que el estudiante puede escoger la programación u otro tipo de trabajos a realizar como proyecto de grado.

2.4 Marco Histórico

Las metodologías de Desarrollo de Software, han experimentado un proceso histórico y evolutivo que inicia aproximadamente en los años 40 con la aparición de las primeras computadoras, entonces no se contaban con parámetros ni estándares. El desarrollo de software era prácticamente empírico y artesanal lo que llevó a que una buena parte de los

proyectos fallan en cubrir las expectativas de los usuarios, así como en entregas extemporáneas y presupuestos excedidos.

La respuesta para superar esta crisis de desarrollo de software fue la adopción de modelos y metodologías clásicas o tradicionales, que progresivamente fueron incorporando estándares, controles y formalidades al desarrollo de software.

Con la llegada de Internet, surgen proyectos caracterizados por requerimientos cambiantes y tiempos de entregas breves, para los que las metodologías tradicionales existentes no se adaptan idóneamente, entonces surgen las metodologías ágiles, enfocadas en el desarrollo de software iterativo e incremental, en equipos de desarrollo, constante comunicación con los usuarios, entregas tempranas y adaptación a los cambios.

Actualmente, existen dos grandes enfoques de metodologías de desarrollo de software: Metodologías ágiles que siguen un enfoque basado en el Manifiesto Ágil y metodologías tradicionales que fueron las primeras en ser utilizadas.

CAPÍTULO III

3 METODOLOGÍA DE LA INVESTIGACIÓN

En este capítulo se menciona la metodología de la investigación que se ha utilizado para realizar esta tesis. Se abordan los elementos básicos que guiarán esta investigación de manera metodológica y didáctica durante el proceso de investigación.

Contiene el Método de Investigación, el Tipo de Investigación, la identificación del Universo o Población de Estudio, la Determinación y Elección de una adecuada Muestra, el Sujeto vinculado a la Investigación, las Fuentes y Diseño del Instrumento de Relevamiento de Información, el Procesamiento y Análisis de Datos; según la Guía Metodológica para la Elaboración de Tesis de Grado Programas Académicos Postgrado.

3.1 Método de Investigación

El método de investigación que se utilizó para esta investigación es el Deductivo Inductivo. Porque: La inducción es una forma de razonamiento a través del cual se pasa de un conocimiento de cosas particulares a un conocimiento más general que va a reflejar lo que hay de común en esos fenómenos individuales. (Ramírez, 2013, p.54) Mientras que la deducción es el proceso que se aplica al pasar de un conocimiento general a uno particular y/o conocimiento menos general. (Ramírez, 2013, p55). Este método se aplicó en la investigación partiendo de hechos generales para llegar a situaciones específicas y viceversa.

3.2 Tipo de Investigación

El tipo de investigación que se utilizó para esta investigación es Propositiva porque se realizó un análisis del problema y después se propone una alternativa teórica.

La Investigación Propositiva después de un análisis del problema, el investigador plantea una alternativa teórica de transformación, para dar una posible solución. Esta propuesta considera en su estructura relaciones esenciales que se constituyen en una propuesta teórica para pretender resolver el problema y cumplir con el objetivo de la investigación. (Ramírez, 2013, p46)

3.3 Universo o Población de Estudio

El Universo o Población de Estudio para esta investigación son 15 estudiantes de la Carrera de Ingeniería de Sistemas que defendieron en los últimos 5 años su Proyecto de Grado desarrollando software, ante un Tribunal, entre hombres y mujeres. Se considera desde 5 años atrás porque la Maestría de Software Libre es de la gestión 2015.

3.4 Determinación y Elección de la Muestra

Por resultar la población muy reducida, no fue necesario realizar un proceso muestral, en este sentido se utilizó el tipo de muestreo no probabilístico porque la muestra se obtuvo atendiendo al criterio del investigador; así mismo dentro de este tipo de muestreo se realizó un muestreo intencional o deliberado, en tal sentido la muestra quedó conformada del mismo tamaño que la población por 15 estudiantes, con las características ya mencionadas.

3.5 Sujetos Vinculados a la Investigación

Los sujetos vinculados a la investigación son estudiantes universitarios de la carrera de Ingeniería de Sistemas de la Facultad de Ciencias y Tecnología de la Universidad Mayor Real y Pontificia de San Francisco Xavier de Chuquisaca.

3.6 Fuentes y Diseño del Instrumento de Relevamiento de Información

3.6.1 Fuentes

La fuente utilizada en esta investigación es de dos tipos: Fuente Primaria, porque se utilizó una encuesta de aplicación directa a los sujetos de la investigación, para obtener información y los reglamentos actualmente vigentes. Fuente secundaria, porque se realizó un análisis bibliográfico relacionado con el contenido de la investigación, para respaldar los fundamentos teóricos.

3.6.2 Diseño del instrumento de relevamiento de Información

Técnica de Investigación: La recogida de datos es una fase importante dentro del proceso de investigación, para ello se utilizó la Técnica de la Encuesta

Encuesta: Es una técnica de adquisición de información mediante un cuestionario previamente elaborado, a través del cual se puede conocer la opinión o valoración del

sujeto seleccionado en una muestra sobre un asunto o tema dado. Para realizar la Encuesta se utilizó como Instrumento el Cuestionario de la Encuesta

Instrumento: Es la herramienta o recurso auxiliar que permite el acopio y colecta de datos, son considerados instrumentos físicos. En este sentido, se diseñó el conjunto de preguntas que conformarán el Cuestionario de la Encuesta.

Cuestionario de la Encuesta: Este cuestionario fue llenado por las personas que conforman la muestra de la investigación. El diseño es el siguiente: Objetivo: Recabar información sobre la realización de proyectos de grado mediante el desarrollo de software.

Estimado (a) estudiante: Esta encuesta que te solicito contestar, solo tiene fines académicos para un estudio investigativo, por lo que la información que proporciones es absolutamente confidencial y sus respuestas tienen el carácter de anónimo y no necesita poner su nombre por lo que puede ser lo más sincero posible. La información que facilite es muy valiosa. Espero tu colaboración. Gracias.

1. ¿Qué metodología o marco de trabajo de desarrollo de software utilizaste en tu Proyecto de Grado?

- Metodología o Marco Ágil
- Metodología o Marco Tradicional
- Metodología o Marco Híbrida
- Ninguno
- Todas

2. ¿Conoce alguna metodología o marco de trabajo ya sea ágil o tradicional de desarrollo de software que esté enfocado específicamente al desarrollo de software de forma individual en 20 semanas?

- Si conozco
- No conozco

3. ¿Conoces todas las funciones reglamentarias que debe realizar un tutor/a de Proyecto de Grado?

- No conozco ninguna
- Conozco algunas

- Si conozco todas

4. ¿Tuviste algún problema durante la realización de tu Proyecto de Grado, en el desarrollo de software, relacionado con el uso de la metodología o marco de trabajo ya sea ágil o tradicional que escogiste utilizar?

- Ningún problema
- Algunos problemas
- Muchos problemas

5. ¿Cuántas horas le dedicas al día para realizar tu Proyecto de Grado?

- Menos de una hora
- 1 - 4
- 5 - 8
- 9 - 12
- Más de 12 horas

6. ¿Con que frecuencia te reúnes con tu tutor/a asignado para tu Proyecto de Grado?

- No sé si tengo tutor asignado
- No tengo tutor asignado
- No me reúno
- Diariamente
- Semanalmente
- Mensualmente
- Dos veces al Semestre
- Otras frecuencias

7. ¿Con qué frecuencia te reúnes con el cliente/usuario del software que estás desarrollando para tu Proyecto de Grado?

- No me reúno
- Diariamente
- Semanalmente
- Mensualmente
- Dos veces al Semestre
- Otras frecuencias

8. ¿Con qué frecuencia autoevalúan el Proyecto de Grado que estás desarrollando y a ti mismo en relación a tu trabajo académico?

- o No me autoevaluo
- o Diariamente
- o Semanalmente
- o Mensualmente
- o Dos veces al Semestre
- o Otras frecuencias

9. ¿La metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado, estaba orientado específicamente al tiempo de realización de un semestre académico de 20 semanas?

- o Si
- o No

10. ¿Realizaste modificaciones, adecuaciones y/o improvisaciones a la metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado?

- o Ninguna
- o Algunas
- o Muchas

11. ¿Conoces el Reglamento de Graduación de la Facultad de Ciencias y Tecnología?

- o Si conozco el contenido
- o Conozco algo del contenido
- o Se que existe, pero no conozco el contenido
- o No sabía que existía y no conozco el contenido

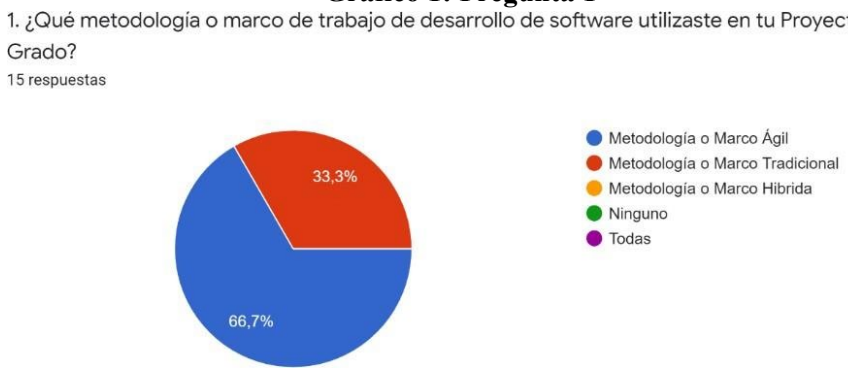
12. ¿El software de tu Proyecto de Grado tiene requisitos del cliente muy cambiantes?

- o Si tiene
- o No tiene

3.7 Procesamiento y Análisis e Interpretación de Información

Pregunta 1: ¿Qué metodología o marco de trabajo de desarrollo de software utilizaste en tu Proyecto de Grado?

Gráfico 1: Pregunta 1



Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 1:

El 66.7% de las respuestas indican que utilizaron en su proyecto de grado una metodología o marco ágil. El 33,3% utilizó una metodología o marco tradicional. Este dato revela que los estudiantes prefieren el enfoque ágil al tradicional.

Pregunta 2: ¿Conoce alguna metodología o marco de trabajo ya sea ágil o tradicional de desarrollo de software que esté enfocado específicamente al desarrollo de software de forma individual en 20 semanas?

Gráfico 2: Pregunta 2

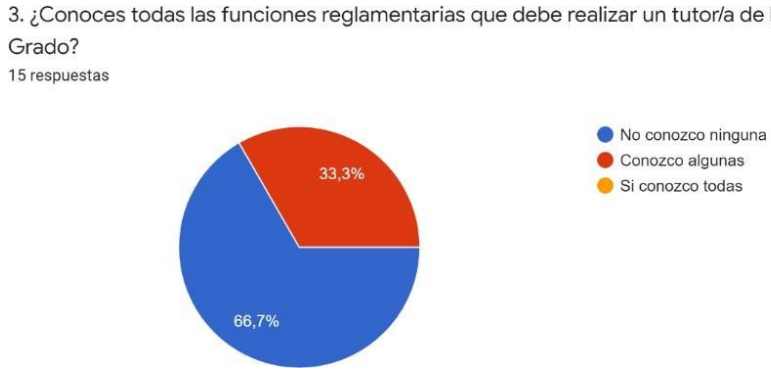
Fuente: Elaboración Propia Análisis de la Gráfica Pregunta 2:

El 100% de las respuestas indican que no conocen específicamente una metodología o marco de trabajo que esté enfocado al desarrollo de software individual en 20 semanas. Este dato revela que los estudiantes durante toda su investigación no encontraron una

metodología o marco de trabajo ya sea ágil o tradicional que esté acorde al reglamento de graduación.

Pregunta 3: ¿Conoces todas las funciones reglamentarias que debe realizar un tutor/a de Proyecto de Grado?

Gráfico 3: Pregunta 3

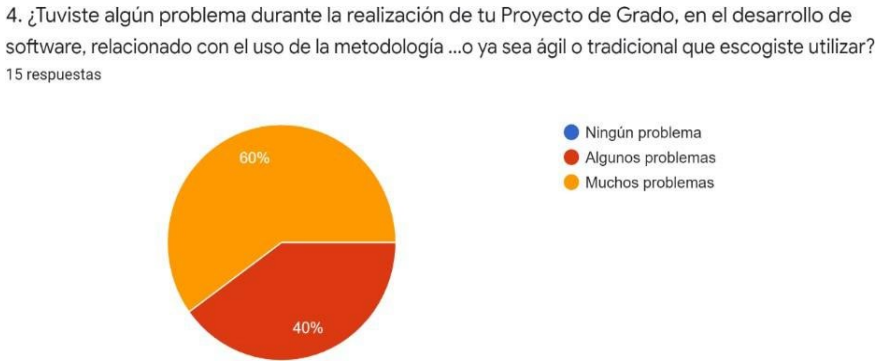


Fuente: Elaboración Propia Análisis de la Gráfica Pregunta 3:

El 66.7% de las respuestas indican que no conocen todas las funciones reglamentarias de un tutor de proyecto de grado. El 33,3% conocen, pero solo algunas funciones. Este dato revela que la mayoría de los estudiantes desconocen las funciones que están establecidas en los reglamentos de graduación para el tutor de proyecto de grado.

Pregunta 4: ¿Tuviste algún problema durante la realización de tu Proyecto de Grado, en el desarrollo de software, relacionado con el uso de la metodología o marco de trabajo ya sea ágil o tradicional que escogiste utilizar?

Gráfico 4: Pregunta 4



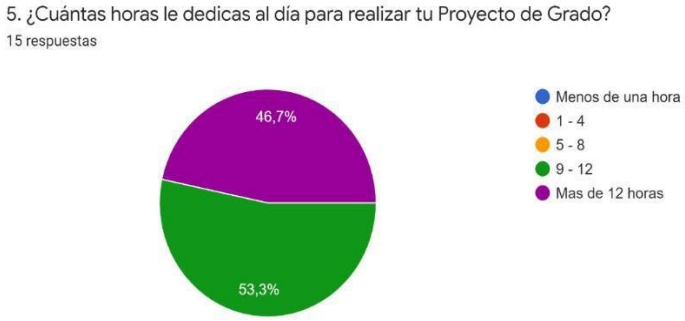
Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 4:

El 60% de las respuestas indican que tuvieron muchos problemas con el uso de la metodología o marco de trabajo ya sea ágil o tradicional que escogieron durante la realización de su proyecto de grado en el desarrollo de software. El 40% indica que tuvieron algunos problemas. Este dato revela en el momento de utilizar la metodología o marco de trabajo ya sea ágil o tradicional que escogió, se enfrentó a problemas al aplicar la misma metodología o marco de trabajo, por la gráfica de la pregunta 2 se puede asumir que uno de los problemas puede ser que estas metodologías o marcos de trabajo específicamente no están orientadas a la realización de software en 20 semanas.

Pregunta 5: ¿Cuántas horas le dedicas al día para realizar tu Proyecto de Grado?

Gráfico 5: Pregunta 5



Fuente: Elaboración Propia Análisis de la Gráfica Pregunta 5:

El 53,3% de las respuestas indican que dedican entre 9 a 12 horas al día en la realización de su proyecto de grado. El 46,7% indica que utilizan más de 12 horas al día. Este dato revela que los estudiantes casi en su mayoría utilizan un promedio de 12 horas al día para realizar su proyecto de grado.

Pregunta 6: ¿Con que frecuencia te reúnes con tu tutor/a asignado para tu Proyecto de Grado?

Gráfico 6: Pregunta 6

Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 6:

El 26,7% de las respuestas indican que se reúnen mensualmente con su tutor. El 26,7% de las respuestas indican que se reúnen semanalmente con su tutor. El 20% indican otras frecuencias de reuniones con su tutor. El 13,3% indican que se reúnen diariamente con su tutor. El 13,3% indican que se reúnen 2 veces al semestre con su tutor. Este dato revela que la mayoría de los estudiantes se reúnen mensual o semanalmente con su tutor, otros prefieren reunirse en otras frecuencias de tiempo, existen algunos que se reúnen muy frecuentemente como es diariamente y otros se reúnen muy pocas veces como ser dos veces al semestre.

Pregunta 7: ¿Con que frecuencia te reúnes con el cliente/usuario del software que estás desarrollando para tu Proyecto de Grado?

Gráfico 7: Pregunta 7

Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 7:

El 26,7% de las respuestas indican que se reúnen mensualmente con el cliente. El 26,7% de las respuestas indican que se reúnen semanalmente con el cliente. El 20% indican otras frecuencias de reuniones con el cliente. El 13,3% indican que se reúnen diariamente con el cliente. El 13,3% indican que se reúnen 2 veces al semestre con el cliente. Este dato revela que la mayoría de los estudiantes se reúnen mensual o semanalmente con el cliente, otros prefieren reunirse en otras frecuencias de tiempo, existen algunos que se reúnen muy frecuentemente como es diariamente y otros se reúnen muy pocas veces como ser dos veces al semestre.

Pregunta 8: ¿Con qué frecuencia autoevalúan el Proyecto de Grado que estás desarrollando y a ti mismo en relación a tu trabajo académico?

Gráfico 8: Pregunta 8

Fuente: Elaboración Propia Análisis de la Gráfica Pregunta 8:

El 26,7% de las respuestas indican que no se autoevalúan a sí mismos ni el proyecto de grado que están realizando. El 20% indican que se autoevalúan mensualmente. El 20% indican que se autoevalúan semanalmente. El 13,3% indican que se autoevalúan diariamente. El 13,3% indican que se autoevalúan dos veces al semestre. El 6,7% indican otras frecuencias de autoevaluación. Este dato revela que el mayor porcentaje de los estudiantes no se autoevalúan a sí mismos ni tampoco evalúan el proyecto de grado que ellos mismos están realizando, el resto de los estudiantes prefiere autoevaluarse semanalmente o mensualmente, algunos lo realizan diariamente y dos veces al semestre, un menor porcentaje se autoevalúa en otras frecuencias.

Pregunta 9: ¿La metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado, estaba orientado específicamente al tiempo de realización de un semestre académico de 20 semanas?

Gráfico 9: Pregunta 9

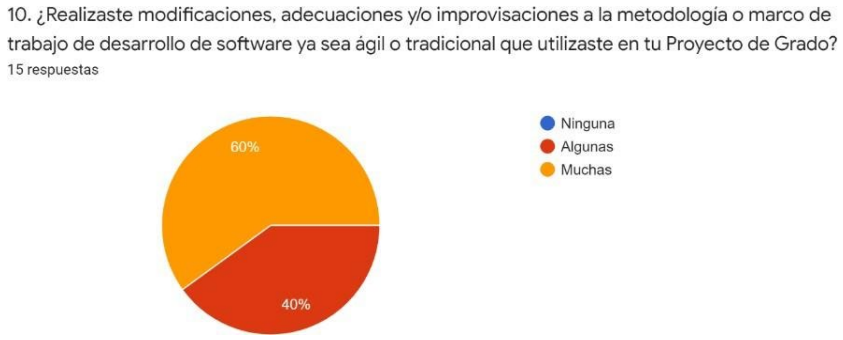
Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 9:

El 100% de las respuestas indican que la metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaron en su proyecto de grado no estaba orientado específicamente al tiempo de realización de un semestre académico de 20 semanas. Este dato revela que los estudiantes no aplicaron una metodología o marco de trabajo que les oriente o guíe en la realización de su proyecto de grado específicamente para 20 semanas.

Pregunta 10: ¿Realizaste modificaciones, adecuaciones y/o improvisaciones a la metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado?

Gráfico 10: Pregunta 10



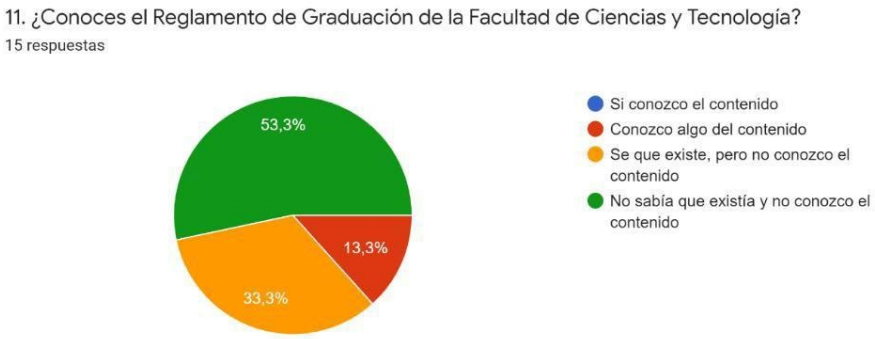
Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 10:

El 60% de las respuestas indican que realizaron muchas modificaciones, adecuaciones y/o improvisaciones a la metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaron en su proyecto de grado. El 40% indica que realizaron algunas. Este dato revela que los estudiantes realizaron modificaciones, adecuaciones y/o improvisaciones ya sea muchas o algunas a la metodología o marco de trabajo ya sea ágil o tradicional de desarrollo de software que utilizaron para poderlo aplicar en su trabajo.

Pregunta 11: ¿Conoces el Reglamento de Graduación de la Facultad de Ciencias y Tecnología?

Gráfico 11: Pregunta 11



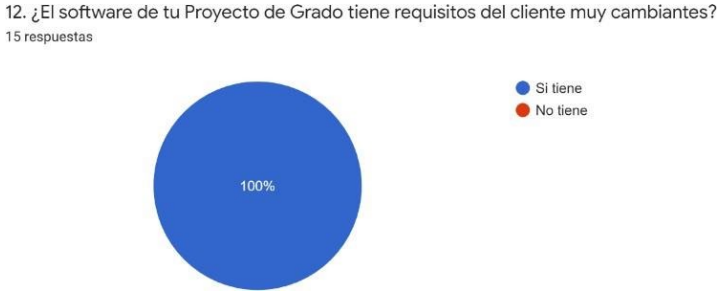
Fuente: Elaboración Propia

Análisis de la Gráfica Pregunta 11:

El 53,3% de las respuestas indican que no sabían que existía y no conocen el contenido del Reglamento de Graduación de la Facultad de Ciencias y Tecnología. El 33,3% indica que saben que existe, pero no conocen el contenido. El 13,3% indica que conocen el contenido. Este dato revela que los estudiantes en su mayoría no conocen que existe un Reglamento de Graduación de la Facultad de Ciencias y Tecnología, algunos si saben de su existencia, pero no conocen el contenido y muy pocos conocen el contenido de dicho reglamento.

Pregunta 12: ¿El software de tu Proyecto de Grado tiene requisitos del cliente muy cambiantes?

Gráfico 12: Pregunta 12



Fuente: Elaboración Propia Análisis de la Gráfica Pregunta 12:

El 100% de las respuestas indican que los requisitos del cliente son muy cambiantes para el software que están desarrollando en su proyecto de grado. Este dato revela que los requisitos del cliente son muy cambiantes, durante el desarrollo del software.

CAPÍTULO IV

4 ESTUDIO COMPARATIVO DE SCRUM, KANBAN Y PROGRAMACIÓN EXTREMA XP

Realizar un estudio comparativo implica un análisis y la síntesis de las similitudes diferencias de Scrum, Kanban y Programación Extrema que comparten un enfoque en común que es el manifiesto ágil.

En este capítulo se realizó un estudio comparativo que evaluará las semejanzas y diferencias existentes entre las metodologías de desarrollo de software analizadas y descritas en el Capítulo Marco Teórico. Los criterios a comparar han sido determinados utilizando como guía el estudio realizado por Abrahamson, Salo, Ronkkainen y Warsta publicado en su libro "Agile Software Development Methods Review and Análisis". Este estudio comparativo permite identificar los elementos de Scrum, Kanban, Programación Extrema que se puedan utilizar en la propuesta.

4.1 Criterios de Comparación

Tabla 6: Criterios de Comparación

Criterios	Sub Criterios	Indicadores	Sub Indicadores
Procesos	Agilidad	Manifiesto Ágil	Utiliza principios del Manifiesto Ágil
		Iterativa	Entregas Iterativas
		Incremental	Desarrollo Incremental
		Flexible a cambios	
		Integración continua de incrementos de software	
	Frecuencia de trabajo.	Tiempo	

	Definición de principios, roles	Principios, Postulados, Valores, Pilares establecidos	
		Roles definidos	Cantidad de Roles
	Definición de entregables y/o artefactos (herramientas)	Cantidad	
		Complejidad	
	Herramientas Adicionales	Compatible con otras herramientas	
Personas	Equipo de trabajo	Número de integrantes	
		Comunicación entre integrantes	Tipo de comunicación predominante
			Formalidad de reuniones
			Frecuencia de retroalimentación
	Clientes / Usuarios	Participación en el proyecto	Tipo de participación
			Disponibilidad a lo largo del proyecto
Organización	Manejo de contratos.	Tipo de Contrato	
Documentación	Generan Documentación	Importancia	
Fuente: Abrahamson, Salo, Ronkkainen y Warsta “Agile Software Development Methods Review and Análisis”			

4.2 Comparativa entre Scrum, Kanban y Programación Extrema

A. CRITERIO PROCESOS

1. Procesos – Agilidad – Manifiesto ágil – Utiliza principios del manifiesto ágil: Los tres están enfocados en el manifiesto ágil por eso se consideran Metodologías o Marcos de Trabajo Ágil.

2. Procesos – Agilidad – Iterativa – Entregas iterativas: Los tres se realizan entregas iterativas.

3. Procesos – Agilidad – Incremental – Desarrollo incremental: Los tres realizan el desarrollo incremental.

4. Procesos – Agilidad – Flexible a cambios: Los tres son flexibles a cambios

5. Procesos – Agilidad – Integración continua de incrementos de software: Los tres realizan integración continua mediante incrementos de software

6. Procesos – Frecuencia de Trabajo – Tiempo: Scrum sugiere realizar sus sprint en máximo un mes, Programación Extrema indica que se debe programar 40 horas a la semana y Kanban no especifica un tiempo determinado.

7. Procesos – Definición de principios, roles – Principios, postulados, valores, pilares establecidos: Los tres tienen principios, pilares, valores o postulados establecidas

8. Procesos – Definición de principios, roles – Roles definidos – Cantidad de roles: Scrum tiene cuatro roles definidos, Programación Extrema tiene siete roles definidos, Kanban no establece roles.

9. Procesos - Definición de entregables y/o artefactos (herramientas) – Cantidad: Scrum tiene tres artefactos definidos, Programación Extrema tiene dos herramientas, Kanban se basa en un Tablero de Actividades

10. Procesos - Definición de entregables y/o artefactos (herramientas) – Complejidad: Los tres tienen entregables y/o artefactos (herramientas) que no tienen alta complejidad

11. Proceso – Herramientas adicionales – Compatible con otras herramientas: Los tres se consideran compatibles para poder utilizar otras herramientas que mejoren su productividad.

B. CRITERIO PERSONAS

1. Personas - Equipo de trabajo – Número de integrantes: En Scrum sugiere que no deben ser equipos muy pequeños ni muy grandes aproximadamente de 5 a 9 la programación la realiza todo el equipo de desarrollo en forma individual; en XP indica no más de 12 personas la programación la realiza todo el equipo de desarrollo en parejas, en Kanban no menciona.

2. Personas - Equipo de trabajo - Comunicación entre integrantes - Tipo de comunicación predominante: En los tres la comunicación predominante es cara a cara en el mismo lugar de trabajo.

3. Personas - Equipo de trabajo - Comunicación entre integrantes - Formalidad de reuniones: En los tres la formalidad de las reuniones de trabajo es baja, se enfocan más en los beneficios de cada reunión.

4. Personas - Equipo de trabajo - Comunicación entre integrantes - Frecuencia de retroalimentación: En los tres la frecuencia de retroalimentación es alta, ya sea entre miembros del equipo de trabajo entre sí y/o con el cliente.

5. Personas – Clientes/usuarios - Participación en el proyecto - Tipo de participación: En Scrum está representado por el Product Owner así que su participación es indirecta, en Programación Extrema su participación es directa porque forma parte del equipo, en Kanban no menciona.

6. Personas – Clientes/usuarios - Participación en el proyecto - Disponibilidad a lo largo del proyecto: En Scrum y XP es alta la disponibilidad de tiempo del cliente o de su representante a lo largo del proyecto, en Kanban no menciona.

C. CRITERIO ORGANIZACIÓN

1. Organización - Manejo de contratos – Tipo de contrato

En Scrum y XP el contrato es de tipo costo objetivo, cronograma objetivo o de beneficios compartidos, en Kanban no menciona

D. CRITERIO DOCUMENTACIÓN

1. Documentación - Generan documentación – Importancia

Los tres realizan la documentación, sin embargo, la importancia que le dan es baja.

E. RESULTADOS DE LA COMPARATIVA

Scrum es un Marco de Trabajo Ágil de Desarrollo de Software que trabaja con un equipo de desarrollo para realizar la programación del software lo que le permite minimizar el tiempo de realización del mismo, no tiene un límite establecido en el cual se debe terminar el producto realizado lo que le da una ventaja de disponer de sus cajas de tiempo con más amplitud, sin embargo, posee herramientas que se pueden adaptar y utilizar en la propuesta.

Programación Extrema es una Metodología Ágil de Desarrollo de Software que trabaja la programación en parejas esto le da una ventaja ya que dos personas juntando sus capacidades de trabajo pueden realizar un mejor proyecto y en menor tiempo del que realizaría una sola persona, no tiene límite establecido en el cual se debe terminar el producto realizado lo que le da una ventaja de realizar todas sus fases en tiempos adecuados, sin embargo, posee herramientas que se pueden adaptar y utilizar en la propuesta.

Kanban es una Metodología Ágil de Desarrollo de Software que trabaja con un Tablero de Kanban basado en Tarjetas de tareas, no menciona la cantidad de personas que forman un equipo, ni menciona tiempo específico, porque se concentra en el flujo de tareas que se debe realizar, visualizando el estado exacto de cada tarea, permitiendo una mejor organización. Este concepto es muy útil y ha sido utilizado en varias metodologías o marcos de trabajo de desarrollo de software, por las ventajas que tiene, por eso es que se consideró también utilizarlo en la propuesta.

4.3 Identificación de los elementos adecuados de Scrum, Programación Extrema y Kanban que se puedan utilizar en la propuesta.

En función al marco teórico, a los resultados de la encuesta, al estudio comparativo y principalmente al propio criterio y experiencia de esta investigación, se identificaron las siguientes herramientas.

A. Scrum

El *Sprint* es el espacio de tiempo que contiene a todos los eventos, es importante que el estudiante tenga un periodo en el que se concentren todas las actividades a realizarse durante el desarrollo del software. Se utilizó el Sprint con el mismo concepto. El nombre

que se le dio es *Periodo* porque por definición esta palabra se refiere a "Espacio de tiempo durante el cual se realiza una acción o se desarrolla un acontecimiento" en el caso de la propuesta también contiene al resto de actividades, sin embargo, el contenido es diferente en relación al Sprint puesto que se propone otros eventos. El objetivo de utilizar el Sprint es que permite agrupar a todas las actividades, eventos para tener mayor claridad y sencillez en la estructura de la propuesta.

El *Sprint Planning* es una reunión de planificación del Sprint, es sustancial que el estudiante organice y planifique la información antes de realizar una Iteración de Trabajo. Se utilizó el Sprint Planning con el mismo concepto, sin embargo, esta reunión se realizará entre el cliente y el estudiante, no con el Scrum Master y Equipo de Desarrollo. El nombre que se le dio es *Reunión Requisitos* porque es una reunión que se realiza con ese motivo de obtener los requisitos del cliente y las prioridades que les asigna a los elementos de la Lista de Requisitos. El objetivo de realizar un Sprint Planning es que el estudiante tenga un espacio de tiempo para coordinar y conocer con detalle los requerimientos del cliente, compartiendo la información cara a cara con el cliente.

El *Daily Scrum* es una reunión donde se inspecciona sobre el trabajo realizado, es primordial que el estudiante inspeccione conscientemente todo el trabajo que ha estado realizando. Se utilizó el Daily Scrum con el mismo concepto, pero esta reunión solo la realiza el estudiante consigo mismo como un espacio de autoevaluación y no con el Scrum Master y el Equipo de Desarrollo. El nombre que se le dio es *Reunión Diaria Autoevaluación*, porque este evento debe realizarse diariamente por el estudiante para autoevaluarse sobre el trabajo que está realizando para su proyecto de grado. El objetivo de la *Reunión Diaria Autoevaluación* es que el estudiante tenga un espacio de tiempo relajado concentrado en sí mismo y evalúe el trabajo realizado, sus capacidades, si existiese alguna dificultad y cómo la podría resolverlo.

El *Sprint Review* es una reunión de revisión del Sprint, es fundamental que el estudiante revise el Incremento del Software realizado en una Iteración de Trabajo para verificar la conformidad del cliente. Se utilizó el Sprint Review con el mismo concepto, sin embargo, esta reunión se realizará entre el cliente y el estudiante, no con el Scrum Master, Equipo de Desarrollo y Product Owner. El nombre que se le dio es *Reunión Retroalimentación* porque es una reunión que se realiza con ese motivo de obtener retroalimentación del

cliente en relación al Incremento de Software; realizado para verificar la conformidad y/o posibles sugerencias de modificación que existiese de parte del cliente. El objetivo de realizar una *Reunión Retroalimentación* es que el estudiante tenga un espacio de tiempo para presentar, explicar con detalle el incremento del software al cliente, compartiendo la información cara a cara, podrá recibir una retroalimentación de parte del cliente para verificar su conformidad y/o posibles sugerencias relacionadas a ese incremento o a nuevos requisitos que pudiesen surgir.

El *Product Backlog* es la lista ordenada de requisitos del software que cambia constantemente, es esencial que el estudiante tenga una única lista de requerimientos del cliente para poderlos ejecutar. Se utilizó el Product Backlog con el mismo concepto, pero para la propuesta no es creada por el Product Owner, sino por el cliente en coordinación con el estudiante. El nombre que se le dio es *Lista de Requisitos*, porque es eso lo que contiene una lista o conjunto de los requisitos cambiantes en el tiempo. Los elementos que contiene la lista también están escritos con el formato de Historias de Usuarios (herramienta de XP). El objetivo de utilizar la *Lista de Requisitos* es que el estudiante tenga documentada una única fuente de información con los requerimientos del cliente que es dinámica porque es flexible a cambios en el tiempo.

El *Sprint Backlog* es la lista ordenada de requisitos de software seleccionados para el Sprint, es fundamental que el estudiante seleccione de toda la Lista de Requisitos según su capacidad de trabajo sólo aquellos elementos que ejecutará en la Iteración de Trabajo. Se utilizó el Sprint Backlog con el mismo concepto, pero se menciona también la realización paralela al software del documento de trabajo de graduación. El nombre que se le dio es *Lista de Requisitos Elegidos*, porque eso es lo que contiene una lista o conjunto de los requisitos elegidos por el estudiante para ser transformados en Incremento de Software y paralelamente ser documentados en el Incremento de Documento. El objetivo de utilizar la *Lista de Requisitos Elegidos* es que el estudiante tenga registrado los elementos que ha seleccionado según su capacidad de trabajo y los tenga concentrados en una única herramienta, para tener mayor control de las actividades que tiene que realizar durante la Iteración de Trabajo.

B. Programación Extrema

Las Historias de Usuario son una adecuada forma de escribir los requisitos de los clientes que son los elementos registrados en la lista de requisitos de la propuesta, ya que tienen una forma sencilla de entender para que el cliente que no necesariamente es del área de programación, pueda redactar y transmitir los requerimientos del software a ser desarrollado.

Se utilizaron estas Historias de Usuario con el mismo formato que tiene. El nombre que se le dio a las Historias de Usuarios es *Requisito del Cliente* porque es eso lo que el estudiante tiene que tener del cliente y escribir para poderlo ejecutar en el desarrollo del software, el cambio del nombre se justifica porque se quiere simplicidad en el léxico de esta propuesta, y porque igual que otras metodologías o marcos de trabajo que utilizan herramientas adicionales de apoyo algunas también les cambian el nombre adecuándose a su propio contexto.

El objetivo de utilizar el *Requisito del Cliente* es que permite redactar de una forma clara, sencilla tanto para el cliente como para el estudiante los requerimientos del software a ser desarrollado.

C. Kanban

El Tablero de Kanban es una adecuada herramienta para visualizar el flujo de trabajo del estudiante mediante las tareas que se colocan en las diferentes columnas, este al igual que sus Tarjetas no tienen un formato específico ya que se puede ir aumentando o disminuyendo la cantidad de columnas acorde a la situación y el detalle de las Tarjetas.

Se utilizó este Tablero de Kanban indicando el contenido de las columnas y el detalle de las tarjetas, que en el caso de la propuesta son los Requisitos del Cliente Elegidos. El nombre que se le dio al Tablero de Kanban es *Tablero de Actividades*, porque refleja las actividades que el estudiante debe ir realizando durante la elaboración de su proyecto de grado y porque el nombre de las columnas y detalle de sus tarjetas han sido adaptadas para el contexto de esta investigación.

El objetivo de utilizar el *Tablero de Actividades* es que permite limitar la cantidad de trabajo en progreso del estudiante, ya que podrá controlarlo de forma visual en un tablero que estará configurado según su capacidad de trabajo.

D. Limitantes encontradas

Scrum, Kanban, ni XP consideran la realización específicamente del documento de trabajo de graduación

Scrum, Kanban, ni XP consideran el tiempo específico de 20 semanas

Scrum, Kanban, ni XP consideran el desarrollo de software con específicamente un solo programador/a, Kanban no menciona la cantidad de personas ni sus roles, pero sí menciona equipo de trabajo

Scrum, Kanban, ni XP consideran específicamente un Tutor/a como parte de su equipo de trabajo

E. Manifiesto Ágil

En relación a los valores del Manifiesto Ágil con la propuesta se tiene:

Tabla 7: Manifiesto Ágil con la Propuesta

N	Valores	Propuesta
1	Individuos e Interacciones sobre Procesos y Herramientas	El Estudiante, El Tutor y El Cliente son muy importantes
2	Software funcionando sobre Documentación Excesiva	El Software y El Documento de trabajo de graduación son importantes, pero este no es extenso ya que está reglamentada.
3	Colaboración con el Cliente sobre Negociación Contractual	El cliente se beneficia con el Software ya que no existe contrato de trabajo ni sueldo para el estudiante.
4	Respuesta ante el Cambio sobre seguir un Plan	Los requisitos del cliente son cambiantes e inestables
Fuente: Elaboración Propia en base al Manifiesto Ágil		

CAPÍTULO V

5 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES DE LA INVESTIGACIÓN

En este capítulo se pretende mostrar un resumen de la información recopilada como resultado de la investigación, las conclusiones a las que se llegaron y algunas recomendaciones sugeridas. Esta estructura de tesis está según el formato establecido por la guía metodológica oficial para la elaboración de tesis de grado, es por eso que este capítulo se encuentra antes del capítulo de propuesta de mejoramiento.

5.1 Resultados de la Investigación

En relación a los aspectos generales de la tesis, se identificaron los objetivos relacionados a la idea a defender y que responderán al planteamiento del problema, considerando los alcances y delimitaciones de la investigación; todo lo mencionado se encuentra detallado en el Capítulo I Aspectos Generales

En relación a la parte teórica se ha recopilado varias definiciones de diferentes autores que se mencionan en la bibliografía, sobre temas relacionados a la investigación como soporte teórico para poder generar una propuesta teóricamente sustentada mediante el Capítulo II Marco Teórico

También se investigó, analizó y enfoco en los vigentes y oficiales reglamentos, para que el uso de la propuesta sea válido para los tribunales en el momento de la defensa pública y los resultados están expuestos en el Capítulo II Marco Teórico (Marco Jurídico o Legal)

En relación a la metodología de la investigación, se aplicó un cuestionario virtual a estudiantes universitarios de la carrera de ingeniería de sistemas para coadyuvar con información útil a la realización de la propuesta y los resultados están explícitos en el Capítulo III Metodología de la Investigación

En relación a XP, Kanban y Scrum, se ha realizado una investigación teórica, luego una análisis y comparación, para poder identificar las potencialidades que serían útiles en el contexto en el cual se utilizara la propuesta y los resultados están explícitos en el Capítulo IV Estudio Comparativo de Scrum, Kanban y Programación Extrema XP

Se ha planteado algunas interrogantes a manera de Resultados de la Investigación

- *¿Es necesario utilizar una metodología ágil para el desarrollo de software en proyectos de titulación de pregrado?*

Actualmente los requisitos de los clientes / usuarios para el desarrollo de software son muy cambiantes, lo que justifica la necesidad de optar por una metodología o marco de trabajo ágil, ya que se adaptaría mejor que otras metodologías, marcos de trabajo o métodos existentes en un entorno cambiante.

- *¿Existe alguna diferencia al utilizar una metodología de desarrollo de software si se usa software libre o no?*

Producto de la investigación se define a una metodología como una serie de métodos y técnicas de rigor científico que se aplican sistemáticamente durante el proceso de investigación para alcanzar un resultado (Knuth, 1997, p.45). En este sentido se entiende que para el desarrollo de software sea este privativo o no, la metodología usada sería la misma, ya que esta es independiente de la parte económica y/o licencias que se le otorguen al producto final que sería el software.

- *¿Existe alguna metodología de trabajo ágil privativa?*

Como resultado de una investigación sobre varias metodologías, se puede afirmar que en la actualidad al 100 % no existe, sin embargo, quizás en el futuro si llegue a existir. Sin embargo, quizás algunas de las muchas metodologías existentes, tengan algunos reglamentos y/o licencias de uso donde algunos de sus enunciados sean de carácter privativo. El estudio de todas las metodologías existentes está fuera del alcance de esta tesis ya que implicaría una nueva tesis.

- *¿Es posible que una persona desarrolle un software en 20 semanas?*

Se debe considerar que todas las personas son diferentes, en cuanto a la velocidad de programación en relación al tiempo, sin embargo. Las personas van adquiriendo conocimiento como producto de su propia experiencia en este sentido un estudiante universitario del décimo semestre ya tiene una cierta experiencia para realizar su software sumando el hecho que a medida que realice un nuevo software para su proyecto de grado también irá aprendiendo día a día.

También se debe tomar en cuenta que una adecuada administración del tiempo, un compromiso de sacrificio, esfuerzo y dedicación de parte del estudiante es un factor clave para que pueda cumplir con este tiempo.

El marco de trabajo ágil DSP es una guía, sin embargo, esta depende mucho de la voluntad, responsabilidad y compromiso del estudiante.

5.2 Conclusiones generales de la Investigación

- Se determinó los aspectos Generales de esta investigación relacionados con los antecedentes, el planteamiento del problema, los objetivos, la idea a defender, el estado del arte, los alcances y delimitaciones de esta tesis.
- Se estructuró el capítulo Marco Teórico considerando un Marco Conceptual, un Marco Jurídico, un Marco Referencial y un Marco Histórico como soporte de esta investigación
- Se definió la Metodología de la Investigación que se utilizó, identificando el método y tipo de investigación, la población, la muestra, el instrumento de relevamiento de información.
- Se aplicó un cuestionario como instrumento de relevamiento de información a la muestra, se procesó y analizó los resultados obtenidos.
- Se realizó un Estudio Comparativo de Scrum, Kanban y Programación Extrema utilizando como guía de referencia el estudio realizado por Abrahamson, Salo, Ronkkainen y Warsta publicado en su libro “Agile Software Development Methods Review and Análisis” para identificar elementos que se puedan utilizar en la propuesta.
- Se diseñó la Propuesta de Mejoramiento en base a un Marco de Trabajo Ágil de desarrollo de software de Proyectos de Grado para que sirva como guía de referencia a los estudiantes universitarios de la carrera de Ingeniería de Sistemas, identificando los componentes, roles, actividades y eventos de la guía.
- Se realizó una ejemplificación resumida aplicando la propuesta, como validación de la misma, de forma puntual, sin detallar todo el contenido que debería tener un proyecto de grado para un estudiante universitario de Ingeniería de Sistemas.
- Se utilizaron los identificadores del Framework para Evaluación de Metodologías Ágiles de Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani como validación de la

propuesta.

5.3 Recomendaciones de la Investigación

- Se recomienda para el futuro, realizar una prueba piloto de aplicación en tiempo real de esta propuesta por un estudiante de la carrera de Ingeniería de Sistemas, ya que por motivos de tiempo no se realizó en esta tesis.
- Se recomienda a otros investigadores continuar con este tema de investigación para encontrar y proponer nuevas formas de solucionar este mismo problema planteado en esta tesis
- Se recomienda considerar esta investigación como punto de partida para futuras investigaciones que en función al resultado de la aplicación de la propuesta en un caso real se pueda proponer mejoras y/o ampliaciones.

CAPÍTULO VI

6 PROPUESTA DE MEJORAMIENTO

Al inicio de esta investigación se realizó el planteamiento del problema expuesta en el Capítulo Aspectos Generales y en respuesta surgen los objetivos de la investigación y se propone como idea a defender la realización de un Marco de Trabajo Ágil de Desarrollo de Software para Proyectos de Grado, que considere la normativa establecida en el Capítulo Marco Teórico y los resultados de los datos obtenidos en el Capítulo Metodología de la Investigación, como una guía de referencia que permita orientar al estudiante durante el desarrollo de su proyecto. En este sentido en este capítulo se encuentra en detalle toda la propuesta de mejoramiento de la idea a defender de esta investigación.

6.1 Propuesta de un Marco de Trabajo Ágil de Desarrollo de Software para Proyectos de Grado

A. Introducción

En esta investigación, se propone como una alternativa un Marco de Trabajo Ágil de Desarrollo de Software para Proyectos de Grado, que sirva de guía de referencia a los estudiantes universitarios desde la aprobación del perfil hasta la defensa final ante el tribunal designado. Considerando que se tiene un tutor/a asignado y un cliente comprometido a cooperar.

B. Información Técnica

Nombre: DSP siglas de “Desarrollo de Software para Proyectos de Grado”

Orientado a: Estudiantes universitarios del décimo semestre de la carrera de Ingeniería de Sistemas que eligieron como modalidad de graduación el Proyecto de Grado mediante el desarrollo de software

Basado en: Reglamento General de Graduación para carreras de Licenciatura y Técnico Superior de la Facultad de Tecnología y el Reglamento General de Tipos y Modalidades de Graduación del Sistema de Universidades Bolivianas

Contexto de Estudio: carrera de Ingeniería de Sistemas de la Facultad de Ciencias y Tecnología de la Universidad Mayor Real y Pontificia de San Francisco Xavier de Chuquisaca

Alcance: El Marco de Trabajo o Framework, es una guía de referencia flexible de modificaciones para poder ser aplicadas en otras universidades, en carreras afines, a nivel nacional y/o para que el estudiante pueda incorporar otras herramientas si así lo desea para complementar y mejorar el desarrollo de software.

Justificación: Existen dos grandes enfoques de metodologías de desarrollo de software las Tradicionales y las Ágiles. Esta investigación se enfoca en las metodologías ágiles y no en las tradicionales principalmente, porque los requerimientos que solicita el cliente/usuario para el software son muy cambiantes.

Prácticas Ágiles - Postulados:

- Colaboración con los clientes/usuarios durante todo el proceso de desarrollo de software
- Desarrollo incremental del software con iteraciones cortas
- El software se construye con una retroalimentación continua durante el proceso de desarrollo por parte del tutor/a y del cliente, con la o el estudiante.

C. Propósito de la Guía

DSP es un Marco de Trabajo adaptable para desarrollar software para proyectos de grado, tiene el propósito de servir como guía de referencia durante toda la elaboración de esta modalidad de graduación. Esta guía contiene roles, eventos, componentes y postulados.

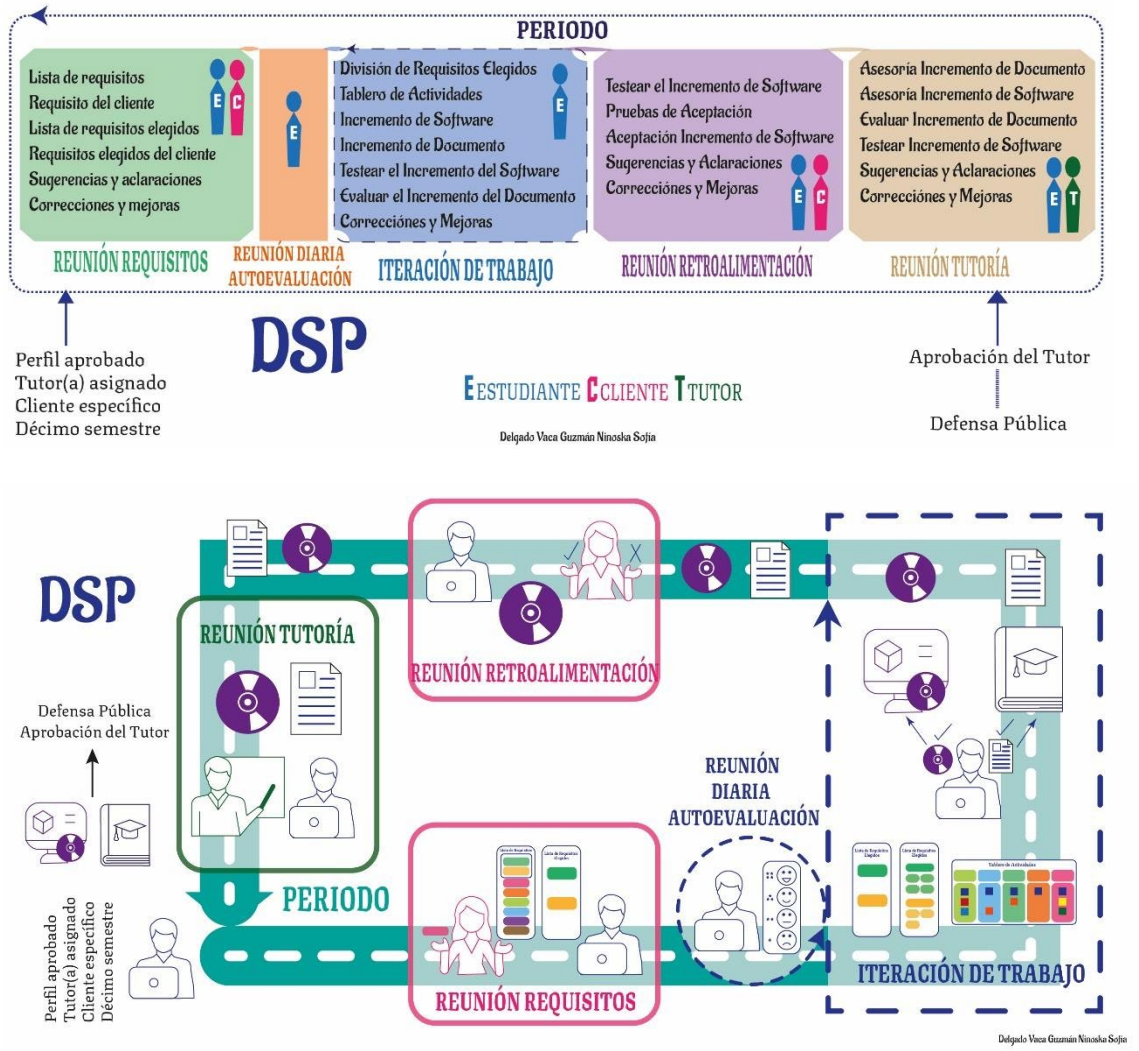
Rol: es la función o el papel asumido por un ser humano en un cierto contexto. Componente: que compone, junto con otros elementos, un todo.

Evento: suceso de importancia que se encuentra planificado

Postulado: principio que se admite como cierto sin necesidad de ser demostrado y que sirve como base para otros razonamientos.

D. Esquema de la Propuesta

Gráfico 13: Esquema del DSP



Fuente: Elaboración Propia

E. Descripción del Esquema de la Propuesta

Inicio del Periodo Inicia en el décimo semestre con el Perfil Aprobado, la asignación según reglamento de un tutor(ra) y el cliente que especificó en su perfil que está relacionado al tema que el estudiante decidió realizar como desarrollo de software para su Proyecto de Grado.

El primer paso es:

Reunión Requisitos entre el cliente y el estudiante donde se realiza la Lista de Requisitos, Lista de Requisitos Elegidos y como resultado de una adecuada comunicación surgen sugerencias, aclaraciones, correcciones y mejoras.

El segundo paso es:

Reunión Diaria Autoevaluación el estudiante solo, se autoevalúa a sí mismo en relación a todo su trabajo realizado hasta ese momento.

El tercer paso es:

Iteración de Trabajo el estudiante solo, divide en tareas acorde a su capacidad de trabajo los elementos de la Lista de Requisitos Elegidos, realiza un Tablero de Actividades colocando como tareas estos elementos seleccionados; empieza con la programación y documentación, según vaya completando el flujo de trabajo que indica en su tablero, el estudiante tendrá el Incremento de Software y el Incremento de Documento, en las tareas también se realiza el testeo, evaluación, correcciones y mejoras pertinentes. Este paso se va iterando hasta finalizar con un Incremento de Software que funcione y que esté debidamente documentado en el Incremento de Documento.

El cuarto paso es:

Reunión Retroalimentación entre el cliente y el estudiante, donde se Testea el Incremento del Software, se realiza pruebas de aceptación, se verifica la aceptación del Incremento de Software por parte del cliente, considerando las sugerencias, aclaraciones, modificaciones, correcciones y/o mejoras que se tiene que hacer.

El quinto paso es:

Reunión Tutoría entre el tutor y el estudiante, donde se realiza el asesoramiento del Incremento del Software y del Incremento del Documento, evaluando el Documento y testeando el Software para dar sugerencias, aclaraciones, correcciones y/o mejoras que se tiene que hacer.

Termina un Periodo se debe repetir un nuevo Periodo. Se realizan los periodos que sean pertinentes acorde al criterio del estudiante en función a su capacidad de trabajo y al tiempo de 20 semanas académicas, se sugiere que el periodo dure una semana, y si existiesen funcionalidades que según el estudiante le parezca difíciles de realizar entonces

debería subdividirse en funcionalidades y/o tareas más pequeñas y menos complejas según el criterio del estudiante.

Nota. Cuando se habla de complejidad se refiere al propio criterio del estudiante, ya que todas las personas son diferentes y lo que para un estudiante es complejo para otro no lo es, por eso tiene mucho que ver la capacidad de trabajo de cada estudiante para decidir personalmente cuando es no complejo un trabajo determinado.

F. Consideraciones previas

En Relación al Tema del Proyecto de Grado:

Según el Artículo 31 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología, establece que: el alcance y contenido del tema a ser estudiado será tal que permita su conclusión en un semestre académico. En este sentido el tema deberá adecuarse considerando que se debe realizar en 20 semanas académicas, correspondiente a un semestre, además debe considerar que el estudiante deba contar con la tecnología e información necesaria.

En Relación al Tutor asignado:

Según los artículos 10, 12 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología, establece que: el tutor/a debe asesorar de forma individualizada, desde la elaboración del perfil hasta la defensa pública del trabajo y su aprobación. También indica que debe establecer reuniones periódicas con sus asesorados; debe efectuar el seguimiento de los trabajos de graduación.

Según los artículos 29 y 33 del Reglamento General de Tipos y Modalidades de Graduación del Sistema de Universidades Bolivianas, establece que: los tutores deberán asumir la tarea de orientar y guiar al postulante, deben orientar y guiar en la elaboración de los trabajos de graduación del postulante, realizar revisiones de todos los trabajos, evaluar y determinar el nivel de suficiencia. También indica que todos los docentes de la universidad tienen la obligación de ser tutores de los estudiantes que así lo soliciten de acuerdo a su especialidad.

En este contexto normativo, se asume que el estudiante debe contar con un tutor/a asignado para su proyecto de grado y se sugiere que el mismo esté plenamente

comprometido. En este sentido el tutor/a es considerado como un rol muy importante en la guía propuesta en esta investigación.

En relación al cliente/usuario escogido:

Según el Artículo 4 y 9 del Reglamento General de Tipos y Modalidades de Graduación, establece que: el Proyecto de Grado para Licenciatura es el trabajo de investigación programación y diseño de objetos de uso social, en este sentido se sobre entiende la necesidad de tener un cliente/usuario que se beneficiara con el desarrollo del software.

En este contexto, se sugiere que el cliente/usuario debe estar comprometido durante todo el desarrollo del software, debe existir una buena comunicación y relación con el estudiante, debe disponer de tiempo para testear y validar el software. En este sentido el cliente/usuario es considerado como un rol muy importante en la guía propuesta en esta investigación.

En relación al eje temporal de uso de la propuesta:

Para la Carrera de Ingeniería de Sistemas; se debe especificar el uso de esta Guía en el Perfil del Proyecto de Grado, según reglamentos de la carrera; en este sentido, el Marco de Trabajo Ágil de Desarrollo de Software propuesto, considera que ya se tiene el perfil aprobado, la asignación de un tutor/a, un cliente comprometido a cooperar y parte desde ese punto para su desarrollo durante el décimo semestre.

En relación al Software realizado:

El software desarrollado por el estudiante tiene como propósito ser un material académico para presentar y defender ante un tribunal de grado para obtener el título de profesional en dicha carrera. No existe ningún contrato del estudiante por la construcción del software con el cliente/usuario, ni retribución financiera. Al no existir ninguna negociación contractual, el estudiante debe ser responsable de la realización

del software desde el perfil hasta la defensa pública ante un tribunal. Dándose la posibilidad de que este software sea implementado o no en el futuro por el cliente/usuario.

En relación a la Jornada de Trabajo:

En la Tesis Doctoral titulada “Tiempo de Trabajo y Tiempo de Descanso”, desarrollada por el Doctor Miguel Basterra Hernández, para obtener el grado de Doctor en Derecho,

en la Universidad de Alicante, en Alicante España, 2016; afirma que los horarios de trabajo diarios o de inter jornadas deben ser mínimo de 8 horas y máximo de 12 horas diarias, para preservar el descanso fisiológico de las personas y mantener la productividad de las mismas. Asimismo, en el Decreto Supremo de 24 de mayo de 1939, Ley General del Trabajo de Bolivia, en el Artículo 46 indica que la jornada efectiva de trabajo nocturno, diurno y trabajos especiales debe ser de 7 horas mínimo hasta 12 horas máximo, según corresponda. En este contexto, se justifica y considera razonable el asignar a la Jornada de Trabajo del Estudiante el tiempo de 12 horas diarias.

G. Roles del DSP

El DSP basada en la reglamentación ya mencionada anteriormente, considera como parte de sus Roles a: Tutor, Cliente y Estudiante.

Tutor(ra)

Según el Artículo 10 y 12 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología. Se define al tutor como un profesional que posee grado académico igual o superior al que postula el estudiante. Este podrá ser un docente de la facultad o un profesional independiente sugerido por el estudiante y aceptado por la dirección de la carrera de Ingeniería de Sistemas.

Funciones: Están establecidas por los reglamentos anteriormente mencionados. En este contexto, el tutor puede:

- Asesorar el Incremento del Software y el Incremento del Documento
- Realizar pruebas o test al software desarrollado
- Evaluar el Incremento del Documento
- Asistir a las Reuniones Tutoría coordinadas con el estudiante

Cliente

Persona o personas interesadas e involucradas en el desarrollo del software que usarán y/o se beneficiarán del mismo. Pueden ser empresas públicas o privadas. A veces se establece un acuerdo académico verbal o escrito entre cliente y la carrera, en este convenio queda claramente establecido que el desarrollo del software tiene una finalidad académica sin retribución financiera.

Funciones:

- Identificar los elementos de la Lista de Requisitos que son los requerimientos del cliente que debe tener el software
- Expresar claramente los elementos de la Lista de Requisitos
- Ordenar los elementos de la Lista de Requisitos por prioridades para crear la Lista de Requisitos Elegidos
- Realizar pruebas funcionales, no funcionales, de aceptación al Incremento del Software
- Proporcionar la información necesaria para el desarrollo del software
- Sugerir correcciones y/o mejoras al Incremento del Software
- Aclarar dudas del estudiante
- Testear el Incremento del Software
- Aceptar el Incremento del Software
- Asistir a las Reuniones Requisitos y Reuniones Retroalimentación coordinadas con el estudiante

Estudiante

Estudiante universitario que cursa el décimo semestre de la carrera de Ingeniería de Sistemas; que ha elegido como modalidad de graduación la realización de un proyecto de grado mediante el desarrollo de software. Debe tener aprobado el perfil de su proyecto de grado; Asignado un Tutor/a para la realización de su proyecto de grado; y un cliente/usuario comprometido a colaborar con el desarrollo del software. Es autoorganizado, elige la mejor forma de gestionar su trabajo. Entrega Incrementos de Software que tienen funcionalidad de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación con el cliente y el tutor.

Funciones:

- Desarrollar el Software con los requerimientos del cliente para su defensa pública ante el tribunal designado
- Elaborar el Documento del Trabajo de Graduación (Documento) acorde al formato establecido por el reglamento de graduación, para su defensa pública ante el tribunal designado

- Aprender constantemente basándose en el empirismo
- Elaborar la Lista de Requisitos con los elementos que identifica el cliente coordinando con este.
- Elaborar la Lista de Requisitos Elegidos con los elementos seleccionados de la Lista de Requisitos para ser implementados en la Iteración de Trabajo
- Dividir los elementos de la Lista de Requisitos Elegidos en tareas a ser realizadas en el Tablero de Actividades
- Realizar un Tablero de Actividades con las tareas priorizadas para visualizar el flujo de trabajo
- Convertir los elementos de la Lista de Requisitos en Incrementos de Software que tengan funcionalidad al final de cada Periodo
- Testear el Incremento de Software
- Realizar el Incremento de Software en cada Iteración de Trabajo
- Realizar el Incremento de Documento en cada Iteración de Trabajo
- Evaluar el Incremento de Software
- Realizar correcciones y mejoras que sean pertinentes

H. Componentes del DSP

Software

Es el software que debe desarrollar el estudiante que cumple con los requisitos establecidos por el cliente. Este componente debe ser defendido para su aprobación ante un tribunal en una defensa pública, para poder culminar el plan de estudios y poder obtener el diploma académico correspondiente a la carrera de Ingeniería de Sistemas. Debe ser desarrollado de forma iterativa e incremental paralelamente al Documento del Trabajo de Graduación. Debe ser asesorado, testeado por el tutor. Debe ser testeado, aceptado por el cliente quien puede sugerir modificaciones. Y también debe ser testeado por el estudiante.

Documento

Es el Documento del Trabajo de Graduación que establece los artículos 23, 40 y 42 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología, que debe elaborar el estudiante, Su estructura y características deben estar en función a lo determinado por los artículos ya mencionados. Este

componente debe ser defendido para su aprobación ante un tribunal en una defensa pública, para poder culminar el plan de estudios y poder obtener el diploma académico correspondiente a la carrera de Ingeniería de Sistemas. Debe ser elaborado por el estudiante de forma iterativa e incremental paralelamente al software. Debe ser documentado por el estudiante con todos los capítulos del proyecto de grado acorde a la reglamentación de la carrera de Ingeniería de Sistemas. Debe ser evaluado y asesorado por el tutor.

Lista de Requisitos

Es una lista ordenada, que está compuesta por todos los requisitos del cliente o lo que se conoce que es necesario en el software. Es la única fuente de requisitos del software.

Esta lista no está completa ya que cambia a medida que el entorno en el que se usará también lo hace. Es dinámica porque cambia constantemente para identificar lo que el cliente necesita para utilizar el software de manera funcional.

Todos los elementos de la Lista de Requisitos son identificados por el cliente y pueden actualizarse en cualquier momento y son escritos por el estudiante en coordinación con el cliente. El nivel de detalle de los elementos de la lista está en función del criterio del estudiante.

A. Elementos de la Lista de Requisitos: Los elementos de la lista de requisitos son descripciones breves y simples del requisito contada desde la perspectiva del cliente que desea que tenga el software. Se escribe siguiendo el siguiente formato sugerido

Tabla 8: Requisito del Cliente

Número:	
Título o Nombre:	
Descripción: Como…Quiero…Para…	
Criterios de Aceptación y/o Condiciones:	
Fuente: Elaboración propia	

- Número: es el número pertinente del Requisito del Cliente
- Título o Nombre: Es el nombre o título que identifique al requisito del cliente
- Descripción: Es una descripción del requisito del cliente
- Como (rol, persona, actor): es la persona o rol de usuario que tiene la necesidad.

¿Quién se beneficia de este requisito?

- Quiero (objetivo, comportamiento, funcionalidad): es lo que se quiere obtener, una funcionalidad, una característica, etc. ¿Qué se quiere hacer con este requisito?
- Para (valor, motivo, razón, beneficios): es el motivo por el que se necesita, es el valor que se obtiene como resultado. El objetivo de este requisito.
- Criterios de Aceptación (considerado como terminado): Son los cambios, las acciones, el comportamiento que se espera a raíz de ejecutar el requisito. Todo lo que se necesita para que sea considerado como terminado, completo.

El Requisito del Cliente, puede ser modificada para aumentar el nivel de detalle acorde al criterio del estudiante. Toda esta información junto con lo que corresponde según reglamento deberá registrarse en el Incremento del Documento paralelamente al desarrollo del Incremento del Software.

Lista de Requisitos Elegidos

Es una lista ordenada de elementos seleccionados, de la Lista de Requisitos para ser programados y documentados en la Iteración de Trabajo. Además, debe tener el objetivo de la Iteración de Trabajo y el trabajo necesario para transformar esos elementos seleccionados en un Incremento de Software terminado que sea útil y funcional.

Cuando se requiere nuevos elementos, el estudiante lo adiciona a la Lista de Requisitos Elegidos para ser ejecutados en esa Iteración de Trabajo o a la Lista de Requisitos para ser implementados después.

1. Elementos de la Lista de Requisitos Elegidos: Se escribe siguiendo el siguiente formato sugerido

Tabla 9: Requisito Elegido del Cliente

Número:	
Título o Nombre:	
Descripción: Como…Quiero…Para…	
Criterios de Aceptación:	
Objetivo de la Iteración:	
Lo que se necesita para transformar el requisito en software útil:	
Fuente: Elaboración propia	

El Requisito Elegido del Cliente puede ser modificado para aumentar el nivel de detalle acorde al criterio del estudiante. Toda esta información junto con lo que corresponde según reglamento deberá registrarse en el Incremento del Documento paralelamente al desarrollo del Incremento del Software.

Incremento de Software

Es la suma de todos los elementos de la Lista de Requisitos completados con funcionalidad durante una Iteración de Trabajo a medida que se itera se terminan los elementos y se unen con el Incremento de Software, este debe estar en condiciones de utilizarse, es parte del software que se va incrementando con cada iteración. Al final el Incremento de Software será el Software, que el estudiante defenderá ante el tribunal. Cada incremento, se integra con todos los incrementos anteriores y es probado de manera exhaustiva asegurando que todos los incrementos funcionen en conjunto.

Incremento de Documento

Es la suma de todos los registros realizados en el documento acorde al formato reglamentario durante una Iteración de Trabajo. De esta forma el estudiante deberá ir documentando todos los capítulos de su proyecto de grado. Este incremento sucede paralelamente al Incremento del Software. Al final el Incremento de Documento será el

documento que el estudiante defenderá ante el tribunal. Cada incremento, se integra con todos los incrementos anteriores y es evaluado de manera exhaustiva asegurando que todos los incrementos registrados en el documento tengan coherencia.

Tablero de Actividades

Es un tablero de actividades, para ver el flujo de acciones y/o tareas realizadas por el estudiante durante el proceso de desarrollo de software, para ver la etapa en la que se encuentra cada Requisito Elegido del Cliente y su prioridad de realización. Debe incluir paralelamente su registro en el Incremento del Documento.

Considerando la capacidad de programación del estudiante, se debe realizar un tablero de actividades para cada Iteración de Trabajo donde el estudiante deberá especificar la cantidad de columnas y filas que desea utilizar que estén acordes al desarrollo de software.

El estudiante debe limitar la cantidad de trabajo en progreso, para concentrarse en los Requisitos Elegidos del Cliente, que sean adecuados en el momento oportuno, ya que dedicarse a muchos requisitos al mismo tiempo causa la pérdida de concentración y ser menos productivo.

Los Requisitos Elegidos del Cliente, son divididos en tareas acorde a la capacidad y criterio del estudiante. La estructura del Tablero de Actividades está compuesta por:

- Fichas o tarjetas, cada una de ellas representará una tarea del Requisito Elegido del Cliente
- Columnas, es una secuencia de estados específicos sucesivos por los que debe transitar una tarea del Requisito Elegido del Cliente, desde que se solicita hasta que está realizada. Cada columna visualiza una fase o etapa del proceso de realización de la tarea. El nombre de la columna está en función del criterio del estudiante
- Filas o carriles, representan diferentes tipos de actividades y/o tareas específicas, se las utiliza para agrupar requisitos según el tipo de trabajo (ejemplo: desarrollo web) o según su prioridad (ejemplo alta, media, baja)

Se escribe siguiendo el siguiente formato sugerido:

Tabla 10: Tablero de Actividades: Periodo N° (se coloca el número que corresponde)

Lista de Requisitos Elegidos	Priorizadas	Ejecutándose	Probándose	Terminadas
Es una lista de todos los Requisitos Elegidos del Clienteque se implementarán y que están a su vez divididos en tareas	Son 1, 2 o 3 Tareas del Requisito Elegido del Cliente priorizadas por el estudiante	Son 1, 2 Tareas del Requisito Elegido del Cliente (máximo 3 simultáneamente) que se encuentran en proceso de ejecución	Son 1, 2 Tareas del Requisito Elegido del Cliente (máximo 3 simultáneamente) que se encuentran en pruebas o testeo	Es una lista de todas Tareas del Requisito Elegido del Cliente que se van terminando
Fuente: Elaboración Propia				

I. Jornada de Trabajo del Estudiante

La Pirámide de Necesidades del ser Humano de Maslow propuesta por Abraham Maslow indica una jerarquía de las necesidades humanas básicas y primordiales, en esta pirámide las necesidades Fisiológicas como el descanso, alimentación entre otras ocupan el primer lugar; en este sentido es importante una adecuada administración del tiempo de trabajo y descanso diario del estudiante.

La realización de un Proyecto de Grado en 20 semanas académicas implica un gran sacrificio, responsabilidad, constancia y disciplina por parte del estudiante. En este sentido en esta investigación se considera como tiempo de trabajo adecuado 12 horas diarias. Este tiempo está justificado por el Decreto Supremo de 1939 y por la Tesis Doctoral del Doctor Miguel Basterra Hernández, mencionadas anteriormente.

Quizás las 12 horas diarias se consideren como una carga excesiva para el estudiante, sin embargo, el contexto en el que tiene que desarrollar el software y el tiempo de 20 semanas académicas exige un factor de compromiso y dedicación de parte del estudiante universitario si realmente quiere terminar a tiempo.

Existen varios autores que plantean el cálculo de la velocidad ideal de un programador que depende de varios factores ya que las personas son diferentes, siendo este cálculo fuera del alcance de esta tesis ya que este tema amerita toda una amplia investigación para la realización de una nueva tesis.

La Técnica Pomodoro desarrollada por Francesco Cirilo se utiliza para fraccionar el tiempo que dura el realizar una actividad. El método divide el tiempo en periodos de 25 minutos denominados Pomodoros separados por pausas de 5 minutos dedicadas al descanso para mejorar la agilidad mental. Una sesión completa de Pomodoro equivale a 4 Pomodoros completos al finalizar el mismo se debe realizar un descanso de 15 minutos como mínimo entre cada sesión.

Tabla 11: Técnica de Pomodoro

Una Sesión completa de Pomodoro equivale a:							
Pomodoro 1		Pomodoro 2		Pomodoro 3		Pomodoro 4	
25 minutos	5 minutos	25 minutos	5 minutos	25 minutos	5 minutos	25 minutos	5 minutos
Trabajo	Descanso	Trabajo	Descanso	Trabajo	Descanso	Trabajo	Descanso
Totales: 25+25+25+25 = 100 minutos de Trabajo 5+5+5+5 = 20 minutos de Descanso 120 minutos dura una sesión completa de Pomodoro							
Fuente: https://pomofocus.io/							

Una Jornada de Trabajo del Estudiante es 12 horas en 1 día Convertimos las horas en minutos: 12 horas = 720 minutos en 1 día Regla de 3 simple:

1 hora equivale a 60 minutos 12 horas equivale a X minutos

Resultado (60*12) /1= 720 minutos

Aplicando la Técnica de Pomodoro a la Jornada de Trabajo del Estudiante Calculamos la cantidad de Pomodoros que contiene una Jornada de Trabajo del Estudiante

Regla de 3 simple:

120 minutos equivale a 1 sesión completa de Pomodoro 720 minutos equivale a X sesiones

Resultado (720*1) /120= 6 sesiones

Una Jornada de Trabajo del Estudiante equivale a realizar 6 sesiones completas de Pomodoro, con descansos de mínimo 15 minutos, entre cada sesión completa de Pomodoro de 120 minutos. Puede ser 15 minutos o más según el criterio del estudiante, pero el tiempo debe estar dentro de las 24 horas de ese día de trabajo.

J. Eventos del DSP

Reunión Requisitos

Es una reunión que se realiza entre el estudiante y el cliente, el tiempo de realización está en función del criterio del estudiante, pero debe estar dentro de la Jornada de Trabajo del Estudiante. Se debe coordinar con el cliente la hora y lugar de la reunión. En esta Reunión Requisitos se realiza:

- La Lista de Requisitos
- La Lista de Requisitos Elegidos
- Se tiene sugerencias, aclaraciones, correcciones y/o mejoras del cliente y estudiante como resultado de la comunicación y acuerdo entre ambos.

Reunión Diaria Autoevaluación

En esta reunión el estudiante debe concentrarse en autoevaluarse a sí mismo diariamente sobre todo el trabajo que realizó, que está realizando y que va a realizar, antes de la Iteración de Trabajo de ese día. El tiempo de realización está en función del criterio del estudiante, pero debe estar dentro de la Jornada de Trabajo del Estudiante.

Iteración de Trabajo

En esta Iteración de Trabajo el estudiante se encuentra solo, el tiempo de realización está en función del criterio del estudiante. El estudiante aprende con cada Iteración de Trabajo necesario para conseguir transformar los requisitos en software útil, basándose en el empirismo. En base a su experiencia, el estudiante irá aprendiendo a estimar el tiempo que le tome desarrollar las diferentes funcionalidades del software que está desarrollando. Debido a que las capacidades y experiencias de los estudiantes son diferentes, esta guía deja abierto el tiempo de realización de cada Iteración de Trabajo.

En la Iteración de Trabajo el estudiante realiza:

- División de Requisitos Elegidos en tareas a realizar en el Tablero de Actividades según su capacidad de trabajo y criterio personal.
- El Tablero de Actividades en función a la Lista de Requisitos Elegidos
- El Incremento del Software
- El Incremento del Documento
- Testear el Incremento de Software
- Evaluar el Incremento del Documento
- Se pueden introducir cambios durante las iteraciones
- Las correcciones y mejoras pertinentes en el Incremento de Software y en el Incremento del Documento.

Reunión Retroalimentación

Es una reunión que se realiza entre el estudiante y el cliente, el tiempo de realización está en función del criterio del estudiante, pero debe estar dentro de la Jornada de Trabajo del Estudiante. Se debe coordinar con el cliente la hora y lugar de la reunión. En esta Reunión Retroalimentación se realiza:

- Pruebas para que el cliente decida la Aceptación y/o Modificación del Incremento del Software
- Testeo del Incremento del Software
- Se tiene sugerencias, aclaraciones del cliente y/o estudiante como resultado de la comunicación y acuerdo entre ambos.
- La solicitud de correcciones y mejoras pertinentes en el Incremento de Software.

Reunión Tutoría

Es una reunión que se realiza entre el estudiante y el tutor, el tiempo de realización está en función del criterio del estudiante, pero debe estar dentro de la Jornada de Trabajo del Estudiante. Se debe coordinar con el tutor la hora y lugar de la reunión.

En esta Reunión Tutoría se realiza:

- La asesoría al incremento del documento y al incremento del software
- Se evalúa el incremento del documento
- Se realiza el testeo del incremento del software
- Se tiene sugerencias, aclaraciones, correcciones y mejoras del tutor y/o estudiante como resultado de la comunicación y acuerdo entre ambos.

Periodo

Es el espacio de tiempo durante el cual se realiza:

- La Reunión Requisitos
- La Reunión Diaria Autoevaluación
- La Iteración de Trabajo
- La Reunión Retroalimentación
- La Reunión Tutoría
- Se realizan correcciones y/o mejoras al incremento del software y al incremento del documento.

Como el tiempo de un semestre académico es de 20 semanas académicas, se sugiere que sea de 7 días de duración, porque el periodo contiene dos reuniones con el cliente y una reunión con el tutor entre otras actividades que implican tiempo de concentración y compromiso de parte del estudiante. Sin embargo, como esta es una guía de referencia para orientar al estudiante, si este ve pertinente realizar en menos o más tiempo, acorde a su capacidad, lo puede realizar así, adecuando las 12 horas diarias de Jornadas de Trabajo del Estudiante a las 20 semanas que tiene para completar todo el Proyecto de Grado.

CAPÍTULO VII

7 VALIDACIÓN MEDIANTE LA EJEMPLIFICACIÓN DE LA PROPUESTA

Este capítulo tiene el propósito de validar el marco de trabajo ágil DSP mediante una pequeña ejemplificación resumida de la aplicación de la propuesta con información simulada. Se realiza la ejemplificación del Marco de Trabajo Ágil DSP, que es una recapitulación de lo más relevante y general de un Proyecto de Grado, no se encuentra detallado toda la documentación que se debería registrar según el reglamento de graduación, ni tampoco se desarrolló la programación del software mencionado, porque eso implicaría realizar toda una tesis completa de proyecto de grado de un estudiante universitario de la carrera de Ingeniería de Sistemas, que está fuera del alcance de esta investigación.

7.1 Ejemplificación del Marco de Trabajo Ágil DSP "Desarrollo de Software para Proyectos de Grado"

A. Roles del Proyecto de Grado

- Estudiante: Adrián Bustamante (persona ficticia)
- Tutor: David Méndez (persona ficticia)
- Cliente: Universidad de Madrid (ejemplo ficticio)

B. Componentes del Proyecto de Grado

Software "Sistema de Registro de Clases": El software final será la suma total de los Incrementos de Software, que se tendrá al finalizar el semestre.

Incremento de Software: El Incremento del Software irá creciendo a medida que se realicen las Iteraciones de Trabajo en los diferentes Periodos.

Documento "Desarrollo de un Sistema de registro de clases para la Universidad de Madrid": El Documento final será la suma total de los Incrementos de Documento, que se tendrá al finalizar el semestre.

Incremento de Documento: El Incremento del Software irá creciendo a medida que se realicen las Iteraciones de Trabajo en los diferentes Periodos.

Lista de Requisitos

Tabla 12: Requisito del Cliente: Debe mostrarse los datos del Estudiante

Número:	1
Título o Nombre:	Datos del Estudiante
Descripción:	Como Estudiante Quiero Visualizar mis datos de Estudiante de la Universidad Para Identificarse con la carrera
Criterios de Aceptación y/o Condiciones:	Los datos del estudiante deben ser con el formato apellido paterno apellido materno nombres Los datos del estudiante deben incluir: fecha de nacimiento, lugar de nacimiento, carrera, carnet de identidad, carnet universitario, fotografía
Fuente: Elaboración propia	

Tabla 13: Requisito del Cliente: Debe adaptarse a la Imagen de la Universidad

Número:	2
Título o Nombre:	Imagen del Sistema
Descripción:	Como Estudiante Quiero Visualizar los datos de la Carrera de la Universidad Para Identificar las asignaturas de la carrera
Criterios de Aceptación y/o Condiciones:	Los colores amarillo y verde deben predominar En la ventana de ingreso debe estar el lema de la Universidad El encabezado debe ser similar al de la página web Debe verse el logo de la Universidad en todo momento en el encabezado
Fuente: Elaboración propia	

Tabla 14: Requisito del Cliente: Debe mostrarse los datos de la Carrera

Número:	3
Título o Nombre:	Datos de la Carrera
Descripción:	Como Usuario del Sistema Quiero Visualizar que el sistema se identifique con la imagen de la Universidad Para Lograr pertenencia hacia la Universidad
Criterios de Aceptación y/o Condiciones:	Los datos de la carrera deben tener el formato nombre de la universidad, nombre de la facultad, nombre de la carrera
	Los datos de la carrera deben incluir: nombre de la materia, sigla de la materia, docente de la materia, cantidad de horas de la materia, si es teórica, práctica o ambas, semestre a la que pertenece la materia.
Fuente: Elaboración propia	

Se completa toda la Lista de Requisitos pertinentes que irá cambiando con el tiempo.

Lista de Requisitos Elegidos

Tabla 15: Requisito elegido del Cliente: Debe mostrarse los datos de la Carrera

Número:	3
Título o Nombre:	Datos de la Carrera
Descripción:	Como Usuario del Sistema Quiero Visualizar que el sistema se identifique con la imagen de la Universidad Para Lograr pertenencia hacia la Universidad
Criterios de Aceptación y/o Condiciones:	Los datos de la carrera deben tener el formato nombre de la universidad, nombre de la facultad, nombre de la carrera Los datos de la carrera deben incluir: nombre de la materia, sigla de la materia, docente de la materia, cantidad de horas de la materia, si es teórica, práctica o ambas, semestre a la que pertenece la materia.
Fuente: Elaboración propia	

Tabla 16: Requisito elegido del Cliente: Debe mostrarse los datos del Estudiante

Número:	1
Título o Nombre:	Datos del Estudiante
Descripción:	Como Estudiante Quiero Visualizar mis datos de Estudiante de la Universidad Para Identificarse con la carrera
Criterios de Aceptación y/o Condiciones:	Los datos del estudiante deben ser con el formato apellido paterno apellido materno nombres Los datos del estudiante deben incluir: fecha de nacimiento, lugar de nacimiento, carrera, carnet de identidad, carnet universitario, fotografía
Fuente: Elaboración propia	

Se completa la Lista de Requisitos Elegidos según el criterio del estudiante.

Tablero de Actividades

Tabla 17: Tablero de Actividades: Periodo N°1

Priorizadas	Ejecutándose	Probándose	Terminadas
Imagen del Sistema			*-
Datos de la Carrera		*	
Datos del Estudiante	*		
…….			
Fuente: Elaboración Propia			

Se realizan los siguientes períodos hasta completar el Proyecto de Grado

C. Jornada de Trabajo del Estudiante en el Proyecto de Grado

Gráfico 14: Tabla de Jornada De Trabajo del Proyecto de Grado

Jornada de Trabajo del Proyecto de Grado				
1 jornada de Trabajo = 12 horas transformando las horas a minutos se tiene = 720 minutos (1 hora equivale a 60 minutos)				
Pomodoro 1	Total	Mañana	Total, tiempo trabajo	Total, tiempo descanso
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	7:00 am a 9:00 am	2 horas	
Sesión completa de Pomodoro; se debe dar un descanso		9:00 am a 9:30 am		0,30 horas
Pomodoro 2				
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	9:30 am a 11:30 am	2 horas	
Dos Sesiones completas de Pomodoro se debe dar un descanso		11:30 am a 13:00 pm		1,30 horas
Pomodoro 3	Total	Tarde		
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	13:00 pm a 15:00 pm	2 horas	
Sesión completa de Pomodoro; se debe dar un descanso		15:00 pm a 15:30 pm		0,30 horas
Pomodoro 4				
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	15:30 pm a 17:30 pm	2 horas	
Dos Sesiones completas de Pomodoro se debe dar un descanso		17:30 pm a 19:00 pm		1,30 horas
Pomodoro 5	Total	Noche		
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	19:00 pm a 21:00 pm	2 horas	
Sesión completa de Pomodoro; se debe dar un descanso		21:00 pm a 21:30 pm		0,30 horas
Pomodoro 6				
25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso 25 min trabajo 5 min descanso	= 120 min transformando en horas = 2 horas	21:30 pm a 23:30 pm	2 horas	
Dos Sesiones completas de Pomodoro se debe dar un descanso		23:30 pm a 7:00 am		7,30 horas
			12 horas Jornada de Trabajo	12 horas de descaso y otras actividades
Fuente: Elaboración Propia				

Fuente: Elaboración Propia

Nota Aclaratoria en la aplicación de la Jornada de Trabajo: Descripción de la Gráfica 14: Se trata de una tabla donde se especifica el uso de los Pomodoros, para administrar el tiempo del estudiante, incluye una columna donde especifica por la mañana, tarde y noche un horario sugerido para realizar el proyecto de grado, además de dos columnas que indican el tiempo de duración del trabajo y el tiempo de duración de descanso. Todo considerando las 12 horas de trabajo propuestas. La justificación jurídica en Relación a la Jornada de Trabajo propuesto de 12 horas diarias ya se mencionó anteriormente en el Inciso F Consideraciones Previas; y la explicación detallada de la jornada de trabajo del proyecto de grado propuesto para la realización de esta gráfica, se encuentra en el Inciso

I Jornada de Trabajo del Estudiante, ambos incisos se encuentran en el Capítulo VI Propuesta de Mejoramiento.

D. Cronograma del Proyecto de Grado

Gráfico 15: Cronograma del Proyecto de Grado

Cronograma del Proyecto de Grado

N°	Desarrollo del Software y Elaboración del Documento	Semanas																		
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
		Agosto				Agosto/Septiembre				Septiembre/Octubre					Octubre/Noviembre				Noviembre/Diciembre		
		1-7	8-14	15-21	22-28	29-4	5-11	12-18	19-25	26-2	3-9	10-16	17-23	24-30	31-6	7-13	14-20	21-27	28-4	5-11	12-18
1	**Periodo 1**																				
2	Tarea 1																				
3	Tarea 2																				
4	Tarea 3																				
5	Tarea 4																				
6	Tarea 5																				
7	**Periodo 2**																				
8	Tarea 1																				
9	Tarea 2																				
10	Tarea 3																				
11	Tarea 4																				
12	Tarea 5																				
13	Tarea 6																				
14	**Periodo 3**																				
15	Tarea 1																				
16	Tarea 2																				
17	Tarea 3																				
18	**Periodo 4**																				
19	Tarea 1																				
20	Tarea 2																				
21	Tarea 3																				
22	Tarea 4																				
23	**Periodo 5**																				
24	Tarea 1																				
25	Tarea 2																				
26	**Periodo 6**																				
27	Tarea 1																				
28	Tarea 2																				
29	Tarea 3																				
30	Tarea 4																				
31	Tarea 5																				
32	**Periodo 7**																				
33	Tarea 1																				
34	Tarea 2																				
35	Tarea 3																				
36	Tarea 4																				
37	Tarea 5																				
38	Tarea 6																				
39	**Periodo 8**																				
40	Tarea 1																				
41	Tarea 2																				
42	Tarea 3																				
43	**Periodo 9**																				
44	Tarea 1																				
45	Tarea 2																				
46	Tarea 3																				
47	Tarea 4																				
48	**Periodo 10**																				
49	Tarea 1																				
50	Tarea 2																				
51	**Periodo 11**																				
52	Tarea 1																				
53	Tarea 2																				
54	Tarea 3																				
55	Tarea 4																				
56	Tarea 5																				
57	**Periodo 12**																				
58	Tarea 1																				
59	Tarea 2																				
60	Tarea 3																				
61	Tarea 4																				
62	Tarea 5																				
63	Tarea 6																				
64	**Periodo 13**																				
65	Tarea 1																				
66	Tarea 2																				
67	Tarea 3																				
68	**Periodo 14**																				
69	Tarea 1																				
70	Tarea 2																				
71	Tarea 3																				
72	Tarea 4																				
73	**Periodo 15**																				
74	Tarea 1																				
75	Tarea 2																				
76	**Periodo 16**																				
77	Tarea 1																				
78	Tarea 2																				
79	Tarea 3																				
80	Tarea 4																				
81	Tarea 5																				
82	**Periodo 17**																				
83	Tarea 1																				
84	Tarea 2																				
85	Tarea 3																				
86	Tarea 4																				
87	Tarea 5																				
88	Tarea 6																				
89	**Periodo 18**																				
90	Tarea 1																				
91	Tarea 2																				
92	Tarea 3																				
93	**Periodo 19**																				
94	Tarea 1																				
95	Tarea 2																				
96	Tarea 3																				
97	Tarea 4																				
98	**Periodo 20**																				
99	Tarea 1																				
100	Tarea 2																				

Fuente: Elaboración Propia

Fuente: Elaboración Propia

E. Periodos del Proyecto de Grado Periodo 1

Inicia el primer periodo del DSP

Reunión Requisitos

- Lugar: Universidad de Madrid
- Fecha: agosto 2021
- Número de Periodo: 1
- Personas que asistieron a la reunión: Cliente y estudiante

Inicia la Reunión Requisitos, el cliente explica detalladamente los requisitos del software, paralelamente surgen una serie de consultas, aclaraciones, sugerencias, entre ambos.

El cliente identifica:

Software: Sistema de registro de clases

Objetivo: se quiere crear un sistema que permita el registro de clases de un estudiante de la Universidad

El estudiante debe Escribir las ideas de Requisitos del Cliente Debe mostrarse los datos del Estudiante

Debe adaptarse a la Imagen de la Universidad Debe mostrarse los datos de la Carrera

Debe validarse en el ingreso que el estudiante esté matriculado Debe mostrar las materias de acuerdo a la carrera y el semestre Debe permitir el registro en el horario deseado

Debe permitir cancelar el registro previamente realizado Debe mostrar los horarios disponibles para cada materia Debe verificar que no se cruce los horarios entre materias

Debe mostrar las materias de acuerdo a la carrera y el semestre Debe mostrar los cupos disponibles para el horario

Debe tener una visualización previa del horario……

El estudiante debe Detallar cada uno de los Requisitos del Cliente

Tabla 18: Requisito del Cliente: Debe mostrarse los datos del Estudiante

Número:	1
Título o Nombre:	Datos del Estudiante
Descripción:	Como Estudiante Quiero Visualizar mis datos de Estudiante de la Universidad Para Identificarse con la carrera
Criterios de Aceptación y/o Condiciones:	Los datos del estudiante deben ser con el formato apellido paterno apellido materno nombres Los datos del estudiante deben incluir: fecha de nacimiento, lugar de nacimiento, carrera, carnet de identidad, carnet universitario, fotografía
Fuente: Elaboración propia	

Tabla 19: Requisito del Cliente: Debe adaptarse a la Imagen de la Universidad

Número:	2
Título o Nombre:	Imagen del Sistema
Descripción:	Como Estudiante Quiero Visualizar los datos de la Carrera de la Universidad Para Identificar las asignaturas de la carrera
Criterios de Aceptación y/o Condiciones:	Los colores amarillo y verde deben predominar En la ventana de ingreso debe estar el lema de la Universidad El encabezado debe ser similar al de la página web Debe verse el logo de la Universidad en todo momento en el encabezado
Fuente: Elaboración propia	

Tabla 20: Requisito del Cliente: Debe mostrarse los datos de la Carrera

Número:	3
Título o Nombre:	Datos de la Carrera
Descripción:	Como Usuario del Sistema Quiero Visualizar que el sistema se identifique con la imagen de la Universidad Para Lograr pertenencia hacia la Universidad
Criterios de Aceptación y/o Condiciones:	Los datos de la carrera deben tener el formato nombre de la universidad, nombre de la facultad, nombre de la carrera Los datos de la carrera deben incluir: nombre de la materia, sigla de la materia, docente de la materia, cantidad de horas de la materia, si es teórica, práctica o ambas, semestre a la que pertenece la materia.
Fuente: Elaboración propia	

Con toda la información anterior el estudiante debe crear la LISTA DE REQUISITOS La LISTA DE REQUISITOS está formada por todos los Requisitos del Cliente

Tabla 21: Requisito del Cliente: Debe mostrarse los datos del Estudiante

Número:	1
Título o Nombre:	Datos del Estudiante
Descripción:	Como Estudiante Quiero Visualizar mis datos de Estudiante de la Universidad Para Identificarse con la carrera
Criterios de Aceptación y/o Condiciones:	Los datos del estudiante deben ser con el formato apellido paterno apellido materno nombres Los datos del estudiante deben incluir: fecha de nacimiento, lugar de nacimiento, carrera, carnet de identidad, carnet universitario, fotografía
Fuente: Elaboración propia	

Tabla 22: Requisito del Cliente: Debe adaptarse a la Imagen de la Universidad

Número:	2
Título o Nombre:	Imagen del Sistema
Descripción:	Como Estudiante Quiero Visualizar los datos de la Carrera de la Universidad Para Identificar las asignaturas de la carrera
Criterios de Aceptación y/o Condiciones:	Los colores amarillo y verde deben predominar En la ventana de ingreso debe estar el lema de la Universidad El encabezado debe ser similar al de la página web Debe verse el logo de la Universidad en todo momento en el encabezado
Fuente: Elaboración propia	

Tabla 23: Requisito del Cliente: Debe mostrarse los datos de la Carrera

Número:	3
Título o Nombre:	Datos de la Carrera
Descripción:	Como Usuario del Sistema Quiero Visualizar que el sistema se identifique con la imagen de la Universidad Para Lograr pertenencia hacia la Universidad
Criterios de Aceptación y/o Condiciones:	Los datos de la carrera deben tener el formato nombre de la universidad, nombre de la facultad, nombre de la carrera Los datos de la carrera deben incluir: nombre de la materia, sigla de la materia, docente de la materia, cantidad de horas de la materia, si es teórica, práctica o ambas, semestre a la que pertenece la materia.
Fuente: Elaboración propia	

El cliente debe priorizar el orden de los elementos de la LISTA DE REQUISITOS para crear la LISTA DE REQUISITOS ELEGIDOS

Tabla 24: Requisito del Cliente: Debe adaptarse a la Imagen de la Universidad

Número:	2
Título o Nombre:	Imagen del Sistema
Descripción:	Como Estudiante Quiero Visualizar los datos de la Carrera de la Universidad Para Identificar las asignaturas de la carrera
Criterios de Aceptación y/o Condiciones:	Los colores amarillo y verde deben predominar En la ventana de ingreso debe estar el lema de la Universidad El encabezado debe ser similar al de la página web Debe verse el logo de la Universidad en todo momento en el encabezado
Fuente: Elaboración propia	

Tabla 25: Requisito del Cliente: Debe mostrarse los datos de la Carrera

Número:	3
Título o Nombre:	Datos de la Carrera
Descripción:	Como Usuario del Sistema Quiero Visualizar que el sistema se identifique con la imagen de la Universidad Para Lograr pertenencia hacia la Universidad
Criterios de Aceptación y/o Condiciones:	Los datos de la carrera deben tener el formato nombre de la universidad, nombre de la facultad, nombre de la carrera Los datos de la carrera deben incluir: nombre de la materia, sigla de la materia, docente de la materia, cantidad de horas de la materia, si es teórica, práctica o ambas, semestre a la que pertenece la materia.
Fuente: Elaboración propia	

Tabla 26: Requisito del Cliente: Debe mostrarse los datos del Estudiante

Número:	1
Título o Nombre:	Datos del Estudiante
Descripción:	Como Estudiante Quiero Visualizar mis datos de Estudiante de la Universidad Para Identificarse con la carrera
Criterios de Aceptación y/o Condiciones:	Los datos del estudiante deben ser con el formato apellido paterno apellido materno nombres Los datos del estudiante deben incluir: fecha de nacimiento, lugar de nacimiento, carrera, carnet de identidad, carnet universitario, fotografía
Fuente: Elaboración propia	

Como resultado de una adecuada comunicación se tiene sugerencias, aclaraciones, correcciones y mejoras de parte del cliente y estudiante.

Reunión Diaria Autoevaluación

- Lugar: Domicilio del Estudiante
- Fecha: agosto 2021
- Número de Periodo: 1
- Personas que asistieron a la reunión: Estudiante

Inicia la Reunión Diaria Autoevaluación, el estudiante debe concentrarse en sí mismo y autoevaluarse para identificar su trabajo hasta ese momento. Para ello se sugiere algunas interrogantes para que el estudiante pueda reflexionar sobre estas: que trabajo ya tiene realizado, que le falta por realizar, qué información, conocimientos, tecnología necesita, que errores ha tenido, que avances positivos ha tenido, como solucionar sus problemas. Como es una guía de referencia el estudiante puede aumentar o quitar elementos de autoevaluación acorde a su criterio.

Iteración de Trabajo

- Lugar: Domicilio del Estudiante
- Fecha: agosto 2021

- Número de Periodo: 1
- Personas que asistieron a la iteración de trabajo: Estudiante

Inicia la Iteración de Trabajo, se realiza la división de Requisitos Elegidos en Tareas para realizar en el Tablero de Actividades durante la Iteración del Trabajo. Se debe dividir en tareas cada uno de los Requisitos del Cliente según su criterio y capacidad de programación

Tabla 27: Requisito del Cliente: Debe adaptarse a la Imagen de la Universidad

Número:	2
Título o Nombre:	Imagen del Sistema
Descripción:	Como Estudiante Quiero Visualizar los datos de la Carrera de la Universidad Para Identificar las asignaturas de la carrera
Criterios de Aceptación y/o Condiciones:	Los colores amarillo y verde deben predominar En la ventana de ingreso debe estar el lema de la Universidad El encabezado debe ser similar al de la página web Debe verse el logo de la Universidad en todo momento en el encabezado
Fuente: Elaboración propia	

Identificar la División de Tareas de: Imagen del Sistema

Tarea 1: Determinar el Tipo de Letra, Paleta de Colores, Dimensionamiento de logos Tarea 2: Definir el espacio del encabezado, cuerpo, pie de página y menú

Tarea 3: API para el manejo de Templates Tarea 4: Diseño del Encabezado

Tarea 5: Diseño Formulario del Login

Se crea un Tablero de Actividades del Periodo 1

Tabla 28: Tablero de Actividades: Periodo N°1

Priorizadas	Ejecutándose	Probándose	Terminadas
Imagen del Sistema			*
Documentar Imagen del Sistema			*
Datos de la Carrera		*	
Documentar Datos de la Carrera		*	
Datos del Estudiante	*		
Documentar Datos del Estudiante	*		
…….			
Fuente: Elaboración Propia			

Al final de la Iteración de Trabajo se tiene un Incremento del Software y un Incremento del Documento. Se realizan correcciones y mejoras si fuesen pertinentes.

Reunión Retroalimentación

- Lugar: Universidad de Madrid
- Fecha: agosto 2021
- Número de Periodo: 1
- Personas que asistieron a la reunión: Cliente y estudiante

Inicia la Reunión Retroalimentación, el estudiante muestra y explica detalladamente el Incremento del Software al Cliente, para después realizar la retroalimentación con diferentes consultas entre el cliente y el estudiante.

El cliente testea el Incremento del Software según su criterio también evalúa la interfaz gráfica, para que paralelamente se intercambien sugerencias, dudas, aclaraciones, y/o modificaciones. Finalmente, el cliente decide aceptar el Incremento de Software quizás con algunas pequeñas modificaciones o tal como está.

Reunión Tutoría

- Lugar: Facultad de Ciencias y Tecnología
- Fecha: agosto 2021

- Número de Periodo: 1
- Personas que asistieron a la reunión: Tutor y estudiante

Inicia la Reunión Tutoría, el estudiante muestra y explica detalladamente el Incremento de Software y el Incremento de Documento, para después realizar el asesoramiento con diferentes consultas entre el tutor y el estudiante.

Se asesora el Incremento del Documento y el Incremento del Software Se evalúa el Incremento del Documento

¿Qué avances se tiene en el Incremento de Documento?

¿Qué problemas se tiene en el Incremento de Documento?

¿Qué dudas se tiene en el Incremento de Documento?

¿Qué sugerencias se tiene en el Incremento de Documento? Se testea el Incremento del Software

¿Qué avances se tiene en el Incremento de Software?

¿Qué problemas se tiene en el Incremento de Software?

¿Qué dudas se tiene en el Incremento de Software?

¿Qué sugerencias se tiene en el Incremento de Software?

¿…….?

Final del Periodo 1

De la misma forma se van realizando los siguientes Períodos según el cronograma del Proyecto de Grado.

CAPÍTULO VIII

8 VALIDACIÓN MEDIANTE UN FRAMEWORK

Este capítulo tiene el propósito de validar la propuesta mediante los indicadores de un Framework que servirá de guía para evaluar el Marco de Trabajo Ágil de Desarrollo de Software DSP. Claramente, se puede verificar el resultado de esta evaluación porque los indicadores son fáciles de interpretar y entender, además que toda la información a la que hacen referencia debe estar o no para acceder a un puntaje pertinente, por lo que se considera por los autores válido ya que se puede comprobar fácilmente la existencia o no de dicha información según el contenido del indicador.

8.1 Framework para Evaluación de Metodologías Ágiles

Para la evaluación de la propuesta se considera un Framework para Evaluación de Metodologías Ágiles, creado por: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani. Profesionales del Laboratorio de I & D en Ingeniería de Software y Sistemas de Información del Departamento de Ciencias e Ingeniería de la Computación de la Universidad Nacional del Sur en Argentina. Este Framework fue aprobado y presentado oficialmente en el XII Workshop de Investigadores en Ciencias de la Computación - WICC en junio del 2010. Actualmente considerado como una investigación muy valiosa para esta área.

Actualmente, las metodologías de desarrollo ágil se basan fundamentalmente en la colaboración con los usuarios de software durante todo el proceso de desarrollo, la facilidad para adaptar el producto a cambios en requerimientos y en la entrega incremental del producto.

Basadas en el Manifiesto Ágil, han sido aceptadas y son utilizadas con éxito en proyectos donde los requerimientos detallados son inicialmente desconocidos y se van construyendo durante el proceso de desarrollo a partir de interacciones con los usuarios y de la retroalimentación obtenida a partir de las mismas.

Los autores en su trabajo proponen un Framework de evaluación para las metodologías ágiles de desarrollo. La definición de este Framework es cuantitativa y novedosa, especialmente porque permite evaluar en cuánto las metodologías ágiles satisfacen los

principios básicos definidos por el Manifiesto Ágil que incluye cuatro postulados y una serie de principios asociados. Sus postulados son:

1) Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas.

Tres premisas sustentan este principio:

- Los integrantes del equipo son el factor principal de éxito de un proyecto;
- Es más importante construir el equipo de trabajo que construir el entorno;
- Es mejor crear el equipo y que éste configure el entorno en base a sus propias necesidades.

2) Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva.

El principio se basa en la premisa que los documentos no pueden sustituir ni ofrecer el valor agregado que se logra con la comunicación directa entre las personas a través de la interacción con los prototipos. Se debe reducir al mínimo indispensable el uso de documentación que genera trabajo y que no aporta un valor directo al producto.

3) Valorar la colaboración con el cliente por sobre la negociación contractual.

En el desarrollo ágil el cliente se integra y colabora con el equipo de trabajo como un integrante más. El contrato en sí no aporta valor al producto, es sólo un formalismo que establece líneas de responsabilidad entre las partes.

4) Valorar la respuesta al cambio por sobre el seguimiento de un plan.

La evolución rápida y continua deben ser factores inherentes al proceso de desarrollo. Se debe valorar la capacidad de respuesta ante los cambios por sobre la capacidad de seguimiento y aseguramiento de planes preestablecidos.

Gráfico 16: Framework de Evaluación de Metodologías Ágiles

P1	*Valorar al individuo y las interacciones del equipo por sobre el proceso y las herramientas*		
P1.1	*Valorar al individuo y las interacciones*	P1.2	*Valorar el proceso y las herramientas*
valor	descripción	valor	descripción
0	No define roles para individuos	-5	Define actividades, entregables, herramientas de desarrollo y de gestión
1	Clara definición de roles para individuos	-3	Define actividades, entregables y herramientas de desarrollo
2	Clara definición de roles y responsabilidades	-2	Define actividades y entregables
3	Clara definición de roles, responsabilidades y conocimientos técnicos	-1	Define actividades para cada iteración
5	Clara definición de roles, responsabilidades, conocimientos técnicos e interacciones entre miembros del equipo de trabajo	0	Define actividades para el proyecto pero no a nivel de cada iteración
P2	***Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva***		
P2.1	*Valorar desarrollo de software que funcione*	P2.2	*Valorar documentación exhaustiva*
valor	descripción	valor	descripción
0	Generar entregable al finalizar el proyecto	-5	Requiere documentación detallada al comienzo del proyecto
3	Generar entregable con testing satisfactorio al finalizar cada iteración	-2	Requiere solo documentación necesaria al comienzo de cada iteración
5	Generar entregable con testing satisfactorio e integrado con el resto de las funciones al finalizar cada iteración	0	No requiere documentación para comenzar a implementar la funcionalidad incluida en una iteración
P3	***Valorar la colaboración con el cliente pro sobre la negociación contractual***		
P3.1	*Valorar la colaboración con el cliente*	P3.2	*Valorar la negociación contractual*
valor	descripción	valor	descripción
0	Cliente colabora a demanda del equipo	-5	Existe contratación detallada al inicio y no se aceptan cambios
3	Cliente es parte del equipo, responde consultas y planifica las iteraciones	-2	La contratación exige contemplar cambios durante el proyecto
5	Cliente es parte del equipo, responde consultas, planifica iteraciones, y colabora en la escritura de requerimientos y pruebas	0	El contrato por la construcción del producto no aporta valor al producto.
P4	***Valorar la respuesta al cambio por sobre el seguimiento de un plan***		
P4.1	*Valorar la respuesta al cambio*	P4.2	*Valorar el seguimiento de un plan*
valor	descripción	valor	descripción
0	No prevé incorporar cambios durante la ejecución del proyecto	-5	Define un plan detallado al inicio del proyecto
1	Prevé introducir sólo cambios de alta prioridad	-3	Define un plan detallado de iteraciones, no acepta cambios durante una iteración
4	Permite la evolución y el cambio, pero no es recomendable en la iteración en curso	-2	Define un plan detallado para cada iteración, que puede ser modificado
5	Permite introducir cambios en la iteración en curso	0	No define planificación alguna

Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani

8.2 Evaluación de la propuesta utilizando el Framework de Evaluación para Metodologías Ágiles

El Framework de Evaluación para Metodologías Ágiles propone cuatro cuadros para realizar la valoración pertinente con los cuatro postulados del manifiesto ágil y sus ponderaciones pertinentes.

Tabla 29: Framework de Evaluación para Metodologías Agiles 1

Postulado 1	*Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas*		
P1.1	*Valorar al individuo y a las interacciones del equipo de desarrollo*	*P1.2*	*Valorar el proceso y las herramientas*
Valor	Descripción	Valor	Descripción
0	No define roles para individuos	-5	Define actividades, entregables, herramientas de desarrollo y de gestión
1	Definición clara de roles para individuos	-3	Define actividades, entregables y herramientas de desarrollo
2	Definición clara de roles y responsabilidades	-2	Define actividades y entregables
3	Definición clara de roles, responsabilidades y conocimientos técnicos	-1	Define actividades para cada iteración
5	Definición clara de roles, responsabilidades, conocimientos técnicos e interacción entre miembros del equipo de trabajo	0	Define las actividades del proyecto, pero no a nivel de iteración
Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani			

Puntaje P1.1 = 5

Definición clara de roles, responsabilidades, conocimientos técnicos e interacción entre miembros del equipo de trabajo.

Validación del Puntaje:

Tutor

Según el Artículo 10 y 11 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología. Se define al tutor como un profesional que posee grado académico igual o superior al que postula el estudiante. Este podrá ser un docente de la facultad o un profesional independiente sugerido por el estudiante y aceptado por el director de carrera.

Funciones: Están establecidas por los reglamentos anteriormente mencionados. En este contexto, el tutor puede:

- Asesorar el Incremento del Software y el Incremento del Documento
- Realizar pruebas o test al software desarrollado
- Evaluar el Incremento del Documento
- Asistir a las Reuniones Tutoría coordinadas con el estudiante

Cliente

Persona o personas interesadas e involucradas en el desarrollo del software que usaran y/o se beneficiaran del mismo. Pueden ser empresas públicas o privadas. A veces se establece un acuerdo académico verbal o escrito entre cliente y la carrera, en este convenio queda claramente establecido que el desarrollo del software tiene una finalidad académica sin retribución financiera.

Funciones:

- Identificar los elementos de la Lista de Requisitos que son los requerimientos del cliente que debe tener el software
- Expresar claramente los elementos de la Lista de Requisitos
- Ordenar los elementos de la Lista de Requisitos por prioridades para crear la Lista de Requisitos Elegidos
- Realizar pruebas funcionales, no funcionales, de aceptación al Incremento del Software
- Proporcionar la información necesaria para el desarrollo del software
- Sugerir correcciones y/o mejoras al Incremento del Software
- Aclarar dudas del estudiante
- Testear el Incremento del Software
- Aceptar el Incremento del Software
- Asistir a las Reuniones Requisitos y Reuniones Retroalimentación coordinadas con el estudiante

Estudiante

Estudiante universitario que cursa el décimo semestre de la carrera de Ingeniería de Sistemas; que ha elegido como modalidad de graduación la realización de un proyecto de grado mediante el desarrollo de software. Debe tener aprobado el perfil de su proyecto de grado; Asignado un Tutor/a para la realización de su proyecto de grado; y un cliente/usuario comprometido a colaborar con el desarrollo del software. Es autoorganizado, elige la mejor forma de gestionar su trabajo. Entrega Incrementos de Software que tienen funcionalidad de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación con el cliente y el tutor.

Funciones:

- Desarrollar el Software con los requerimientos del cliente para su defensa publica ante el tribunal designado
- Elaborar el Documento del Trabajo de Graduación (Documento) acorde al formato establecido por el reglamento de graduación, para su defensa publica ante el tribunal designado
- Aprender constantemente basándose en el empirismo
- Elaborar la Lista de Requisitos con los elementos que identifica el cliente coordinando con este.
- Elaborar la Lista de Requisitos Elegidos con los elementos seleccionados de la Lista de Requisitos para ser implementados en la Iteración de Trabajo
- Dividir los elementos de la Lista de Requisitos Elegidos en tareas a ser realizadas en el Tablero de Actividades
- Realizar un Tablero de Actividades con las tareas priorizadas para visualizar el flujo de trabajo
- Realizar el Incremento de Software en cada Iteración de Trabajo
- Realizar el Incremento de Documento en cada Iteración de Trabajo
- Convertir los elementos de la Lista de Requisitos en Incrementos de Software que tengan funcionalidad al final de cada Periodo
- Testear el Incremento de Software
- Evaluar el Incremento de Software
- Realizar correcciones y mejoras que sean pertinentes

Puntaje P1.2 = -2

Define actividades y entregables Validación del puntaje:

Entregables:

Software

Es el software que debe desarrollar el estudiante que cumple con los requisitos establecidos por el cliente. Este componente debe ser defendido para su aprobación ante un tribunal en una defensa publica, para poder culminar el plan de estudios y poder obtener el diploma académico correspondiente a la carrera de Ingeniería de Sistemas. Debe ser desarrollado de forma iterativa e incremental paralelamente al Documento del Trabajo de Graduación. Debe ser asesorado, testeado por el tutor. Debe ser testeado, aceptado por el cliente quien puede sugerir modificaciones. Y también debe ser testeado por el estudiante.

Documento

Es el Documento del Trabajo de Graduación que establece los artículos 23, 40 y 42 del Reglamento General de Graduación para carreras de Licenciatura y Técnico superior de la Facultad de Tecnología, que debe elaborar el estudiante, Su estructura y características deben estar en función a lo determinado por los artículos ya mencionados. Este componente debe ser defendido para su aprobación ante un tribunal en una defensa publica, para poder culminar el plan de estudios y poder obtener el diploma académico correspondiente a la carrera de Ingeniería de Sistemas. Debe ser elaborado por el estudiante de forma iterativa e incremental paralelamente al software. Debe ser documentado por el estudiante con todos los capítulos del proyecto de grado acorde a reglamentación de la carrera de Ingeniería de Sistemas. Debe ser evaluado y asesorado por el tutor.

Actividades:

Tablero de Actividades

Es un tablero de actividades, para ver el flujo de acciones y/o tareas realizadas por el estudiante durante el proceso de desarrollo de software, para ver la etapa en la que se encuentra cada Requisito Elegido del Cliente y su prioridad de realización. Debe incluir paralelamente su registro en el Incremento del Documento.

Considerando la capacidad de programación del estudiante, se debe realizar un tablero de actividades para cada Iteración de Trabajo donde el estudiante deberá especificar la cantidad de columnas y filas que desea utilizar que estén acordes al desarrollo de software. El estudiante debe limitar la cantidad de trabajo en progreso, para concentrarse en los Requisitos Elegidos del Cliente, que sean adecuados en el momento oportuno, ya que dedicarse a muchos requisitos al mismo tiempo causa la perdida de concentración y ser menos productivo.

Los Requisitos Elegidos del Cliente, son divididos en tareas acorde a la capacidad y criterio del estudiante.

La estructura del Tablero de Actividades está compuesta por:

- Fichas o tarjetas, cada una de ellas representara una tarea del Requisito Elegido del Cliente
- Columnas, es una secuencia de estados específicos sucesivos por los que debe transitar una tarea del Requisito Elegido del Cliente, desde que se solicita hasta que está realizada. Cada columna visualiza una fase o etapa del proceso de realización de la tarea. El nombre de la columna está en función del criterio del estudiante
- Filas o carriles, representan diferentes tipos de actividades y/o tareas específicas, se las utiliza para agrupar requisitos según el tipo de trabajo (ejemplo: desarrollo web) o según su prioridad (ejemplo alta, media, baja)

Se escribe siguiendo el siguiente formato sugerido

Tablero de Actividades: Período N°__				
Lista de Requisitos Elegidos	Priorizadas	Ejecutándose	Probándose	Terminadas
Es una lista de todos los Requisitos Elegidos del Cliente que se implementaran y que están a su vez divididos en tareas	Son 1, 2 o 3 Tareas del Requisito Elegido del Cliente priorizadas por el estudiante	Son 1, 2 Tareas del Requisito Elegido del Cliente (máximo 3 simultáneamente) que se encuentran en proceso de ejecución	Son 1, 2 Tareas del Requisito Elegido del Cliente (máximo 3 simultáneamente) que se encuentran en pruebas o testeo	Es una lista de todas Tareas del Requisito Elegido del Cliente que se van terminando

Puntaje Total P1 = 5 + (-2) = 3

Tabla 30: Framework de Evaluación para Metodologías Ágiles 2

Postulado 2	*Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva*		
P2.1	*Valorar el desarrollo de software que funcione*	*P2.2*	*Valorar la documentación exhaustiva*
Valor	Descripción	Valor	Descripción
0	Generar entregable al finalizar el proyecto	-5	Requiere documentación detallada al comienzo del proyecto
3	Generar entregable con pruebas satisfactorias al finalizar cada iteración	-2	Solo requiere documentación necesaria al comienzo de cada iteración
5	Genera entregable con pruebas satisfactorias e integrado con el resto de las funciones al finalizar cada iteración	0	No requiere documentación para comenzar a implementar la funcionalidad definida para una iteración
Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani			

Puntaje P2.1 = 3

Generar entregable con pruebas satisfactorias al finalizar cada iteración Validación del

Iteración de Trabajo

En esta Iteración de Trabajo el estudiante se encuentra solo, el tiempo de realización está en función del criterio del estudiante. El estudiante aprende con cada Iteración de Trabajo necesario para conseguir transformar los requisitos en software útil, basándose en el empirismo. En base a su experiencia, el estudiante ira aprendiendo a estimar el tiempo que le tome desarrollar las diferentes funcionalidades del software que está desarrollando. Debido a que las capacidades y experiencias de los estudiantes son diferentes, esta guía deja abierto el tiempo de realización de cada Iteración de Trabajo.

En la Iteración de Trabajo el estudiante realiza:

- División de Requisitos Elegidos en tareas a realizar en el Tablero de Actividades según su capacidad de trabajo y criterio personal.
- El Tablero de Actividades en función a la Lista de Requisitos Elegidos
- El Incremento del Software
- El Incremento del Documento
- Testear el Incremento de Software
- Evaluar el Incremento del Documento
- Se pueden introducir cambios durante las iteraciones
- Las correcciones y mejoras pertinentes en el Incremento de Software y en el Incremento del Documento.

Puntaje:

Puntaje P2.2 = -2

Solo requiere documentación necesaria al comienzo de cada iteración Validación del Puntaje:

Lista de Requisitos Elegidos

Es una lista ordenada de elementos seleccionados, de la Lista de Requisitos para ser programados y documentados en la Iteración de Trabajo. Además, debe tener el objetivo de la Iteración de Trabajo y el trabajo necesario para transformar esos elementos seleccionados en un Incremento de Software terminado que sea útil y funcional.

Cuando se requiere nuevos elementos, el estudiante lo adiciona a la Lista de Requisitos Elegidos para ser ejecutados en esa Iteración de Trabajo o a la Lista de Requisitos para ser implementados después.

A. Elementos de la Lista de Requisitos Elegidos: Se escribe siguiendo el siguiente formato sugerido

Requisito Elegido del Cliente	
Título o Nombre:	
Descripción: Como…Quiero…Para…	
Criterios de Aceptación:	
Objetivo de la Iteración:	
Lo que se necesita para transformar el requisito en software útil:	
Fuente: Elaboración propia	

El Requisito Elegido del Cliente puede ser modificada para aumentar el nivel de detalle acorde al criterio del estudiante. Toda esta información junto con lo que corresponde según reglamento deberá registrarse en el Incremento del Documento paralelamente al desarrollo del Incremento del Software.

Puntaje Total P2 = 3 + (-2) = 1

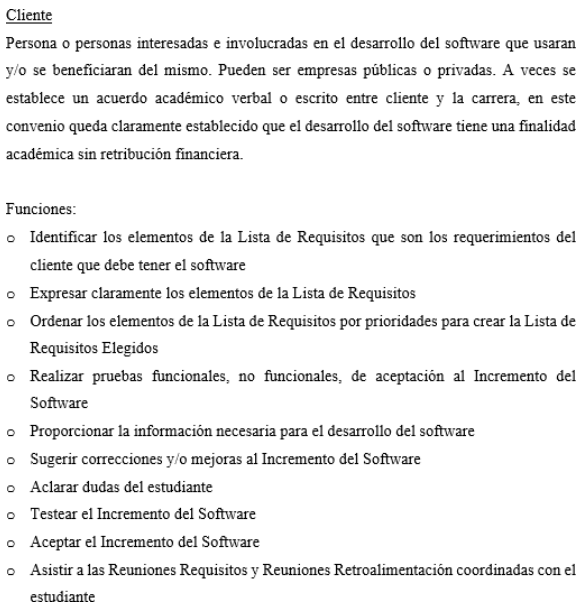
Tabla 31: Framework de Evaluación para Metodologías Ágiles 3

Postulado 3	*Valorar la colaboración con el cliente por sobre la negociación contractual*		
P3.1	Valorar la colaboración con el cliente	P3.2	Valorar la negociación contractual
Valor	Descripción	Valor	Descripción
0	El cliente colabora a petición del equipo	-5	Existe un contrato detallado y no se aceptan cambios
3	El cliente es parte del equipo. Responde preguntas y planifica iteraciones	-2	El contrato exige considerar cambios durante el proyecto
5	El cliente es un miembro del equipo, responde preguntas, planifica iteraciones y colabora en la redacción de requisitos y pruebas	0	El contrato no agrega ningún valor para la construcción de los productos del proyecto
Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani			

Puntaje P3.1 = 5

El cliente es un miembro del equipo, responde preguntas, planifica iteraciones y colabora en la redacción de requisitos y pruebas

Validación del puntaje:



Cliente

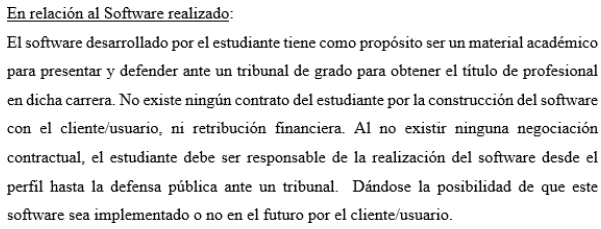
Persona o personas interesadas e involucradas en el desarrollo del software que usaran y/o se beneficiaran del mismo. Pueden ser empresas públicas o privadas. A veces se establece un acuerdo académico verbal o escrito entre cliente y la carrera, en este convenio queda claramente establecido que el desarrollo del software tiene una finalidad académica sin retribución financiera.

Funciones:

- Identificar los elementos de la Lista de Requisitos que son los requerimientos del cliente que debe tener el software
- Expresar claramente los elementos de la Lista de Requisitos
- Ordenar los elementos de la Lista de Requisitos por prioridades para crear la Lista de Requisitos Elegidos
- Realizar pruebas funcionales, no funcionales, de aceptación al Incremento del Software
- Proporcionar la información necesaria para el desarrollo del software
- Sugerir correcciones y/o mejoras al Incremento del Software
- Aclarar dudas del estudiante
- Testear el Incremento del Software
- Aceptar el Incremento del Software
- Asistir a las Reuniones Requisitos y Reuniones Retroalimentación coordinadas con el estudiante

Puntaje P3.2 = 0

El contrato no agrega ningún valor para la construcción de los productos del proyecto
Validación del puntaje:



En relación al Software realizado:

El software desarrollado por el estudiante tiene como propósito ser un material académico para presentar y defender ante un tribunal de grado para obtener el título de profesional en dicha carrera. No existe ningún contrato del estudiante por la construcción del software con el cliente/usuario, ni retribución financiera. Al no existir ninguna negociación contractual, el estudiante debe ser responsable de la realización del software desde el perfil hasta la defensa pública ante un tribunal. Dándose la posibilidad de que este software sea implementado o no en el futuro por el cliente/usuario.

Puntaje Total P3 = 5 + 0 = 5

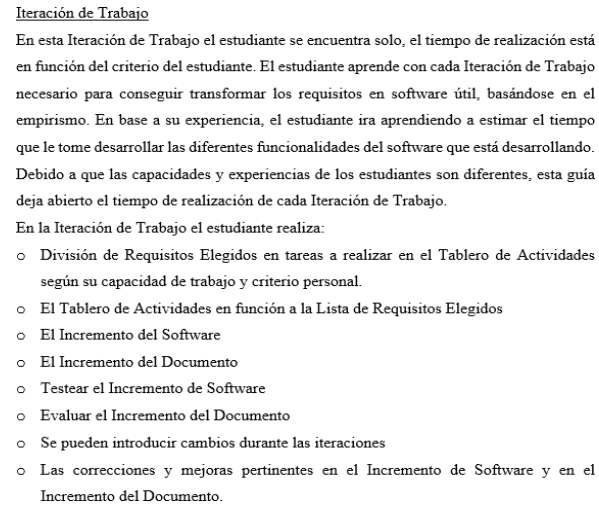
Tabla 32: Framework de Evaluación para Metodologías Ágiles 4

Postulado 4	*Valorar la respuesta al cambio por sobre el seguimiento de un plan*		
P4.1	*Valorar la respuesta al cambio*	*P4.2*	*Valorará el seguimiento de un plan*
Valor	Descripción	Valor	Descripción
0	Se permiten N cambios durante la ejecución del proyecto	-5	Define un plan detallado al inicio del proyecto
1	Solo se puede introducir cambios de alta prioridad durante la ejecución del proyecto	-3	Define un plan detallado de iteraciones y no acepta cambios durante una iteración
4	Se recomienda considerar la evolución y el cambio durante las iteraciones	-2	Define un plan detallado para cada iteración que puede ser modificado

5	Se pueden introducir cambios durante las iteraciones del proyecto	0	No define planificación alguna
Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani			

Puntaje P4.1 = 5

Se pueden introducir cambios durante las iteraciones del proyecto Validación del puntaje:



Iteración de Trabajo

En esta Iteración de Trabajo el estudiante se encuentra solo, el tiempo de realización está en función del criterio del estudiante. El estudiante aprende con cada Iteración de Trabajo necesario para conseguir transformar los requisitos en software útil, basándose en el empirismo. En base a su experiencia, el estudiante ira aprendiendo a estimar el tiempo que le tome desarrollar las diferentes funcionalidades del software que está desarrollando. Debido a que las capacidades y experiencias de los estudiantes son diferentes, esta guía deja abierto el tiempo de realización de cada Iteración de Trabajo.

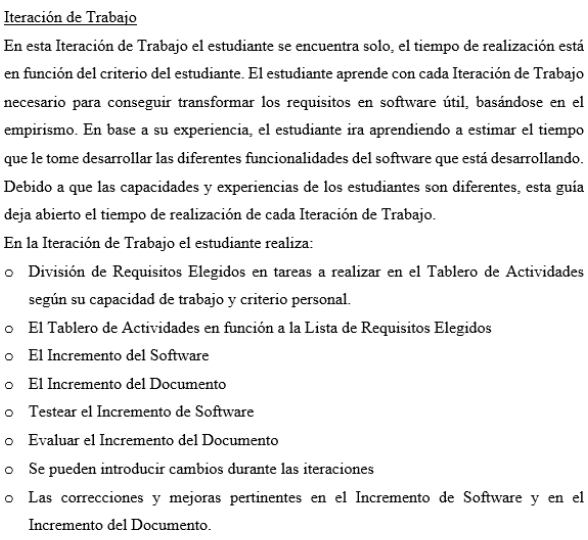
En la Iteración de Trabajo el estudiante realiza:

- División de Requisitos Elegidos en tareas a realizar en el Tablero de Actividades según su capacidad de trabajo y criterio personal.
- El Tablero de Actividades en función a la Lista de Requisitos Elegidos
- El Incremento del Software
- El Incremento del Documento
- Testear el Incremento de Software
- Evaluar el Incremento del Documento
- Se pueden introducir cambios durante las iteraciones
- Las correcciones y mejoras pertinentes en el Incremento de Software y en el Incremento del Documento.

Puntaje P4.2 = -2

Define un plan detallado para cada iteración que puede ser modificado

Validación del puntaje:



Iteración de Trabajo

En esta Iteración de Trabajo el estudiante se encuentra solo, el tiempo de realización está en función del criterio del estudiante. El estudiante aprende con cada Iteración de Trabajo necesario para conseguir transformar los requisitos en software útil, basándose en el empirismo. En base a su experiencia, el estudiante ira aprendiendo a estimar el tiempo que le tome desarrollar las diferentes funcionalidades del software que está desarrollando. Debido a que las capacidades y experiencias de los estudiantes son diferentes, esta guía deja abierto el tiempo de realización de cada Iteración de Trabajo.

En la Iteración de Trabajo el estudiante realiza:

- División de Requisitos Elegidos en tareas a realizar en el Tablero de Actividades según su capacidad de trabajo y criterio personal.
- El Tablero de Actividades en función a la Lista de Requisitos Elegidos
- El Incremento del Software
- El Incremento del Documento
- Testear el Incremento de Software
- Evaluar el Incremento del Documento
- Se pueden introducir cambios durante las iteraciones
- Las correcciones y mejoras pertinentes en el Incremento de Software y en el Incremento del Documento.

Puntaje Total P4 = 5 + (-2) = 3

Tabla 33: Framework de Evaluación para Metodologías Ágiles Resumen

Aplicación del Framework de Evaluación para Metodologías Ágiles al Marco de Trabajo DSP													
Postulados	P1			P2			P3			P4			Total 3+1+5+3
Marco de Trabajo	P1.1	P1.2	Total	P2.1	P2.2	Total	P3.1	P3.2	Total	P4.1	P4.2	Total	
Puntaje del DSP	5	-2	**3**	3	-2	**1**	5	0	**5**	5	-2	**3**	12
Fuente: Elaboración Propia													

Utilizando el Framework esta propuesta tiene el siguiente puntaje: DSP: P1(3) + P2(1) + P3(5) + P4(3) = 12

Gráfico 17: Aplicación del Framework por los Autores a Scrum y XP

Postulados	P1			P2			P3			P4		
Metodología	P1.1	P1.2	total	P2.1	P2.2	total	P3.1	P3.2	total	P4.1	P4.2	total
Scrum	5	-3	2	4	-2	2	5	0	3	4	-3	1
XP	5	-2	3	5	-2	3	5	0	5	5	-2	3

Fuente: Karla Mendes Calo, Elsa Estevez y Pablo Fillottrani

Según los autores aplicando el Framework a Scrum y XP tiene los siguientes puntajes:
Scrum: P1(2) + P2(2) + P3(3) + P4(1) = 8

XP: P1(3) + P2(3) + P3(5) + P4(3) = 14

Esta evaluación cuantitativa nos indica de qué manera las metodologías y/o marcos de trabajo cumplen con el manifiesto ágil para favorecer al desarrollo del software. En este contexto, la propuesta queda con un puntaje total de 12 puntos y se encuentra entre Scrum que tiene 8 puntos y XP que tiene 14 puntos.

BIBLIOGRAFÍA

Allen D. (2006). Organízate con eficacia: Llega más lejos de lo que nunca hubieras imaginado. Canadá. Urano

Allen D. (2015). Getting Things Done. The Arts Of Stress-Free Productivity. Canadá. Piatkus

Allen D. (2015). Organízate con eficacia. Gestión del conocimiento. El arte de la productividad sin estrés. Canadá. Penguin Books

Beck K. y Andres C (2004) Extreme Programming Explained: Embrace Change. Canadá. Addison Wesley

Beck K. y Fowler M. (2000). Planning Extreme Programming. España. Prints

Benson J. y Tonianne D, (2011). Personal Kanban. Mapping Work Navigating Life. España. Modus Operandi Press

Bourque P. y Dupuis R. (2004). Guide to the Software Engineering Body of Knowledge. Uruguay. Casanovas Editors

Cockburn A. (2006) Agile Software Development: The Cooperative Game Canadá. Addison Wesley

Espinoza A. (2013). Manual para elegir una metodología de desarrollo de software dentro de un proyecto informático. Tesis de Licenciatura. Piura Perú. Universidad de Piura, Facultad de Ingeniería, Programa Académico de Ingeniería Industrial y de Sistemas]. Repositorio Institucional.

García J. (2017). Propuesta de metodología ágil de desarrollo de software con integración de TAM – Z-ÁGIL. [Tesis de Maestría, Pontificia Universidad Católica de Valparaíso Facultad de Ingeniería Escuela de Ingeniería Informática]. Repositorio Institucional.

García W. (2016). Metodología en el Desarrollo de Software. [Informe del trabajo práctico de Licenciatura, Iquitos Perú]. Repositorio Institucional.

Hunt A. y Thomas D. (1999) Pragmatic Programmer From Journeyman to Master. Canadá. Addison Wesley

Joyanes L (2015). Fundamentos de Programación. España. Mc Graw Hill

Knuth D. (1997) Art of Computer Programming, The: Volume 1: Fundamental Algorithms. Canadá. Addison Wesley

Lawrence P. (1998). Software Engineering: Theory and Practice. España, Prints

Levine M. (2000). Analyzing the Deliverables Produced in the Software Development Life Cycle. Paraguay. Fernandez Editors

Morales R. (2015) Gestión de Tareas con Kanban: Introducción a la gestión visual del trabajo. España. Ediciones Rainer.

Paciencia J. (2015). Metodologías de Desarrollo de Software. Tesis de Licenciatura, Pontificia Universidad Católica Argentina Santa María de los Buenos Aires. Buenos Aires Argentina]. Repositorio Institucional.

Pineda M. (2015). Diseño e implantación de una tecnología para el desarrollo de aplicaciones enfocadas en la utilización de las metodologías ágiles. [Monografía de Licenciatura, Medellín Colombia. La Universidad EAFIT departamento de informática y sistemas]. Repositorio Institucional.

Pressman R. (2018). Ingeniería de Software Un enfoque práctico. España. Mc Graw Hill Interamericana.

Pressman, R. 2010. Ingeniería del Software Un Enfoque Práctico. Connecticut. McGraw-Hill Interamericana Editores.

Ramirez F (2013) Apuntes de Metodología de la investigación. Bolivia. Prisma.

Robert C. (2001). Agile Software Development, Principles, Patterns, and Practices. Canadá. Addison Presley

Schenone M. (2017). Diseño de una Metodología Ágil de Desarrollo de Software. Tesis de Licenciatura, Buenos Aires Argentina. Facultad de Ingeniería. Universidad de Buenos Aires]. Repositorio Institucional.

Scholz R. y Tietje O. (2001) Embedded Case Study Methods: Integrating Quantitative and Qualitative Knowledge. Canadá. Addison Wesley. Estados Unidos. SAGE Publications

Schwaber K. Beedle M. (2008). Agile Software Development with SCRUM. España. Prints

Schwaber K. y Beedle M. (2001) Desarrollo de Software Ágil con Scrum. España. Pearson

Schwaber K. y Beedle M. (2001). Agile Software Development with Scrum. España. Prentice Hall

Schwaber K. y Sutherland J. (2011). The Scrum Guide. The definitive guide to scrum: the rules of the game. Canadá. Addison Wesley

Schwaber M. y Sutherland J. (2012). Rubin Essential Scrum: A practical guide to the most popular agile process. Canadá. Addison Wesley

Sommerville, (2015). Ingeniería de Software. Uruguay. Pearson

Stephen R. (2016). Los 7 hábitos de la gente altamente efectiva. Uruguay. Grupo planeta.

Vernon V. (2013) Implementing Domain-Driven Design. Canadá. Addison Wesley

Vliet H. (2002). Software Engineering. Principles and Practice. España. Casanovas Editors

Weitzenfeld A. (2004) Ingeniería de software orientada a objetos con UML, Java e Internet/ Software Engineering applied to UML, Java and Internet Object. Cengage. España. Learning Latin América.

ANEXOS

Los anexos solo se hace referencia al nombre del archivo pdf que lo contiene, el mismo se adjuntará a la tesis en formato pdf ya que son muy voluminosos para incluirlos completamente.

Anexo 1: XII Congreso Nacional de Universidades – Informes, Resoluciones, Reglamentos (vigente a la fecha)

http://www.uajms.edu.bo/deva/wp-content/uploads/sites/42/2018/05/XII-Congreso- de-Universidades-CEUB.pdf

Adjunto archivo digital en pdf: XII-Congreso-de-Universidades-CEUB.pdf

Anexo 2: Plan N° 9 de Estudios de la Carrera de Ingeniería de Sistemas (vigente a la fecha)

https://tecnologia.usfx.bo/principal/ingenieria-de-sistemas/plan-de-estudios-ingenieria-de-sistemas/

Adjunto archivo digital en pdf PLAN 9 DE ESTUDIOS INGENIERÍA SISTEMAS.pdf

Anexo 3: Reglamento General de Graduación para carreras de licenciatura y técnico superior de la Facultad de Tecnología (vigente a la fecha)

http://procesos.tecnologia.usfx.bo/wp-content/uploads/2018/01/A53- REGLAMENTO-GENERAL-GRADUACION-TECNOLOGIA-2015-completo.pdf

Adjunto archivo digital en pdf A53-REGLAMENTO-GENERAL-GRADUACION-TECNOLOGÍA-2015-completo.pdf

Anexo 4: Reglamento General y Reglamentos Específicos de Universidades Privadas

https://www.minedu.gob.bo/files/publicaciones/vesfp/dgesu/REGLAMENTO-U-PRIVADAS.pdf

Adjunto archivo digital en pdf REGLAMENTO-U-PRIVADAS.pdf

Anexo 5: Manifiesto Ágil

https://agilemanifesto.org/

Adjunto archivo digital en pdf agile-manifesto.pdf

Anexo 6: Framework para Evaluación de Metodologías Ágiles

https://core.ac.uk/download/pdf/301041546.pdf

https://www.researchgate.net/publication/242584795_Un_Framework_para_Evaluacion_de_Metodologias_Agiles

Adjunto archivos digitales en pdfs: VALORACIÓN METODO AGIL 1.pdf VALORACIÓN METODO AGIL 2.pdf

Documento Completo.pdf

Un Framework para Evaluación de Metodologías Agile.pdf

Anexo 7: Cuestionario Aplicado a la Muestra mediante Google Forms vía Internet

Cuestionario

OBJETIVO: Recabar información sobre la realización de proyectos de grado mediante el desarrollo de software, para la Tesis de Maestría en Software Libre.
Estimado (a) estudiante: Esta encuesta que te solicito contestar, solo tiene fines académicos para un estudio investigativo, por lo que la información que proporciones es absolutamente confidencial y sus respuestas tienen el carácter de anónimo y no necesita poner su nombre, ni correo electrónico, por lo que puede ser lo más sincero posible. La información que facilite es muy valiosa. Espero tu colaboración. Gracias.

*Obligatorio

1. ¿Qué metodología o marco de trabajo de desarrollo de software utilizaste en tu Proyecto de Grado? *

- ◯ Metodología o Marco Ágil
- ◯ Metodología o Marco Tradicional
- ◯ Metodología o Marco Hibrida
- ◯ Ninguno
- ◯ Todas

2. ¿Conoce alguna metodología o marco de trabajo ya sea ágil o tradicional de desarrollo de software que este enfocado específicamente al desarrollo de software de forma individual en 20 semanas? *

- ◯ Si conozco
- ◯ No conozco

3. ¿Conoces todas las funciones reglamentarias que debe realizar un tutor/a de Proyecto de Grado? *

- ◯ No conozco ninguna
- ◯ Conozco algunas
- ◯ Si conozco todas

4. ¿Tuviste algún problema durante la realización de tu Proyecto de Grado. en el desarrollo de software. relacionado con el uso de la metodología o marco de trabajo ya sea ágil o tradicional que escogiste utilizar? *

- ◯ Ningún problema
- ◯ Algunos problemas
- ◯ Muchos problemas

5. ¿Cuántas horas le dedicas al día para realizar tu Proyecto de Grado? *

- ◯ Menos de una hora
- ◯ 1 - 4
- ◯ 5 - 8
- ◯ 9 - 12
- ◯ Mas de 12 horas

6. ¿Con que frecuencia te reúnes con tu tutor/a asignado para tu Proyecto de Grado? *

- ◯ No se si tengo tutor asignado
- ◯ No tengo tutor asignado
- ◯ No me reúno
- ◯ Diariamente
- ◯ Semanalmente
- ◯ Mensualmente
- ◯ 2 veces al Semestre
- ◯ Otras frecuencias

7. ¿Con que frecuencia te reúnes con el cliente/usuario del software que estas desarrollando para tu Proyecto de Grado? *

- ◯ No me reúno
- ◯ Diariamente
- ◯ Semanalmente
- ◯ Mensualmente
- ◯ 2 veces al Semestre
- ◯ Otras frecuencias

8. ¿Con que frecuencia autoevalúas el Proyecto de Grado que estas desarrollando y a ti mismo en relación a tu trabajo académico? *

- ◯ No me autoevaluó
- ◯ Diariamente
- ◯ Semanalmente
- ◯ Mensualmente
- ◯ 2 veces al Semestre
- ◯ Otras frecuencias

9. ¿La metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado. estaba orientado específicamente al tiempo de realización de un semestre académico de 20 semanas? *

- ◯ Si
- ◯ No

10. ¿Realizaste modificaciones. adecuaciones y/o improvisaciones a la metodología o marco de trabajo de desarrollo de software ya sea ágil o tradicional que utilizaste en tu Proyecto de Grado? *

- ◯ Ninguna
- ◯ Algunas
- ◯ Muchas

11. ¿Conoces el Reglamento de Graduación de la Facultad de Ciencias y Tecnología? *

- ◯ Si conozco el contenido
- ◯ Conozco algo del contenido
- ◯ Se que existe, pero no conozco el contenido
- ◯ No sabía que existía y no conozco el contenido

12. ¿El software de tu Proyecto de Grado tiene requisitos del cliente muy cambiantes? *

- ◯ Si tiene
- ◯ No tiene

Enviar