Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Sistema de Recomendación de Metodologías de Desarrollo de Software

Memoria para optar al título de Ingeniero Civil en Informática

Concepción, 2018

Autores

Fabián Alejandro Ávila Cáceres
Richard Andres Fica Inzunza

Profesor guía
Elizabeth Eliana Grandón Toledo

Resumen

El proyecto titulado "Sistema de Recomendación de Metodologías de Desarrollo de Software" se presenta para dar conformidad a los requisitos exigidos por la Universidad del Bío-Bío en el proceso de titulación para la carrera Ingeniería Civil en Informática.

En las últimas décadas se han desarrollado diferentes metodologías de desarrollo de software (SDM en su sigla en inglés, Systems Development Metholodology) con la finalidad de mejorar y optimizar los procesos de desarrollo. Una de las principales razones de la variedad de metodologías de desarrollo de software es la necesidad de desarrollar diferentes tipos de sistemas y disminuir sus costos asociados.

Dada la importancia de adoptar una óptima metodología de desarrollo de software en la creación de nuevos proyectos informáticos y la necesidad de garantizar un mayor éxito en la distribución de recursos, se ha desarrollado un sistema de recomendación de metodologías que permite escoger y orientar a desarrolladores con bajo nivel de experiencia o conocimientos a seleccionar un grupo de metodologías de desarrollo de software adecuada basada en diferentes criterios. Estos criterios se obtuvieron mediante encuestas realizadas a empresas consideradas relevantes en Chile junto con las de diferentes fuentes de literatura las que serán expuestas más adelante. El sistema permite al usuario revisar información de diferentes grupos de metodologías de desarrollo de software y prácticas que podrá implementar con las metodologías ya definidas y comparar resultados prácticos con la teoría.

Este proyecto fue desarrollado con el fin de ayudar y orientar a profesionales que necesiten aplicar, conocer y/o reforzar más sobre el uso e implementación de metodologías de desarrollo de software en sus futuros proyectos, además con las intención de comparar resultados prácticos con las teoría para así dar a conocer la realidad de las empresas dedicadas al desarrollo de sistemas en Chile.

Abstract

The project entitled "Software Development Methodology Recommendation System" is presented to comply with the requirements of the Universidad Del Bío-Bío in the process of qualification for the Civil Engineering degree in Computer Science.

In the last decades, different software development methodologies (SDM) have been developed in order to improve and optimize the development processes. One of the main reasons for the variety of software development methodologies is the need to develop different types of systems and lower their associated costs.

Given the importance of adopting optimal software development methodologies in the creation of new IT projects and the need to guarantee greater success in the distribution of resources, a method recommendation system has been developed that allows to choose and guide developers with Low level of experience or knowledge to select a group of appropriate software development methodologies based on different criteria. These criteria were obtained through surveys conducted with companies considered relevant in Chile along with those from different literature sources, which will be discussed later. The system allows the user to review information from different groups of software development methodologies and practices that can be implemented with the already defined methodologies and compare practical results with the theory.

This project was developed in order to help and guide professionals who need to apply, know and / or reinforce more about the use and implementation of software development methodologies in their future projects, also with the intention of comparing practical results with theories in order to make known the reality of the companies dedicated to the development of systems in Chile.

Índice General

Índice Tablas

Índice Figuras

INTRODUCCIÓN

La informática hoy en día ha avanzado enormemente con el paso de los años, también se han ido incorporando nuevas formas de trabajo y desarrollo. En el ámbito del desarrollo de sistemas se han ido estableciendo diversas metodologías que sirven como una guía y presentan prácticas que se deberían realizar durante el desarrollo de un sistema para que este tenga éxito, según diversos criterios y formas de trabajo. Al pasar los años se han ido creando nuevas metodologías de desarrollo que han agregado o modificado puntos en la forma de trabajo, ya sea enfocando nuevas áreas, el desarrollo comunicativo con los clientes o el privilegiar los avances funcionales. Debido a esto, las metodologías de desarrollo de software se han convertido en una parte fundamental para realizar distintos proyectos de software, las empresas que presentan áreas de desarrollo han ido utilizando y adaptando según sus necesidades.

En la Universidad del Bío-Bío, para las carreras de Ingeniería Civil Informática e Ingeniería (E) Computación en Informática, se dispone de ramos como Metodología de Desarrollo de Software e Ingeniería de Software, donde se presentan y se puede ver la importancia de estas metodologías en el área. Sin embargo, no existen estudios que muestren qué tipos de metodologías están utilizando las empresas chilenas y menos aún un sistema informático que recomiende el uso de un grupo de metodologías de acuerdo a la realidad nacional.

Este documento constara de doce capítulos:

En el primer capítulo se dará a conocer un estudio que demuestra la importancia del uso de las metodologías para proyectos exitosos, conceptos claves, así como la descripción de la problemática y área de estudio.

En el segundo capítulo se encontrara la definición del proyecto junto a los objetivos a cumplir.

En el tercer capítulo conoceremos las diferentes metodologías pertenecientes a sus respectivos grupos, además de los criterios fundamentales según la literatura, los cuales son la base para el desarrollo del instrumentó de captación de datos (Encuesta).

En el cuarto capítulo detallaremos es proceso de recolección de datos y los sujetos de estudio.

En el quinto capítulo conoceremos los resultados de la aplicación de la encuesta, junto con las reglas definidas tras el análisis de datos obtenidos.

En el sexto capítulo encontramos detalles sobre el alcance del software, descripción global del producto y requerimientos de este.

En el séptimo capítulo se detallara la factibilidad, tanto técnica como operativa y económica.

En el octavo capítulo encontraremos detalles del análisis, dentro de lo encontramos diagramas de flujo de datos, casos de uso, especificación de estos, entre otros.

En el noveno capítulo se encuentra toda la información asociada al diseño; base de datos, sistema de navegación, e interface.

En el décimo capítulo se detallan las pruebas realizadas al sistema.

En el undécimo capítulo se menciona la futura puesta en marcha que nuestro sistema tendrá una vez habilitado el servidor.

En el duodécimo capítulo encontraremos una breve conclusión de lo que ha sido la investigación y desarrollo del proyecto.

1 MARCO TEORICO

En las últimas décadas se han desarrollado diferentes metodologías de desarrollo de software (SDM en su sigla en inglés, Systems Development Metholodology) con la finalidad de mejorar y optimizar los procesos de desarrollo. Una de las principales razones de las variedades de SDM es la necesidad de desarrollo de diferentes tipos de sistemas de software.

Dado que las SDM son fundamentales para la creación de proyectos informáticos es necesario utilizar la más adecuada para este propósito. Se pueden utilizar dependiendo del tipo de proyecto que se desee llevar a cabo, sin embargo no se utiliza siempre la más adecuada tomando en cuenta las herramientas que se dispone para el desarrollo de este, la experiencia y conocimientos de los desarrolladores. En el informe de caos (Chaos Report, 2015) se muestra la alta tasa de fracaso en el desarrollo e implementación de proyectos de software. Por ejemplo, para el año 2015 se han estudiado alrededor de 50.000 proyectos en todo el mundo desde mantenimientos pequeños hasta gigantescos proyectos de reingeniería, de los cuales solo el 29% fueron exitosos, el 52% discutidos y el 19% fallidos (ver Figura 1).

Exitosos: Son aquellos proyectos en lo que no hay duda que fueron un éxito.
Discutidos: Son aquello proyectos en los que hay duda si tuvieron éxito o fueron un fracaso.
Fallidos: Son aquellos en los que no hay duda de que fueron un fracaso.

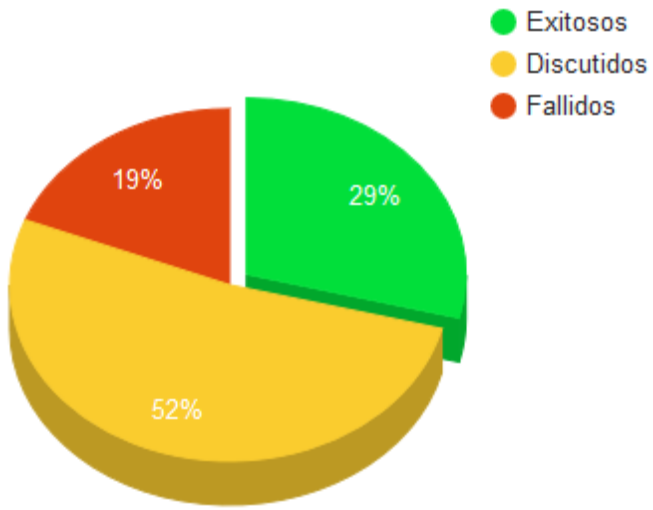


FIGURA N° 1: Proyectos 2015
Fuente: Chaos Report (2015)

Los desarrolladores de software así como las empresas del rubro, deben utilizar las SDM constantemente, pero el uso de estas no siempre es el adecuado. Al obtener resultados buenos según sus criterios, no se busca la optimización ni mejora de estos resultados, quedando atrás la posibilidad de crear software en menor tiempo, más eficientemente, con menos costos de recursos humanos, económicos, entre otros. Por otra parte los trabajadores evitan el realizar tareas más complejas y prácticas formales que las SDM establecen. Además no se busca una capacitación para obtener mayor conocimiento en las nuevas prácticas que se llevan a cabo en el desarrollo de software.

En base a la información obtenida de diferentes fuentes, como lo son literaturas e investigaciones, hemos concluido que la selección de metodologías por parte de equipos de desarrollo no siempre es la adecuada. Al seleccionar una metodología adecuada se obtendrían mejores resultados al finalizar el desarrollo, en comparación a los que se obtienen actualmente con las elecciones que realizan las empresas basándose en utilizar frecuentemente la que les ha dado resultados, en lugar de escoger la que mejor se adapte a las características de desarrollo de su proyecto.

Es por esto que desarrollamos un sistema de recomendación de metodologías que permitirá escoger u orientar a los desarrolladores con bajo nivel de experiencia o conocimientos a seleccionar un grupo de metodologías adecuado basado en diferentes criterios, que se obtuvieron de la recopilación de información obtenida directamente de las empresas de desarrollo de software y de la revisión de la literatura. Esto nos permitió comparar resultados prácticos con la teoría, la cual sirvió como base para el desarrollo del sistema de recomendación.

1.1 Definiciones de conceptos y criterios

Software:

Recurriendo al diccionario de informática publicado originalmente por la Oxford University Press (1993) el término software o programa se aplica a aquellos componentes de un sistema informático que no son tangibles, es decir, que físicamente no se pueden tocar.

Para Freedman (1984) el término software o programa es sencillamente el conjunto de instrucciones que contiene la computadora, ya sean instrucciones para poner en funcionamiento el propio sistema informático (software de sistema) o instrucciones concretas dirigidas a programas particulares del usuario (software específico).

Según Sánchez Montoya (1995) el término software o programa supone un conjunto de pasos que indican a la máquina (hardware) aquello que debe hacer.

Metodología:

Conjunto de métodos que se siguen en una investigación científica, un estudio o una exposición doctrinal. (Oxford University Press, 2017).

Metodología de la investigación:

Conjunto de procedimientos y técnicas que se aplican de manera ordenada y sistemática en la realización de un estudio. (Eyssautier de la Mora, 2006).

Desarrollo de Software:

Es la construcción de un software mediante su descripción. (Pressman, 2010).

Metodología de Desarrollo de Software:

Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. (Departament of Health& Human Service -USA, 2005).

Desarrollador de software:

Un desarrollador de software es una persona interesada en las facetas del proceso de desarrollo de software, incluida la investigación, el diseño, la programación y las pruebas de software. (Enciclopedia libre Wikipedia, 2017).

Presupuesto:

Cálculo anticipado del coste para el desarrollo de un proyecto. (Oxford University Press, 2017).

Requerimientos:

Los requerimientos son declaraciones que identifican atributos, capacidades, características y/o cualidades que necesita cumplir un sistema (o un sistema de software) para que tenga valor y utilidad para el usuario. (Alegsa, 1998 - 2017).

Adaptabilidad a cambios:

Capacidad para adaptarse y avenirse a los cambios, modificando si fuera necesario la propia conducta para alcanzar determinados objetivos cuando surgen dificultades, nueva información o cambios en el medio, ya sean del entorno exterior, de la propia organización, del cliente o de los requerimientos del trabajo en sí.(Universidad de Santiago, 2013).

Tiempo Estimado:

Tiempo que se estima que durará el proceso de desarrollo del proyecto antes de su ejecución. (Pressman, 2010).

Tiempo Real:

Tiempo que duró el desarrollo del proyecto. (Pressman, 2010).

Riesgo:

Posibilidad de que se produzca un contratiempo que retrase o impida la ejecución del proyecto final. (Oxford University Press, 2017).

1.2 Descripción de la problemática

Las metodologías de desarrollo de software actualmente se han ido utilizando cada vez más, a su vez se han ido desarrollando distintas versiones con distintos enfoques. Sin embargo, no existe una forma clara para saber qué metodología puede funcionar mejor en cada proyecto, nada más que el criterio y la experiencia de los desarrolladores adquirida durante los años de trabajo.

Este proyecto busca ayudar a orientar a las personas, ya sea de empresas y/o emprendedores que no tienen la experiencia suficiente, a escoger y guiar para tomar una elección de metodología más adecuada según la situación en la que se encuentre. Para esto, en esta investigación se contrasta la experiencia proporcionada por las empresas en Chile con la teoría que establece criterios de selección de metodologías de desarrollo de software.

1.3 Descripción del área de estudio

En términos institucionales, la enseñanza del desarrollo de software normalmente se realiza en el Departamento de Sistemas de Información y en el Departamento de Ciencias de la Computación y Tecnologías de la Información de la Universidad del Bío-Bío. Actualmente en la universidad se imparten cursos de metodologías de desarrollo de software y desarrollo de sistemas de información dictado para las carreras de Ingeniería Civil Informática e Ingeniería (E) en Computación e Informática. Estas tienen como objetivo primordial entregar los conocimientos y habilidades necesarias para permitir al estudiante elaborar y proponer soluciones relacionadas al desarrollo de sistemas considerando aspectos claves para la toma de óptimas decisiones que no comprometan el desarrollo del proyecto. Las Tablas 1 y 2 muestran la lista de contenidos considerados en el curso de Metodología de Desarrollo de Software y de Desarrollo de Sistemas de Información respectivamente.

Unidades
• Introducción
• Métodos y modelos lineales
• Métodos y modelos evolutivos - incrementales
• Evaluación de los métodos y modelos de desarrollo

TABLA N° 1: Curso Metodología de Desarrollo de Software

Unidades
• Metodología SCRUM (Adaptación)
• Planificación de proyecto
• Ingeniería de requerimientos
• Configuración de entornos de desarrollo
• Uso de frameworks de desarrollo
• Desarrollo de proyecto

TABLA N° 2: Curso Desarrollo de Sistemas de Información

2 DEFINICIÓN PROYECTO

El presente proyecto consta de la generación de un sistema de recomendación de metodologías de desarrollo de software que ayude a seleccionar de mejor forma un grupo de metodologías de desarrollo a profesionales del área informática, en base a un estudio teórico y empírico en empresas de desarrollo de software en Chile.

2.1 Objetivos del proyecto

En este apartado se dará a conocer los principales objetivos a cumplir con el desarrollo de nuestro sistema de recomendación de metodologías de desarrollo de software.

2.1.1 Objetivo General

Generar un sistema de recomendación que ayude a seleccionar un grupo de metodologías de desarrollo de software, en base a un estudio teórico y empírico en empresas de desarrollo de software.

2.1.2 Objetivos Específicos

- Conocer metodologías de desarrollo de software usadas en las empresas Chilenas.
- Investigar qué metodologías presentan mayor índice de éxito/fracaso.
- Establecer factores críticos de las metodologías implementadas exitosamente.
- Crear una herramienta de software que facilite la elección de metodologías para proyectos futuros.

2.2 Ambiente de Ingeniería de Software

2.2.1 Técnicas y notaciones

A continuación se presentan las técnicas y notaciones utilizadas para el desarrollo de este proyecto de los cuales se detalla lenguaje unificado de modelado (UML), modelamiento de datos, patrón de diseño modelo vista controlador (M.V.C), paradigma de orientación a objetos, además de características propias de cada uno de estas.

Lenguaje unificado de modelado

Para la documentación y modelado de este proyecto de software, se utiliza el Lenguaje unificado de Modelado (UML), específicamente para la elaboración de los Diagramas de Casos de Usos y Diagrama de Clases. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema, cabe destacar que este lenguaje de modelos sirve para especificar o describir métodos o procesos.

Modelamiento de datos

Para la creación y modelado de la base de datos para este proyecto de software, se utiliza el modelo de entidad-relación que es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades. También se utiliza el modelo relacional que es para la creación de la base de datos.

Patrón de diseño Modelo Vista Controlador (M.V.C.)

M.V.C es un patrón de diseño que separa la lógica del problema de negocio en tres partes esenciales: el Modelo, encargado de la lógica de los datos, su integridad y coherencia; la vista, encargada de lo que se entrega al usuario, con lo que interactúa; y el controlador, que se encarga de traer los datos, aplicar la algoritmia en ellos y entregarlos al usuario en vistas. Este patrón es ideal para este trabajo, dado su base en orientación a objetos, y la utilidad que tiene al permitir documentar y modularizar el software, de modo que se trabaje de manera óptima en equipo y por versiones.

Paradigma de orientación a objetos

En el desarrollo del software se trabaja en base al paradigma de orientación a objetos. Este paradigma fomenta en gran medida la mantenibilidad, escalabilidad y la reutilización de componentes de software, ya que los programas son fáciles de diseñar y mantener. También permiten un alto nivel de abstracción mediante la representación de objetos, además permite crear sistemas más complejos y escalables, ya que el código fuente de un objeto puede inscribirse y mantenerse independiente del código fuente del resto de los objetos, de esta forma se puede ampliar el dominio del negocio sin verse afectada la totalidad del software.

2.2.2 Herramientas de apoyo utilizadas

La siguiente tabla muestra las herramientas de apoyo utilizadas para el desarrollo de este en este proyecto.

Nombre	**Descripción**
PHP	Abreviación de PHP Hypertext Pre-processor, es un lenguaje de programación que permite orientación a objetos, interpretado en el servidor y permite trabajar con toda la parte lógica del software. Versión utilizada 5.6.28
XAMPP	Es un servidor independiente de plataformas, con el que se podrá gestionar bases de datos MySql, el servidor web Apache y los intérpretes para lenguajes de scripts: PHP y Pearl.
Yii2	Framework Orientado a Objetos de alto rendimiento basado en componentes PHP. Versión utilizada 2.0.12
Sybase PowerDesigner	Herramienta Case que permite diseñar, guardar y modificar los modelos de datos, los diagramas de casos de usos y el modelo racional, todo esto en base reglas de UML. Versión utilizada es la 16.1.0.3637

Sublime Text	Editor de texto con resaltado de sintaxis que entre sus características más relevantes están: su compatibilidad con múltiples lenguajes, autocompletado de sintaxis y fácil sistema para manejar proyectos grandes. Versión utilizada es la Build.3126
PHPMyAdmin	Sistema administrador de bases de datos relacionales MySQL.
Google Docs.	Es un procesador de texto online que permite crear y dar formato a documentos de texto, además de colaborar con otras personas en tiempo real.
Formularios de Google Docs.	Formularios que permite crear encuestas para recopilar, administrar y distribuir la información automáticamente, así como controlar las personas que han respondido, y crear gráficos para usos estadísticos.
Firefox	Firefox es un navegador web diseñado por Mozilla, de software libre.
Google Chrome	Navegador web desarrollado por Google y compilado con base en varios componentes e infraestructuras de desarrollo de aplicaciones.

TABLA N° 3: Herramientas de apoyo

2.2.3 Definiciones, siglas y abreviaciones

La Tabla 4 muestra las definiciones, siglas y abreviaciones de los principales conceptos incluidos en este informe.

Concepto	Descripción
SDM	Metodología de desarrollo de software (en su sigla en inglés, Systems Development Metholodology)
MER	Un diagrama o Modelo Entidad-Relación es una herramienta para el modelado de datos que permite representar las entidades relevantes de un sistema de información, así como sus interrelaciones y propiedades.
MR	El modelo relacional utilizado para la gestión de una base de datos, es un modelo de datos basado en la lógica de predicados y en la teoría de conjuntos.
SQL	Abreviación de StructuredQueryLanguage (Lenguaje Estructurado de Consultas), es el lenguaje actual para la gestión de base de datos relacionales.
UML	Lenguaje Unificado de Modelado, es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
Wireframe	Guía visual que representa el esqueleto o estructura visual del sitio web.
Herramientas CASE	Abreviación de ComputerAided Software Engineering, Ingeniería de Software Asistida por Computadora.
HTML	Abreviación de HypertextMarkupLanguage (Lenguaje de marcado de hipertexto), el cual es utilizado para la parte visible de la página.
HTTP	Abreviación de Hypertext Transfer Protocol (protocolo de

	transferencia de hipertexto), el cual es utilizado para las aplicaciones web en una arquitectura cliente - servidor.
CSS	Abreviación de Cascading Style Sheet (Hojas de Estilo en Cascada) Lenguaje de maquetación web que define los elementos de interfaz de usuario en pantalla.
Framework	Nombre dado a los marcos de trabajo usados en el desarrollo de aplicaciones, en este caso para Yii Framework.

TABLA N° 4: Definiciones, siglas, abreviaciones

3 METODOLOGÍA DE DESARROLLO DE SOFTWARE

3.1 Clasificación de grupos de metodología

A continuación se presentan los cuatro grandes grupos de metodologías de desarrollo de software (lineales, evolutivas, incrementales y ágiles) en los cuales se mencionan las metodologías o modelos pertenecientes a dichos grupos.

3.1.1 Grupo de Metodologías Lineales

En este grupo se encuentra una serie de metodologías que en cierta forma, comparten algunas características. Dentro de las que se discutirán en este informe se encuentran el modelo cascada, cascada con iteración, cascada con sub-proyectos, structured system analysis and desingn method (SSADM) y Case de Oracle.

3.1.2 Grupo de Metodologías Evolutivas

Dentro del grupo de metodologías evolutivas, se han considerado las siguientes: prototipo evolutivo, entrega por etapas, modelo espiral, cascada con reducción de riesgo.

3.1.3 Grupo de Metodologías Incrementales

En este grupo se encuentra una serie de metodologías que comparten algunas características. Dentro de las que se encuentran: modelo incremental, rational unified process (RUP), entrega por etapas y diseño por planificación.

3.1.4 Grupo de Metodologías Ágiles

Dentro del grupo de metodologías ágiles, se han considerado las siguientes: scrum, extreme programming (XP), dynamic systems development method (DSMD), adaptive software development (ASD), crystal clear, feature driven development (FDD), lean software development (LSD), kanban, open up.

Para obtener mayor información respecto de las metodologías pertenecientes a los diferentes grupos revisar anexo: METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

3.2 Justificación de selección de criterios

La literatura indica una serie de criterios a considerar al momento de seleccionar una metodología de desarrollo. A continuación se describe los principales criterios incorporados en este estudio:

a) Presupuesto apropiado: Pressman (2010) indica que el presupuesto debe estar acorde al proyecto evitando el parar el desarrollo retrasando y aumentar plazos establecidos.

b) Requerimientos específicos: Pressman (2010) indica que contar con requerimientos específicos al inicio del proyecto permite definir tareas, alcance y la naturaleza del problema que se va a resolver, mientras se define lo que se requiere en el proceso de elaboración donde se refina y modifican los requerimientos.

c) Variabilidad de requerimientos: Pressman (2010) afirma que si bien los cambios son inevitables estos deben tomarse en cuenta y con mucha importancia, ya que a medida que avanza el tiempo, el cambio puede afectar en mayor medida al desarrollo del proyecto, sobre todo en metodologías tradicionales como es cascada.

d) Participación del cliente: En el estudio de Ahimbisibwe, Cavana&Daellenbach (2015) se recolectaron diversos factores críticos de éxito, y entre los factores que competen al cliente se agregó la participación de este.

e) Adaptabilidad a cambios: Pressman (2010) indica que los cambios suceden, pero es relevante tener esto en cuenta ya que los cambios al transcurrir el tiempo de desarrollo afectan en mayor medida al progreso.

f) Tiempo desarrollo: Basado en el estudio de Ahimbisibwe, Cavana&Daellenbach (2015) señala que la calendarización del proyecto se vuelve un punto importante al momento de querer reducir el tiempo de desarrollo del proyecto.

g) Tamaño proyecto: Pressman (2010) indica que el tamaño del proyecto es otro factor importante que puede afectar la precisión y la eficacia de las estimaciones.

h) Experiencia desarrolladores - Problemática similar - Requerimientos similares: Germain & Robillard (2005) hacen notar que la experiencia de los equipos en el uso de los lenguajes de programación, problemática y requerimientos similares son de alta relevancia, ya que el desconocimiento puede causar el retraso del desarrollo de un sistema. Dentro de la experiencia de los equipos, se considera también la experiencia del jefe de proyecto.

i) Riesgos del proyecto: Pressman (2010) indica que los riesgos del proyecto amenazan el plan del proyecto, es decir, si los riesgos del proyecto se vuelven reales, es probable que el calendario del proyecto se deslice y que los costos aumenten.

j) Complejidad de desarrollo: Basado en Ozturk (2013) un estudio relacionado a la selección con la lógica difusa, la complejidad del desarrollo se seleccionó como importante al momento de escoger un tipo de metodología. También están dentro de los criterios que usaron para generar una tabla para comparar las diversas metodologías.

k) Documentación de proyecto: Pressman (2010) indica que comparan la documentación generada por los métodos tradicionales con los métodos ágiles, diciéndonos que métodos tradicionales se enfocan en documentación exacta sin errores, que un software funcional, por otro lado los métodos ágiles se enfocan en entregar un producto funcional dejando en segundo plano la documentación exhaustiva.

l) Tamaño del equipo desarrollo: Basado en el estudio de Ahimbisibwe, Cavana & Daellenbach (2015) dentro de sus parámetros se toma en cuenta el tamaño de los equipos, ya que como conclusión en la revisión de documentos que realizaron establecieron de acuerdo a la teoría que en el desarrollo tradicional se usan grandes equipos, así como en el desarrollo ágil los equipos que trabajan son más pequeños, convirtiendo este criterio en un punto importante a la hora de seleccionar un tipo de metodología.

De la información presentada anteriormente se genera una tabla de resumen de los criterios seleccionados junto a su autor.

Criterio	Autor (año)
Presupuesto apropiado	Pressman (2010)
Requerimientos específicos	Pressman (2010)
Variabilidad de requerimientos	Pressman (2010)
Participación del cliente	Ahimbisibwe, Cavana&Daellenbach (2015)
Adaptabilidad a cambios	Pressman (2010)
Tiempo desarrollo	Ahimbisibwe, Cavana&Daellenbach (2015)
Tamaño proyecto:	Pressman (2010)
Experiencia desarrolladores	Germain & Robillard (2005)
Problemática similar	Germain & Robillard (2005)
Requerimientos similares	Germain & Robillard (2005)
Riesgos del proyecto	Pressman (2010)
Complejidad de desarrollo	Ozturk (2013)
Documentación de proyecto	Pressman (2010)
Tamaño del equipo desarrollo	Ahimbisibwe, Cavana & Daellenbach (2015)

TABLA N° 5: Resumen criterios de selección

En base a la teoría se generó un instrumento para medir la importancia que dan las empresas dedicadas al desarrollo de software a los diferentes criterios seleccionados (Anexo Encuesta).

4 METODOLOGÍA DELA INVESTIGACIÓN

4.1 Recolección de datos

Para la obtención de datos empresariales se analizó una base de datos perteneciente al directorio de empresas y ejecutivos de Chile del año 2015, en la que se encontraban más de 20.000 ejecutivos de diferentes áreas. Luego del análisis se encontraron alrededor de 170 ejecutivos de diferentes empresas Chilenas pertenecientes a áreas de informática.

En nuestro estudio piloto se realizó un análisis de la validez de contenido de la encuesta, de la cual participaron cuatro expertos en el área de metodologías y desarrollo de software.

Para recolectar información necesaria, se examinan diversas alternativas de software y herramientas online para crear una encuesta que pueda ser usada en la recolección de datos, tales como; Survio, Typeform, ZohoSurvey, Lime Survey y formularios de Google drive. Se opta por Google Drive ya que ofrece las mismas características de los otros software de encuestas pero de forma gratuita y sin restricciones.

Una vez enviadas las encuestas por correo a las empresas Chilenas que cuentan con área o departamento de informática se determinó una espera de una semana para obtener las respuestas. Pasado el plazo de espera se realiza recordatorios por correos y llamados telefónicos. Además se realizó un estudio de campo en las empresas más cercanas de la zona de Concepción dentro de las cuales se visitó empresas de desarrollo de software, de áreas de salud, educación y comercio automotriz. Finalmente se obtienen respuesta de 35 empresas de las cuales se obtuvieron 54 proyectos diferentes.

4.2 Sujetos de investigación

Dentro de los 170 ejecutivos existen 50 cargos diferentes en las empresas que presentan áreas de desarrollo o departamentos de informática. Entre ellos se encuentran:

- Gerente de Desarrollo
- Gerente Área Quality
- Gerente de Tecnología
- Gerente Corporativo de Procesos y Tecnologías de la Información
- Jefe de Computación
- Jefe de Informática

- Encargado de Computación
- Gerente de Informática
- Subgerente de Informática
- Gerente de Tecnologías de Información
- Gerente de Innovación, Tecnología y Operaciones
- Gerente de Operaciones y Tecnología
- Analista de Sistemas
- Director de Informática
- Ingeniería / Sistemas
- Director de Desarrollo
- Jefe de Sistemas y Comunicaciones
- Gerente de Informática y Administración
- Gerente de Sistemas
- Gerente Informática
- Jefe de Departamento de Informática
- Gerente de Contenidos y Programación
- Jefe Sub-departamento de Informática
- Jefe Sección Informática y Telecomunicaciones
- Jefe División Administración Interna e Informática (S)
- Subgerente de Tecnologías de la Información y las Comunicaciones
- Jefe de Sistemas
- Gerente de Ingeniería e Informática
- Encargado de Informática
- Subgerente de Informática
- Jefe de Ventas - Encargado de Informática
- Encargada de Informática
- Gerente de Tecnología y Operación
- Director de Operaciones y Sistemas
- Gerente Negocios Tecnológicos
- Gerente de Tecnología y Sistemas
- Gerente Tecnologías de la Información
- Gerente de Planificación, Ingeniería y Sistemas
- Encargado de Informática / Administrador
- Jefe de Informática y Soporte Técnico
- Jefe Departamento Informática - Personal
- Jefe de Personal e Informática
- Gerente Área de Informática
- Gerente División Servicios
- Gerente División Aplicaciones
- Gerente Servicios Data center
- Gerente División Sistemas de Gestión
- Gerente de Servicios Soporte de Software
- Gerente de Servicios de Soporte de Infraestructura
- Gerente de Servicios de Cloud Computing.

5 RESULTADOS DE LA ENCUESTA

5.1 Resultados del estudio empírico

Como resultado de la recopilación de información a través de las encuestas realizadas de forma online y presencialmente en los alrededores de la ciudad de Concepción, se logró obtener 35 respuestas de diversas empresas, tales como, universidades e instituciones gubernamentales, en las cuales se realiza o ha realizado el desarrollo de algún sistema, obtuvimos información relacionada al tipo de metodología utilizada y la percepción que las personas encuestas tuvieron respecto de ella durante el desarrollo de dicho sistema, además debemos señalar que aunque sean las respuestas de 35 empresas se analizaron 54 proyectos diferentes.

5.1.1 Resultados demográficos.

- **Género**

Dentro de la demografía obtenida se refleja que el área de desarrollo está dominada ampliamente por el género masculino equivalente a 28 personas que equivalen al 80% de la muestra obtenida en esta encuesta y las mujeres solo presentan un 20%.

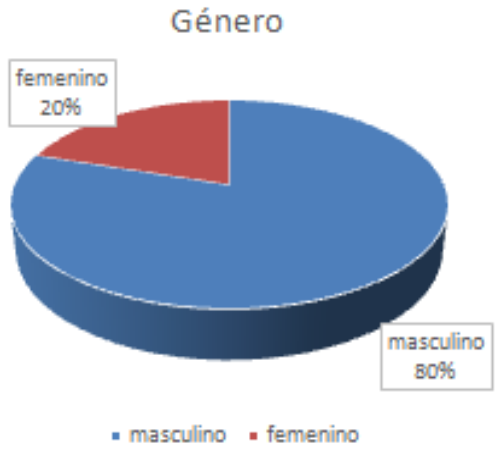


FIGURA N° 2: Género Encuestados

➢ **Edad**

El promedio de edad de los entrevistados lo dividimos en 4 grupos, menor de 30 años, entre 30 y 40 años, entre 40 y 50 años y mayores de 50 años, para comprobar el cambio en la elección de metodologías según el rango de edad, y si estas diferencias afectan en la elección de metodologías.

En estos resultados obtuvimos que los porcentajes de encuestados reflejan que el mayor grupo contempla 37% en la edad ente 40 y 50 años, le siguen los grupos de entre 30 y 40 años con un 26%, los menores de 30 años con un 23% respectivamente, y con solo un 14% los mayores de 50 años.

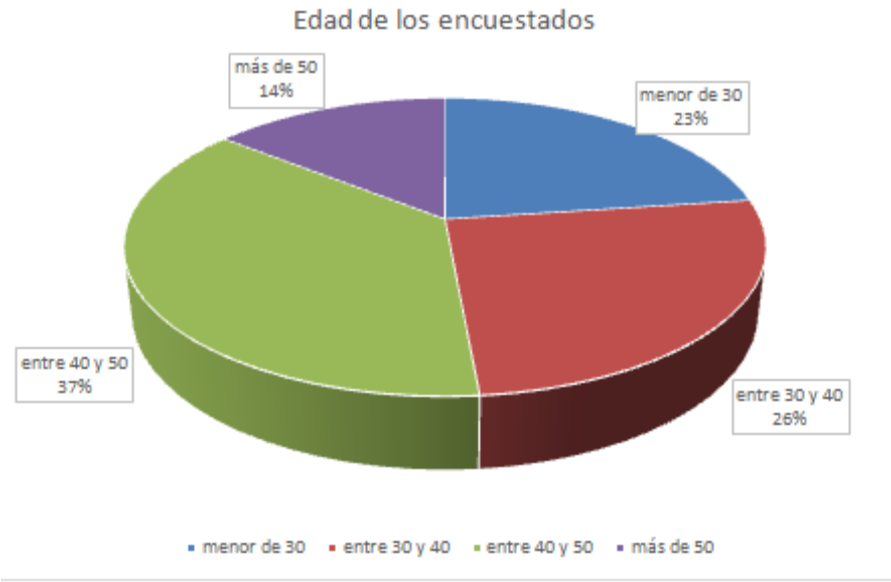


FIGURA N° 3: Edad Encuestados

➢ **Nivel educacional**

Para el nivel educacional de los entrevistados obtuvimos que el 60% tiene un nivel universitario representando la mayoría en esta área, seguido de los técnicos profesionales con un 20%, los magister con 17% y solo una persona con doctorado representando un 3% en este área.

De este resultado podemos inferir mas no asegurar debido al tamaño de la muestra que las personas que poseen mayor nivel educacional están dedicados a otras áreas de la informática fuera del desarrollo.

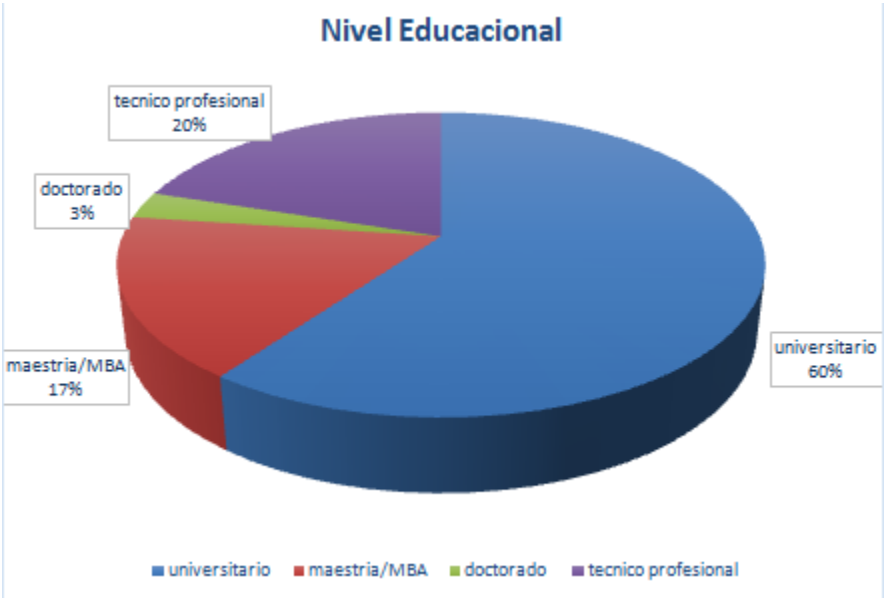


FIGURA N° 4: Nivel educacional Encuestados

➢ **Cargos**

Entre las distintas instituciones que entrevistamos tanto como universidades, institutos, instituciones de salud, instituciones gubernamentales y empresas de desarrollo, también encontramos distintos cargos a los cuales pertenecían los entrevistados tales como, directores de informática, jefes de proyectos, desarrolladores, analistas, distintos tipos de gerencia, gerentes de tecnologías, gerentes de innovación e informática, y encargados del área.

Para ordenar estos datos de manera legible establecimos 4 grupos distintos que agruparían los distintos cargos encontrados, donde tenemos gerente de informática que contempla las distintas gerencias y subgerencias encontradas, encargados de área de informática que incluye directores y encargados del área que están sobre los equipos de desarrollo, tercero tenemos a los jefes de equipo de desarrollo que influyen en la toma de decisión y tienen una comunicación más directa con los encargados, y el grupo de los desarrolladores que pertenecen a un equipo.

En los resultados obtenidos tenemos una amplia variedad de respuestas de los distintos grupos señalados, donde tenemos un 20% de respuestas de gerentes y 37 % de encargados del área, por el lado del desarrollo tenemos un 34% de respuestas de desarrolladores, 9% de equipos de desarrollo que nos da un total de 43% de respuestas de quienes trabajan y llevan a cabo el uso de la metodología.

FIGURA N° 5: Cargo Encuestados

➢ **Edad y años en la empresa**

Para el siguiente grafico que compara la edad de los encuestados con los años que han permanecido trabajando en la misma empresa o institución, notamos que el rango de entre 40 y 50 años de edad son los que tienen mayor presencia en todos los rangos de permanencia en el trabajo donde podemos inferir que hay posibilidad de que las personas se establezcan en una empresa y/o continuamente buscan otros empleos, también que los menores de 30 solo aparecen en la comparación de 5 años o menos.

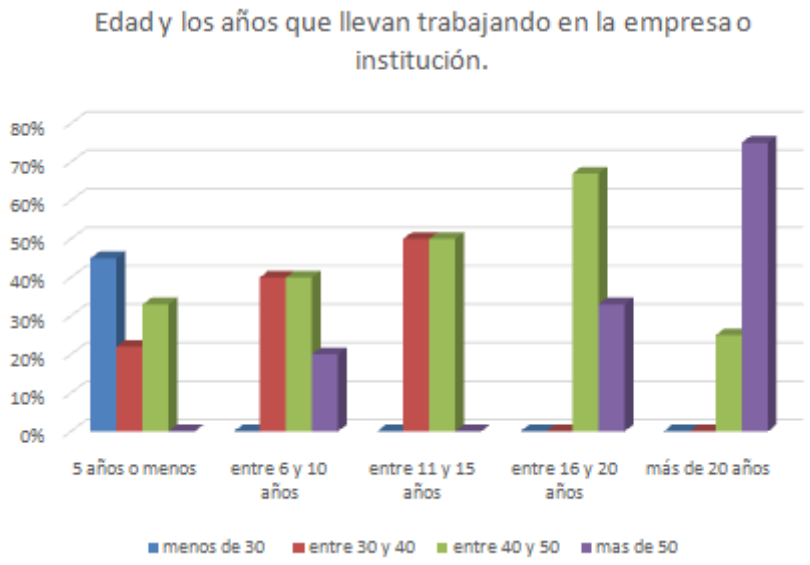


FIGURA N° 6: Edad y años de trabajo Encuestados

5.1.2 Información de las metodologías de desarrollo

- **Tamaño de los proyectos.**

En el siguiente grafico se muestra que el tamaño de los proyectos no causa gran diferencia en los tipos de metodologías, a excepción de agiles donde destaca por amplia diferencia el uso en proyectos denominados medianos.

La medida utilizada por la mayoría de los encuestados para definir el tamaño de los proyectos fue las funcionalidades que debían ser implementadas

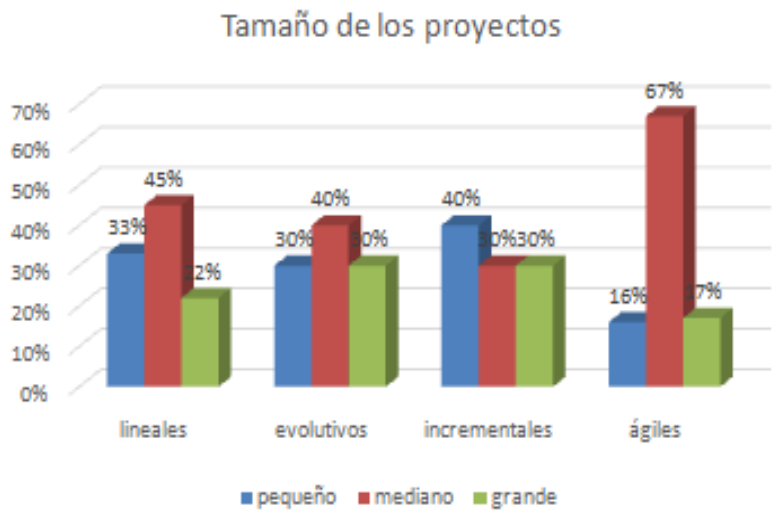


FIGURA N° 7: Tamaño de los proyectos

- **Tiempo del Proyecto y metodología usada.**

En el siguiente grafico tenemos como resultado que los proyectos que se usó metodología lineal fueron en su mayoría proyectos de plazo medio entre 6 meses y 1 año, mientras que en las evolutivas tenemos que los proyectos más pequeños de menos de 6 meses son los que destacaron, aunque este resultado era más esperado en las metodologías agiles donde destaco sobresaliendo los proyectos de plazo medio, y los incrementales tienen un resultado similar a las lineales, para los casos que utilizaron practicas agiles y no utilizaron metodologías se realizaron proyectos de corto plazo con un 71% y 100% respectivamente.

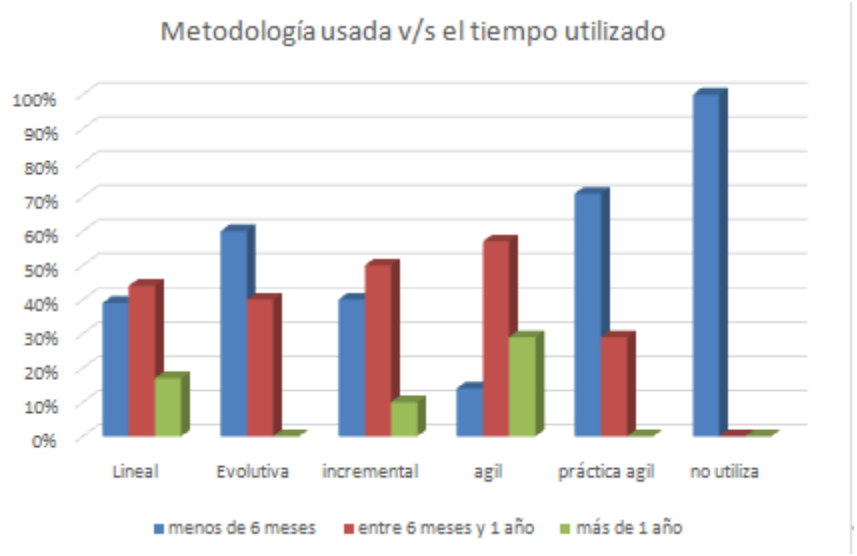


FIGURA N° 8: Tiempo v/s Metodología

- **Tamaño de los equipos de trabajo.**

También recopilamos información del tamaño de los equipos de desarrollo donde para el total de metodologías, sin excluir el tipo al cual pertenecían los resultados muestran que la mayoría de los equipos eran de 4 o menos personas.

FIGURA N° 9: Tamaño de los equipos de trabajo

➢ **Información de las metodologías usadas.**

Dentro del estudio dividimos las metodologías en 4 grupos o categorías distintas que abarcan las distintas metodologías, y también añadimos el uso de las prácticas ágiles, esto se refiere al uso de prácticas que presentan algunas metodologías ágiles, pero no significa el uso o implementación completa de una metodología en sí.

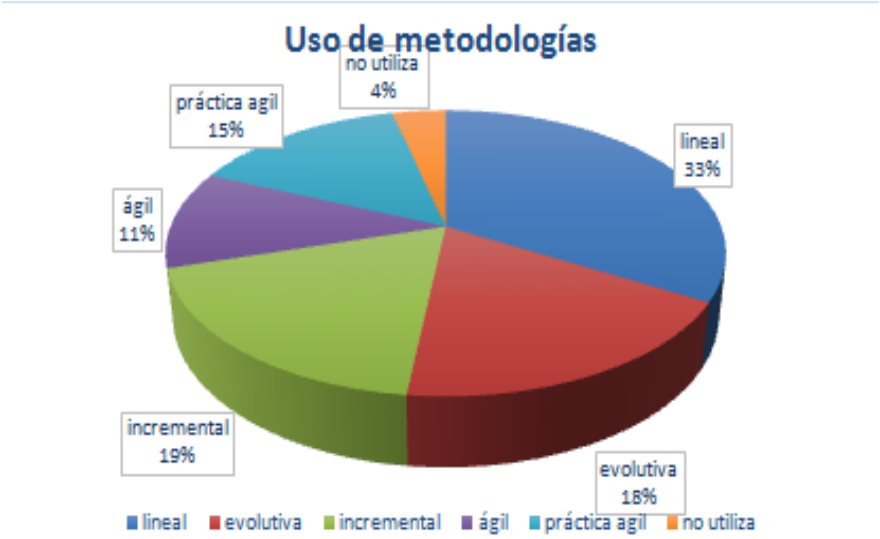


FIGURA N° 10: Uso de Metodologías

Como se puede apreciar en el grafico el mayor uso de las metodologías se centró en el grupo de las lineales e incrementales con un 34% y 19% respectivamente.

Dentro de lo esperado esto se considera una sorpresa, porque antes del estudio se esperaba que las metodologías ágiles junto a las lineales fueran las más destacadas, pero el resultado nos ha demostrado que las agiles están por detrás, y tiene un mayor valor el uso de las practicas agiles sobre una metodología ágil completa.

Aunque este resultado podría suponer que el uso menor de las metodologías agiles se debe al poco conocimiento y a lo que se conoce como resistencia al cambio, que evita que las personas que trabajan con una metodología más tradicional y obtienen buenos resultados eviten inclinarse a utilizar otro tipo de metodología.

Otra forma de contrastar esta información fue separar la elección de los grupos de metodologías en los distintos grupos de edad de los entrevistados, donde podemos ver otro enfoque y también tenemos un cambio en la elección cada grupo.

Primero en el grupo de los menores de 30 años notamos una tendencia al grupo de metodologías lineales seguido de prácticas agiles con un 37%, dejando inesperadamente a un cuarto lugar las metodologías ágiles, esto contradice nuestras expectativas iníciales que presumíamos antes del estudio, pero refleja el hecho de que las metodologías más tradicionales son las más utilizadas, aunque puede deberse a la fácil aplicación de esta metodología para las personas que tienen menor experiencia en el área, porque como logramos ver en el siguiente grupo los porcentajes se equiparan.

En el grupo de edades entre 30 años y 40 años, cambia la diferencia respecto al grupo anterior, donde las metodologías incrementales y agiles se emparejan en el uso comparado a las lineales con un 22%.

En el rango de edades entre 40 años y 50 años vuelve a estar con mayor uso las metodologías del tipo lineal donde presenta un 38% de uso y cabe también destacar que tenemos un 0% de metodología ágil, aunque debido al tamaño de la muestra conseguida no podemos aseverar una conclusión es punto que merece ser mencionado.

Finalmente en el grupo de mayores de 50 años existe otro cambio no esperado donde lo más esperado era que las metodologías tradicionales lideran en uso, pero se ve reflejado el uso es más distribuido destacando el grupo incremental, las conclusiones que se pueden obtener de este resultado es que las personas de este grupo, al tener más experiencia saben que metodología usar en cada caso y se atreven a experimentar el uso de las nuevas metodologías.

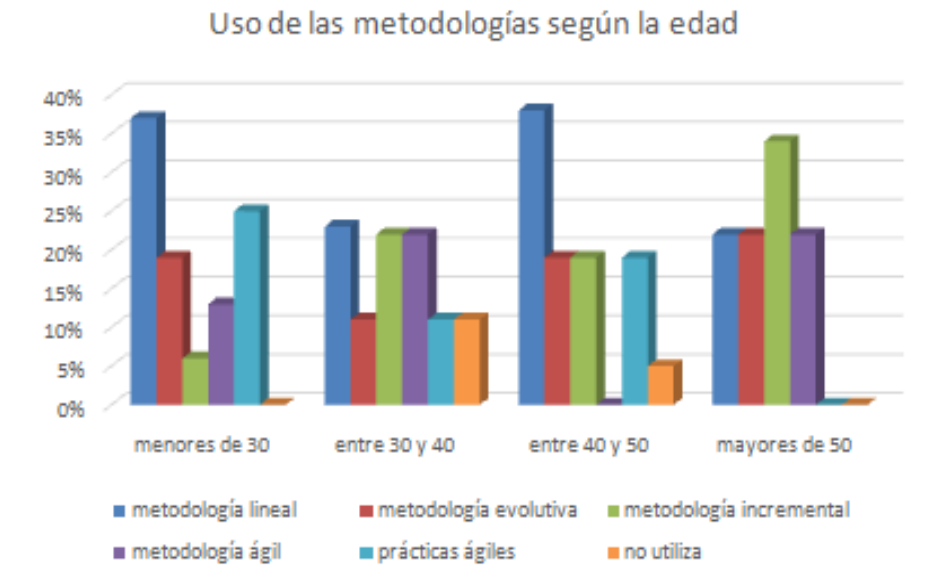


FIGURA N° 11: Uso de Metodologías según Edad

➢ **Metodologías usadas según años en la empresa y/o institución.**

Como podemos notar en este grafico que refleja la metodología que utilizan los encuestados según la cantidad de años que llevan en una empresa se puede ver que a la menor cantidad de años se nota una principal selección de las metodologías lineales dejando a un tercer lugar las metodologías agiles, y al ver los encuestados con más de 20 años en la empresa tienen un uso de las metodologías similares para cada grupo.

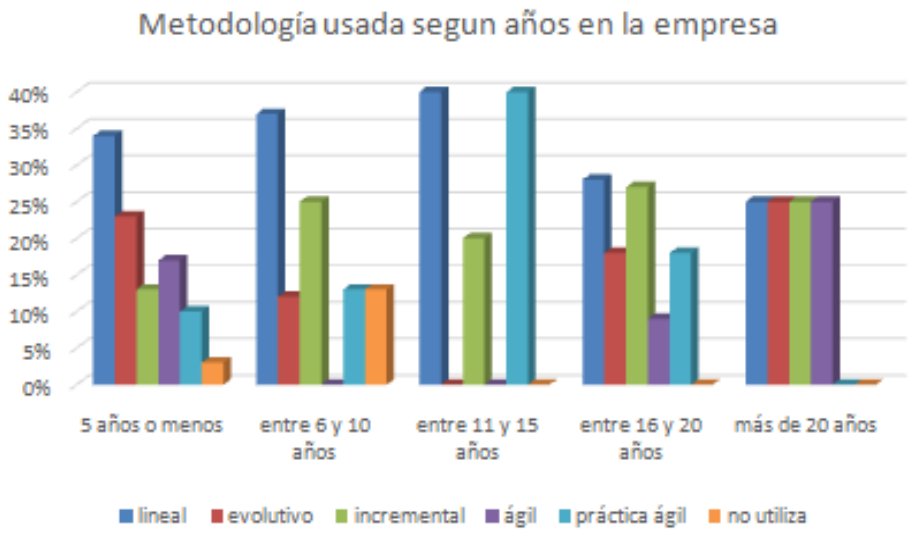


FIGURA N° 12: Metodologías usadas según años en la empresa

Metodologías usadas

Dentro de la encuesta solicitamos una especificación de la metodología usada dentro de cada grupo que haya señalado para saber cuál metodología es la más utilizada dentro de la selección inicial.

Los datos que hemos obtenido los hemos traspasado a gráficos divididos según los distintos grupos de metodologías señalados anteriormente, para reflejar de mejor manera el uso de cada metodología.

- **Metodologías lineales usadas.**

En el grupo de las metodologías lineales tenemos como resultado un 45% de uso de cascada en su versión de cascada con iteración con mucha diferencia sobre las otras, aunque se debe destacar que existe un 32% que no especificaron una metodología.

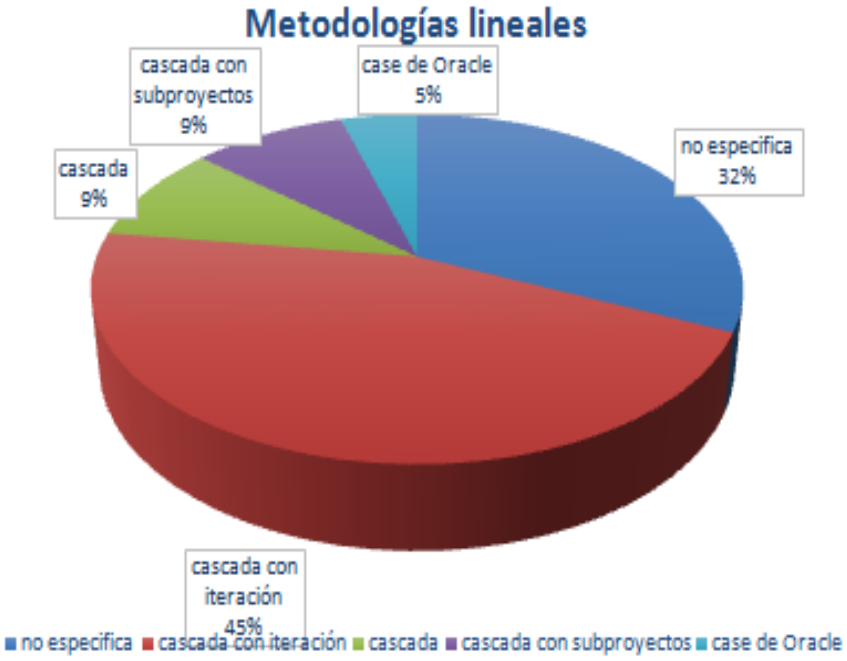


FIGURA N° 13: Metodología Lineal

- **Metodologías evolutivas usadas.**

En las metodologías evolutivas señaladas destaca el prototipo evolutivo con 54%, por otro lado metodologías como cascada con fases solapadas no fueron mencionadas.

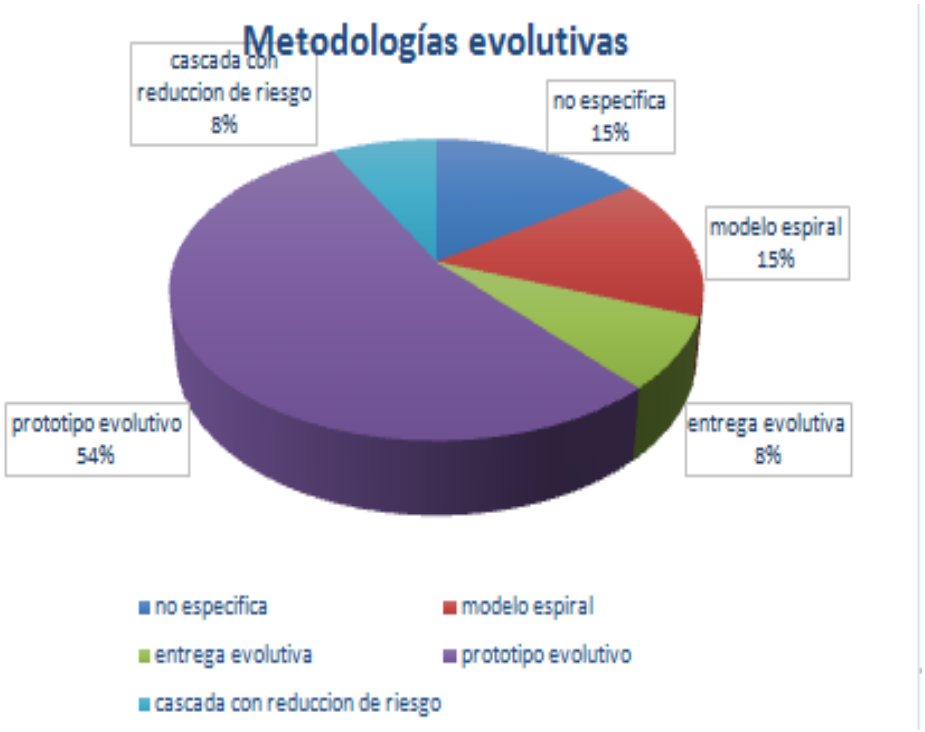


FIGURA N° 14: Metodología Evolutiva

➢ **Metodologías incrementales usadas**

En las metodologías incrementales, que como vimos anteriormente son las más usadas por las personas con más de 50 años predomina la selección de la entrega por etapas en el desarrollo de sistemas.

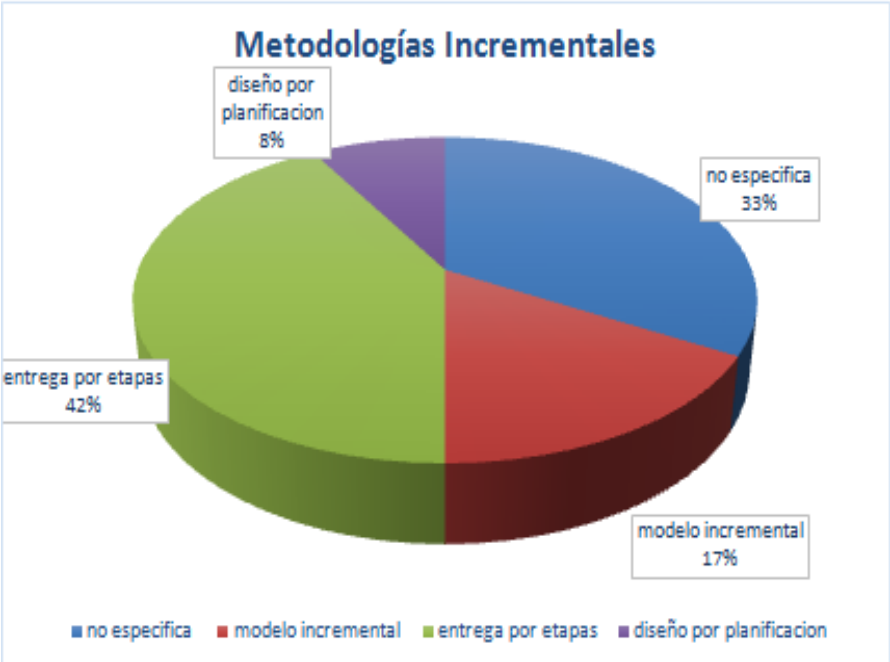


FIGURA N° 15: Metodología Incremental

- **Metodologías ágiles usadas**

En las metodologías ágiles predomina el uso de SCRUM con el 50% de selección seguido de XP y Kanban, dentro de la variada cantidad de metodologías denominadas ágiles estas son las predominantes.

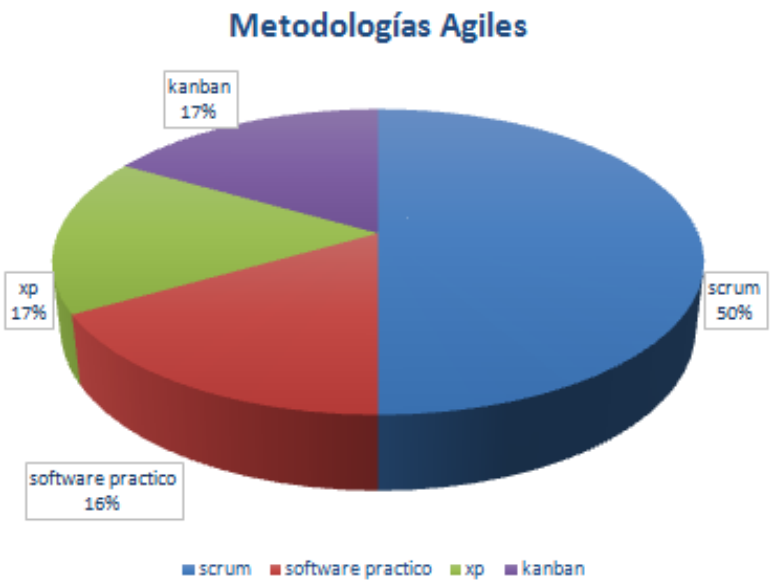


FIGURA N° 16: Metodología Ágil

- **Prácticas ágiles usadas**

Para el uso las practicas agiles obtuvimos resultados que mezclaban estas prácticas con una metodología y otros que solo utilizaban estas en el desarrollo de sus proyectos sin agregarlas o combinarlas con otra metodología.

Para el total de respuestas de prácticas agiles el 50% de ellos utilizaba únicamente las prácticas agiles sin utilizar ningún tipo de metodología.

Luego si examinamos el total de los grupos de metodologías vemos que el 22,2% de los que utilizaron metodologías lineales utilizaron en combinación practicas agiles, para el grupo de las evolutivas e incrementales tenemos un 20% y 10% respectivamente que usaron prácticas ágiles durante el uso de estas metodologías y finalmente las personas que utilizaban metodologías ágiles no señalaron utilizar alguna practica ágil independiente a la que establecía la metodología que usaban.

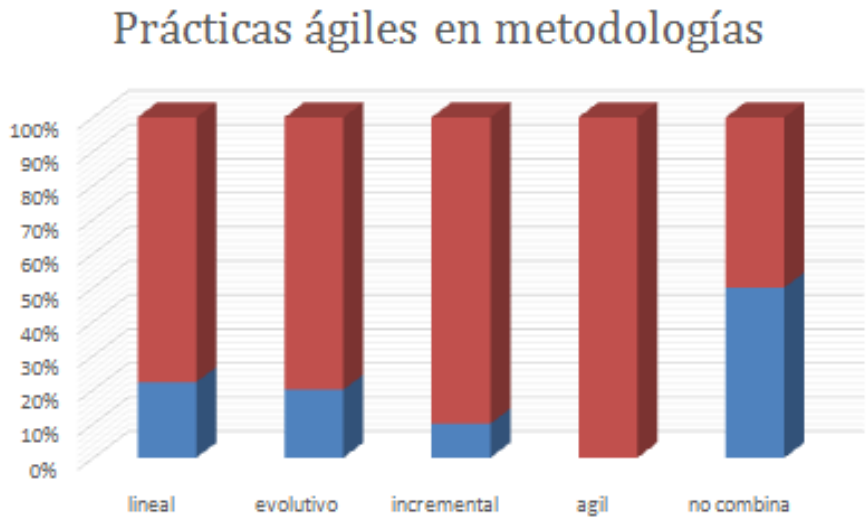


FIGURA N° 17: Prácticas ágiles en Metodologías

5.1.3 Resultados de los criterio

Los siguientes resultados muestran la importancia de los distintos criterios que fueron consultados en las encuestas, donde se puede notar que todos presentan una alta importancia, pero con algunas excepciones en ciertos puntos, los cuales serán relevantes a la hora de compararlos con la teoría, y luego establecer una forma de crear una recomendación bajo distintas reglas.

También se debe destacar que estos criterios fueron seleccionados luego de examinar la literatura relacionada a las metodologías, donde se sabe que al menos un criterio o más tienen mayor relevancia en la teoría, como lo son documentación en las metodologías lineales, o el aporte del usuario en las metodologías ágiles.

- **Metodologías lineales**

Para las metodologías lineales tenemos que los criterios que más destacaron con un 94% de importancia fueron el tener que trabajar con una problemática similar a una que hayan usado antes, así como la claridad de los requerimientos que está dentro de lo esperado según la literatura de las metodologías, seguido de la complejidad que es un punto importante en todos los resultados obtenidos de los distintos grupos, y el que hayan trabajado requerimientos similares anteriormente cada uno con 89%. Por otro lado según la teoría la documentación es uno de los puntos más importantes en las metodologías lineales pero en los resultados obtenidos su valor de importancia es el más bajo con 61% aunque sigue siendo superior a los resultados de otros grupos y es el más alto en los que consideraron la documentación poco importante con 28%.

También podemos considerar que el criterio del tamaño del proyecto está entre los más bajos de nivel de importancia, pero esto a diferencia de los otros criterios da a notar que no influye tanto el tamaño que pueda tener y funciona bien en proyectos grandes.

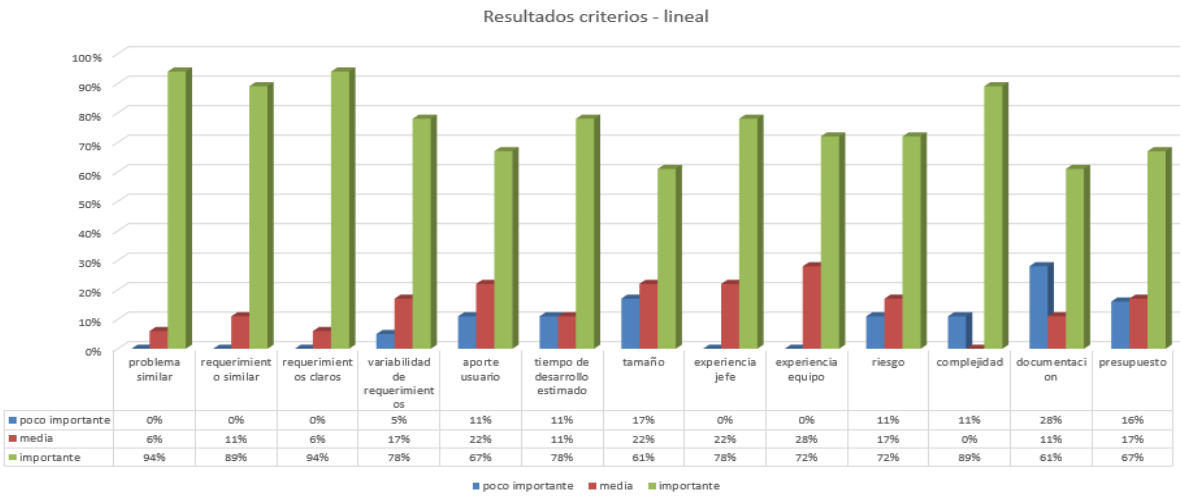
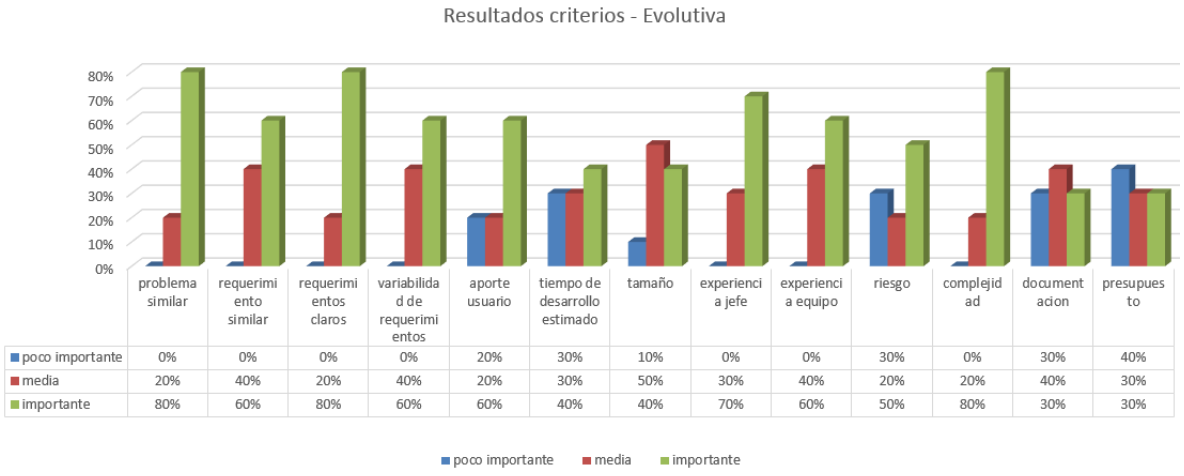


FIGURA N° 18: Criterios Metodologías Lineales

➢ **Metodologías evolutivas**

En las metodologías evolutivas obtuvimos un resultado similar al de las lineales para los criterios de complejidad, problemática similar y requerimientos claros, con 80% de respuestas que los consideraron importante, la experiencia del jefe de proyecto adquiere un valor de 70% siendo el siguiente valor más alto respecto a los mencionados antes, esto puede dar a entender que se necesita algo más de conocimientos por parte del líder de grupo al momento de escoger alguna metodología de este grupo, también vemos que la documentación y el presupuesto presentan los porcentajes de importancia más bajos, con una mayoría que considero el presupuesto poco importante con un 40%.



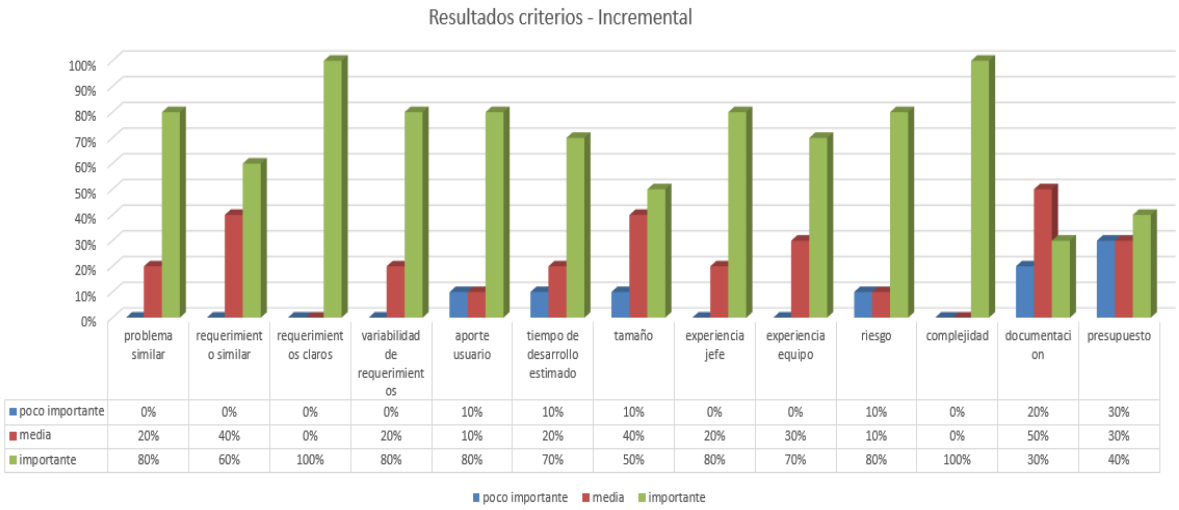
	problema similar	requerimiento similar	requerimientos claros	variabilidad de requerimientos	aporte usuario	tiempo de desarrollo estimado	tamaño	experiencia jefe	experiencia equipo	riesgo	complejidad	documentacion	presupuesto
poco importante	0%	0%	0%	0%	20%	30%	10%	0%	0%	30%	0%	30%	40%
media	20%	40%	20%	40%	20%	30%	50%	30%	40%	20%	20%	40%	30%
importante	80%	60%	80%	60%	60%	40%	40%	70%	60%	50%	80%	30%	30%

FIGURA N° 19: Criterio Metodologías Evolutivas

➢ **Metodologías incrementales**

La complejidad y la claridad de los requerimientos tienen un 100% de importancia y destaca como en todos los grupos de metodologías, también la documentación presenta un valor bajo de importancia con un 30%.

La experiencia del jefe de proyecto junto a la experiencia del equipo toma valores a tomar en cuenta ya que nos permite inferir que para el uso de estas metodologías se necesita un conocimiento en el trabajo con ellas.

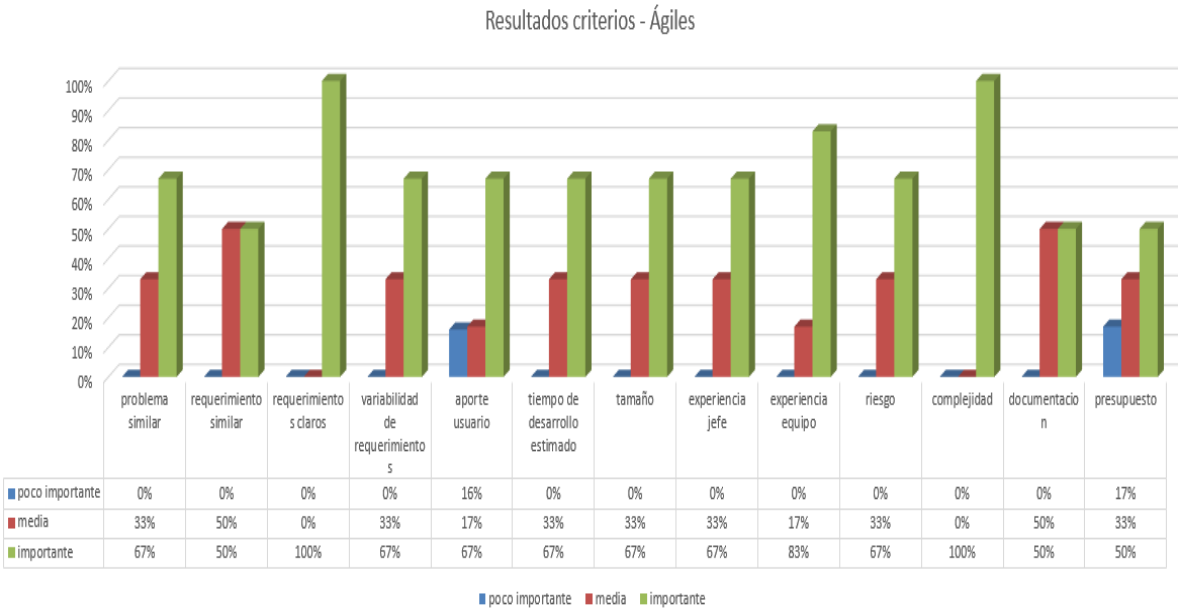


	problema similar	requerimiento similar	requerimientos claros	variabilidad de requerimientos	aporte usuario	tiempo de desarrollo estimado	tamaño	experiencia jefe	experiencia equipo	riesgo	complejidad	documentacion	presupuesto
poco importante	0%	0%	0%	0%	10%	10%	10%	0%	0%	10%	0%	20%	30%
media	20%	40%	0%	20%	10%	20%	40%	20%	30%	10%	0%	50%	30%
importante	80%	60%	100%	80%	80%	70%	50%	80%	70%	80%	100%	30%	40%

FIGURA N° 20: Criterios Metodologías Incrementales

➢ **Metodologías ágiles**

La complejidad nuevamente está entre los mayores valores, junto a la claridad de los requerimientos, se tiene que el aporte que puede dar el usuario tiene un menor valor 67% esto es un valor bajo, considerando que uno de los principales puntos de las metodologías ágiles destaca la participación del usuario por lo tanto se esperaba que este criterio fuera un valor destacable en este grupo de metodologías, también podemos señalar que la experiencia del grupo o equipo de trabajo tiene un valor de 83% lo que nos puede dar a entender que el manejo de conocimientos entre el equipo debe ser similar entre sí para que tenga un buen desarrollo el proyecto que deseen abordar.



	problema similar	requerimiento similar	requerimientos claros	variabilidad de requerimientos	aporte usuario	tiempo de desarrollo estimado	tamaño	experiencia jefe	experiencia equipo	riesgo	complejidad	documentacion	presupuesto
poco importante	0%	0%	0%	0%	16%	0%	0%	0%	0%	0%	0%	0%	17%
media	33%	50%	0%	33%	17%	33%	33%	33%	17%	33%	0%	50%	33%
importante	67%	50%	100%	67%	67%	67%	67%	67%	83%	67%	100%	50%	50%

FIGURA N° 21: Criterios Metodologías Ágiles

- **Criterios más importantes**

Finalmente se tomó el porcentaje de importancia de los criterios y los comparamos entre los 4 grupos de metodologías para poder relacionar y comparar que criterios tenían mayor porcentaje en cada grupo de metodologías, donde podemos ver que las metodologías lineales destacan en "problema similar", "requerimiento similar", "tiempo de desarrollo estimado", "documentación" y "presupuesto".

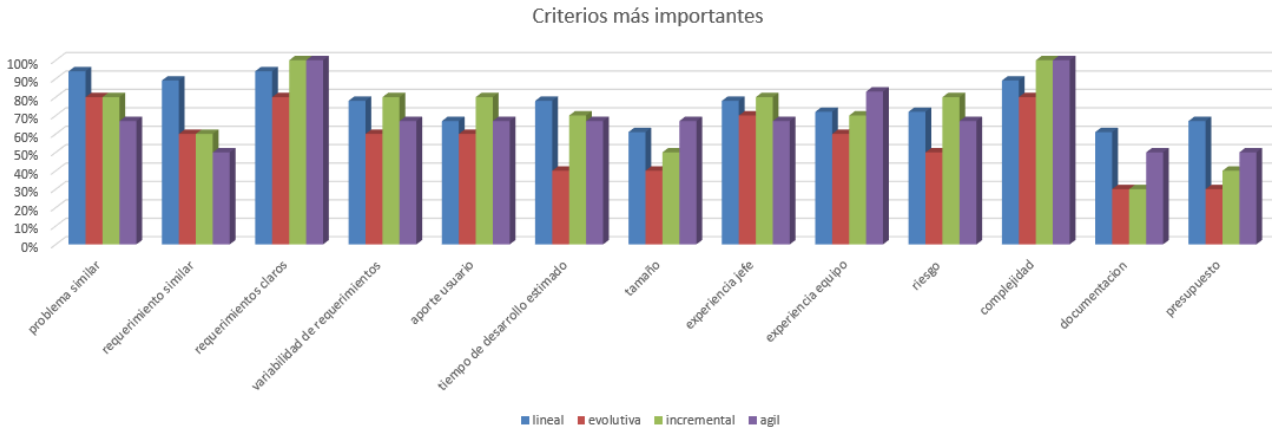


FIGURA N° 22: Criterios más Importantes

5.2 Reglas

A continuación se realiza una comparación entre los criterios presentados en la literatura (sección anterior 3.2) con los criterios obtenidos como importantes en el estudio de campo, asociadas a cada pregunta de la encuesta (en las siguientes tablas se representa con una X las respuestas asociadas a cada grupo de metodología según la pregunta).

1. ¿Desea resolver un problema ya conocido o enfrentado anteriormente?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Conocido en su totalidad	X	X			X	X	X	X
Conocido parcialmente	X				X		X	X
No es conocido			X	X	X			

2. ¿Ha implementado requerimientos similares en un proyecto anterior?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si	X	X			X	X		
No	X		X	X			X	X

3. ¿Los requerimientos obtenidos fueron detallados en su totalidad?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	X	X	X	X	
No		X				X	X	X

4. ¿Es posible que los requerimientos cambien en el desarrollo del proyecto?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si		X	X	X		X	X	X
No	X		X		X	X	X	

5. ¿Considerará la participación del usuario durante el desarrollo del proyecto?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si, al inicio	X				X			
Si, al final		X	X					
Si, Durante el proyecto			X	X		X	X	X
No	X							

6. ¿Tiene un margen de tiempo establecido para el desarrollo del proyecto?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	X	X	X	X	
No	X	X	X	X			X	X

7. ¿Cómo considera el tamaño de su proyecto en cuanto a funcionalidades del sistema a realizar?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Pequeño	X	X	X	X	X			X
Mediano	X	X	X	X	X	X	X	
Grande	X	X	X	X	X	X	X	

8. ¿Qué tan experimentado es el jefe de proyecto en cuanto a metodologías de desarrollo de software?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Experto	X	X	X	X	X	X	X	X
Medianamente experimentado	X	X	X	X	X	X	X	X
No tiene experiencia					X			

9. ¿Qué tan experimentado es el equipo de desarrollo en cuanto a metodologías de desarrollo de software?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Experto	X	X	X	X	X	X	X	X
Medianamente experimentado	X	X	X	X	X	X	X	X
No tiene experiencia	X				X			

10. ¿Se considerará el riesgo asociado en el proyecto a realizar (tiempo, costo, etc.)?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X			X		
No		X		X	X		X	X

11. ¿Qué tan complejo considera el proyecto que realizara?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Muy complejo	X		X	X	X		X	
Medianamente complejo	X	X	X	X	X	X	X	X
Nada complejo	X	X	X	X	X	X	X	X

12. ¿Considera importante respaldar el proceso de desarrollo del proyecto que realizara mediante documentación?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	X	X	X	X	
No			X	X		X	X	X

13. ¿Cómo considera el presupuesto disponible para realizar el proyecto?

	Práctica				Teoría			
	Lineal	Evolutiva	Incremental	Ágil	Lineal	Evolutiva	Incremental	Ágil
Estrecho							X	X
Holgado (Apropiado)	X				X	X	X	X
No tiene presupuesto	X	X	X	X	X	X	X	X

Resultados finales de comparación para reglas del sistema.

En la presente sección se resume la aplicabilidad de la metodología en función de los resultados arrojados por la revisión de la literatura y la información provista por las empresas.

En algunos casos se priorizó lo que indicaba la teoría, ya que algunas respuestas contradecían a lo establecido por los autores.

1. ¿Desea resolver un problema ya conocido o enfrentado anteriormente?

	Lineal	Evolutiva	Incremental	Ágil
Conocido en su totalidad	X	X	X	X
Conocido parcialmente	X			X
No es conocido	X		X	

2. ¿Ha implementado requerimientos similares en un proyecto anterior?

	Lineal	Evolutiva	Incremental	Ágil
Si	X	X		
No			X	X

3. ¿Los requerimientos obtenidos fueron detallados en su totalidad?

	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	
No		X	X	X

4. ¿Es posible que los requerimientos cambien en el desarrollo del proyecto?

	Lineal	Evolutiva	Incremental	Ágil
Si		X	X	X
No	X	X	X	

5. ¿Considerará la participación del usuario durante el desarrollo del proyecto?

	Lineal	Evolutiva	Incremental	Ágil
Si, al inicio	X			
Si, al final		X	X	
Si, Durante el proyecto			X	X
No	X			

6. ¿Tiene un margen de tiempo establecido para el desarrollo del proyecto?

	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	
No			X	X

7. ¿Cómo considera el tamaño de su proyecto en cuanto a funcionalidades del sistema a realizar?

	Lineal	Evolutiva	Incremental	Ágil
Pequeño	X			X
Mediano	X	X	X	
Grande	X	X	X	

8. ¿Qué tan experimentado es el jefe de proyecto en cuanto a metodologías de desarrollo de software?

	Lineal	Evolutiva	Incremental	Ágil
Experto	X	X	X	X
Medianamente experimentado	X	X	X	X
No tiene experiencia	X			

9. ¿Qué tan experimentado es el equipo de desarrollo en cuanto a metodologías de desarrollo de software?

	Lineal	Evolutiva	Incremental	Ágil
Experto	X	X	X	X
Medianamente experimentado	X	X	X	X
No tiene experiencia	X			

10. ¿Se considerará el riesgo asociado en el proyecto a realizar (tiempo, costo, etc.)?

	Lineal	Evolutiva	Incremental	Ágil
Si		X		
No	X		X	X

11. ¿Qué tan complejo considera el proyecto que realizara?

	Lineal	Evolutiva	Incremental	Ágil
Muy complejo	X		X	
Medianamente complejo	X	X	X	X
Nada complejo	X	X	X	X

12. ¿Considera de alta prioridad respaldar el proceso de desarrollo del proyecto que realizara mediante documentación?

	Lineal	Evolutiva	Incremental	Ágil
Si	X	X	X	
No			X	X

13. ¿Cómo considera el presupuesto disponible para realizar el proyecto?

	Lineal	Evolutiva	Incremental	Ágil
Estrecho			X	X
Holgado (Apropiado)	X	X	X	X
No tiene presupuesto	X	X	X	X

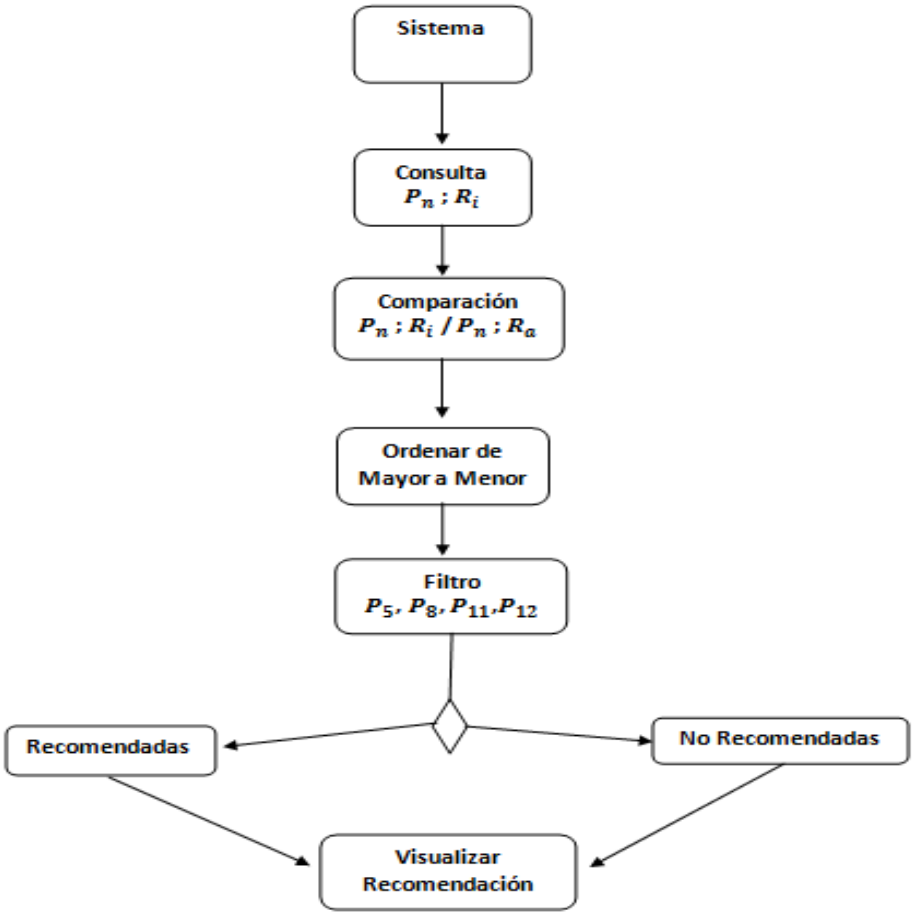


FIGURA N° 23: Representación Reglas

La Figura 23 es presentada con la finalidad de facilitar la interpretación de las reglas que tendrá el sistema de recomendación de metodologías. El proceso para llevar a cabo una recomendación de un grupo de metodologías es el siguiente:

$\boldsymbol{P_n}$ **:** Pregunta del cuestionario.

$\boldsymbol{R_i}$ **:** Respuesta seleccionada por el usuario en el cuestionario.

$\boldsymbol{R_a}$ **:** Respuesta almacenada del cuestionario.

$\boldsymbol{P_5}$ **:** ¿Considerará la participación del usuario durante el desarrollo del proyecto?

$\boldsymbol{P_8}$ **:** ¿Qué tan experimentado es el jefe de proyecto en cuanto a metodologías de desarrollo de software?

$\boldsymbol{P_{11}}$ **:** ¿Qué tan complejo considera el proyecto que realizara?

$\boldsymbol{P_{12}}$ **:** ¿Considera de alta prioridad respaldar el proceso de desarrollo del proyecto que realizara mediante documentación?

L: Metodología Lineal.

E: Metodología Evolutiva.

I: Metodología Incremental.

A: Metodología Ágil.

Sistema: El sistema realiza un análisis interno de la información correspondiente al cuestionario ingresado respondido por el usuario.

Consulta: El sistema por cada pregunta registrada consulta por su respuesta selecciona por el usuario.

Comparación: El sistema realiza una comparación de cada respuesta ingresada por el usuario con cada respuesta almacenada en el sistema, las cuales están asociadas a un grupo de metodología, además, la respuesta ingresada por el usuario incrementara un contador por cada grupo de metodologías (en total 4 contadores, los cuales en ningún caso excederán la cantidad total de preguntas, que en este caso son 13) dependiendo del caso según tenga coincidencia con las respuestas almacenadas.

Orden de Mayor a Menor: Una vez realiza la comparación de respuestas ingresadas con las almacenadas el sistema realiza el ordenamiento según el valor de las coincidencias encontradas.

Filtro: Una vez ordenadas las preferencias de selección estas deben pasar por un filtro el cual determinara si dicho grupo de metodologías podrá ser recomendado o no.

El filtro que se aplicará será el siguiente:

for(x=1; x<=4; x++){

-> Si Orden x ! = L y opción $\boldsymbol{P_5}$ = No (Considerar NO utilizar Metodología Lineal)

-> Si Orden x ! = L y opción $\boldsymbol{P_8}$ = No tiene experiencia. (Considerar utilizar Metodología Lineal)

-> Si Orden x ! = L ó I y opción $\boldsymbol{P_{11}}$ = Muy complejo. (Considera NO utilizar Metodología Evolutiva ó Ágil)

-> Si Orden x ! = E ó L y opción $\boldsymbol{P_{12}}$! = Sí. (Considera NO utilizar Metodología Lineal ó Evolutiva)

-> Si Orden x ! = A y opción $\boldsymbol{P_{12}}$! = No. (Considera NO utilizar Metodología Ágil)

}

Recomendadas: En este paso se encuentran las metodologías que han pasado satisfactoriamente el filtro aplicado.

No Recomendadas: En este paso se encuentran las metodologías que no han pasado el filtro aplicado.

Visualizar Recomendación: En este paso el usuario puede visualizar de forma grafica los valores obtenidos del orden de preferencia, además, de distinguir los grupos de metodologías recomendados y no recomendados junto a sus causas.

5.3 Conclusión resultados

Dentro de los resultados obtenidos de la muestra se tiene un pequeño y variado reflejo del estado del desarrollo informático, que permite compararlo con lo que es enseñado a través de la teoría. Luego de comparar el estudio de la teoría contrastándolo con los resultados obtenidos de la encuesta, a partir de la comparación de los criterios que fueron seleccionados del estudio teórico, se generaron las reglas de selección para el sistema a desarrollar, así como las preguntas y posibles respuestas para cada una de ellas. Estas reglas son un filtro que permitirán al sistema generar una recomendación, y por lo tanto son la base del sistema. Si se hubiese obtenido una muestra mayor de empresas se podría haber generado una recomendación más específica con una o varias metodologías de desarrollo. Por lo anterior se concluye que la recomendación se realizará de forma general a través de los 4 grupos de metodologías expuestos en la sección 3 y detallados en el anexo de METODOLOGÍAS DE DESARROLLO DE SOFTWARE.

6 ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

6.1 Alcances

El sistema proporcionará una recomendación de un grupo de metodología de desarrollo al usuario, con la cual se pretende optimizar el desarrollo de futuros proyectos, además de permitir distribuir de mejor manera los recursos presentes.

Algunas de las características del software son:
El software realizará al usuario diversas preguntas de las cuales se establecerá una recomendación asociada a lo que haya respondido y entregará retroalimentación al usuario dándole a conocer cuál grupo de metodología debería usar para lo que desea desarrollar.

El software permitirá al usuario revisar y verificar la información sobre las metodologías ingresadas en el sistema, viendo sus características principales, así como el plan de desarrollo que presentan y tipos de proyectos en los que se usa.

El sistema permitirá al usuario seleccionar un grupo de metodologías que se ha recomendado para su posterior calificación.

El software permitirá al usuario ingresar sugerencias sobre la recomendación o funcionalidad del sistema.

El software permitirá al administrador del sistema ingresar información al sistema una vez este se autentique a través de un login.

6.2 Objetivo del software

El objetivo del software es implementar una solución informática, que permita optimizar la toma de decisiones respecto a qué metodología de desarrollo de software implementar al momento de desarrollar un sistema, además de otorgar un eficiente manejo de información que permite al usuario conocer sobre las diferentes metodologías y prácticas que puede implementar.

El sistema realizará preguntas predefinidas al usuario con respuestas limitadas que estarán asociadas a criterios principales de los diversos grupos de metodologías, para realizar una recomendación en base a las respuestas ingresadas.

El sistema almacenará y permitirá a los usuarios ver y examinar información detallada de las metodologías de desarrollo de software para el conocimiento de los usuarios de sistema.

6.3 Descripción Global del Producto

Nuestro producto es un sistema web de recomendaciones de metodologías, que permitirá a usuarios no experimentados en metodologías de desarrollo de software a realizar una óptima distribución de sus recursos para iniciar sus nuevos proyectos y conocer sobre otras metodologías que podrá adaptar a futuro.

6.3.1 Interfaz de usuario

Para el usuario el sistema mostrara en pantalla principal un breve mensaje sobre el sistema y tendrá en la barra de menú la opción de crear un nuevo registro o iniciar sesión, en caso que ya cuente con un registro en el sistema deberá de ingresar los datos solicitados. El sistema mostrara una nueva pantalla de principal en la que encontrara una breve descripción del funcionamiento del sw y un botón principal que nos derive a la pantalla de un cuestionario que nos permitirá generar una recomendación.

El sistema contará con una interfaz intuitiva para el usuario con accesos directos que permiten el fácil uso del sistema, contará con una barra de menú en la que encontrará botones

de acceso directo para conocer los distintos grupos de metodologías, prácticas ágiles entre otras, con la finalidad de que el usuario pueda examinarlas y obtener la información o recomendación de algún grupo de metodología.
Para el administrador el sistema debe tener en la barra de menú un botón de inicio de sesión que le permitirá acceder directamente a realizar algún cambio de información que estime conveniente para que el usuario pueda visualizar.

Luego de iniciar sesión el administrador contará con una pantalla principal en la cual contará con accesos directos para realizar alguna actualización, agregación o eliminación de información.
Además el sistema luego de cada cambio mostrará el resultado de estos o mostrará una alerta de error al administrador según corresponda el caso.

6.3.2 Interfaz De Hardware

El sistema que se implementará no necesita la interacción con periféricos hardware, ya que toda la información proporcionada será mostrada por pantalla y será usada inmediatamente por el usuario o administrador del sistema.

El sistema puede ser usado con equipos de las siguientes características:

- Notebook, Netbook, Computador de escritorio.
  Deben tener al menos instalada una memoria RAM de 2 Gb.
  Procesador Intel Celeron E1200 equivalente a 1.60 GHz, o superior
  Mouse básico
  Teclado
  Monitor (para PC de escritorio)
- Tablet, Smarphone, iphone.
  Deben de contar al menos con las siguientes características:
  Android versión recomendada 5.1 o superior
  ios versión 8.0 o superior
  1 Gb de RAM
  Procesador quad-core
- Router

Para acceder a la aplicación web debe de contar con una conexión a internet la cual será distribuida por un router el cual debe de contar con al menos estas características.

Puerto LAN para proporcionar una conexión directa entre el router y computador que utilizará para acceder a la página web.

Wi-Fi (opcional) si desea conectar equipos inalámbricos para acceder a la página web ya sean Tablet, Smarphone, iphone, Notebook, Netbook, entre otros.

Botones encendido/apagado (On/Off), reset y otro para activar/desactivar el Wi-Fi.

6.3.3 Interfaz Software

Nuestro sistema de recomendación de metodologías de desarrollo de software no se relaciona con otros sistemas ya que todo su proceso se realiza internamente.

Solo se necesita los programas básicos que posee un computador de usuario común, tales como:

- Nombre: Windows
- Abreviación: Windows
- Version:
  - Windows 7 Ultimate Service Pack 1 (32 bits)
  - Windows 8.1 Professional (64 bits)
  - Windows 10Home (64 bits)
- Fuente: https://www.microsoft.com/es-cl/windows

- Nombre: Navegador web Mozilla Firefox
- Abreviación: Firefox
- Versión: 55.0.3 (32-64 bit)
- Fuente: https://www.mozilla.org

- Nombre: Navegador web Google Chrome
- Abreviación: Google Chrome
- Versión: 60.0.3112.113 (32-64 bit)
- Fuente:https://www.google.es/chrome/browser/desktop/index.html

- Nombre: Yii2 Framework
- Abreviación: Yii2
- Versión: 2.0.12
- Fuente: http://www.yiiframework.com/download/
- Nombre: XAMPP
- Abreviación: XAMPP
- Versión: 5.6.31
- Fuente: https://www.apachefriends.org/es/download.html

6.3.4 Interfaces de comunicación

En nuestro sistema se utilizará el diseño de Cliente-Servidor de 3 capas

- capa de presentación
- capa de aplicación
- capa de datos.

Ya que nuestro sistema realiza la totalidad de sus funcionalidades en un servidor web, por consiguiente, como se debe acceder desde cualquier computadora o dispositivo que tenga conexión a internet, a través de un navegador web, Firefox o Google Chrome, por ejemplo, es necesario realizar esta separación entre el cliente y servidor.

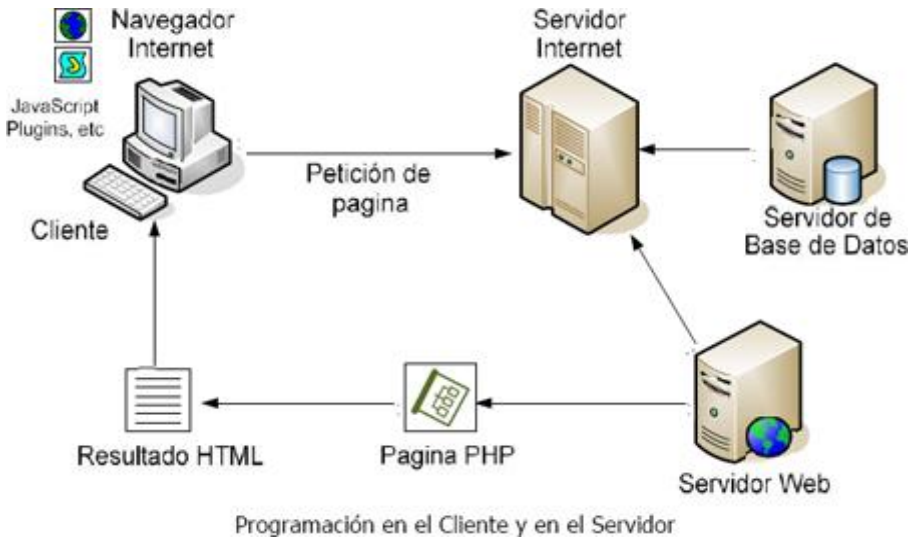


FIGURA N° 24: Interfaces de comunicación

6.4 Requerimientos Específicos

6.4.1 Requerimientos Funcionales del sistema

ID	Nombre	Descripción
RF01	Ingresar al sistema (Administrador - Usuario)	El sistema debe de permitir acceder desde un navegador web.
RF02	Iniciar sesión en el sistema (Administrador - Usuario)	El sistema debe solicitar al administrador o usuario nombre y contraseña para acceder a las opciones disponibles según perfil.
RF03	Emitir alerta de error (Administrador - Usuario)	El sistema debe enviar una alerta de error al usuario o administrador si éste ingresa o realizar alguna actividad errónea.
RF04	Registrar usuario	El sistema debe de permitir a aquellos no tengan cuenta crear un nuevo perfil de usuario.
RF05	Modificar grupo de metodologías (Administrador)	El sistema debe permitir al administrador modificar información perteneciente a los distintos grupos de metodologías.
RF06	Visualizar grupo de metodologías (Administrador - Usuario)	El sistema debe permitir al administrador o usuario visualizar grupos de metodologías con su respectiva información. El administrador además tendrá la opción a actualizar directamente en la vista.
RF07	Ingresar metodologías (Administrador)	El sistema debe permitir al administrador ingresar nuevas metodologías con su respectiva información.
RF08	Modificar metodologías (Administrador)	El sistema debe permitir al administrador modificar metodologías con su respectiva información.
RF09	Eliminar metodologías	El sistema debe permitir al administrador eliminar

	(Administrador)	metodologías con su respectiva información.
RF10	Visualizar metodologías (Administrador - Usuario)	El sistema debe permitir al administrador o usuario visualizar metodologías con su respectiva información. El administrador además tendrá la opción de actualizar o eliminar directamente en la vista.
RF11	Ingresar prácticas ágiles (Administrador)	El sistema debe permitir al administrador ingresar nuevas prácticas ágiles con su respectiva información.
RF12	Modificar prácticas ágiles (Administrador)	El sistema debe permitir al administrador modificar prácticas ágiles con su respectiva información.
RF13	Eliminar prácticas ágiles (Administrador)	El sistema debe permitir al administrador eliminar prácticas ágiles con su respectiva información.
RF14	Visualizar prácticas ágiles (Administrador - Usuario)	El sistema debe permitir al administrador o usuario visualizar prácticas ágiles con su respectiva información. El administrador además tendrá la opción de actualizar o eliminar directamente en la vista.
RF15	Visualizar cuestionario (Administrador - Usuario)	El sistema debe permitir al administrador o usuario visualizar preguntas con sus posibles respuestas.
RF16	Responder Cuestionario (Administrador - Usuario)	El sistema debe de permitir al administrador o usuario seleccionar respuesta para responder cuestionario.
RF17	Solicitar recomendación (Administrador - Usuario)	El sistema debe de permitir al administrador o usuario una recomendación en base a sus respuestas tras presionar un botón.
RF18	Visualizar recomendación (Administrador - Usuario)	El sistema debe de permitir al administrador o usuario visualizar la recomendación en base a respuestas ingresadas.
RF19	Modificar acerca de (Administrador)	El sistema debe permitir al administrador modificar acerca de.

RF20	Visualizar acerca de (Administrador - Usuario)	El sistema debe permitir al administrador o usuario visualizar acerca de. El administrador además tendrá la opción de actualizar directamente en la vista.
RF21	Ingresar sugerencias (Usuario)	El sistema debe permitir al usuario ingresar sugerencias del sistema o recomendación.
RF22	Visualizar sugerencias (Administrador - Usuario)	El sistema debe de permitir al administrador o usuario visualizar sugerencias ingresadas. El administrador podrá visualizar todas las sugerencias ingresadas por los usuarios. El usuario podrá visualizar todas sus sugerencias ingresadas.
RF23	Ingresar calificación (Usuario)	El sistema debe permitir al usuario ingresar calificación de recomendaciones realizadas.
RF24	Visualizar calificación (Administrador - usuario)	El sistema debe de permitir al administrador o usuario visualizar calificaciones ingresadas. El administrador podrá visualizar calificación de todos los usuarios que obtuvieron alguna recomendación. El usuario podrá visualizar calificación que ha dado a recomendaciones generadas por sistema.
RF25	Cerrar sesión (Administrador - Usuario)	El sistema debe de permitir al administrador o usuario cerrar su sesión en el sistema.
RF26	Salir del sistema (Administrador - Usuario)	El sistema debe permitir al administrador o usuario salir del sistema.
RF27	Generar recomendación	El sistema debe permitir generar una recomendación de metodologías de desarrollo de software a partir de información de respuestas ingresadas.

TABLA N° 6: Requerimientos funcionales

6.4.2 Interfaces externas de entrada

ID	Nombre del ítem	Detalle de Datos contenido en ítem
IE_01	Datos de registro de usuarios	nombreusuario, email, contraseña.
IE_02	Datos de inicio de sesión	nombreusuario, contraseña.
IE_03	Datos de metodologías	nombre_de_metodologia, descripcion_de_metodología, imagen_de_metodologia.
IE_04	Datos de prácticas ágiles	nombre_practica_agil, descripcion_practica_agil, imagen_practica_agil.
IE_05	Datos de sugerencias	nombre (usuario), sugerencia.
IE_06	Datos de calificación	calificacion.
IE_07	Datos de cuestionario	respuesta.

TABLA N° 7: Interfaces externas de entrada

6.4.3 Interfaces externas de Salida

ID	Nombre del ítem	Detalle de Datos contenido en ítem	Medio salida
IS_01	Grupo de metodologías	nombre_grupo_metodologia, descripcion_del_grupo_de_metodologia.	Pantalla
IS_02	Metodologías	nombre_de_metodologia, descripcion_de_metodologia, imagen_de_metodologia.	Pantalla
IS_03	Prácticas ágiles	nombre_practica_agil, descripcion_practica_agil, imagen_practica_agil.	Pantalla
IS_04	Acerca de	Descripcion	Pantalla
IS_05	Cuestionario	pregunta, respuesta.	Pantalla
IS_06	Calificación	fecharegistro, pregunta, respuesta, calificacion.	Pantalla
IS_07	Sugerencias	nombre (usuario), sugerencia.	Pantalla

TABLA N° 8: Interfaces externas de salida

6.4.4 Atributos del producto

- USABILIDAD- OPERABILIDAD: El sistema presenta mensajes claros sobre errores que pudiesen ocurrir durante el uso e ingreso de datos como por ejemplo:

  Registro de usuario: si el usuario ingresa un nombre de usuario ya registrado en la bases de datos el sistema mostrara un mensaje que indique que nombre de usuario ya está registrado.

  Inicio de sesión: Si se ingresa nombre de usuario o contraseña incorrecta el sistema mostrara un mensaje donde indica que el nombre de usuario o contraseña son incorrectos.

  Eliminar información: Si el administrador desea eliminar información ya ingresada en el la base de datos el sistema mostrara un mensaje donde el administrador tendrá que confirmar para proceder con la eliminación.

- FUNCIONALIDAD-SEGURIDAD: El sistema protege la información ingresada por el administrador mediante método login password con la finalidad de que un usuario común no tenga acceso a modificar o eliminar información, además el sistema permite el ingreso de acuerdo al perfil con el cual se ingresa al sistema.

7 FACTIBILIDAD

7.1 Factibilidad técnica

La factibilidad técnica permite obtener la información necesaria respecto a, si existe o está al alcance la tecnología necesaria para la realización del sistema a implementar, chequeando si se cuenta con los equipos y programas mínimos para la el desarrollo y utilización de este.

Equipo o Producto	**Cantidad**	**Especificación**
Sistema operativo	2	Windows 10/8.1
Notebook	2	Procesador: Intel Core i3 de 2,3 GHz, Memoria RAM: 4GB, Disco duro: 500GB Entrada HDMI: 1 Entradas USB: 3 Teclado: Tipo isla Unidad óptica
Office	2	Microsoft® Office 2016 Hogar y Estudiantes. Word, Excel, PowerPoint y OneNote para 1 PC. Almacena archivos en la nube con OneDrive.
Powerdesigner	2	2 licencias de Powerdesigner por un año, para el Modelamiento de datos.
Sublime text 3	2	2 licencias para el editor de código multiplataforma SublimeText 3
BalsamiqMockups	2	Licencia para 3 pc, para el desarrollo de los mockups o wireframe del sistema

Framework	2	Yii2 framework para el desarrollo del sistema
Gestor de servidor y base de datos	2	Xampp que nos proporciona motor de servidor apache y gestor de base de datos MySql
Drive y documentación online	1	Google Drive y Google Docs. almacenamiento y desarrollo de documentación online

TABLA N° 9: Factibilidad técnica

7.2 Factibilidad operativa

El sistema web que se desarrollo permite a usuarios no experimentados obtener información de una forma directa y en un solo lugar respecto a los diferentes grupos de metodologías y practicas aplicadas de lo cual se generan aspectos positivos y negativos detallados a continuación.

Positivo

- Orienta a personas poco experimentadas en el área de desarrollo de software.
- Orienta a desarrolladores poco experimentados en el área de desarrollo de software.
- Orienta a la toma de decisiones sobre la implementación de algún grupo de metodología.
- Informa en un solo lugar (página web) sobre los diferentes grupos de metodologías.
- Informa en un solo lugar (página web) sobre las diferentes metodologías específicas.
- Informa en un solo lugar (página web) sobre las diferentes prácticas ágiles.
- El usuario puede acceder al sistema mediante algún pc o dispositivo móvil que cuente con acceso a internet (ej. tablet, notebook, smartphone, etc.).

Negativo

- El usuario puede tomar la recomendación como respuesta absoluta.
- Al ser una orientación no siempre puede dar el resultado esperado.
- Al ser un sistema web el usuario siempre deberá disponer de acceso a internet.

7.3 Factibilidad económica

Costos de personal y desarrollo

La propuesta no estima que se deba realizar un gasto adicional en el costo del personal, puesto que estará a cargo de dos alumnos de la Universidad del Bío-Bío con el objetivo de optar al título de Ingeniero Civil en Informática.

A continuación se presenta una aproximación de los posibles costos asociados al desarrollo del sistema de recomendación. Para ello se ha considerado el valor de la UF a **$**26.800 y el costo de investigación y desarrollo por una sola persona.

Actividad	**Duración (Horas)**	**Costo por Hora (UF)**	**Costo HH por hora**	**Costo total**
Investigación teórica	30	0.190	$5.092	$152.760
Desarrollo herramienta de obtención de datos.	90	0.110	$2.948	$265.320
Investigación empírica	60	0.210	$5.628	$337.680
Diseño	50	0.122	$3.269	$163.450
Desarrollo	150	0.140	$3.752	$562.800
Implementación	10	0.150	$4.002	$40.020
Pruebas	5	0.140	$3.752	$18.760
Total	395			$1.538.790

TABLA N° 10: Factibilidad económica

Costos de hardware y software

Se considero un desktop con características adecuadas para el desarrollo donde su valor no sobrepasa los $800.00 incluyendo licencia de Windows.

Costo total

Investigación y Desarrollo **$1.538.790.-** por una persona

Hardware y Software **$800.000.-** por una persona

Costo total del desarrollo del proyecto: **$1.538.790 x 2 + $800.000 x2 = $4.677.580.-**

8 ANÁLISIS

8.1 Procesos de Negocios futuros

8.1.1 Administrador - Sistema

En este sistema web el administrador podrá llevar el control total de la información que desee transmitir, para esto la aplicación web le solicitará identificación y clave de acceso con la finalidad de no permitir ingreso a personal no autorizado, una vez que el sistema valida los datos ingresados desplegara un menú con opciones dentro de las cuales el administrador podrá elegir.

Dentro de las opciones que desplegara el sistema tenemos; actualizar - modificar o visualizar los diferentes grupos de metodologías y acerca de, visualizar cuestionario y posibles respuestas, agregar, actualizar - modificar, eliminar o visualizar información correspondiente a: metodologías, prácticas ágiles.

8.1.2 Usuario - Sistema

En este sistema web el usuario podrá realizar todas las acciones disponibles con la obligación de registrarse e iniciar sesión, una vez que el usuario se encuentre en el sistema este podrá visualizar el menú de las opciones que dispone.

Dentro de las opciones que mostrara el sistema tenemos; grupos de metodologías, prácticas agiles, acerca de y contara con dos botón centrales los cuales permitirán iniciar el proceso de recomendación de un grupo de metodología y calificar una recomendación ya realizada.

8.2 Diagrama de Flujo de Datos

8.2.1 DFD contexto

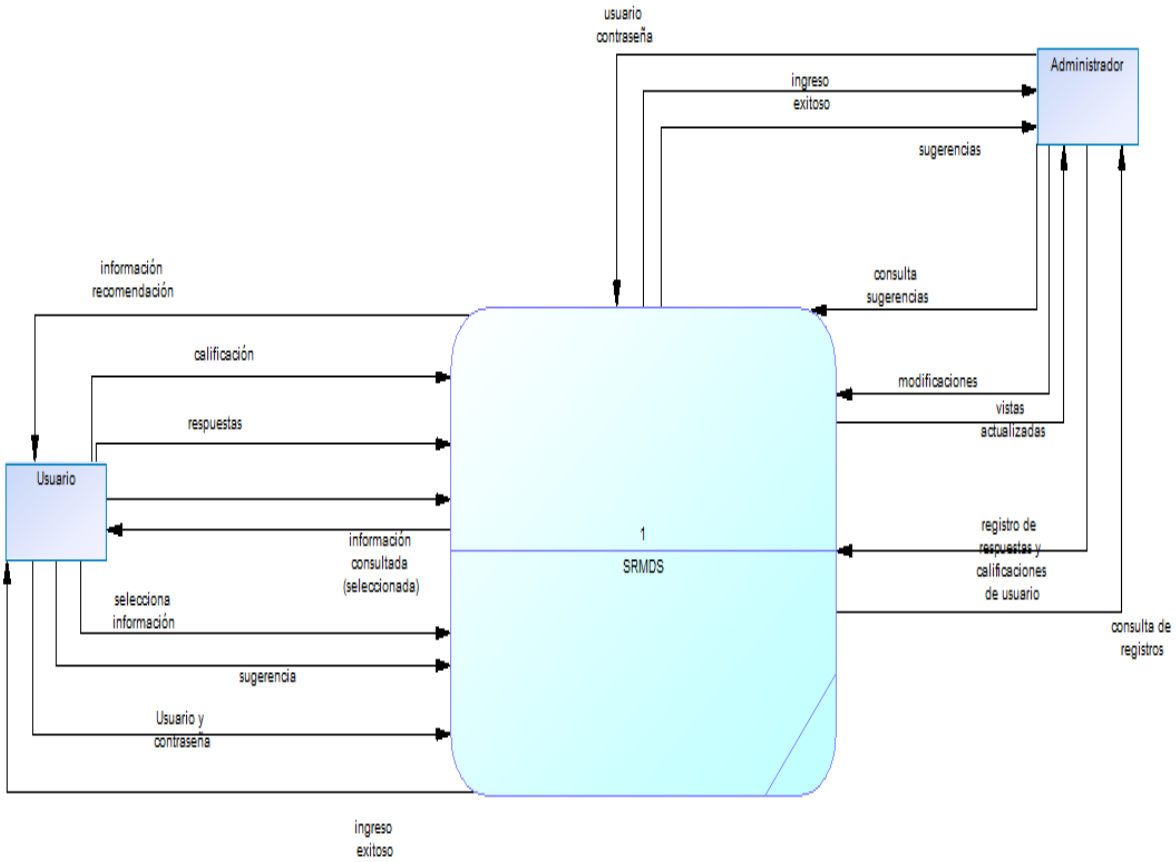


FIGURA N° 25: DFD contexto

8.2.2 DFD superior

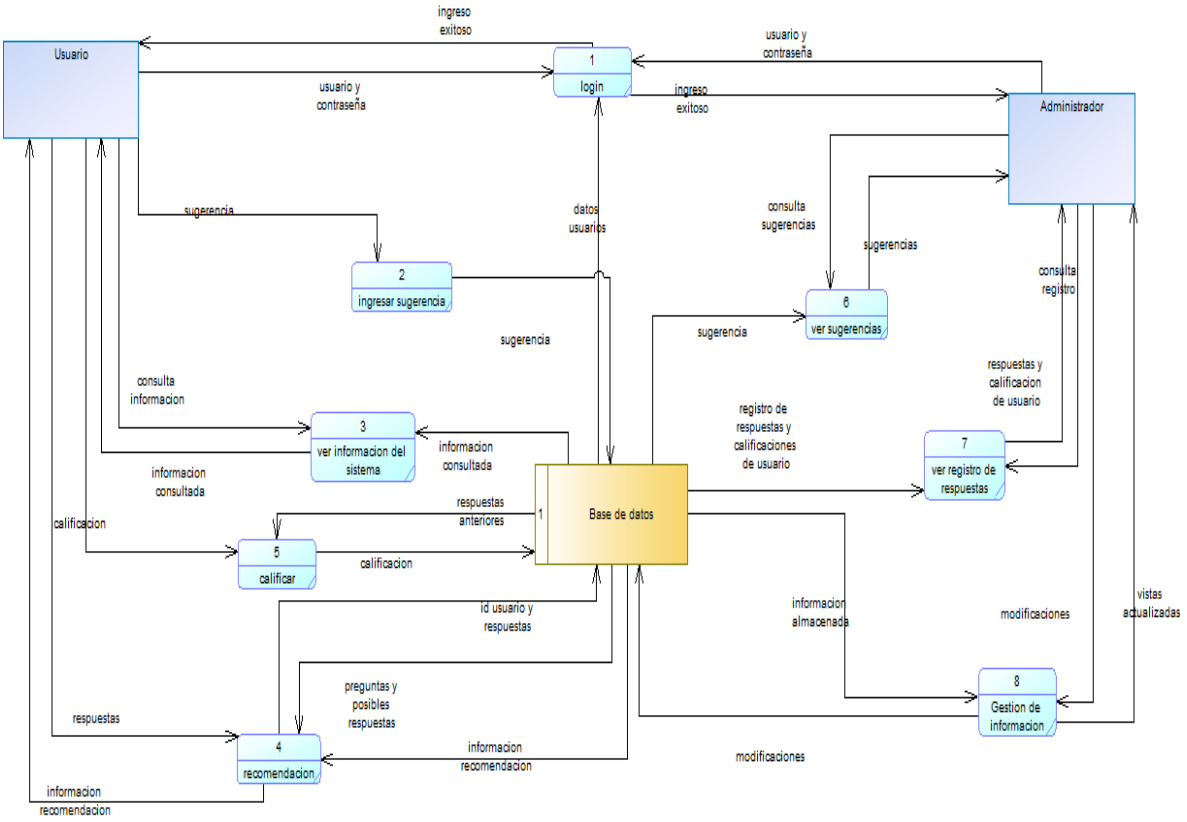


FIGURA N° 26: DFD Superior

8.2.3 DFD detalle

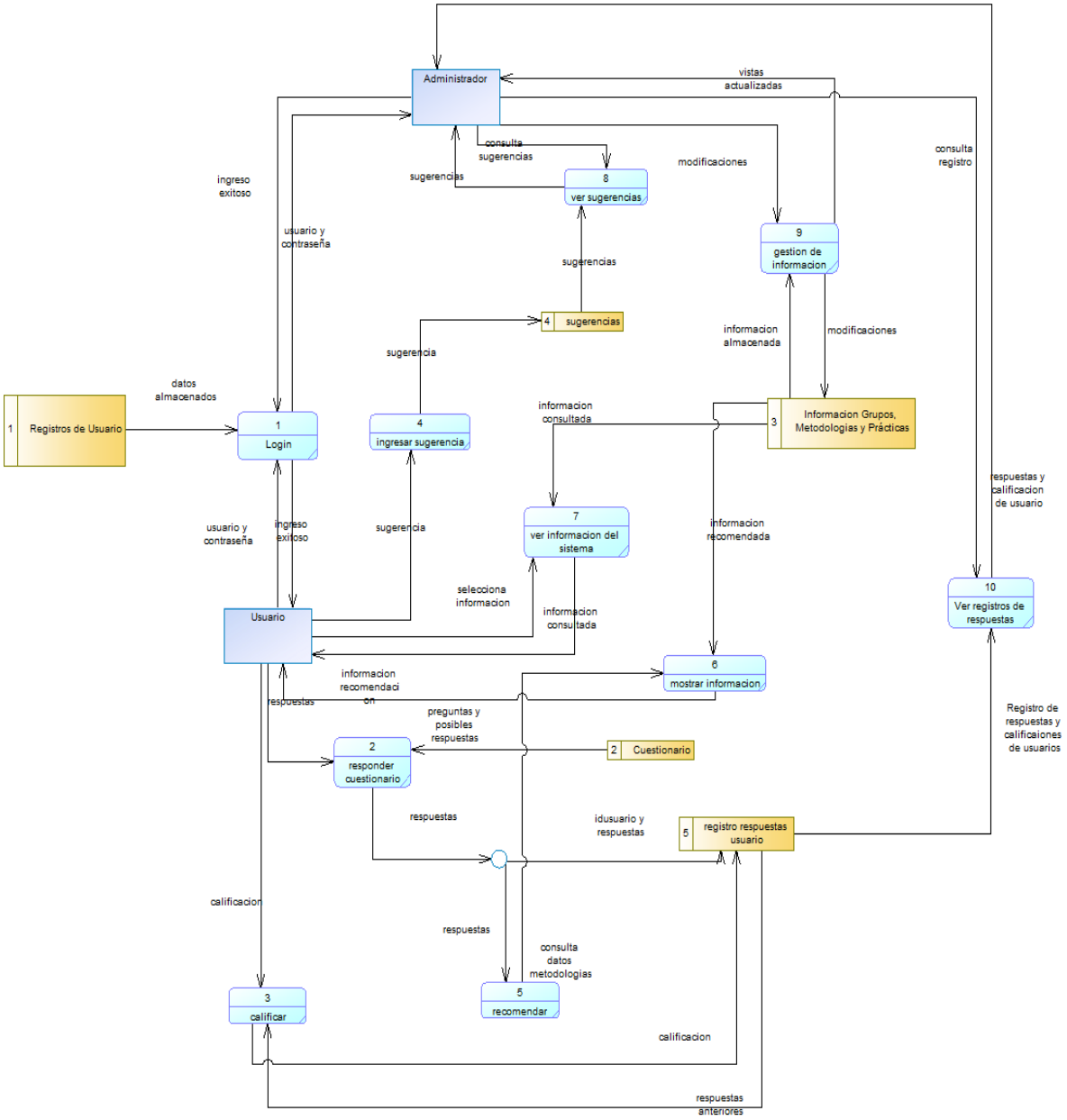


FIGURA N° 27: DFD Detalle

8.3 Diagrama de casos de uso

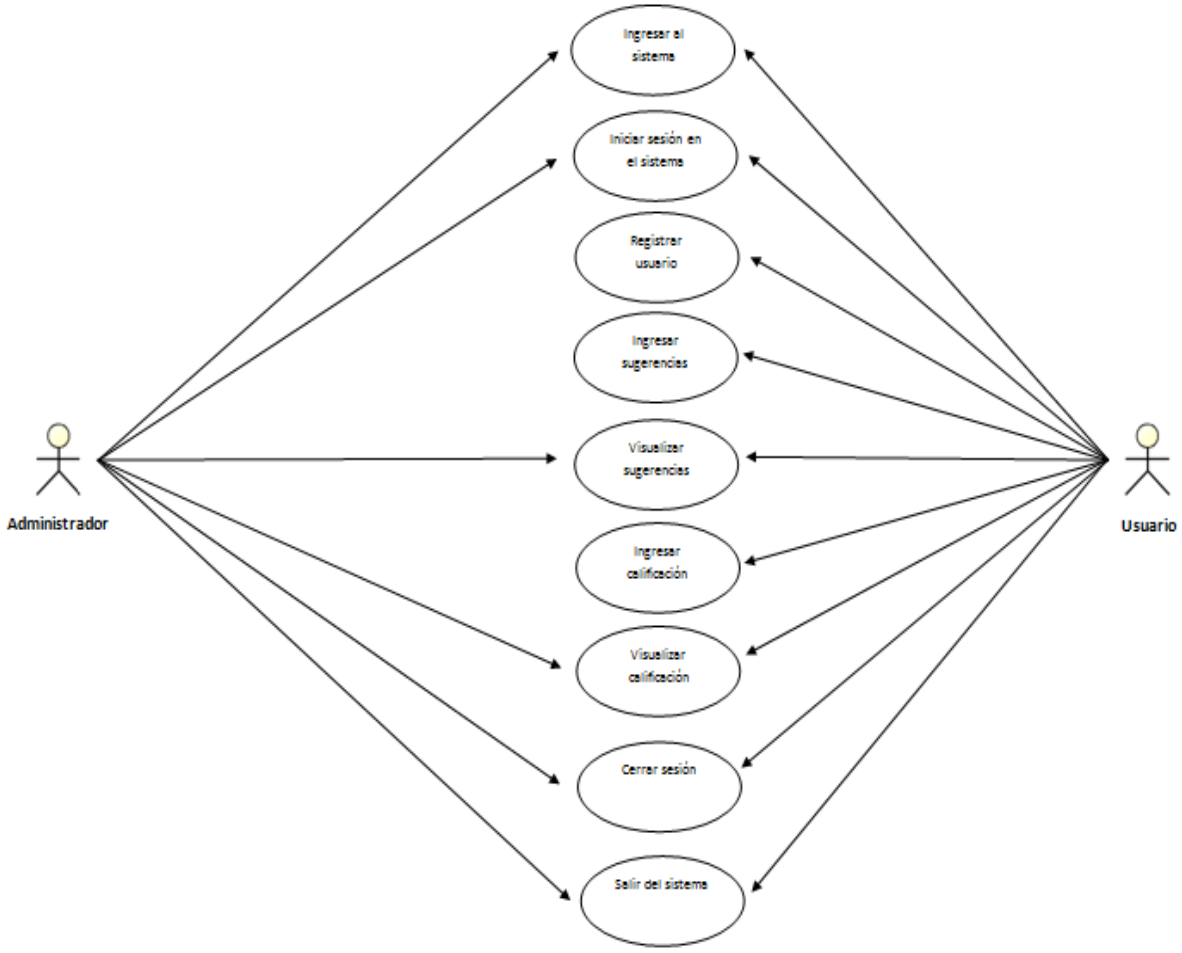


FIGURA N° 28: Caso de uso 1 de 2

FIGURA N° 29: Casos de uso 2 de 2

8.3.1 Actores

Para el diagrama de casos de uso consideramos a dos actores que interactúan con el sistema, en este caso el administrador y el usuario.

Actor: Administrador

- **Rol:** Corresponde al encargado de gestionar la información correspondiente a:

Los diferentes grupos de metodologías, las diferentes metodologías, las diferentes prácticas ágiles, acerca de. Supervisa el correcto funcionamiento del sistema de recomendación, lo hace mediante la correcta actualización de la información de la base de datos.

- **Nivel de conocimientos técnicos requeridos:** Conocimientos nivel usuario en aplicaciones funcionales de oficina y en específico navegación web.

- **Nivel de privilegios del sistema:** Tiene acceso a todas las funcionalidades del sistema, tales como agregar información de diferentes metodologías, prácticas ágiles, editar información de diferentes metodologías, grupos de metodologías, prácticas ágiles, acerca de, además podrá eliminar información de metodologías o prácticas ágiles que estén ingresadas .

Administrador

Realiza "Iniciar Sesión" por medio de un login (User y Password), posteriormente efectúa una tarea determinada, puede modificar o visualizar información correspondiente a los diferentes: grupos de metodologías de desarrollo de software, acerca de, además puede agregar, modificar, eliminar o visualizar información correspondiente a metodología, prácticas ágiles.

Para realizar la salida del sistema tras haber finalizado todas sus actividades, se cierra la sesión mediante el caso de uso "Salir del Sistema".

Actor: Usuario

- **Rol:** Corresponde al usuario final quien utilizara el sistema de recomendación de metodologías de desarrollo de software, lo hace mediante el inicio de sesión al sistema.

- **Nivel de conocimientos técnicos requeridos:** Conocimientos nivel usuario en aplicaciones funcionales de oficina y en específico navegación web.

- **Nivel de privilegios del sistema:** Tiene acceso a información disponible respecto a las metodologías de desarrollo de software, tales como visualizar información de diferentes grupos de metodologías, metodologías pertenecientes al los distintos grupos, prácticas ágiles, cuestionario, además podrá responder el cuestionario con la respuesta ya definidas para posteriormente visualizar recomendación de un grupo de metodologías de desarrollo de software.

Usuario

Realiza "Iniciar Sesión" por medio de un login (User y Password), posteriormente puede efectuar diferentes acciones como visualizar información correspondiente a los diferentes grupos de metodologías de desarrollo de software donde además encontrara información de las diferentes metodologías que pertenecen al determinado grupo, visualizar preguntas y posibles respuesta de las que tendrá que selecciona solo una para generar posteriormente la recomendación de metodologías de desarrollos de software, además podrá calificar las recomendaciones que de el sistema junto con agregar sugerencias de este.

Para realizar la salida del sistema tras haber finalizado todas sus actividades, se cierra la sesión mediante el caso de uso "Salir del Sistema".

8.3.2 Casos de Uso y descripción

ID caso de uso	Nombre caso de uso
CU01	< Ingresar al sistema >
CU02	< Salir del sistema >
CU03	< Iniciar sesión en el sistema >
CU04	< Cerrar sesión >
CU05	< Registrar usuario >
CU06	< Modificar grupo de metodologías >
CU07	< Visualizar grupo de metodologías >
CU08	< Ingresar metodologías >
CU09	< Modificar metodologías >
CU10	< Eliminar metodologías >
CU11	< Visualizar metodologías >
CU12	< Ingresar prácticas ágiles >
CU13	< Modificar prácticas ágiles >
CU14	< Eliminar prácticas ágiles >
CU15	< Visualizar prácticas ágiles >
CU16	< Visualizar cuestionario >
CU17	< Responder cuestionario >
CU18	< Modificar acerca de >
CU19	< Visualizar acerca de >

CU20	< Solicitar recomendación >
CU21	< Visualizar recomendación >
CU22	< Ingresar sugerencias >
CU23	< Visualizar sugerencias >
CU24	< Ingresar calificación >
CU25	< Visualizar calificación >

TABLA N° 11: Casos de uso

8.3.3 Especificación de los Casos de Uso

8.3.3.1 Caso de Uso: <Ingresar al sistema>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario ingresan al sistema de recomendación de metodologías de desarrollo de software.

Pre-Condiciones: Se debe contar con una conexión a internet e ingresar a un navegador web para cargar la página del sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario cargan la página web en un navegador.	2.- El sistema inmediatamente muestra su inicio, en donde estarán disponibles diversas opciones.

Flujo de Eventos Alternativo:

Al actor	El sistema
1(a).- El administrador o usuario modifican la ruta de la página.	2(a).- El sistema mostrará un error de página no encontrada.

Post-Condiciones: El administrador o usuario pueden consultar diversa información disponible en el sistema.

8.3.3.2 Caso de Uso: <Salir del sistema>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario salen completamente del sistema.

Pre-Condiciones: El administrador o usuario deben de hacer cerrado sesión con anterioridad en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario presiona el cuadro rojo que contiene una 'x' en su interior.	2.- El sistema se cerrará automáticamente.

Flujo de Eventos Alternativo:

Al actor	El sistema
1(a).- El administrador o usuario presentan problemas de navegación.	2(a).- El sistema se cerrará y no guardará los datos que no se alcanzaron a actualizar.

8.3.3.3 Caso de Uso: <Iniciar sesión en el sistema >

Actores: Administrador, Usuario.

Descripción: El administrador o usuario inicia sesión (se autentica) en el sistema.

Pre-Condiciones: El administrador o usuario debe haber ingresado al sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "Iniciar sesión".	2.- El sistema despliega el formulario de inicio de sesión, solicitando el nombre de usuario y contraseña.
3.- El administrador o usuario ingresa su nombre de usuario y contraseña, presiona el botón "Iniciar sesión" del formulario de ingreso.	4.- El sistema autentica al administrador o usuario, desplegando vista inicial según corresponda perfil.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador o usuario ingresa el nombre de usuario y/o contraseña incorrecta.	4(a).- El sistema emite una alerta de; Nombre de usuario o contraseña invalida.
3(b).- El administrador o usuario no ingresa el nombre de usuario y/o contraseña.	4(b).- El sistema emite una alerta advirtiendo que no puede dejar campos vacíos.

Post-Condiciones: El sistema despliega la vista inicial del administrador o usuario y en la esquina superior derecha donde se ubica el botón de inicio de sesión se muestra un nuevo botón "cerrar sesión (nombre administrador)" o "cerrar sesión (nombre usuario)" según el caso.

8.3.3.4 Caso de Uso: <Cerrar sesión>

Actores: administrador, Usuario.

Descripción: El administrador o usuario cierra la sesión del sistema.

Pre-Condiciones: El administrador o usuario deben de haber iniciado sesión en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona la opción "cerrar sesión" de la barra de menú.	2.-El sistema cierra la sesión.

Post-Condiciones: Se cierra la sesión actual, el sistema re-direcciona a la página principal de inicio.

8.3.3.5 Caso de Uso: <Registrar usuario>

Actores: Usuario (No registrados).

Descripción: Se crea un nuevo perfil de usuario en el sistema, para que pueda hacer uso de este.

Pre-Condiciones: Ingresar al sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando se presiona el botón "regístrate".	2.-El sistema despliega un formulario donde solicita un nombre de usuario, email y contraseña.
3.- El usuario ingresa la información solicitada y presiona el botón registrar.	4.- El sistema almacena la información ingresada, y despliega una notificación de registro exitoso.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El usuario ingresa un nombre de usuario y/o email ya registrados en el sistema.	4(a).- El sistema muestra una alerta de existencia del nombre de usuario y/o email que desea registrar, no permitiendo el registro en el sistema.
3(b).- El usuario ingresa un correo invalido.	4(b).- El sistema mostrará una alerta destacando que el formato del email ingresado no es válido, no permitiendo el registro en el sistema.
3(c).- El usuario ingresa confirmación de contraseña distinta o que no están dentro del límite de caracteres (entre 5 y 16).	4(c).- El sistema mostrará una alerta advirtiendo el problema que se presenta en el campo de las contraseñas, no permitiendo el registro en el sistema.

Post-Condiciones: El sistema guarda la información ingresada en la base de datos, y el nuevo usuario podrá ingresar al sistema a través del sistema de ingreso con su nombre de usuario y contraseña.

8.3.3.6 Caso de Uso: <Modificar grupo de metodologías>

Actores: Administrador.

Descripción: El administrador modifica descripción de un grupo de metodologías ingresada en el sistema: El administrador debe hacer iniciado sesión en el sistema.

Pre-Condiciones: El administrador debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador presiona el botón "modificar"(icono en forma de lápiz) del grupo de metodologías que desea.	2.- El sistema despliega formulario donde podrá modificar la descripción de esta.
3.- El administrador modifica la información del formulario, y presiona el botón "actualizar".	4.- El sistema actualiza y guarda la información modificada en la base de datos, mostrando una vista con la información ingresada.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador modifica y sale sin presionar el botón actualizar.	4.- El sistema no realiza cambios.

Post-Condiciones: El sistema actualiza y guarda la nueva información en la base de datos y muestra una vista donde se encuentra la información ingresada, dando la opción de modificarla nuevamente.

8.3.3.7 Caso de Uso: <Visualizar grupo de metodologías>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario podrá visualizar la información ya ingresada en la base de datos.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario selecciona la opción de visualizar un grupo de metodologías	2.- El sistema muestra la información almacenada en la base de datos al usuario. El administrador además tendrá la opción de modificar dicha información ya ingresada.

Post-Condiciones: El sistema mantiene al administrador o usuario con su sesión iniciada a la espera de nuevas acciones, no se presentan cambios.

8.3.3.8 Caso de Uso: <Ingresar metodologías>

Actores: Administrador.

Descripción: El administrador ingresa una nueva metodología a la base de datos asociada a un grupo de metodologías en el sistema para que esta pueda ser visualiza en el sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador selecciona la opción de "nuevo registro".	2.- El sistema solicitara ingresar datos relacionados con la nueva metodología (Grupo de metodología, nombre de metodología, descripción, imagen).
3.- El administrador ingresa la información solicitada.	4.- El sistema da la opción de crear nueva metodología (guardar).
5.- El administrador selecciona la opción de crear.	6.- El sistema guarda la información ingresada en la base de datos y muestra una vista en la que se encuentra la información ingresada por el administrador.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador no ingresa datos solicitados.	4(a).- El sistema muestra campos sin datos ingresados.
5(a).- El administrador selecciona crear sin haber ingresados datos.	6(a).- El sistema mostrará una alerta donde resaltara los campos obligatorios que se encuentran vacios.

Post-Condiciones: El sistema guarda la información ingresada en la base de datos, luego una vista donde el administrador podrá visualizar la información ingresada con la opción de modificar o eliminar dicha metodología.

8.3.3.9 Caso de Uso: <Modificar metodologías>

Actores: Administrador.

Descripción: El administrador modifica una metodología ingresada en el sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión en el sistema, debe existir al menos una metodología ingresada en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador visualizar el listado de metodologías o entra en la vista de una metodología y presiona el botón actualizar.	2.- El sistema muestra un formulario donde puede editar lo que ya está guardado en la base de datos.
3.- El administrador modifica la información del formulario, y presiona el botón "actualizar".	4.- El sistema actualiza y guarda la nueva información en la base de datos y muestra una vista con la información ingresada.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador no ingresa el "id del grupo de metodologías", y presiona el botón "actualizar".	4(a).- El sistema emite una alerta indicando que el campo no puede estar vacío.

Post-Condiciones: El sistema actualiza y guarda la nueva información en la base de datos y muestra una vista donde muestra la información ingresada, dando la opción de modificar nuevamente o eliminar.

8.3.3.10 Caso de Uso: <Eliminar metodologías>

Actores: Administrador.

Descripción: El administrador elimina una metodología ingresada en el sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión en el sistema, debe de existir al menos una metodología ingresada en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador visualiza el listado de metodologías o entra en la vista de una metodología y presiona el botón eliminar (Icono en forma de basurero).	2.- El sistema emite una alerta de confirmación.
3.- El administrador confirma la eliminación.	4.- El sistema elimina el registro de la base de datos.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador cancela la eliminación.	4(a).- El sistema permanece en la pantalla actual y no ejecuta ninguna acción.

Post-Condiciones: El sistema elimina la metodología seleccionada en la base de datos y muestra el listado actualizado de metodologías.

8.3.3.11 Caso de Uso: <Visualizar metodologías>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario podrá visualizar la información ya ingresada en la base de datos.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad, debe de existir al menos una metodología ingresada en el sistema para poder visualizar la información.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario selecciona la opción de visualizar una metodología.	2.- El sistema muestra los datos almacenados en la base de datos al usuario. El administrador además tendrá la opción de modificar o eliminar la información ya ingresada.

Post-Condiciones: El sistema mantiene al administrador o usuario con su sesión iniciada, no se presentan cambios.

8.3.3.12 Caso de Uso: <Ingresar prácticas ágiles>

Actores: Administrador.

Descripción: El administrador ingresa una nueva práctica ágil a la base de datos para que esta puede ser visualizada en la página del sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador selecciona la opción de ingresar metodología.	2.- El sistema solicitará ingresar datos relacionados con la nueva práctica (nombre, descripción, imagen).
3.- El administrador ingresa la información solicitada.	4.- El sistema muestra la información ingresada y da la opción de "crear"(guardar).
5.- El administrador selecciona crear.	6.- El sistema guarda la información ingresada en la base de datos, muestra una vista donde se encuentra la información ingresada por el administrador.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador no ingresa datos solicitados.	4(a).- El sistema muestra los campos sin datos ingresados y no realiza cambios.
5(a).- El administrador selecciona la opción "crear".	6(a).- El sistema mostrará una alerta donde se resaltan los campos obligatorios que se encuentran vacios.

Post-Condiciones: El sistema guarda la información ingresada en la base de datos, luego muestra una vista donde el administrador podrá visualizar la información ingresada con la opción de modificar o eliminar en caso de algún error.

8.3.3.13 Caso de Uso: <Modificar prácticas ágiles>

Actores: Administrador.

Descripción: El administrador modifica una práctica ágil ingresada en el sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión en el sistema, debe existir al menos una práctica ágil ingresada en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador visualiza el listado de prácticas ágiles o entra en la vista de una y presiona el botón actualizar.	2.- El sistema muestra un formulario con los campos disponibles para la actualización.
3.- El administrador modifica la información del formulario, y presiona el botón "actualizar".	4.- El sistema actualiza y guarda la nueva información en la base de datos, muestra una vista con la información ingresada.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador no ingresa los campos solicitados y presiona el botón "actualizar".	4(a).- El sistema emite una alerta diciendo que este campo no puede quedar vacío.

Post-Condiciones: El sistema actualiza y guarda la nueva información en la base de datos, muestra una vista donde está la información ingresada dando la opción de modificar o eliminar.

8.3.3.14 Caso de Uso: <Eliminar prácticas ágiles>

Actores: Administrador.

Descripción: El administrador elimina una práctica ágil ingresada en el sistema.

Pre-Condiciones: El administrador debe haber iniciado sesión en el sistema, debe existir al menos una práctica ágil ingresada en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador visualiza el listado de prácticas o entra en la vista de una práctica ágil y presiona el botón eliminar (icono en forma de basurero)	2.- El sistema emite una alerta de confirmación de eliminación.
3.- El administrador conforma la eliminación.	4.- El sistema elimina el registro de la base de datos.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador cancela la eliminación.	4(a).- El sistema permanece en la pantalla actual y no ejecuta ninguna acción.

Post-Condiciones: El sistema elimina la práctica ágil seleccionada en la base de datos y muestra el listado actualizado de las prácticas ágiles.

8.3.3.15 Caso de Uso: <Visualizar prácticas ágiles>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario podrá visualizar la información ya ingresada en la base de datos.

Pre-Condiciones: El administrador o usuario debe haber iniciado sesión con anterioridad, debe de existir al menos una práctica ágil ingresada para poder visualizar la información.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario selecciona la opción de visualizar una práctica ágil.	2.- El sistema muestra los datos almacenados en la base de datos al usuario. El administrador además tendrá la opción de modificar o eliminar la información ya ingresada.

Post-Condiciones: El sistema mantiene al administrador o usuario con su sesión iniciada, no presenta cambios.

8.3.3.16 Caso de Uso: <Visualizar cuestionario>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario podrá visualizar el cuestionario que presenta el sistema para generar la recomendación.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el administrador o usuario presiona el botón "iniciar cuestionario", ubicado en la pantalla principal.	2.- El sistema muestra las preguntas del cuestionario, cada una con su respectivo botón para desplegar las posibles respuestas y un botón para generara la recomendación.

Post-Condiciones: El sistema mantiene al administrador o usuario con su sesión iniciada, no presentan cambios.

8.3.3.17 Caso de Uso: < Responder cuestionario>

Administrador, Usuario.

Descripción: El administrador o usuario responden las preguntas del cuestionario.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "responder" de una de las preguntas.	2.- El sistema abre una "pop-up" que muestra la pregunta y una lista desplegable con las posibles respuestas a la pregunta.
3.- El administrador o usuario selecciona una respuesta y presiona el botón "actualizar".	4.- El sistema actualiza y guarda la respuesta, muestra la nueva respuesta almacena.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador o usuario no presiona el botón "actualizar"	4(a).- El sistema no realiza ningún cambio.

Post-Condiciones: El sistema actualiza y guarda las respuestas seleccionadas.

8.3.3.18 Caso de Uso: < Modificar acerca de>

Actores: Administrador.

Descripción: El administrador modifica la información que se presenta en el acerca de.

Pre-Condiciones: El administrador debe haber iniciado sesión con anterioridad en el sistema.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- En la visualización del acerca de el administrador presiona el botón "actualizar" (icono en forma de lápiz).	2.- El sistema muestra la información del acerca de, predefinida y modificable.
3.- El administrador realiza los cambios y presiona el botón "actualizar".	4.- El sistema actualiza y guarda la información en la base de datos, muestra una vista donde se ven los cambios, además de un botón para modificar nuevamente.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El administrador no presiona el botón "actualizar".	4(a).- El sistema no realiza ningún cambio.

Post-Condiciones: El sistema actualiza y guarda la información en la base de datos, se mantiene la sesión del administrador conectada.

8.3.3.19 Caso de Uso: < Visualizar acerca de>

Actores: Administrador, Usuario.

Descripción: El administrador o usuario podrá visualizar la información ingresada anteriormente.

Pre-Condiciones: El administrador o usuario debe haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "acerca de" en la barra de menú.	2.- El sistema muestra la información correspondiente al acerca de, del sistema al usuario. El administrador además tendrá un botón actualizar.

Post-Condiciones: El sistema no presenta cambios.

8.3.3.20 Caso de Uso: < Solicitar recomendación >

Actores: Administrador, Usuario.

Descripción: Luego de responder el cuestionario el administrador o usuario solicita una recomendación del grupo de metodologías.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad, debe de responder el cuestionario.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "recomendar".	2.- El sistema muestra un gráfico que representa los porcentajes en que sería recomendable un grupo de metodologías en base a las respuestas ingresadas, en caso de descartar un grupo se agrega una notificación y se aparta el grupo informando porque ha sido descartado.

Flujo de Eventos Alternativo:

Al actor	El sistema
1(a).- El administrador o usuario no presiona el botón "recomendar".	2(a).- El sistema no realiza ninguna acción.

Post-Condiciones: El sistema muestra recomendación, se mantiene la sesión del administrador o usuario conectada.

8.3.3.21 Caso de Uso: < Visualizar recomendación >

Actores: Administrador, Usuario.

Descripción: Luego de solicitar recomendación el administrador o usuario visualizarán recomendación del grupo de metodologías.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad, debe de haber solicitado recomendación.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "solicitar recomendación".	2.- El sistema muestra la información correspondiente a la recomendación de metodologías.

Post-Condiciones: El sistema no presenta cambios.

8.3.3.22 Caso de Uso: <Ingresar sugerencias>

Actores: Usuario.

Descripción: El usuario ingresa una sugerencia al sistema.

Pre-Condiciones: El usuario debe haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el usuario selecciona el botón de ingresar sugerencias.	2.- El sistema solicitará ingresar datos relacionados a la sugerencia (nombre, sugerencia).
3.- El usuario ingresa la información solicitada.	4.- El sistema muestra la información ingresada y da la opción de "enviar"(guardar).
5.- El usuario selecciona enviar.	6.- El sistema guarda la información ingresada en la base de datos, muestra una vista donde se encuentra la información ingresada por el usuario.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El usuario no ingresa datos solicitados.	4(a).- El sistema muestra los campos sin datos ingresados y no realiza cambios.
5(a).- El usuario selecciona la opción "enviar".	6(a).- El sistema mostrará una alerta donde se resaltan los campos obligatorios que se encuentran vacios.

Post-Condiciones: El sistema guarda la información ingresada en la base de datos, luego muestra una vista donde el usuario podrá visualizar la información ingresada.

8.3.3.23 Caso de Uso: < Visualizar sugerencias >

Actores: Administrador, Usuario.

Descripción: Luego de ingresar una sugerencia el usuario visualizará la información ingresada, el administrador podrá ver todas las sugerencias ingresadas por los distintos usuarios.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "sugerencias".	2.- El sistema muestra la información correspondiente a la sugerencia ingresada por el usuario luego que este la envía. El administrador podrá visualizar todas las sugerencias ingresas en el sistema.

Post-Condiciones: El sistema no presenta cambios.

8.3.3.24 Caso de Uso: <Ingresar calificación>

Actores: Usuario.

Descripción: El usuario ingresa una calificación al sistema.

Pre-Condiciones: El usuario debe haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- Este caso de uso comienza cuando el usuario selecciona el botón de calificar.	2.- El sistema mostrará la información de la o las recomendaciones realizadas para su calificación.
3.- El usuario ingresa la información calificación.	4.- El sistema muestra la información ingresada y da la opción de guardar.
5.- El usuario selecciona guardar.	6.- El sistema guarda la información ingresada en la base de datos, muestra una vista donde se encuentra la información ingresada por el usuario.

Flujo de Eventos Alternativo:

Al actor	El sistema
3(a).- El usuario no ingresa datos de calificación.	4(a).- El sistema muestra el campo sin datos ingresados y no realiza cambios.
5(a).- El usuario selecciona la opción "guardar".	6(a).- El sistema guardara de todos modos aun que este calificado.

Post-Condiciones: El sistema guarda la información ingresada en la base de datos, luego muestra una vista donde el usuario podrá visualizar la información ingresada.

8.3.3.25 Caso de Uso: < Visualizar calificación >

Actores: Administrador, Usuario.

Descripción: el usuario luego de calificar podrá visualizar sus calificaciones ingresadas al sistema, el administrador podrá visualizar las calificaciones de todos los usuarios.

Pre-Condiciones: El administrador o usuario debe de haber iniciado sesión con anterioridad.

Flujo de Eventos Básicos:

Al actor	El sistema
1.- El administrador o usuario presiona el botón "calificar".	2.- El sistema muestra al usuario las calificaciones que este ha realizado. El sistema muestra al administrador todas las calificaciones ingresadas

Post-Condiciones: El sistema no presenta cambios.

8.4 Modelamiento de datos

El siguiente modelo presenta las características necesarias que debe tener la base de datos para el sistema, donde se ha dividido en dos áreas importantes y unas extras que ayudan a complementar el sistema.

Entre las áreas tenemos la de usuario y registro, área de cuestionario y recomendación, en el primero guardamos información básica para mantener el registro de las actividades del usuario, en la entidad de registro almacenaremos las respuestas que haya ingresado el usuario donde podrá calificarla recomendación, esta información nos servirá como punto de retroalimentación.

En el área de cuestionario y recomendación utilizamos las tablas que almacenan las preguntas y respuestas definidas en el sistema, donde cada una de estas posibles respuestas estará asociada a una o más posibles grupos de metodologías, a las cuales nuestro usuario podrá acceder para ver información de dichos grupos, así como también información de las metodologías asociadas al grupo anteriormente mencionado. Para complementar la información del sistema agregaremos información de las prácticas ágiles la que se representa de la misma forma que las metodologías, además se agregó un ítem de sugerencias para el sistema.

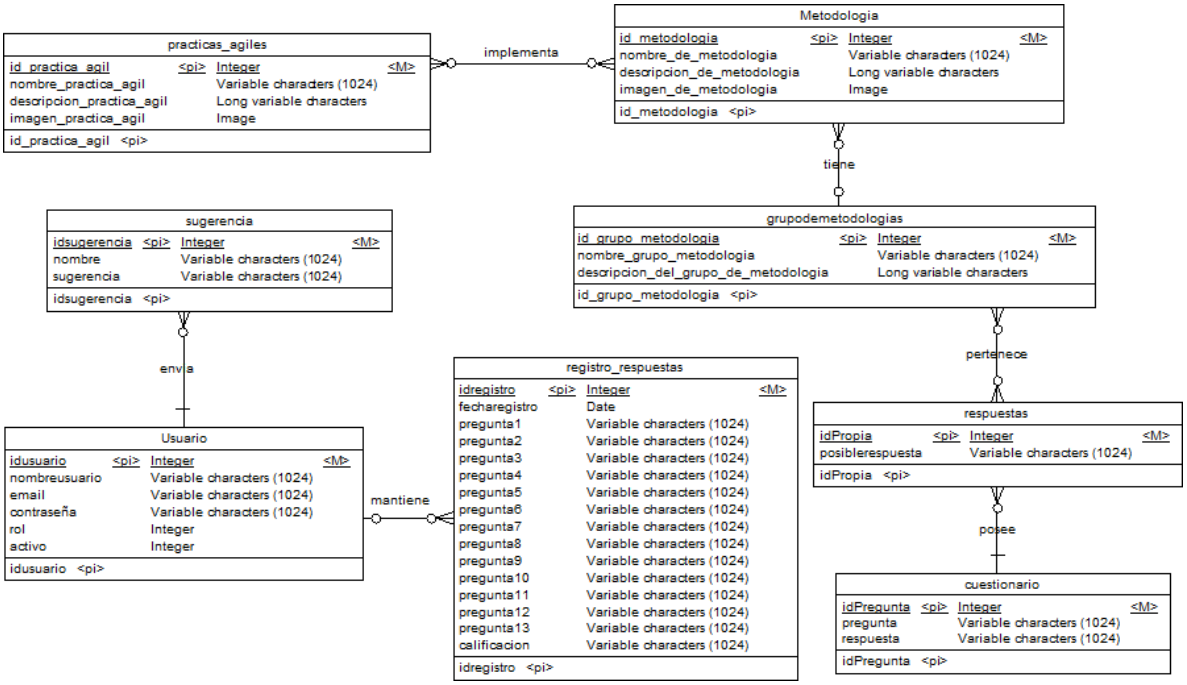


FIGURA N° 30: Modelo entidad - relación

8.4.1 Detalle modelado de datos

cuestionario			
idPregunta	<pi>	Integer	<M>
pregunta		Variable characters (1024)	
respuesta		Variable characters (1024)	
idPregunta <pi>			

FIGURA N° 31: Detalle MER - cuestionario

Atributos:

- idPregunta: identificador de la pregunta
- pregunta: almacena la pregunta realizada al usuario
- respuesta: almacena la respuesta seleccionada por el usuario

respuestas			
idPropia	<pi>	Integer	<M>
posiblerespuesta		Variable characters (1024)	
idPropia <pi>			

FIGURA N° 32: Detalle MER - respuestas

Atributos:

- idPropia: identificador de la respuesta
- posiblerespuesta: son las posibles respuestas que puede tener determinada pregunta

grupodemetodologias			
id_grupo_metodologia	<pi>	Integer	<M>
nombre_grupo_metodologia		Variable characters (1024)	
descripcion_del_grupo_de_metodologia		Long variable characters	
id_grupo_metodologia <pi>			

FIGURA N° 33: Detalle MER - grupo metodologías

Atributos:

- id_grupo_metodologia: identificador del grupo de metodologías
- nombre_grupo_metodologia: nombre del grupo de metodología
- descripcion_del_grupo_de_metodologia: breve descripción con información general de los grupos de metodologías

Metodologia			
id_metodologia	<pi>	Integer	<M>
nombre_de_metodologia		Variable characters (1024)	
descripcion_de_metodologia		Long variable characters	
imagen_de_metodologia		Image	
id_metodologia <pi>			

FIGURA N° 34: Detalle MER - Metodología

Atributos:

- id_metodologia: identificador de las metodologías
- nombre_de_metodologia: nombre de las metodologías
- descripcion_de_metodologia: información descriptiva de la metodología
- imagen_de_metodologia: imagen representativa de la metodología

practicas_agiles			
id_practica_agil	<pi>	Integer	<M>
nombre_practica_agil		Variable characters (1024)	
descripcion_practica_agil		Long variable characters	
imagen_practica_agil		Image	
id_practica_agil <pi>			

FIGURA N° 35: Detalle MER - practicas agiles

Atributos:

- id_practica_agil: identificador prácticas ágiles
- nombre_practica_agil: nombre de las prácticas ágiles
- descripcion_practica_agil: información descriptiva de la práctica ágil
- imagen_practica_agil: imagen representativa de la practica ágil

Usuario			
idusuario	<pi>	Integer	<M>
nombreusuario		Variable characters (1024)	
email		Variable characters (1024)	
contraseña		Variable characters (1024)	
rol		Integer	
activo		Integer	
idusuario <pi>			

FIGURA N° 36: Detalle MER - Usuario

En un inicio no se tenía contemplado tener usuarios registrados, esta tabla se agregó posteriormente durante el desarrollo al igual que la tabla registro

Atributos:

- idusuario: identificador del usuario
- nombreusuario: nombre que utilizara el usuario
- email: correo electrónico del usuario registrado
- contraseña: contraseña para iniciar sesión al sistema
- rol: indica si el usuario es administrador o usuario común del sistema
- activo: indica si el usuario está activo o no

registro_respuestas			
idregistro	<pi>	Integer	<M>
fecharegistro		Date	
pregunta1		Variable characters (1024)	
pregunta2		Variable characters (1024)	
pregunta3		Variable characters (1024)	
pregunta4		Variable characters (1024)	
pregunta5		Variable characters (1024)	
pregunta6		Variable characters (1024)	
pregunta7		Variable characters (1024)	
pregunta8		Variable characters (1024)	
pregunta9		Variable characters (1024)	
pregunta10		Variable characters (1024)	
pregunta11		Variable characters (1024)	
pregunta12		Variable characters (1024)	
pregunta13		Variable characters (1024)	
calificacion		Variable characters (1024)	
idregistro <pi>			

FIGURA N° 37: Detalle MER - registro respuestas

Atributos:

- idregistro: identificador del registro
- fecharegistro: fecha en que se responde cuestionario
- pregunta1-13: registro de respuestas del usuario a cada una de las 13 preguntas
- calificacion: calificación que puede agregar el usuario a la recomendación

registrorecomendacion			
idregistrorecomendacion	<pi>	Integer	<M>
recolineal		Float	
recoevolutivo		Float	
recoincremental		Float	
recoagil		Float	
gruposeleccionado		Variable characters (1024)	
metodologiaseleccionada		Variable characters (1024)	
idregistrorecomendacion <pi>			

FIGURA N° 38: Detalle MER - registro recomendación

Esta tabla fue agregada al final del desarrollo después de haber implementado el usuario y registro de respuestas y para evitar rehacer base de datos completa se agrego aparte.

Atributos:

- idregistrorecomendacion: identificador del registro de recomendación
- recolineal: porcentaje que fue recomendado el grupo de metodologías lineales
- recoevolutivo: porcentaje que fue recomendado el grupo de metodologías evolutivas
- recoincremental porcentaje que fue recomendado el grupo de metodologías incrementales
- recoagil: porcentaje que fue recomendado el grupo de metodologías ágiles
- gruposeleccionado: grupo de metodologías utilizado por el usuario
- metodologiaseleccionada: metodología utilizada por el usuario

sugerencia			
idsugerencia	<pi>	Integer	<M>
nombre		Variable characters (1024)	
sugerencia		Variable characters (1024)	
idsugerencia <pi>			

FIGURA N° 39: Detalle MER - sugerencia

Esta tabla se agregó al final del desarrollo después de haber implementado el usuario y registro de respuestas.

Atributos:

- idsugerencia: identificador de las sugerencias
- nombre: nombre de usuario que envía sugerencia esto es opcional
- sugerencia: descripción de la sugerencia

9 DISEÑO

9.1 Diseño Físico de la Base de datos

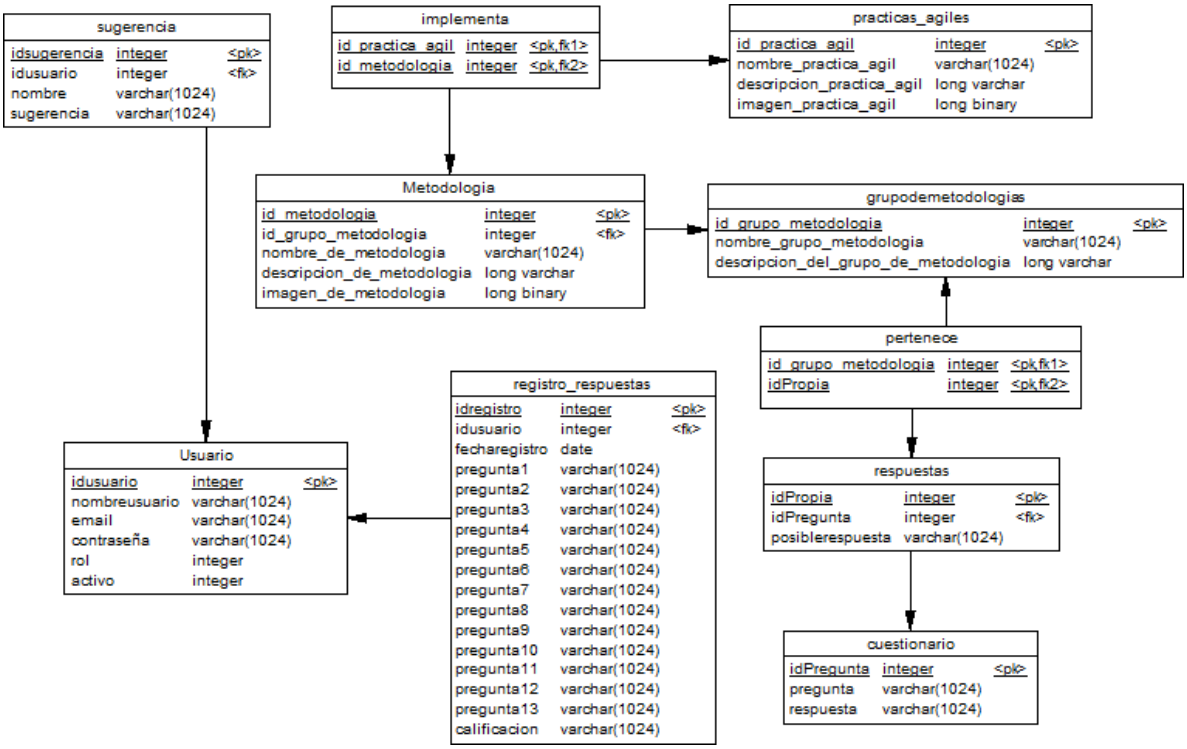


FIGURA N° 40: Modelo relacional

9.1.1 Detalle diseño físico de base de datos

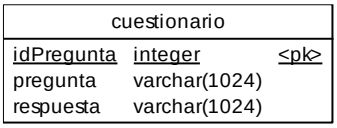


FIGURA N° 41: Detalle MR - cuestionario

Atributos:

- idPregunta: identificador de la pregunta
- pregunta: almacena la pregunta realizada al usuario
- respuesta: almacena la respuesta seleccionada por el usuario

respuestas		
idPropia	integer	<pk>
idPregunta	integer	<fk>
posiblerespuesta	varchar(1024)	

FIGURA N° 42: Detalle MR - respuestas

Atributos:

- idPropia: identificador de la respuesta
- idPregunta: clave foránea que asocia la respuesta a una pregunta
- posiblerespuesta: son las posibles respuestas que puede tener determinada pregunta
-

grupodemetodologias		
id_grupo_metodologia	integer	<pk>
nombre_grupo_metodologia	varchar(1024)	
descripcion_del_grupo_de_metodologia	long varchar	

FIGURA N° 43: Detalle MR - grupo de metodologías

Atributos:

- id_grupo_metodologia: identificador del grupo de metodologías
- nombre_grupo_metodologia: nombre del grupo de metodología
- descripcion_del_grupo_de_metodologia: breve descripción con información general de los grupos de metodologías
-

pertenece		
id_grupo_metodologia	integer	<pk,fk1>
idPropia	integer	<pk,fk2>

FIGURA N° 44: Detalle MR - pertenece

Esta tabla es una relación entre los grupos de metodologías y las posibles respuestas de cada pregunta, donde cada posible respuesta pertenece a un grupo de metodologías

Atributos:

- id_grupo_metodologia: identificador del grupo de metodologías
- idPropia: identificador de la respuesta

Metodologia		
id_metodologia	integer	<pk>
id_grupo_metodologia	integer	<fk>
nombre_de_metodologia	varchar(1024)	
descripcion_de_metodologia	long varchar	
imagen_de_metodologia	long binary	

FIGURA N° 45: Detalle MR - Metodologia

Atributos:

- id_metodologia: identificador de las metodologías
- id_grupo_meodologia: clave foránea que asocia cada metodología a un grupo de metodologías
- nombre_de_metodologia: nombre de las metodologías
- descripcion_de_metodologia: información descriptiva de la metodología
- imagen_de_metodologia: imagen representativa de la metodología

implementa		
id_practica_agil	integer	<pk,fk1>
id_metodologia	integer	<pk,fk2>

FIGURA N° 46: Detalle MR - Implementa

Esta tabla de relación representa las prácticas ágiles asociadas a las metodologías

- esta tabla no fue implementada en el sistema ya que se decidió dejar las prácticas ágiles como un módulo aparte de las metodologías.
-

practicas_agiles		
id_practica_agil	integer	<pk>
nombre_practica_agil	varchar(1024)	
descripcion_practica_agil	long varchar	
imagen_practica_agil	long binary	

FIGURA N° 47: Detalle MR - practicas agiles

Atributos:

- id_practica_agil: identificador prácticas ágiles
- nombre_practica_agil: nombre de las prácticas ágiles
- descripcion_practica_agil: información descriptiva de la práctica ágil
- imagen_practica_agil: imagen representativa de la practica ágil

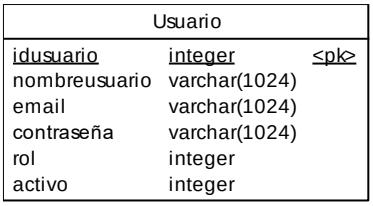


FIGURA N° 48: Detalle MR - Usuario

En un inicio no se tenía contemplado tener usuarios registrados, esta tabla se agregó posteriormente durante el desarrollo al igual que la tabla registro

Atributos:

- idusuario: identificador del usuario
- nombreusuario: nombre que utilizara el usuario
- email: correo electrónico del usuario registrado
- contraseña: contraseña para iniciar sesión al sistema
- rol: indica si el usuario es administrador o usuario común del sistema
- activo: indica si el usuario está activo o no

registro_respuestas		
idregistro	integer	<pk>
idusuario	integer	<fk1>
idregistrorecomendacion	integer	<fk2>
fecharegistro	date	
pregunta1	varchar(1024)	
pregunta2	varchar(1024)	
pregunta3	varchar(1024)	
pregunta4	varchar(1024)	
pregunta5	varchar(1024)	
pregunta6	varchar(1024)	
pregunta7	varchar(1024)	
pregunta8	varchar(1024)	
pregunta9	varchar(1024)	
pregunta10	varchar(1024)	
pregunta11	varchar(1024)	
pregunta12	varchar(1024)	
pregunta13	varchar(1024)	
calificacion	varchar(1024)	

FIGURA N° 49: Detalle MR - registro respuestas

Atributos:

- idregistro: identificador del registro
- idusuario: clave foránea que asocia el registro al usuario
- idregistrorecomendacion: clave foránea que asocia el registro de respuestas al registro de la recomendación
- fecharegistro: fecha en que se responde cuestionario
- pregunta1-13: registro de respuestas del usuario a cada una de las 13 preguntas
- calificacion: calificación que puede agregar el usuario a la recomendación

registrorecomendacion		
idregistrorecomendacion	integer	<pk>
idregistro	integer	<fk>
recolineal	float	
recoevolutivo	float	
recoincremental	float	
recoagil	float	
gruposeleccionado	varchar(1024)	
metodologiaseleccionada	varchar(1024)	

FIGURA N° 50: Detalle MR - registro recomendación

Esta tabla fue agregada al final del desarrollo después de haber implementado el usuario y registro de respuestas y para evitar rehacer base de datos completa se agregó aparte.

Atributos:

- idregistrorecomendacion: identificador del registro de recomendación
- idregistro: clave foránea que asocia el registro de la recomendación con el registro de las respuestas
- recolineal: porcentaje que fue recomendado el grupo de metodologías lineales
- recoevolutivo: porcentaje que fue recomendado el grupo de metodologías evolutivas
- recoincremental porcentaje que fue recomendado el grupo de metodologías incrementales
- recoagil: porcentaje que fue recomendado el grupo de metodologías ágiles
- gruposeleccionado: grupo de metodologías utilizado por el usuario
- metodologiaseleccionada: metodología utilizada por el usuario

sugerencia		
idsugerencia	integer	<pk>
idusuario	integer	<fk>
nombre	varchar(1024)	
sugerencia	varchar(1024)	

FIGURA N° 51: Detalle MR - sugerencia

Esta tabla se agregó al final del desarrollo después de haber implementado el usuario y registro de respuestas.

- En el sistema no se agregó la clave foránea ya que se decidió dejar las sugerencias de forma anónima y el nombre que ingresa el usuario es opcional

Atributos:

- idsugerencia: identificador de las sugerencias
- nombre: nombre de usuario que envía sugerencia esto es opcional
- sugerencia: descripción de la sugerencia

9.2 Diseño de arquitectura funcional

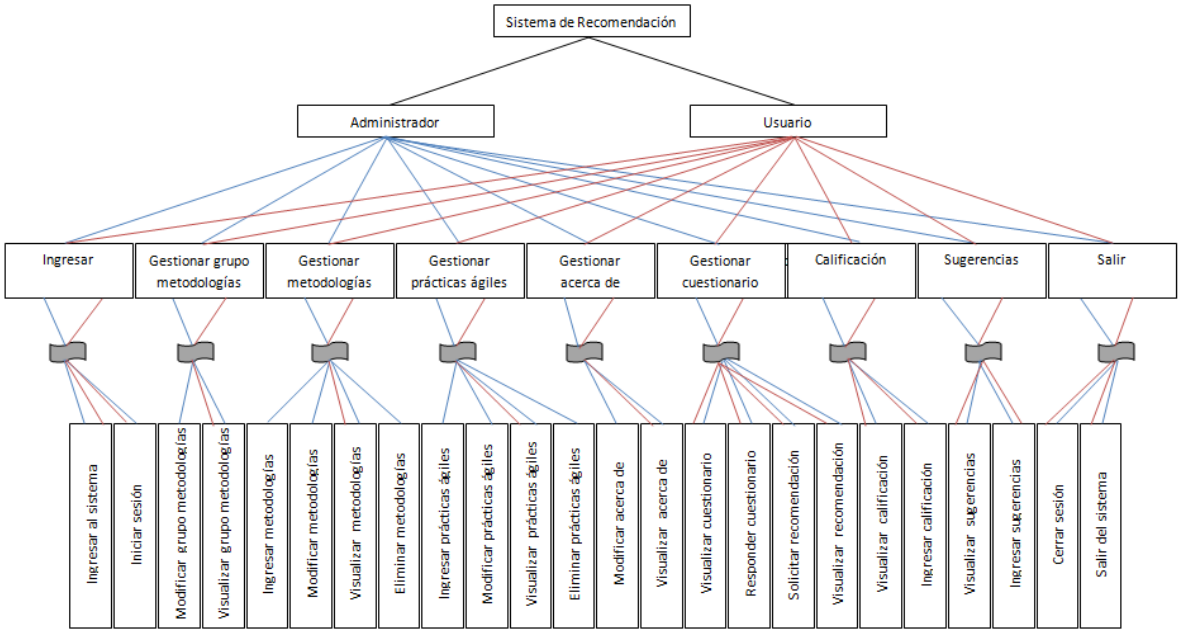


FIGURA N° 52: Diseño arquitectónico

9.3 Diseño interfaz y navegación

Pantalla principal

Como prototipo de interfaz del diseño y navegación, presentaremos la vista inicial que tendrá el usuario y el administrador en el sistema web de recomendación de metodologías de desarrollo de software.

En esta imagen podemos apreciar un logo que será definido y presentado en el producto final, además contará con una barra de menú que muestra las distintas opciones que se podrán visualizar si no se encuentra logueado, cuenta con un mensaje de bienvenida al Sistema de Recomendación de Metodología de Desarrollo de Software.

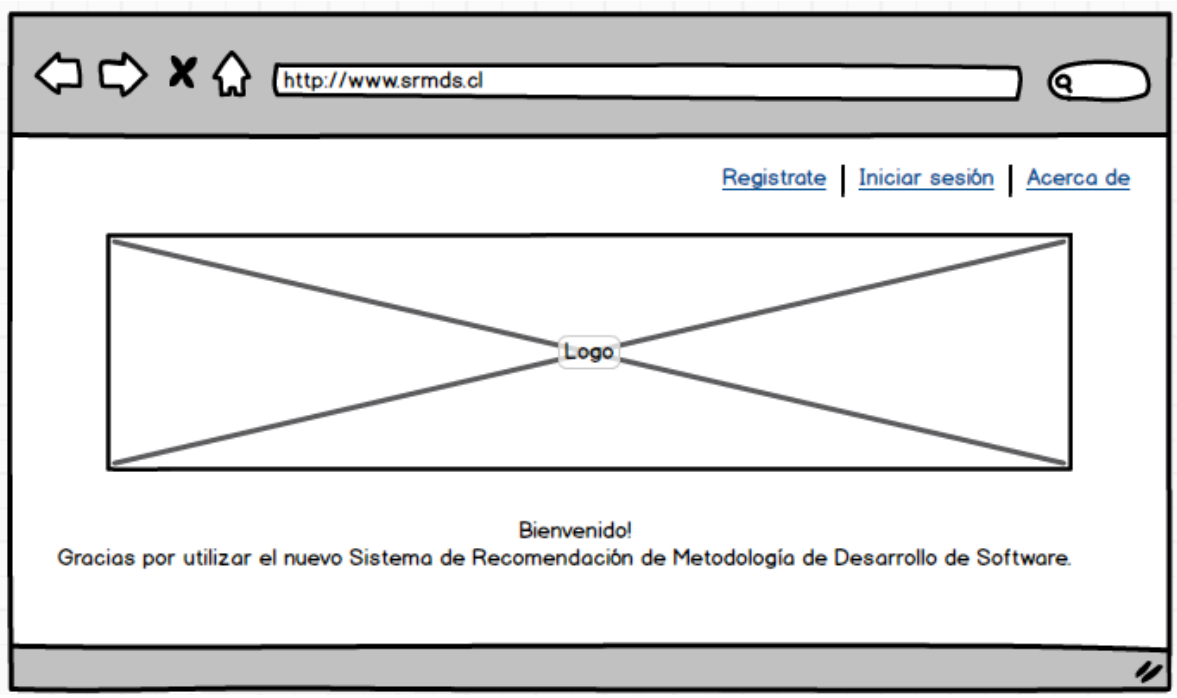


FIGURA N° 53: Página inicio

Barra de Menú

Regístrate

Los futuros usuarios que no cuenten con registro podrán hacerlo mediante la opción Regístrate, la cual solicitara nombre del usuario, correo, contraseña, confirmar contraseña.

Si se ingresa un nombre o correo ya registrado en la base de datos el sistema emitirá una alerta donde indicara que le nombre de usuario o el correo ya están registrados en el sistema, por lo que se deberá intentar el registro con otros datos.

Si confirmar contraseña no coincide con la ingresa anteriormente el sistema emitirá otra alerta donde indicara que las contraseñas no inciden.

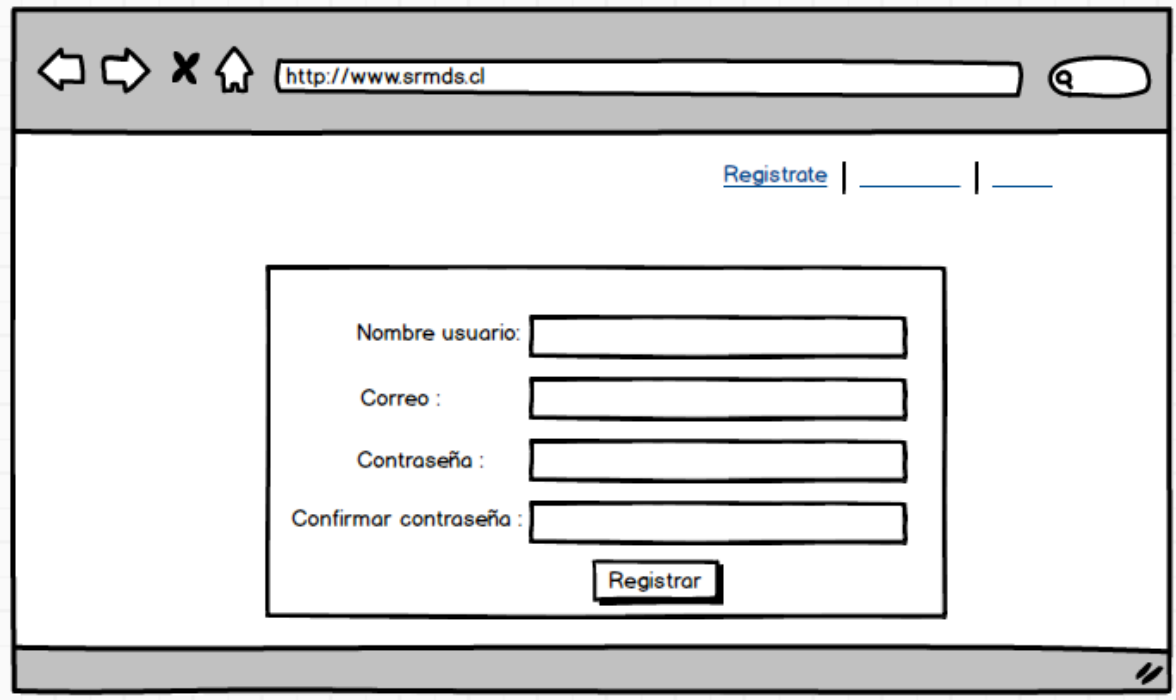


FIGURA N° 54: Registro usuario

Iniciar Sesión

Si se encuentra registrado en el sistema puede iniciar sesión como su usuario y contraseña, el sistema reconocerá su perfil automáticamente (Usuario común o Administrador).

FIGURA N° 55: Inicio de sesión

Acerca de

En la siguiente imagen podemos apreciar que en la barra de menú se selecciona la opción de "Acerca de" la cual mostrará información relevante sobre el sistema.

Esta información podrá ser vista también desde la una cuenta de usuario logueado.

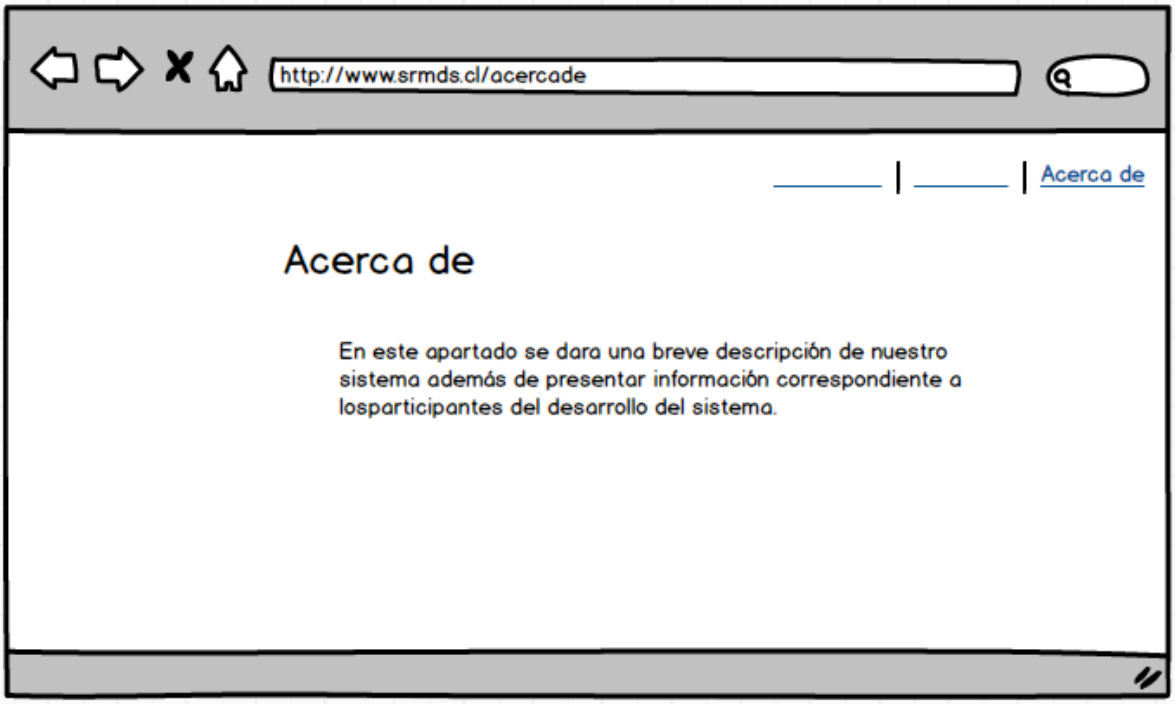


FIGURA N° 56: Acerca de

Opciones del Usuario - Administrador (logueado)

Una vez logueado el sistema mostrara la siguiente pantalla con las opciones que tendrán disponibles el usuario y administrador

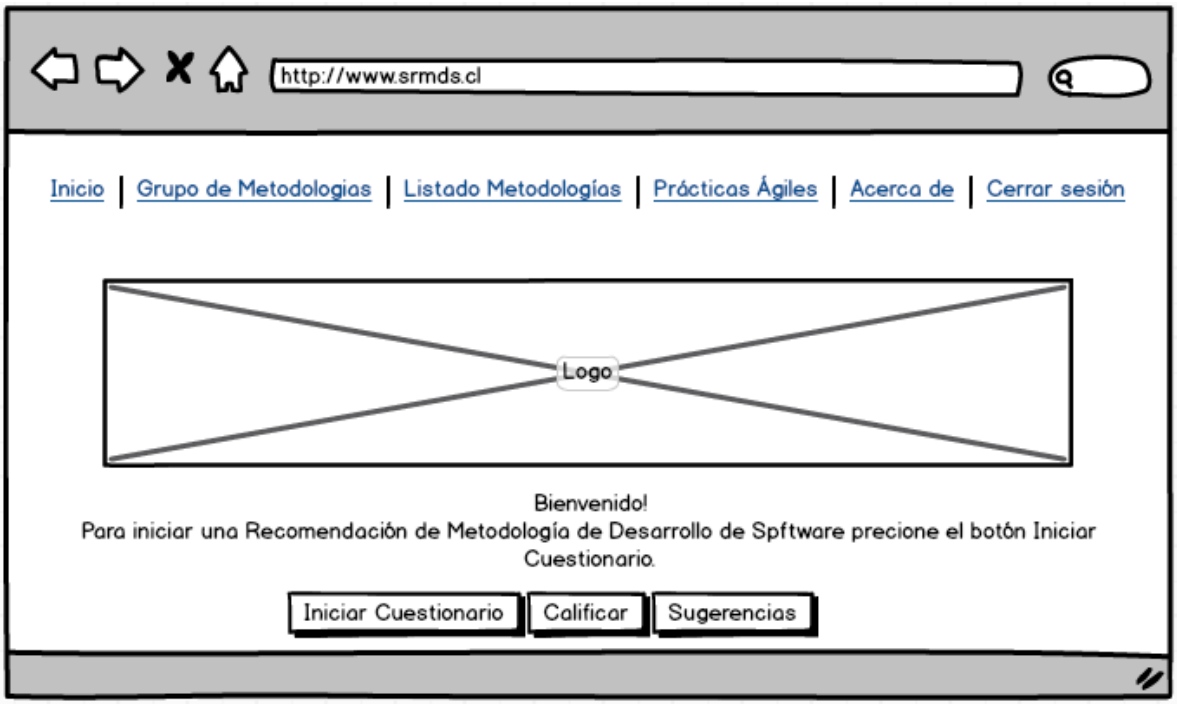


FIGURA N° 57: Página inicio usuario logeado

Iniciar Cuestionario

A continuación como se presenta en la imagen existe un botón de "Iniciar Cuestionario" el cual dirigirá al usuario o administrador a un cuestionario que cuenta con múltiples preguntas cerradas de las cuales debe seleccionar solo una de las opciones de respuestas que se les dará, esta debe de ser la más representativa a su proyecto a realizar.

Para la posterior realizar la generación de la recomendación de la metodología de desarrollo de software debe presionar el botón de "Generar Recomendación" el cual le mostrara resultados gráficos y opciones a entrar de forma directa al grupo de metodología(la navegación del grupo de metodología y metodología perteneciente al grupo serán detalladas más adelante).

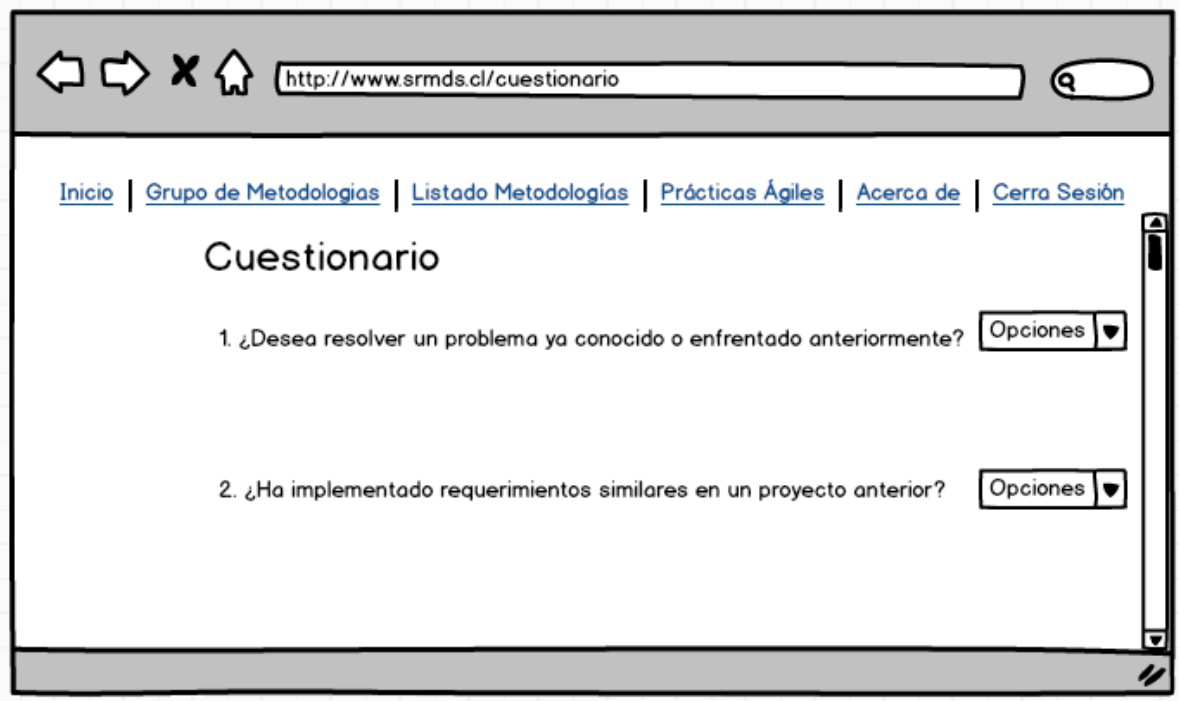


FIGURA N° 58: Cuestionario 1 de 2

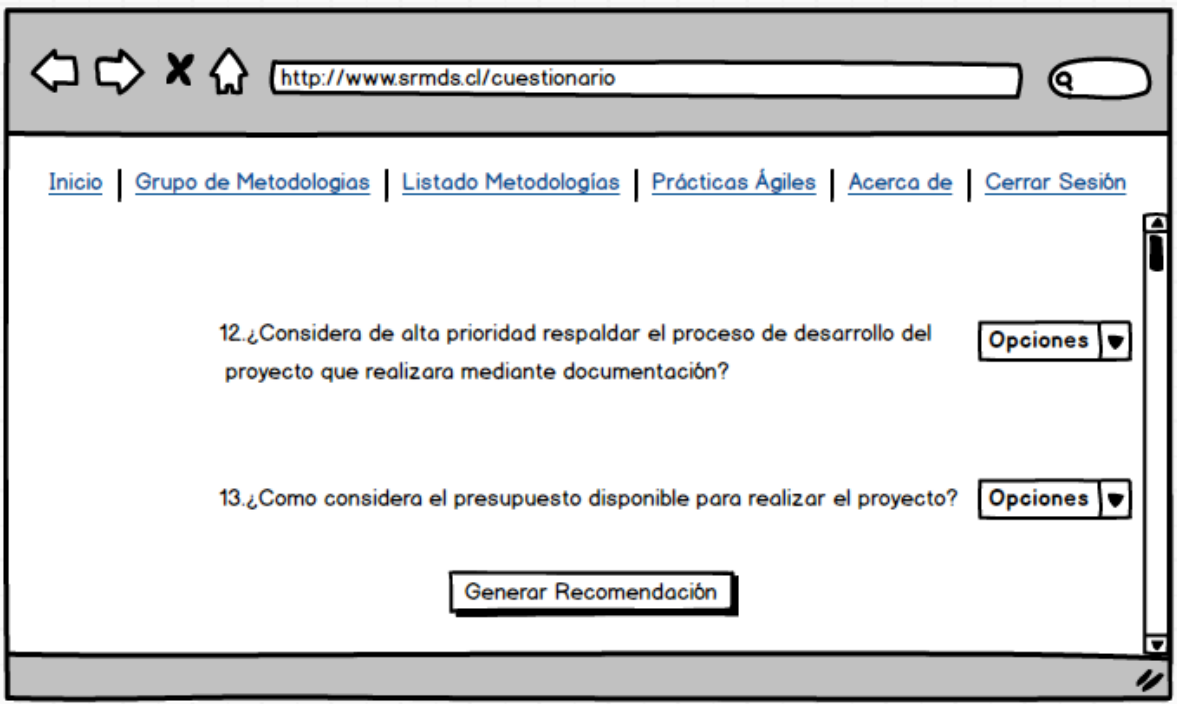


FIGURA N° 59: Cuestionario 2 de 2

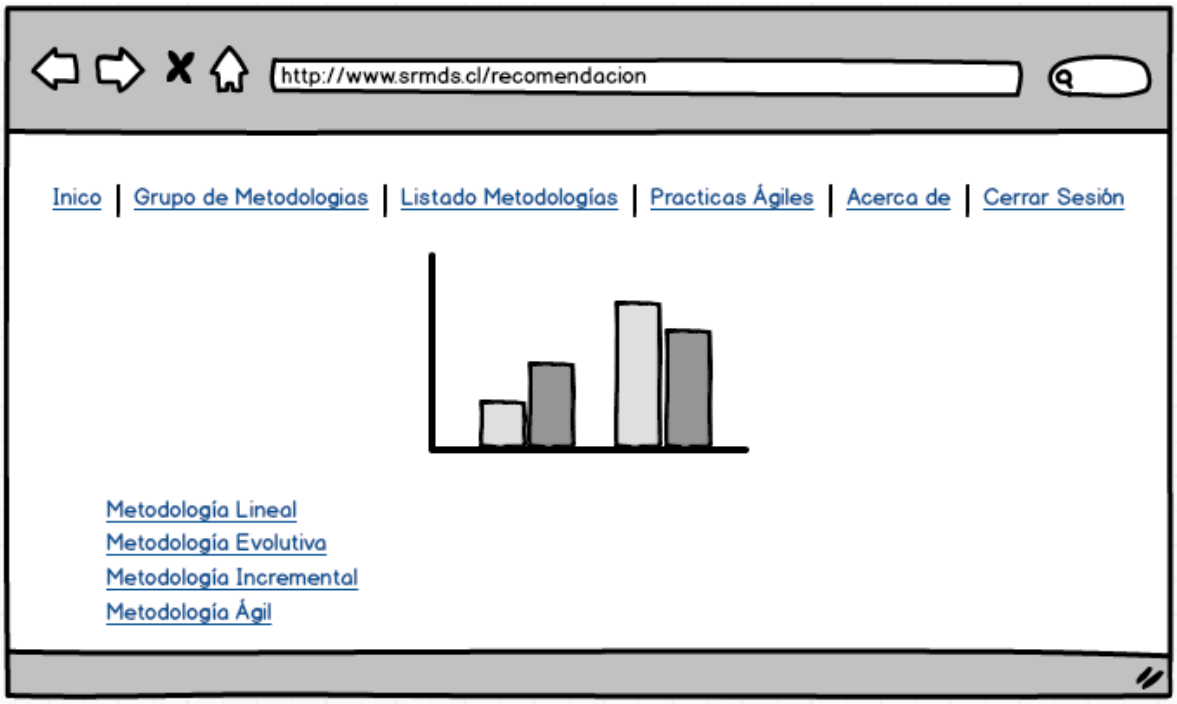


FIGURA N°60: Recomendación

Opciones de Usuario

Calificar

En la siguiente imagen podemos apreciar que el usuario presiona el botón calificar, el sistema le mostrará una nueva página donde podrá ver todas las recomendación por fecha que ha solicitador en el sistema, al entrar más en detalle podrá ver las preguntas con sus respectivas respuesta ingresadas, además un espacio donde podrá calificar la recomendación con opciones ya definidas.

FIGURA N°61: Vista calificación

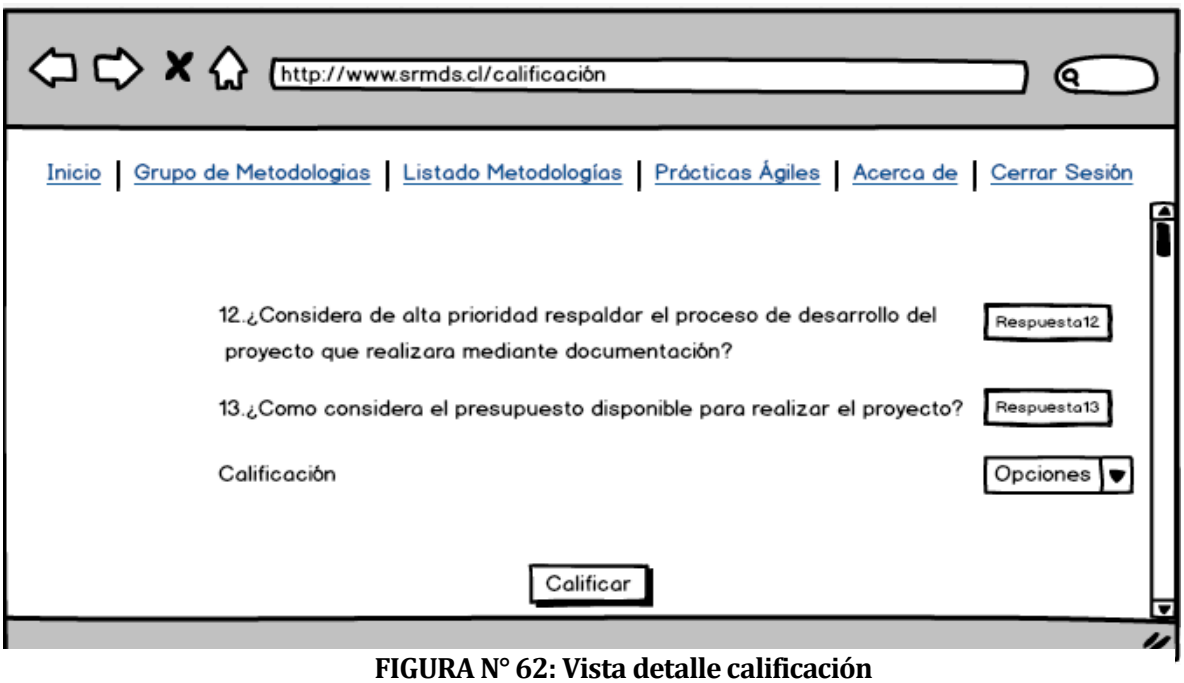


FIGURA N° 62: Vista detalle calificación

Sugerencias

En la siguiente imagen podemos apreciar que el usuario presiona el botón sugerencias, el sistema desplegara una nueva página donde el usuario podrá ingresar su nombre junto a la sugerencia, para finalizar el usuario presiona el botón enviar, el sistema le mostrara una vista que la sugerencia ha sido enviada junto a la información que ingreso.

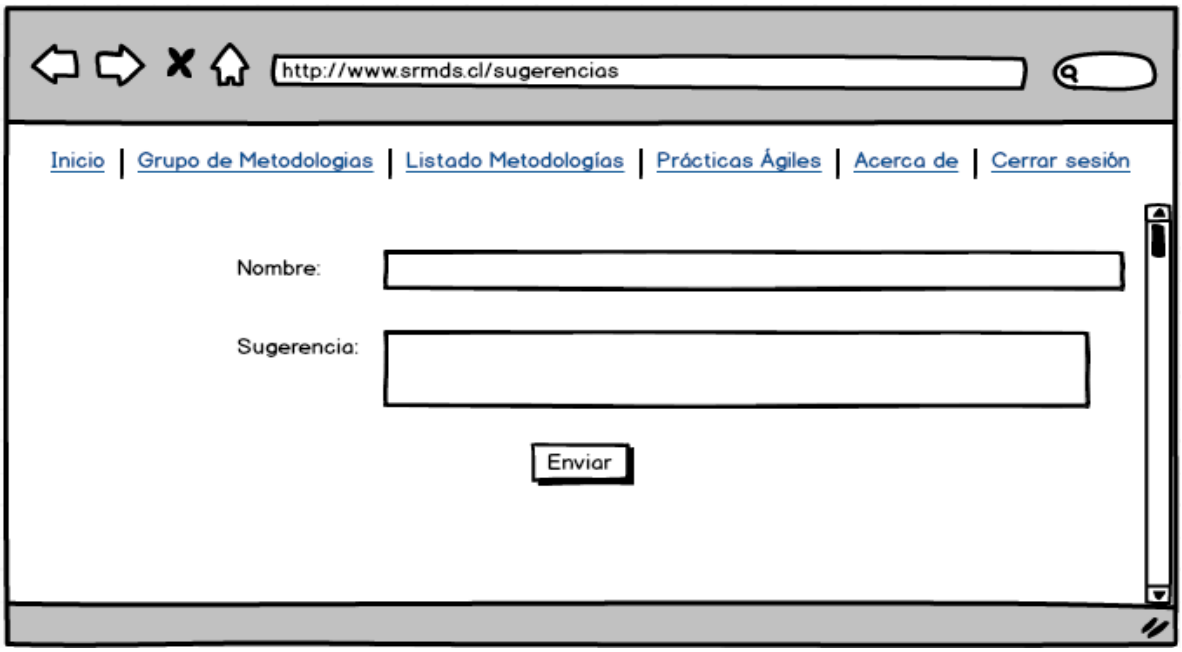


FIGURA N°63: Ingreso sugerencias

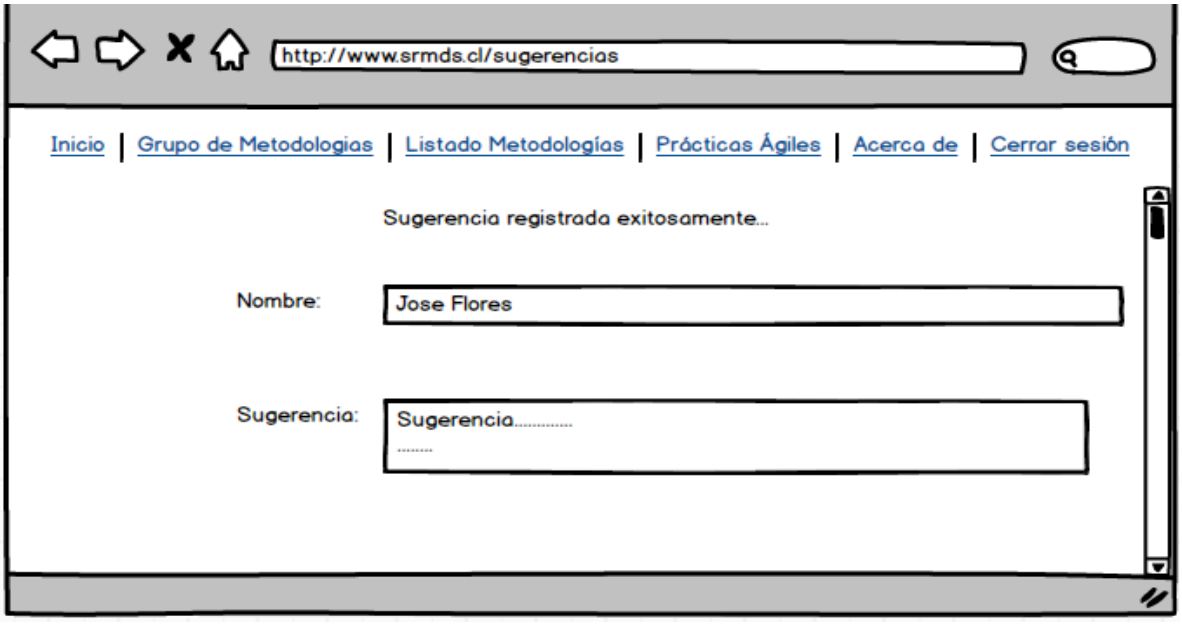


FIGURA N° 64: Vista sugerencia

Grupo de metodologías

En la siguiente imagen podemos apreciar que el usuario está en la barra de menú y selecciona la opción de "Grupo de metodologías", la cual mostrara los diferentes grupos de metodologías que podrá seleccionar.

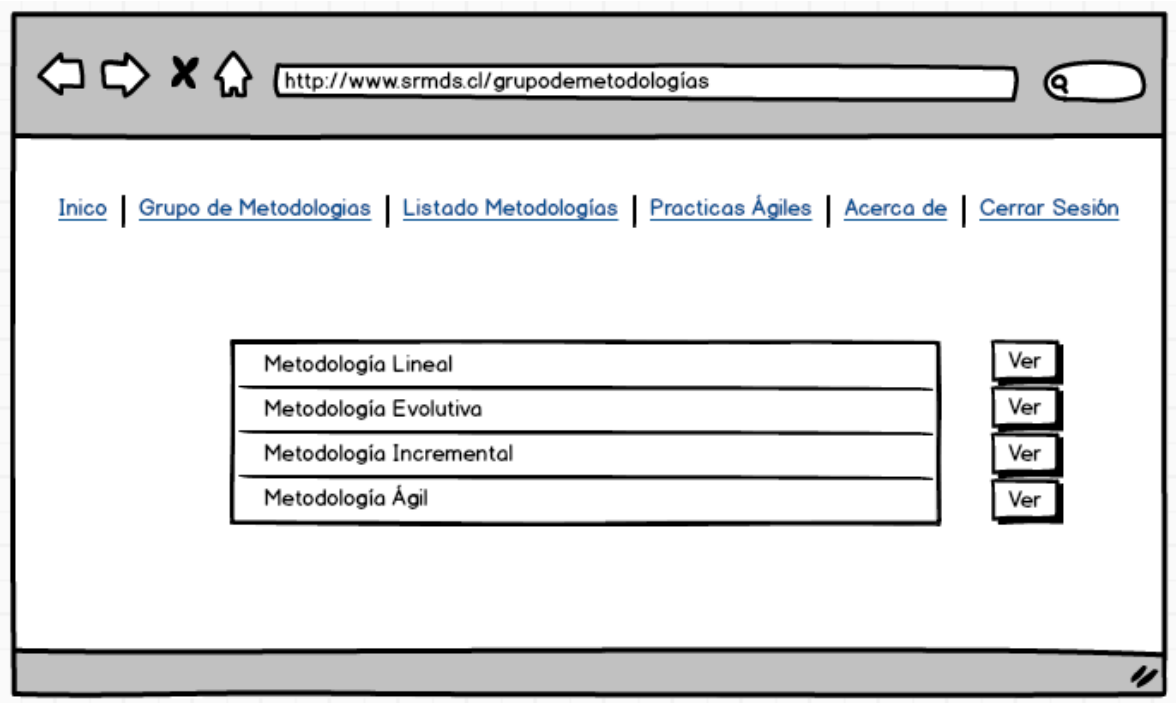


FIGURA N° 60: Vista grupo de metodologías

A modo de ejemplo supondremos que el usuario selecciona la opción de "Metodología Lineal". El usuario al seleccionar la opción "Metodología Lineal" el sistema mostrará información acerca del grupo incluyendo las diferentes metodologías o modelos pertenecientes al grupo de las metodologías lineales como se muestra a continuación.

Nota: En las imágenes se presentará la navegación e interfaz para conocer una determinada metodología en detalle. El usuario puede seleccionar cualquiera de las cuatro opciones y el procesamiento siguiente será de la misma manera cambiando solamente la información de acuerdo al grupo de la metodología seleccionada.

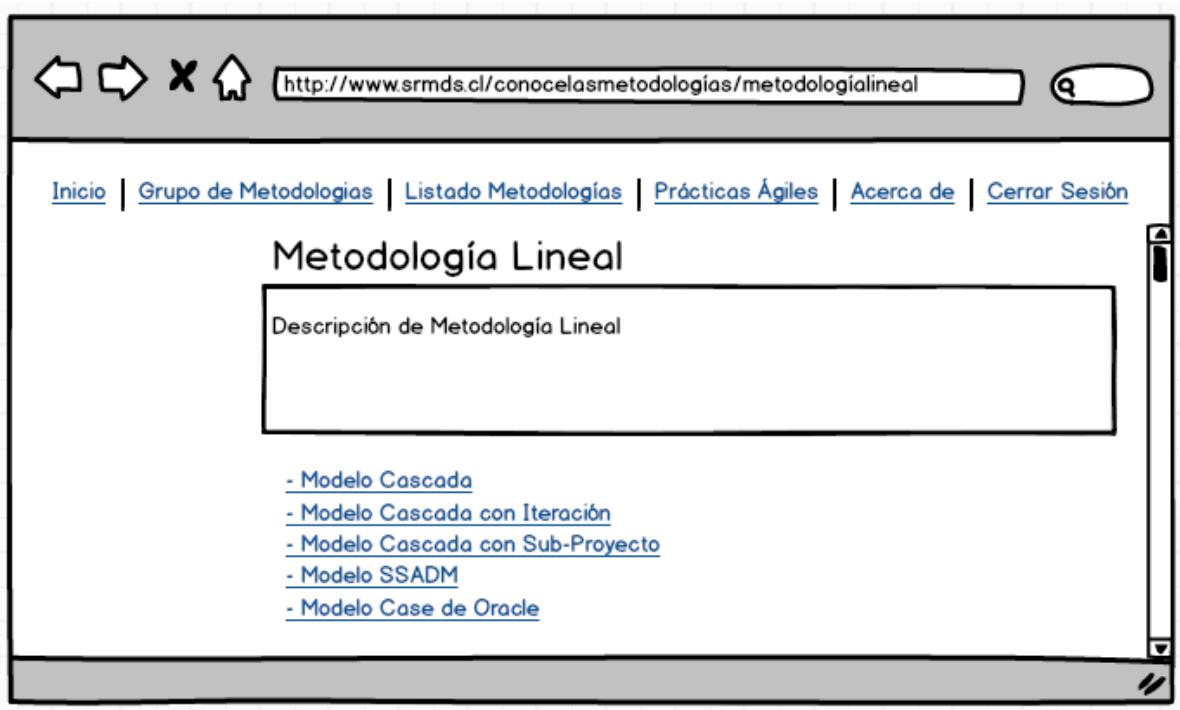


FIGURA N° 61: Vista detalle de grupo de metodología

La siguiente imagen muestra en detalle la opción de la metodología que ha seleccionado el usuario, para este ejemplo seleccionaremos "Modelo Cascada".

Nota: El usuario podrá revisar la información de un modelo específico y si desea conocer en detalle el resto de los modelos pertenecientes al grupo de la metodología seleccionada puede hacerlo utilizando el botón retroceder del navegador y seleccionando nuevamente.

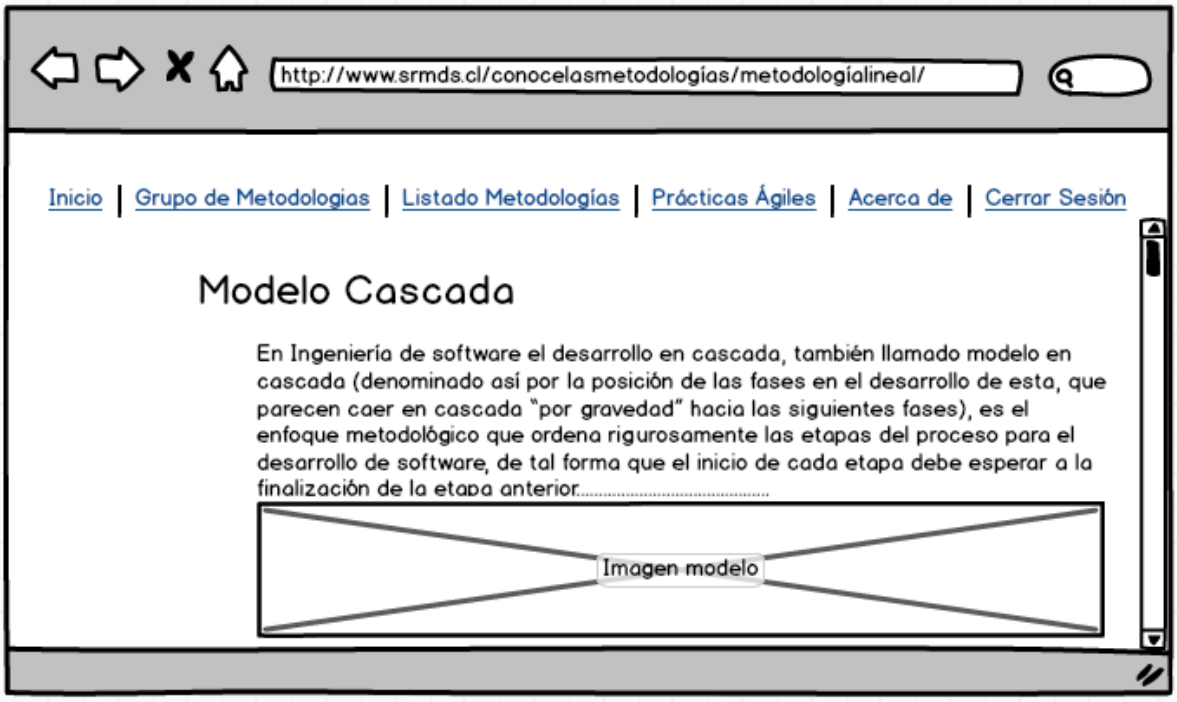


FIGURA N° 62: Vista detalle metodología

Nota: El usuario podrá conocer el resto de los grupos de metodologías repitiendo los pasos anteriores, para ello deberá volver a seleccionar un grupo de metodología del menú "Grupo de Metodologías".

Listado Metodologías

En la siguiente imagen podemos apreciar que el usuario está en la barra de menú y selecciona la opción de "Listado Metodologías", la cual mostrara las diferentes metodologías que podrá seleccionar.

FIGURA N° 63: Listado metodologías

La siguiente imagen muestra en detalle la opción de la metodología que ha seleccionado el usuario, para este ejemplo seleccionaremos "Modelo Cascada".

Nota: El usuario podrá revisar la información de un modelo específico y si desea conocer en detalle el resto de los modelos o metodologías puede hacerlo utilizando el botón retroceder del navegador y seleccionando nuevamente.

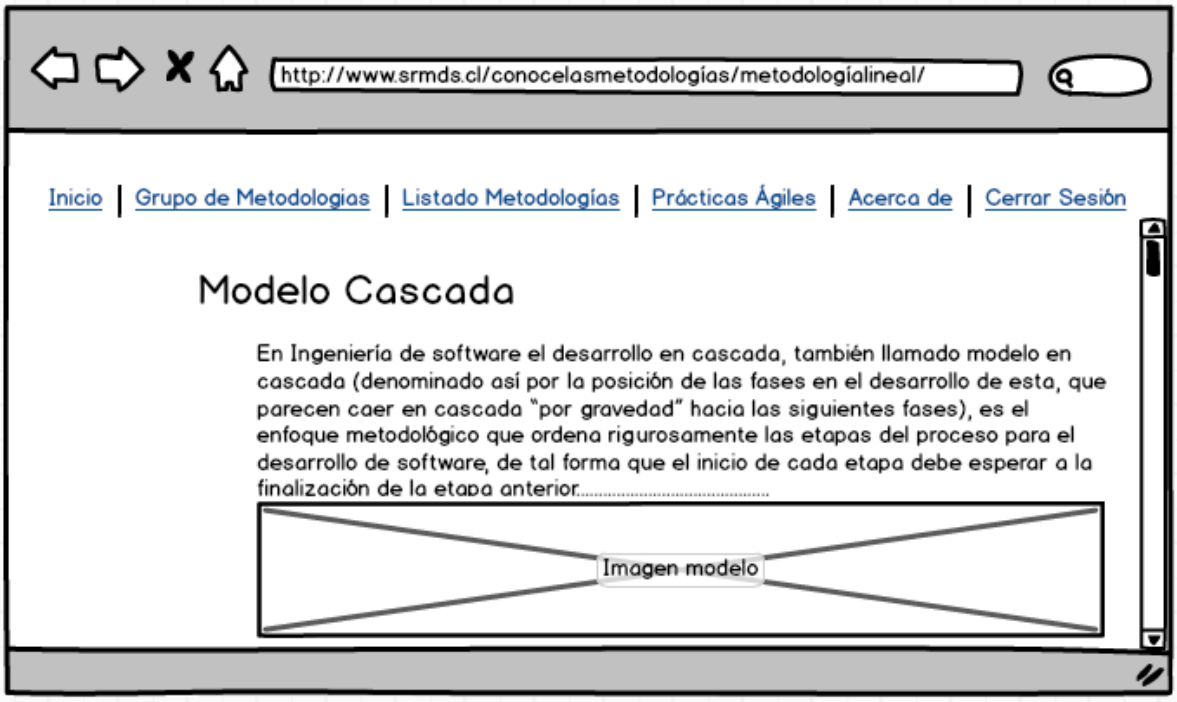


FIGURA N° 64: Vista detalle metodología

Prácticas Ágiles

En la siguiente imagen podemos apreciar que el usuario está en la barra de menú y selecciona la opción de "Prácticas Ágiles", el sistema mostrará las diferentes prácticas que podrá incorporar en sus metodologías o modelos como se muestra a continuación.

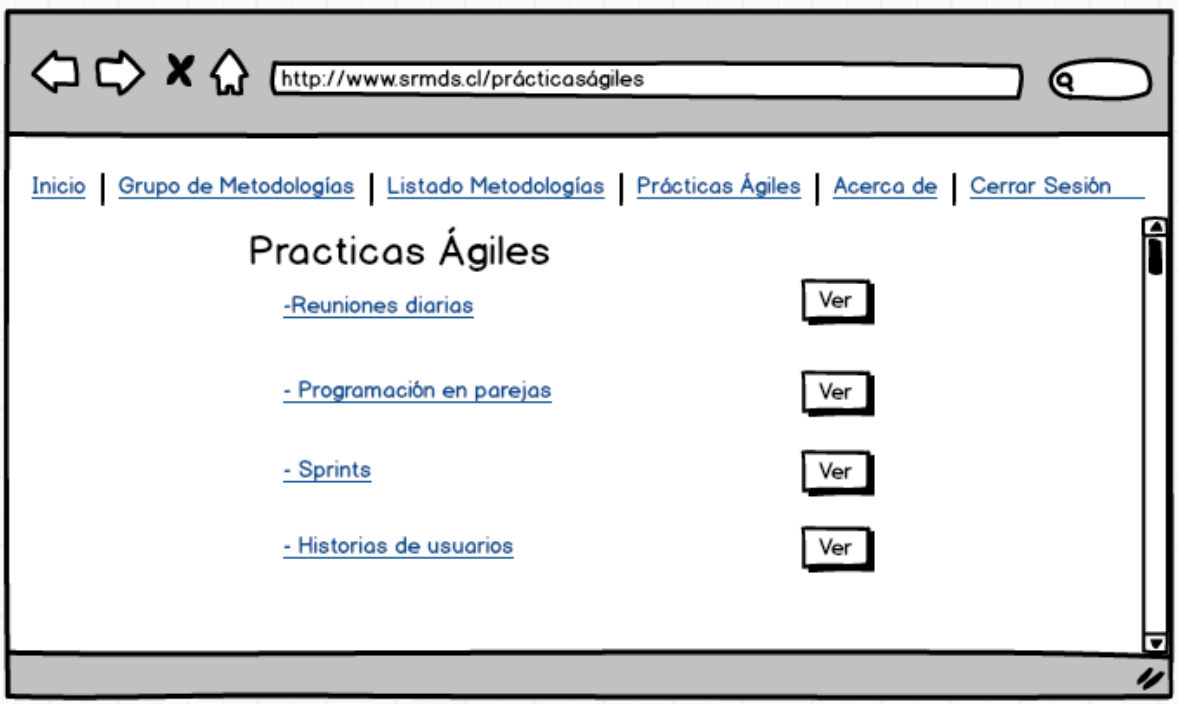


FIGURA N° 65: Listado prácticas ágiles

La siguiente imagen muestra en detalle la opción de la práctica ágil que ha seleccionado el usuario, para este ejemplo seleccionaremos "Reuniones diarias".

Nota: El usuario podrá revisar la información de una práctica específica y si desea conocer en detalle el resto de las prácticas puede hacerlo utilizando el botón retroceder del navegador y seleccionando nuevamente.

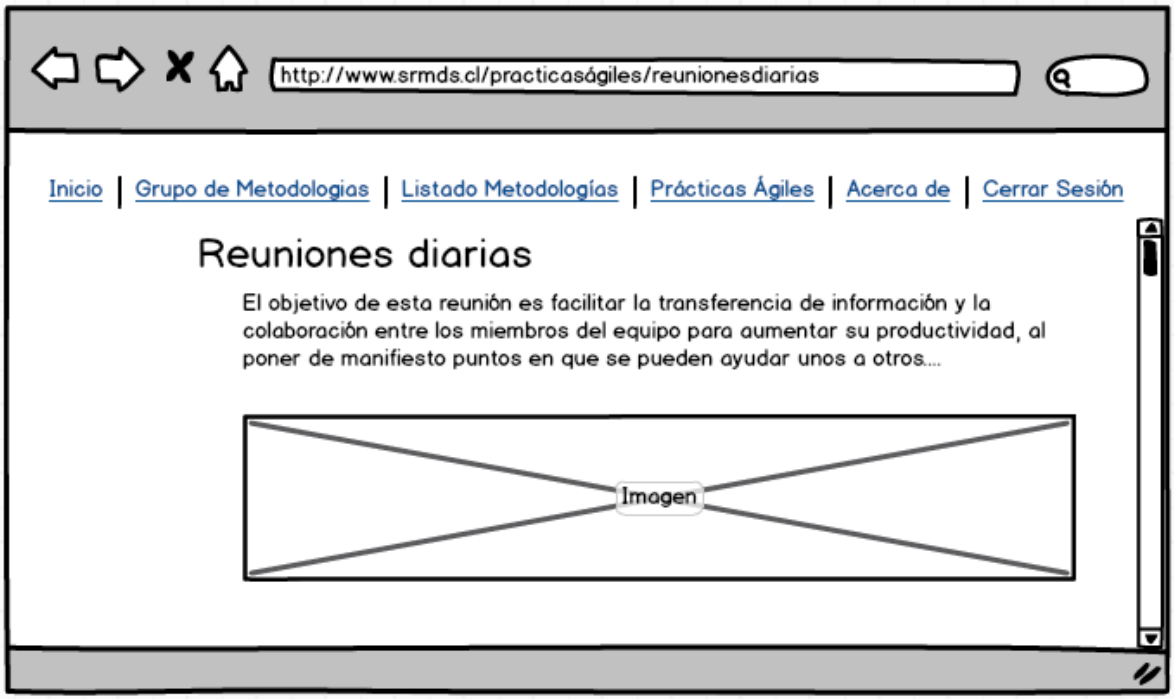


FIGURA N° 66: Detalle práctica ágil

Opciones de Administrador

Calificar

En la siguiente imagen podemos apreciar que el administrador presiona el botón calificar, el sistema le mostrará una nueva página donde podrá ver todas las calificaciones de los usuarios por fecha que se han ingresado en el sistema, al entrar más en detalle podrá ver las preguntas con sus respectivas respuesta ingresadas.

FIGURA N°72: Vista calificaciones

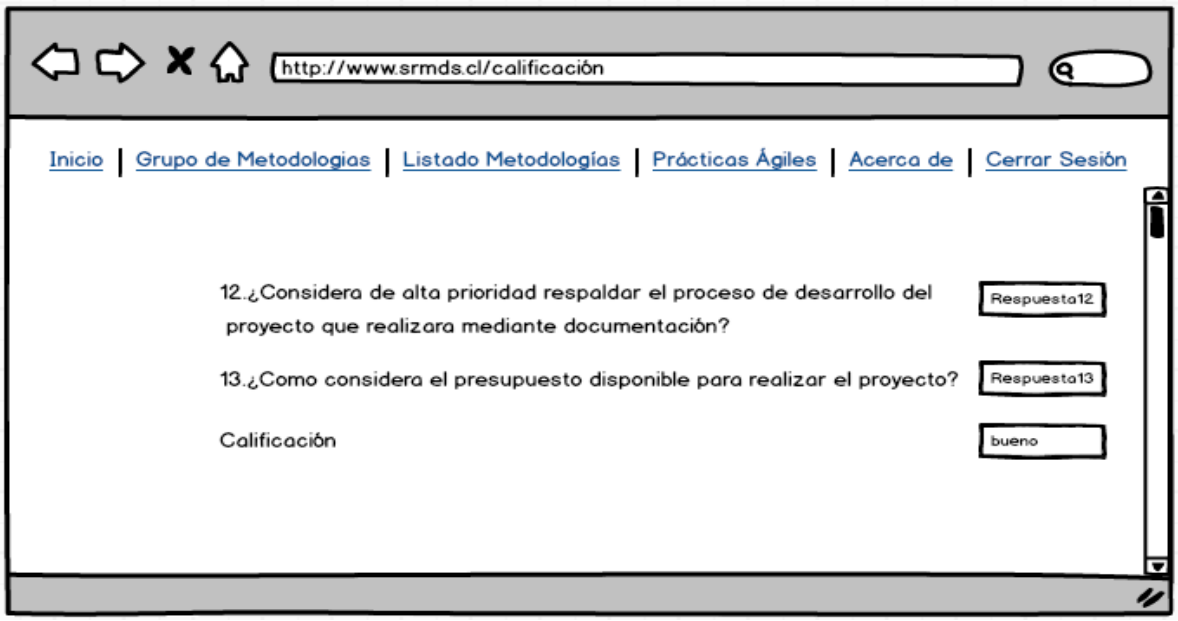


FIGURA N° 73: Detalle calificación

Sugerencias

En la siguiente imagen podemos apreciar que el administrador presiona el botón sugerencias, el sistema desplegara una nueva página donde el administrador podrá ver todas las sugerencia ingresadas por los usuarios.

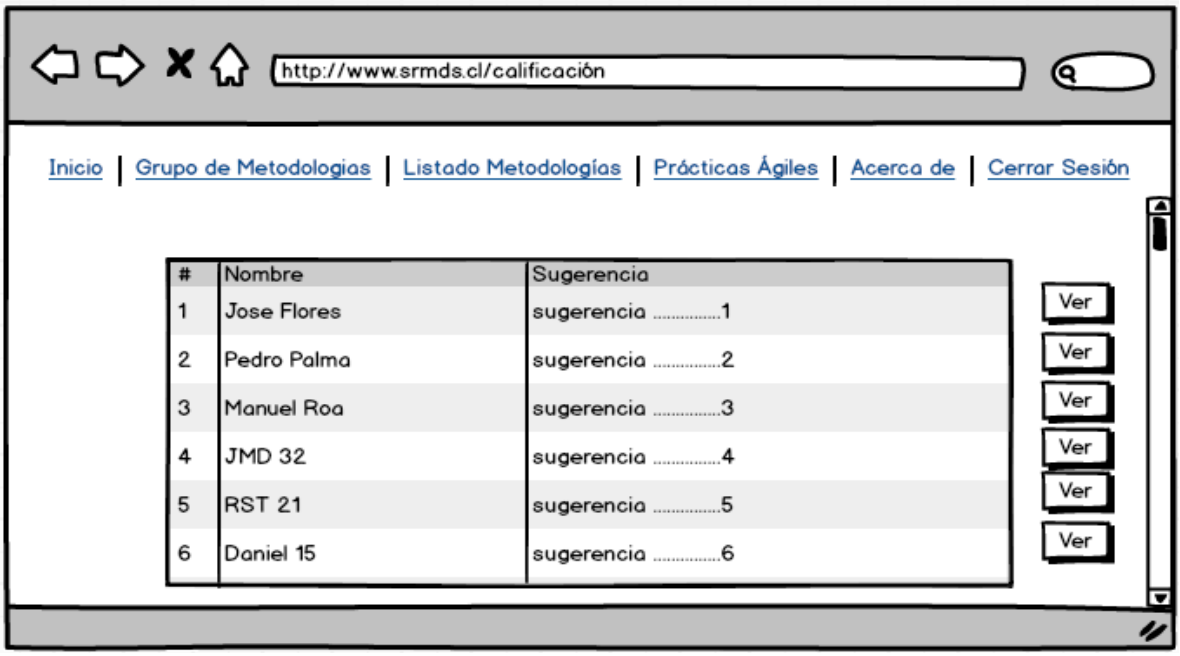


FIGURA N°74: Listado sugerencias

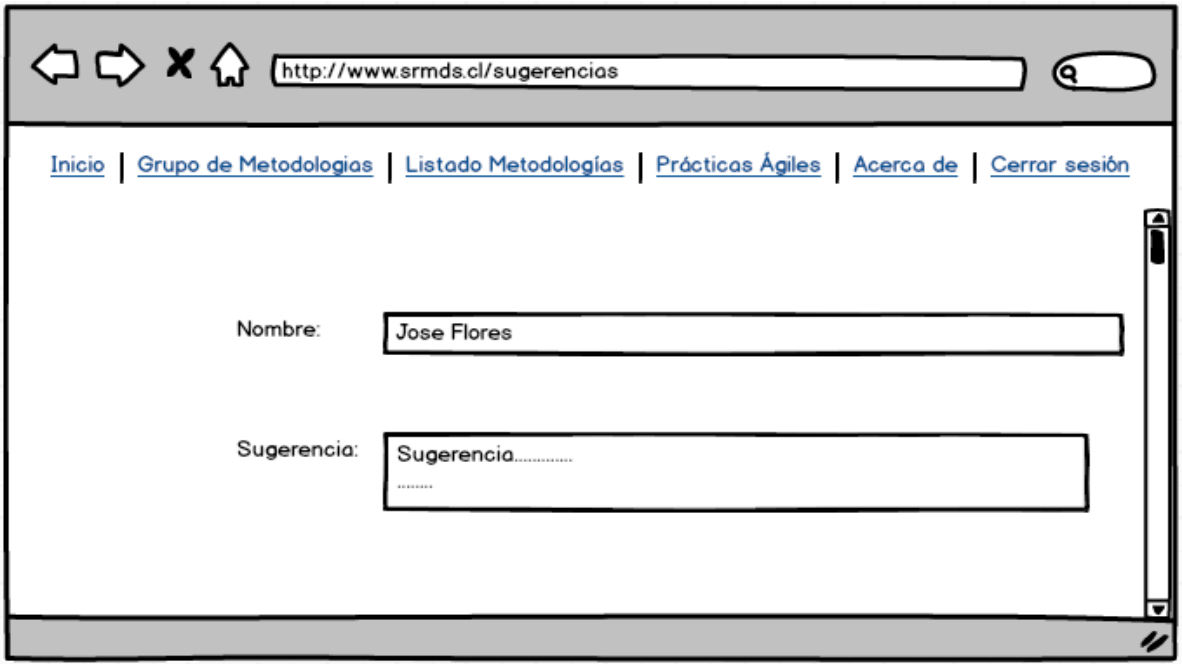


FIGURA N°75: Detalle sugerencia

Grupo de metodologías

Al igual que le usuario el administrador tendrá una navegación similar con la diferencia que más adelante aparecerán nuevas opciones.

El administrador selecciona la opción de la barra de menú "Grupo de Metodologías", se le mostraran el grupo de metodologías con sus respectivas opciones, las que serán visualizar o modificar.

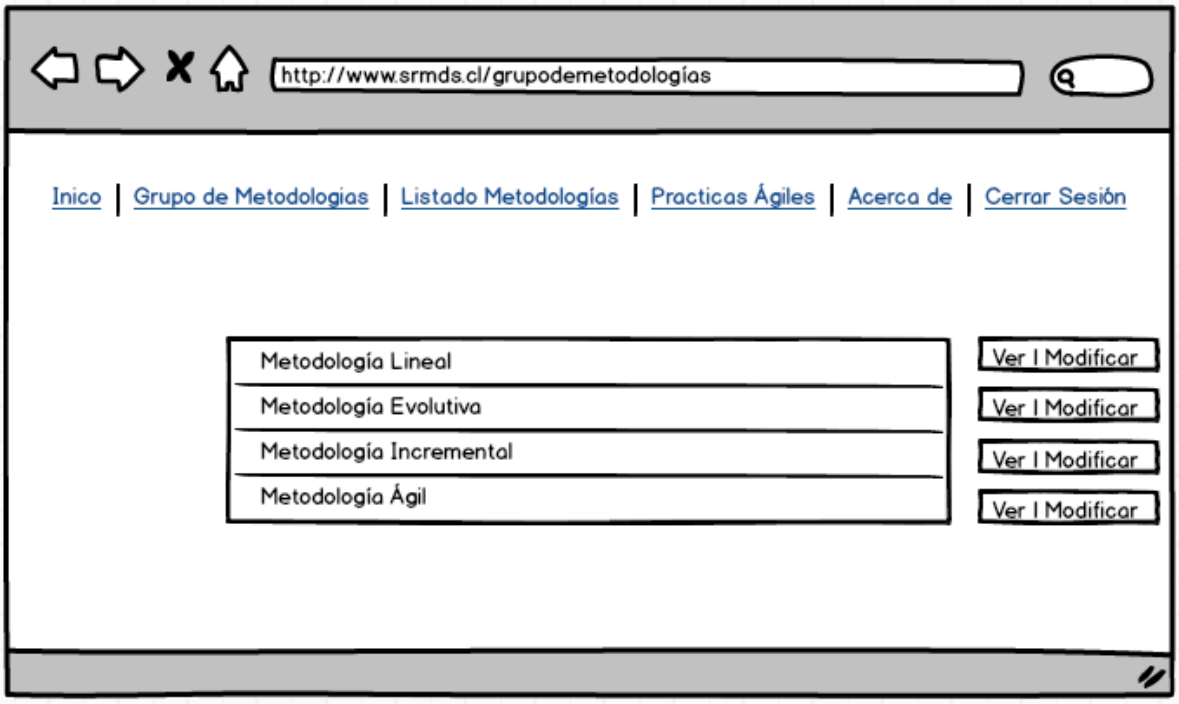


FIGURA N° 67: Gestión grupo de metodologías

Visualizar - Modificar

El administrador podrá visualizar la información del grupo de metodología seleccionada.

Para este ejemplo tomaremos el grupo de metodología Lineal, donde el administrador tendrá la vista completa de la información que mostrará, además se le da la opción de actualizar la información mediante el botón actualizar que le permitirá modificar la descripción de la metodología al igual que el botón modificar que estaba en la página anterior.

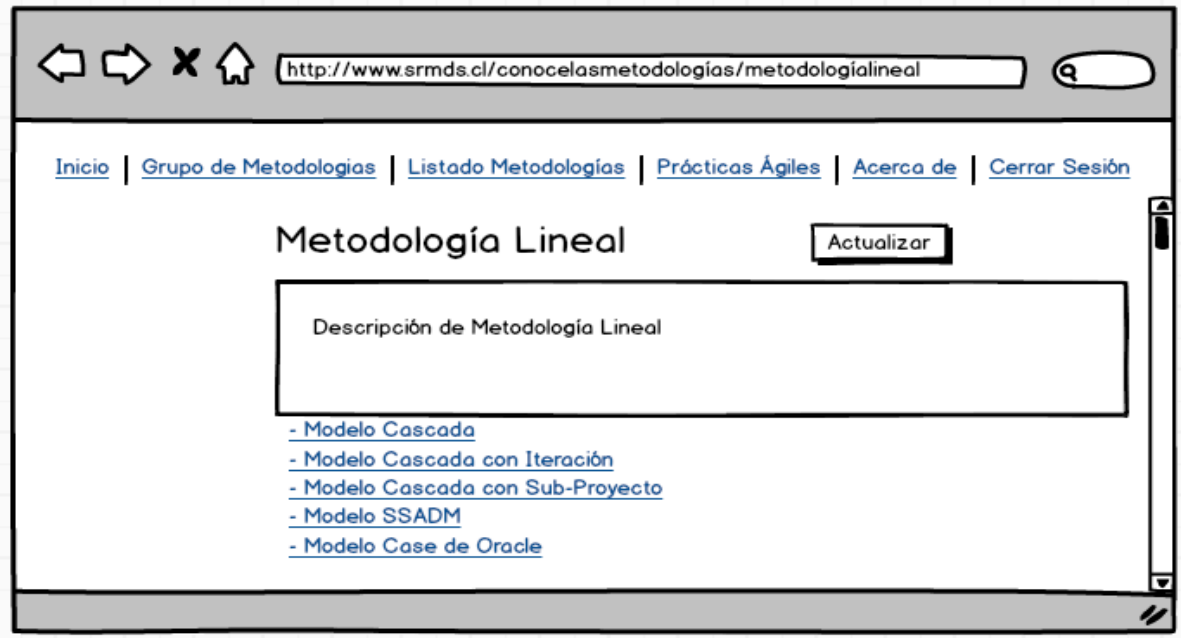


FIGURA N° 68: Detalle grupo de metodologías

FIGURA N° 69: Modificar grupo de metodologías

Listado Metodologías

En la siguiente imagen podemos apreciar que el administrador está en la barra de menú y selecciona la opción de "Listado Metodologías", la cual mostrara las diferentes metodologías ingresadas, además se le dará la opción de generar nuevos registros, visualizar modificar o eliminar alguna.

FIGURA N° 70: Gestión metodologías

Nuevo Registro

Para iniciar un nuevo registro debe de indicar a qué grupo de metodologías pertenece, luego ingresar la información que se solicita.

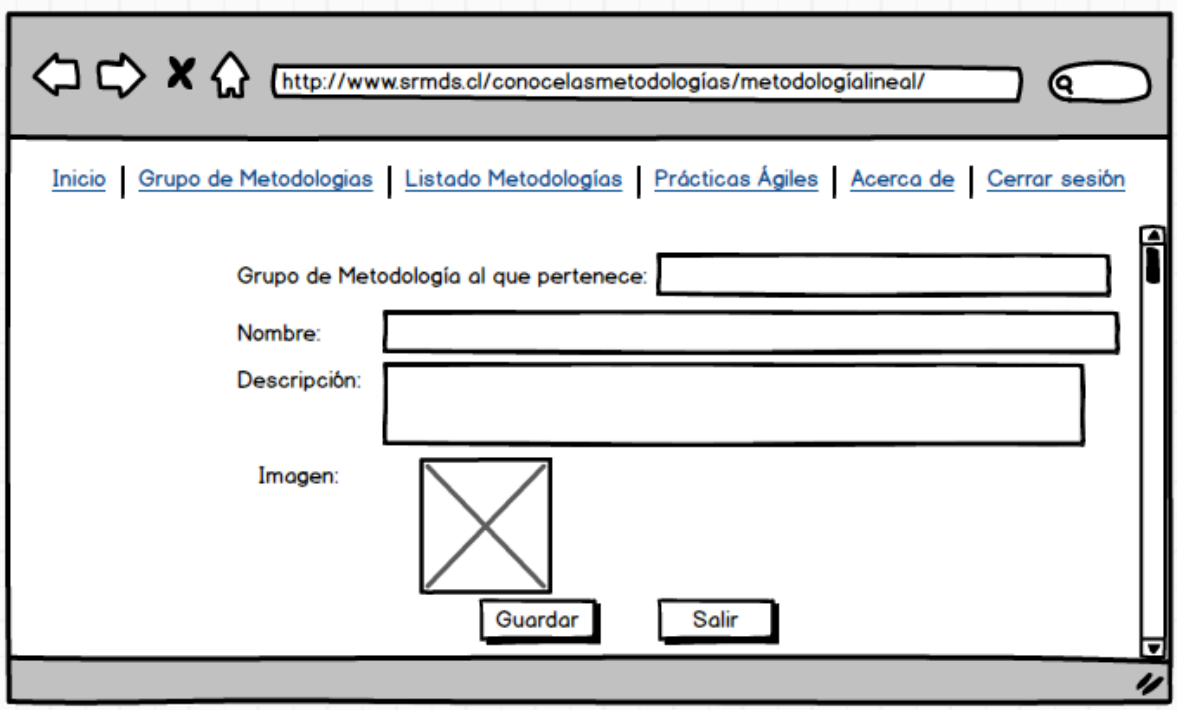


FIGURA N° 71: Registro metodología

Visualizar

Para Visualizar tomaremos como ejemplo el "Modelo Cascada".

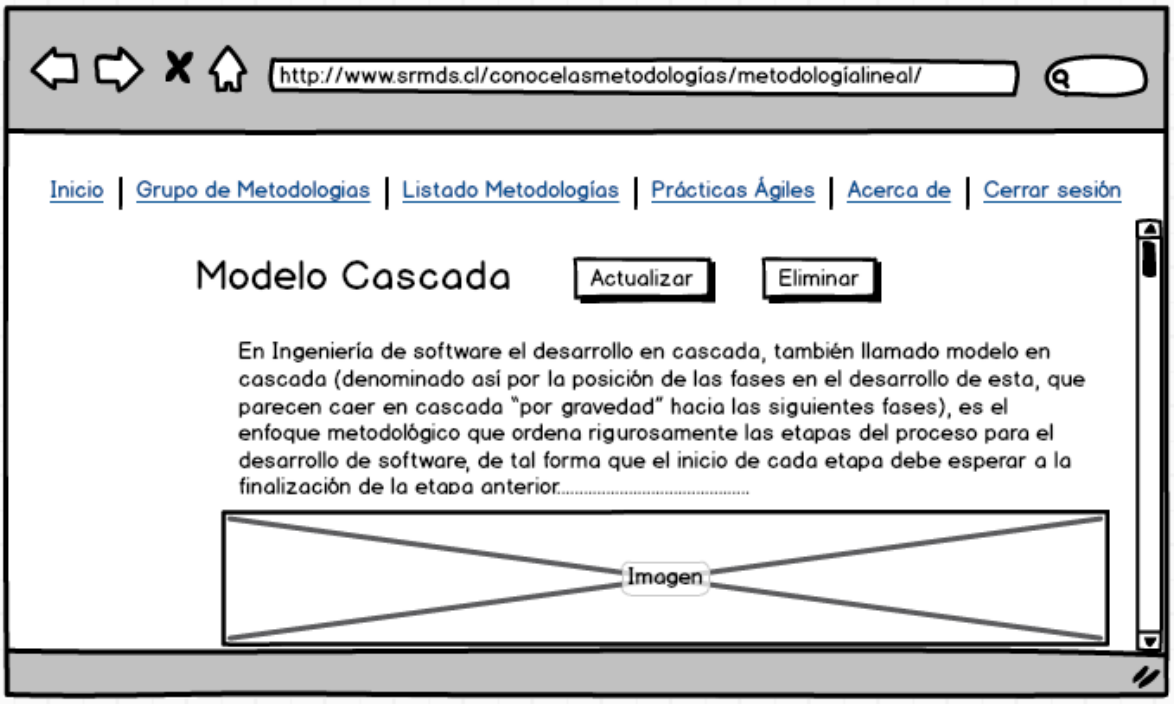


FIGURA N° 72: Detalle metodología

Podemos ver que la opción de visualizar del administrador también nos entrega dos opciones más que serian el actualizar información y eliminar que serán definidos en las próximas imágenes.

Modificar

Permite modificar todos los campos disponibles.

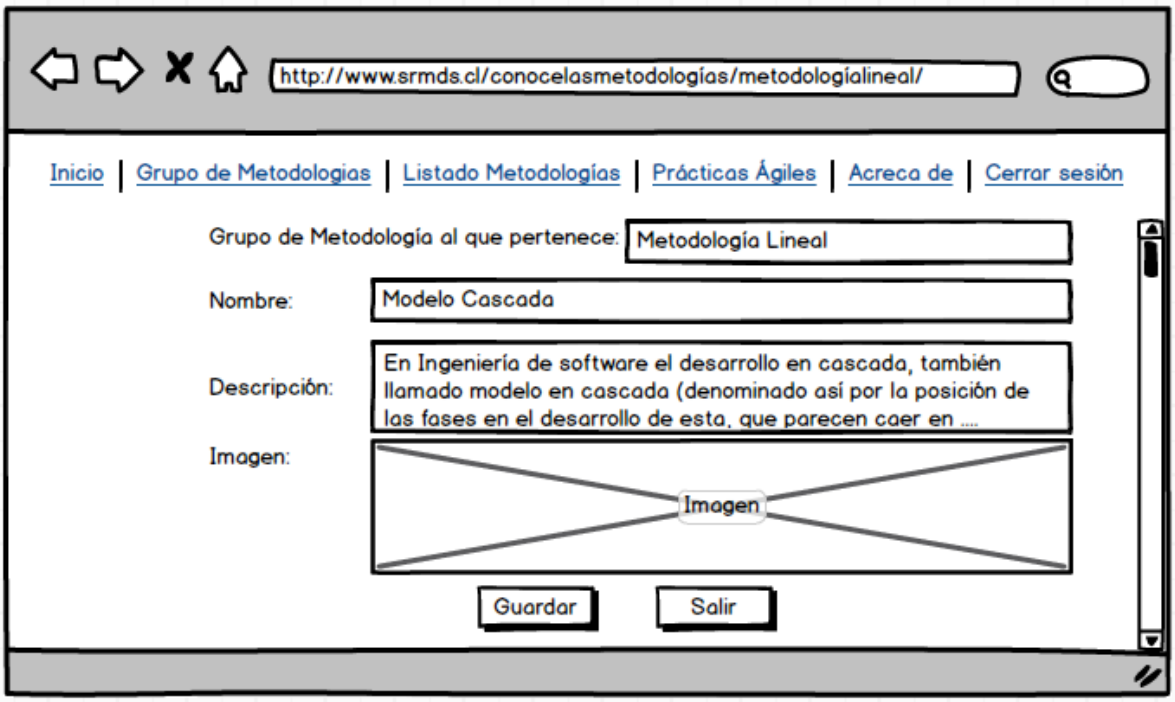


FIGURA N° 73: Modificar metodología

Eliminar

Permite eliminar los registros de una metodología ingresada.

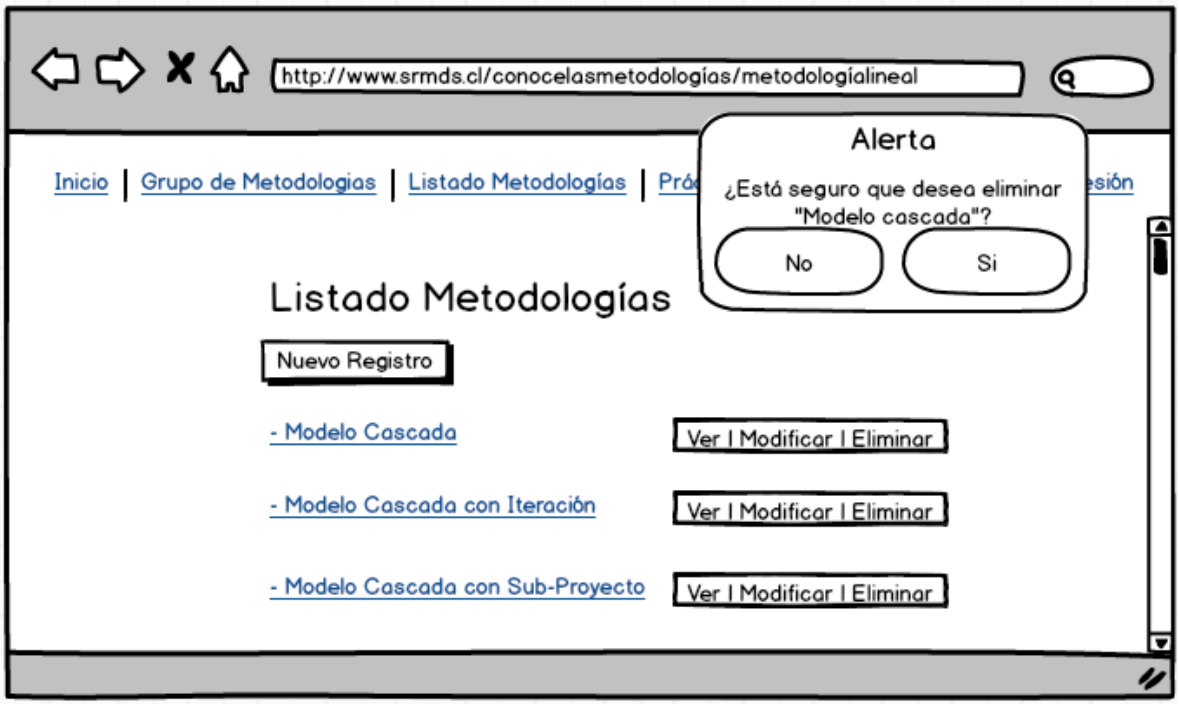


FIGURA N° 74: Eliminar metodología

Prácticas Ágiles

En la siguiente imagen podemos apreciar que el administrador está en la barra de menú y selecciona la opción de "Prácticas Ágiles", la cual mostrara las diferentes prácticas ágiles ingresadas, además se le dará la opción de generar nuevos registros, visualizar modificar o eliminar alguna.

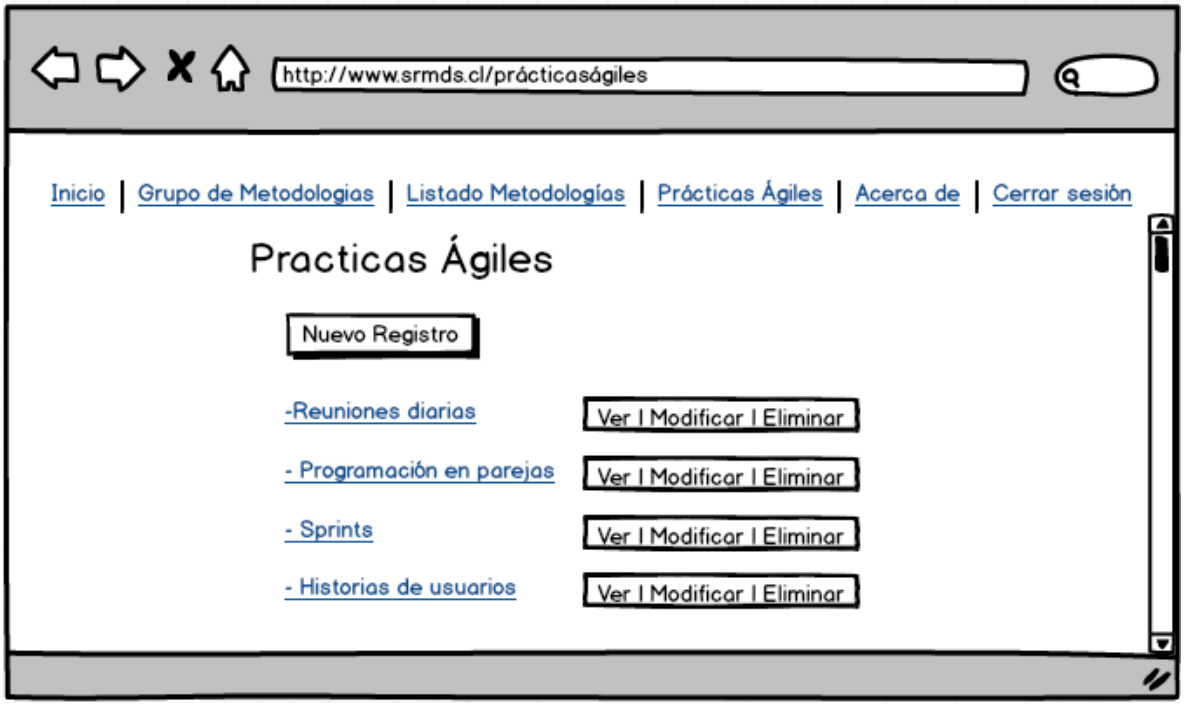


FIGURA N° 75: Gestión prácticas ágiles

Nuevo Registro

Permite ingresar nuevos registro de prácticas agiles al sistema.

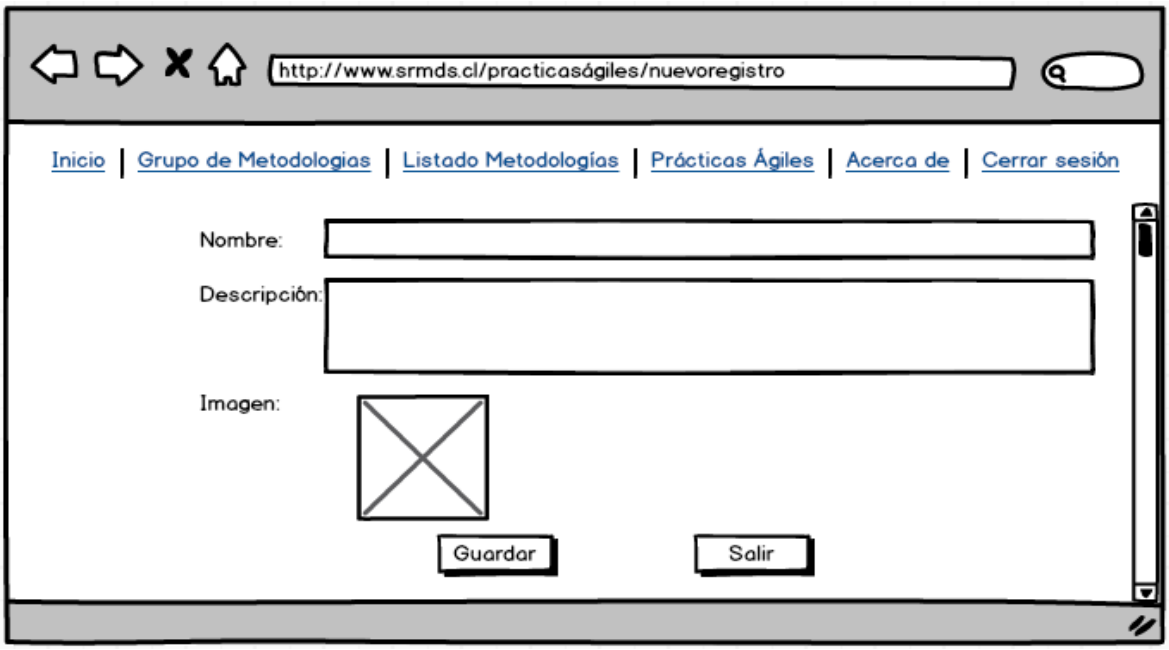


FIGURA N° 76: Registro práctica ágil

Visualizar

Para este ejemplo tomaremos la práctica ágil "Reuniones diarias", dentro de la cual a su vez nos dará dos opciones que son actualizar o eliminar la información almacenada.

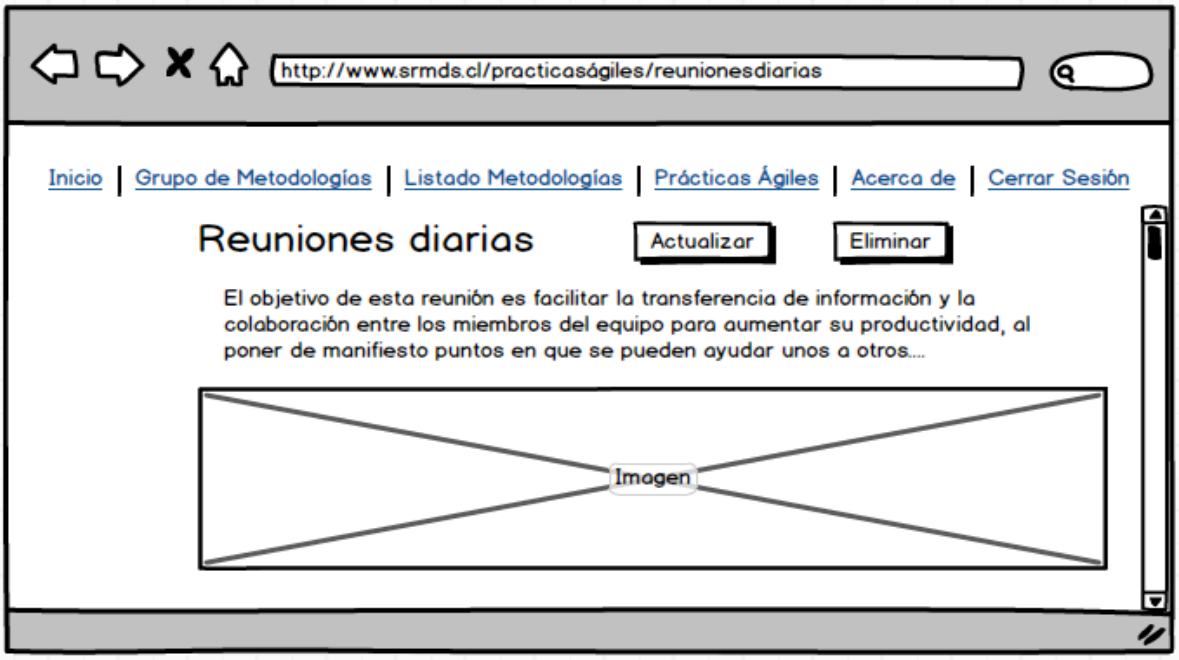


FIGURA N° 77: Detalle práctica ágil

Modificar

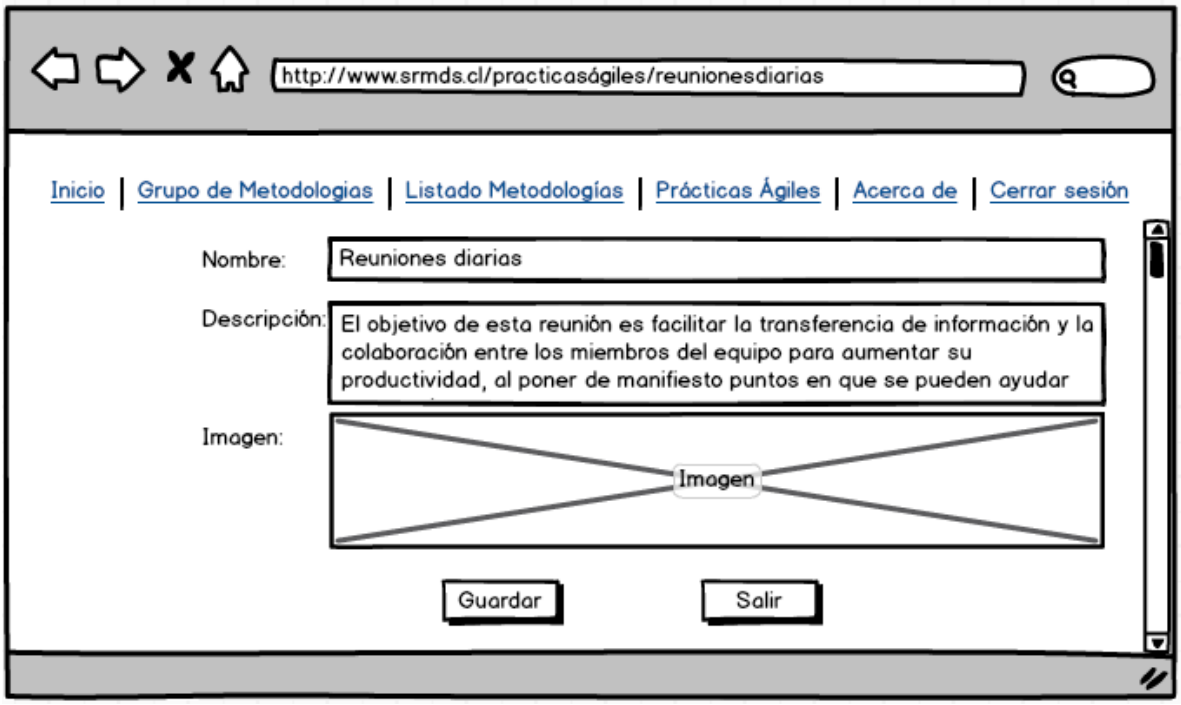


FIGURA N° 78: Modificar práctica ágil

Eliminar

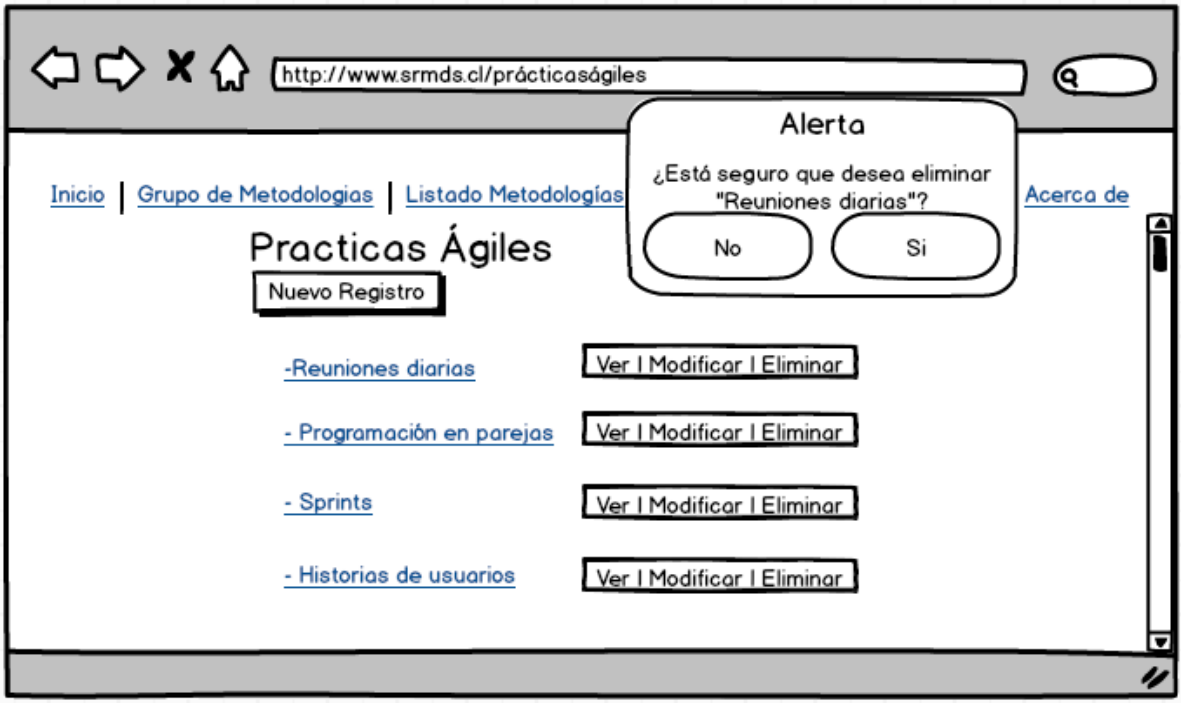


FIGURA N° 79: Eliminar práctica ágil

Cerrar Sesión Administrador - Usuario

El administrador o usuario pueden cerrar sesión de la misma forma, al presionar en la barra de menú la opción de "Cerrar Sesión" serán re-direccionados al página principal.

9.3.1 Jerarquía de menú

9.3.1.1 Jerarquía de menú Usuario

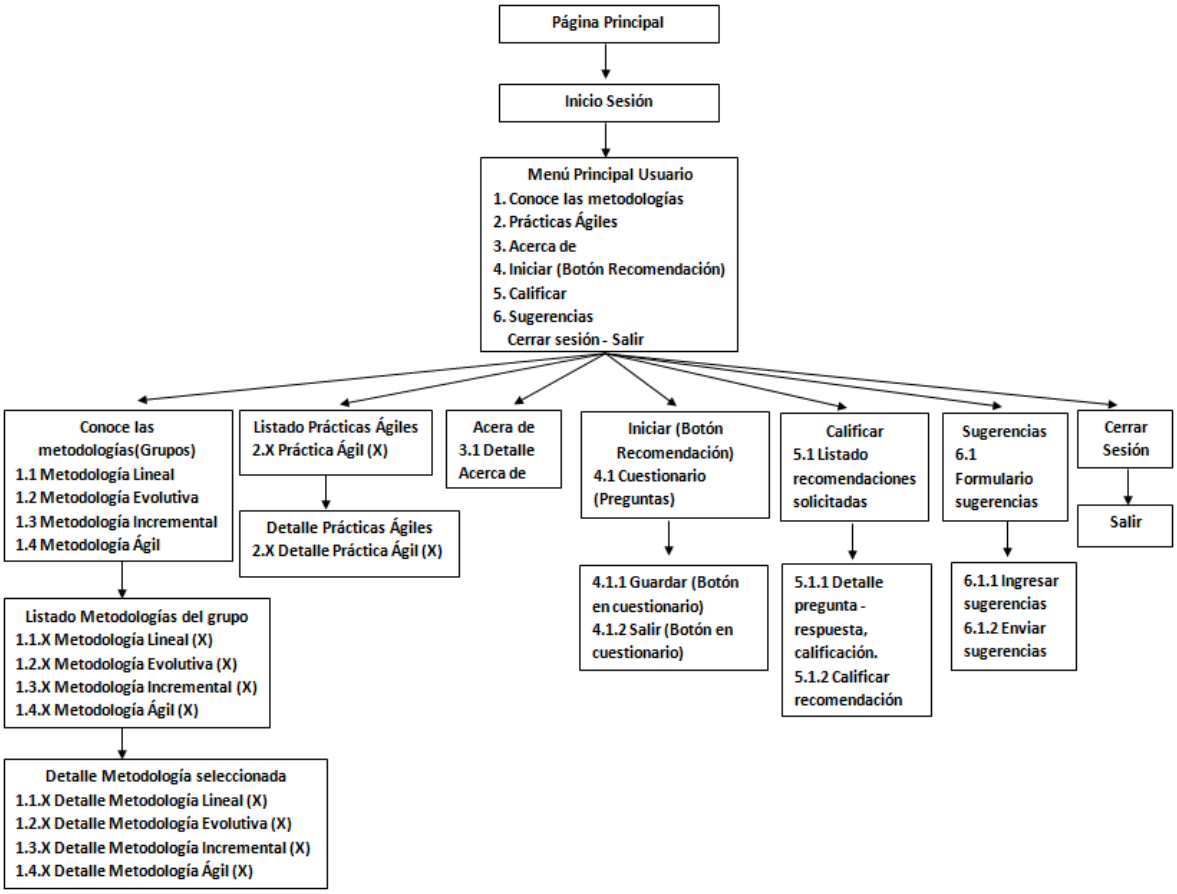


FIGURA N° 80: Jerarquía menú usuario

9.3.1.2 Jerarquía de menú Administrador

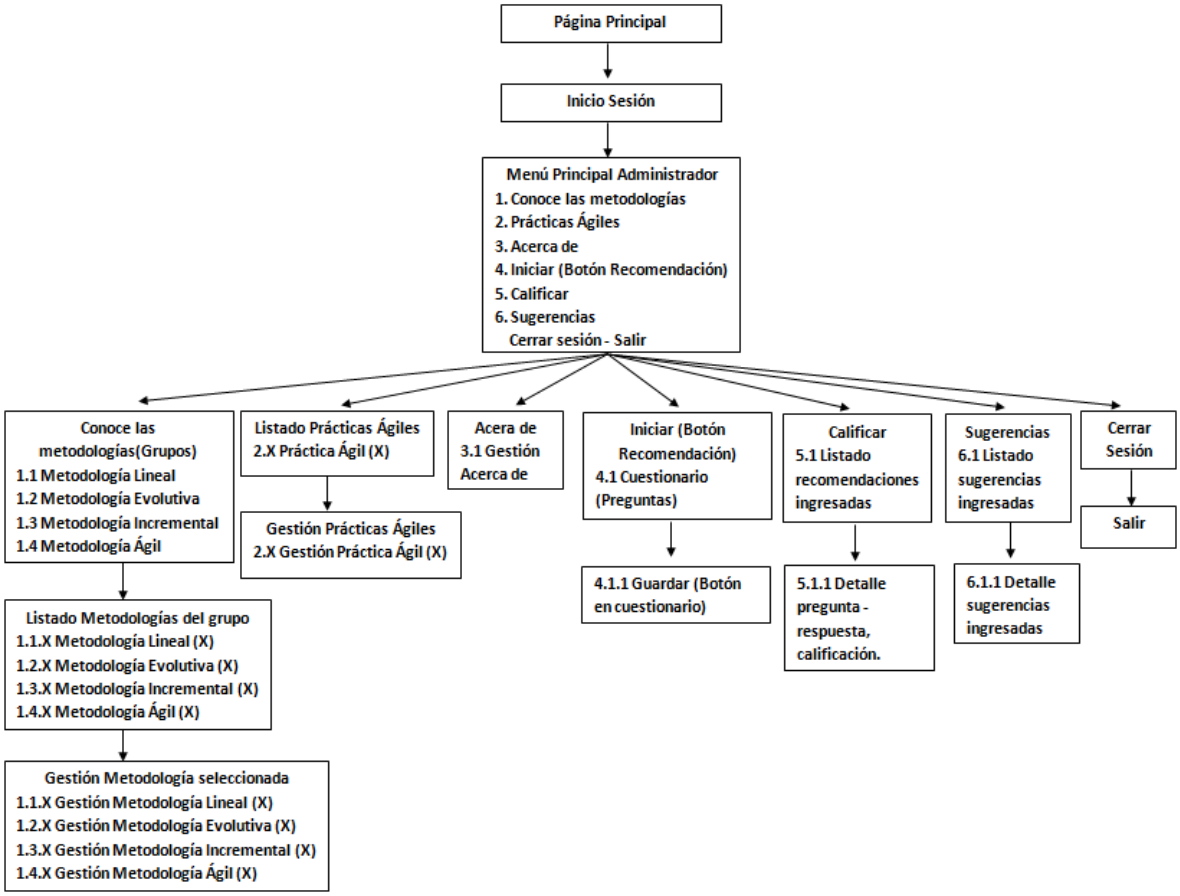


FIGURA N° 81: Jerarquía menú administrador

9.3.2 Mapa de navegación

9.3.2.1 Mapa de navegación Usuario

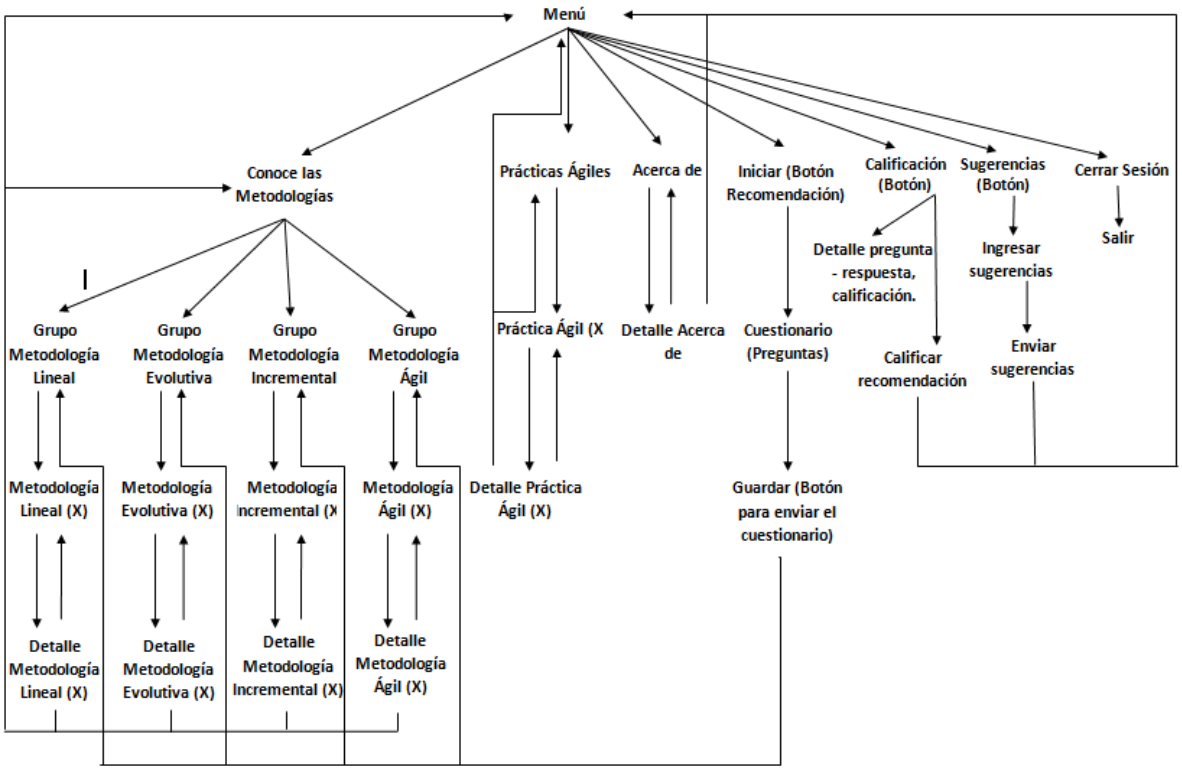


FIGURA N° 82: Mapa navegación usuario

9.3.2.2 Mapa de navegación Administrador

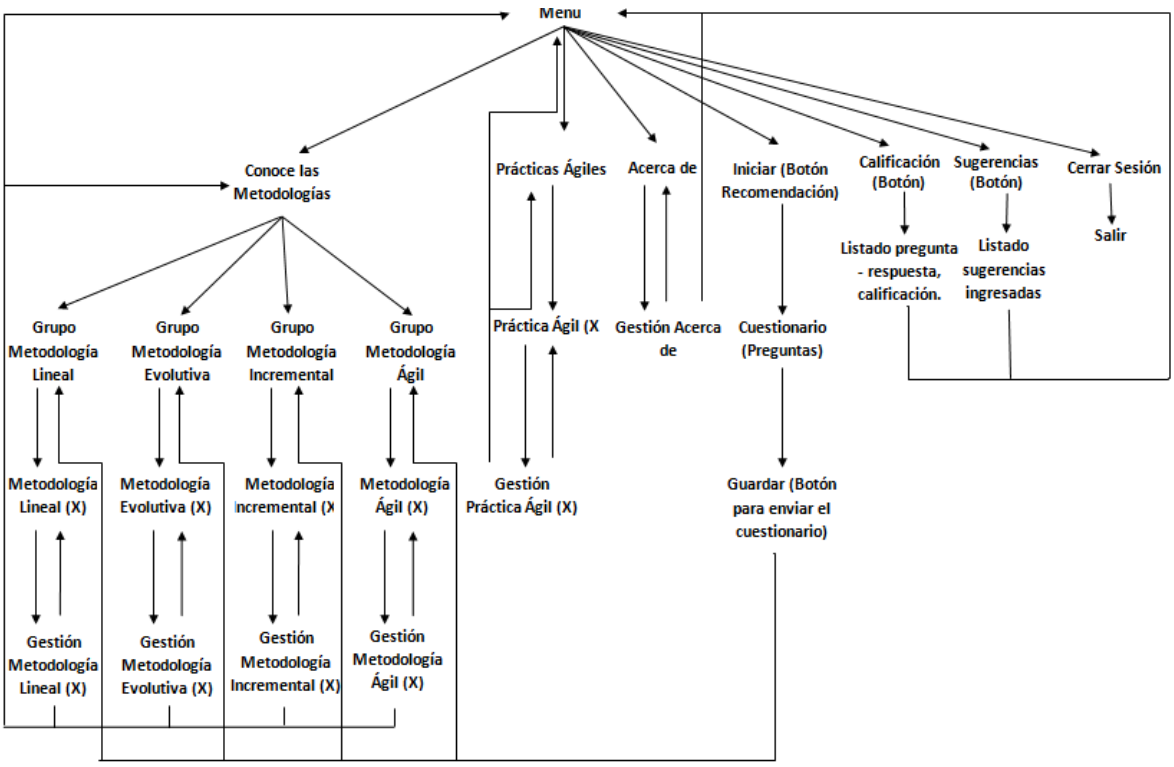


FIGURA N° 83: Mapa navegación administrador

9.4 Especificación de módulos

N° Módulo: 1		**Nombre Módulo**: Iniciar sesión	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Nombreusuario	Varchar	Pantalla index correspondiente	boolean
Contraseña	Password		

N° Módulo: 2		**Nombre Módulo**: modificar grupo de metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Descripcion_del_grupo_de_metodologia	Long text/longvarchar	Nombre_grupo_metodologia	varchar
		Descripcion_del_grupo_de_metodologia	Long text/long varchar

N° Módulo: 3		**Nombre Módulo**: visualizar grupo de metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón vista	Boolean	Nombre_grupo_de_metodologia	varchar
		Descripcion_del_grupo_de_metodologia	Long text/long varchar

N° Módulo: 4		**Nombre Módulo**: Ingresar metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Id_grupo_de_metodologia	Integer	Nombre_de_metodologia	Varchar
Nombre_de_metodologia	Varchar	Descripcion_de_metodologia	Long text/long varchar
Descripcion_de_metodologia	Long text/longvarchar	Imagen_de_metodologia	Image
Imagen_de_metodologia	Image		

N° Módulo: 5		**Nombre Módulo**: Modificar metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Id_grupo_de_metodologia	Integer	Nombre_de_metodologia	Varchar
Nombre_de_metodologia	Varchar	Descripcion_de_metodologia	Long text/long varchar
Descripcion_de_metodologia	Long text/longvarchar	Imagen_de_metodologia	Image
Imagen_de_metodologia	Image		

N° Módulo: 6		**Nombre Módulo**: Visualizar metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón vista	Boolean	Nombre_de_metodologia	Varchar
		Descripcion_de_metodologia	Long text/long varchar
		Imagen_de_metodologia	Image

N° Módulo: 7		**Nombre Módulo**: Eliminar metodologías	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón eliminar	Boolean	Index actualizado	Boolean
"click" botón confirmar	Boolean		

N° Módulo: 8		**Nombre Módulo**: Ingresar practicas agiles	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Nombre_practica_agil	Varchar	Nombre_practica_agil	Varchar
Descripcion_practica_agil	Long text/longvarchar	Descripcion_practica_agil	Long text/long varchar
Imagen_practica_agil	Image	Imagen_practica_agil	Image

N° Módulo: 9		**Nombre Módulo**: Modificar practicas agiles	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Nombre_practica_agil	Varchar	Nombre_practica_agil	Varchar
Descripcion_practica_agil	Long text/longvarchar	Descripcion_practica_agil	Long text/long varchar
Imagen_practica_agil	Image	Imagen_practica_agil	Image

N° Módulo: 10		**Nombre Módulo**: Visualizar practicas agiles	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón vista	Boolean	Nombre_practica_agil	Varchar
		Descripcion_practica_agil	Long text/long varchar
		Imagen_practica_agil	Image

N° Módulo: 11		**Nombre Módulo**: Eliminar practicas agiles	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón eliminar	Boolean	Index actualizado	Boolean
"click" botón confirmar	Boolean		

N° Módulo: 12		**Nombre Módulo**: modificar acerca de	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Descripción_acercade	Long text/longvarchar	Descripción_acercade	Long text/long varchar

N° Módulo: 13		**Nombre Módulo**: visualizar acerca de	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón vista	Boolean	Descripción_acercade	Long text/long varchar

N° Módulo: 14		**Nombre Módulo**: visualizar cuestionario	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón iniciar cuestionario	Boolean	Preguntas	Varchar

N° Módulo: 15		**Nombre Módulo**: responder cuestionario	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón iniciar cuestionario	Boolean	Preguntas	Varchar
		Posible_respuesta	Varchar

N° Módulo: 16		**Nombre Módulo**: solicitar recomendación	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón recomendar	Boolean		

N° Módulo: 17		**Nombre Módulo**: visualizar recomendación	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
		Gráficos	Imagen
		Información	Text

N° Módulo: 18		**Nombre Módulo**: visualizar calificación	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón visualizar calificaciones	Boolean	Respuestas almacenadas	Varchar
		Calificación	Varchar

N° Módulo: 19		**Nombre Módulo**: ingresar calificación	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
Calificación	Varchar	Respuestas almacenadas	Varchar
		Calificación	Varchar

N° Módulo: 20		**Nombre Módulo**: visualizar sugerencias	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón ver sugerencias	Boolean	Sugerencias	Long text/ longvarchar

N° Módulo: 21		**Nombre Módulo**: ingresar sugerencias	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" botón sugerencias	Boolean	Sugerencias	Long text/ longvarchar

N° Módulo: 22		**Nombre Módulo**: cerrar sesión	
Parámetros de entrada		**Parámetros de Salida**	
Nombre:	**Tipo de dato:**	**Nombre:**	**Tipo de dato:**
"click" cerrar sesión	Boolean	Pantalla inicio	Boolean

10 PRUEBAS

10.1 Elementos de prueba

Módulo	Descripción
Registrarse en el sistema	En este módulo se verifica que se permita crear nuevas cuentas de usuario.
Ingresar al sistema	En este módulo se verifica que se permita el ingreso al administrador y los usuarios registrados (con sus respectivos privilegios).
Visualizar Acerca de (autenticado y sin autenticar)	En este módulo se verifica que cualquier persona haya o no iniciado sesión pueda visualizar la sección Acerca De del sistema.
Gestionar grupos de metodologías	En este módulo el administrador puede visualizar y modificar la información correspondiente a los grupos de metodologías establecidos.
Gestionar metodologías	En este módulo el administrador puede realizar el CRUD (create, read, update y delete) correspondiente a las metodologías.
Gestionar prácticas ágiles	En este módulo el administrador puede realizar el CRUD (create, read, update y delete) correspondiente a las prácticas ágiles.
Visualizar Sugerencias	En este módulo se verifica que el administrador pueda visualizar las sugerencias registradas.
Visualizar Registros y Calificaciones	En este módulo se verifica que el administrador pueda visualizar los registros de respuestas y calificaciones registradas.
Visualizar grupos de metodologías	En este módulo se verifica que el usuario pueda visualizar la información de los grupos de metodologías almacenados del sistema.
Visualizar metodologías	En este módulo se verifica que el usuario pueda visualizar la

	información de las metodologías almacenadas del sistema.
Visualizar prácticas ágiles	En este módulo se verifica que el usuario pueda visualizar la información de las prácticas ágiles almacenados del sistema.
Responder Cuestionario y recibir recomendación	En este módulo se verifica que el usuario pueda realizar el cuestionario y recibir la recomendación correspondiente.
Ver registro y calificar	En este módulo se verifica que el usuario pueda ver sus respuestas realizadas en una recomendación anterior, y calificarla.
Enviar Sugerencia	En este módulo se verifica que el usuario pueda realizar una sugerencia para sistema.

TABLA N° 12: Elementos de Pruebas

10.2 Especificación de las pruebas

Características a probar	**Nivel de prueba**	**Objetivo de la Prueba**	**Enfoque para la definición de casos de prueba**	**Técnicas para la definición de casos de prueba**	**Actividades de prueba**	**Criterios de cumplimiento**
Funcionalidad	Sistema	Que se cumplan los requerimientos planteados o expuestos.	Caja Negra	Valores, Límites y Particiones	1.- Ejecutar Script MySQL de la base de datos.	Que se cumplan los requerimientos solicitados.
Funcionalidad	Aceptación	Que el administrad	Caja negra	Valores, Límites y	1.- Ingresar al perfil	Que el administrador

		or realice las diferentes tareas y que se cumplan los requerimientos planteados.		Particiones	administrador. 2.- Gestionar grupos de metodologías. 3.- Gestionar metodologías. 4.- Gestionar prácticas ágiles. 5.- Visualizar sugerencias, registros y calificaciones.	se cerciore que los requerimientos han sido cumplidos.
Funcionalidad	Aceptación	Que el usuario realice las diferentes tareas y que se cumplan los requerimientos planteados.	Caja negra	Valores, Límites y Particiones	1.- Ingresar al perfil usuario. 2.- Visualizar grupos de metodologías. 3.- Visualizar metodologías. 4.- Visualizar prácticas ágiles. 5.- Responder	Que el usuario verifique los requerimientos han sido cumplidos satisfactoriamente.

					cuestionario y ver recomendación. 6.- Ver registro y calificar 7.-Enviar sugerencia.	

TABLA N° 13: Especificación de Pruebas

10.3 Responsables de las pruebas

Actividad de Prueba	**Responsable**
Ejecutar Script MySQL de la base de datos.	Fabián Ávila Cáceres.
Ingresar al perfil administrador.	Fabián Ávila Cáceres.
Gestionar grupos de metodologías.	Fabián Ávila Cáceres.
Gestionar metodologías.	Richard Fica Inzunza
Gestionar prácticas ágiles.	Fabián Ávila Cáceres.
Visualizar sugerencias, registros y calificaciones.	Richard Fica Inzunza
Ingresar al perfil usuario.	Fabián Ávila Cáceres.
Visualizar grupos de metodologías.	Richard Fica Inzunza
Visualizar metodologías.	Richard Fica Inzunza
Visualizar prácticas ágiles.	Fabián Ávila Cáceres.
Responder cuestionario y ver recomendación.	Richard Fica Inzunza

Ver registro y calificar	Richard Fica Inzunza
Enviar sugerencia	Richard Fica Inzunza

TABLA N° 14: Responsables de Pruebas

10.4 Calendario de pruebas

Actividad Prueba	**Fecha**
Caso de prueba N°1	18-12-2017
Caso de prueba N°2	19-12-2017
Caso de prueba N°3	20-12-2017

TABLA N° 15: Calendario de Pruebas

10.5 Detalle de las pruebas

Las siguientes figuras muestran el detalle de las pruebas efectuadas. En verde se representa el correcto funcionamiento del sistema durante la prueba, mientras que en rojo un ingreso erróneo de datos.

Ejecutar Script MySQL de la base de datos.

Id	**Descripción Requerimiento Funcional**	**Entrada**					**Salida esperada**	**Salida Obtenida**	**Evaluación**	
									Éxito / Fracaso	**Criticidad en caso Fracaso**
1	Ejecutar Script	Script					Base de datos creada		Éxito	

Ingresar al perfil administrador.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre usuario	Contraseña							Éxito / Fracaso	Criticidad en caso Fracaso
1	Autenticar administrador	Admin 2	*******					Usuario Incorrecto	Usuario Incorrecto	Éxito	
2	Autenticar administrador	Admin	********					Contraseña Incorrecta	Contraseña Incorrecta	Éxito	
3	Autenticar administrador	Admin 2	********					Usuario y/o Contraseña Incorrectas	Usuario y/o Contraseña Incorrectas	Éxito	
4	Autenticar administrador	Admin	*******					Correcto	Correcto	Éxito	

Gestionar grupos de metodologías.

Modificar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_grupo_metodologia	Descripción_grupo_metodologia							Éxito / Fracaso	Criticidad en caso Fracaso
1	Modificar Grupo metodología	1	Descripción.					Correcta	Correcta	Éxito	

Visualizar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre Grupo Metodología								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar grupo de Metodología	Metodologías lineales						Mostrar grupo de Metodología	Mostrar grupo de Metodología	Éxito	
2	Visualizar grupo de Metodología	Metodo li						Grupo de Metodología no existe	Grupo de Metodología no existe	Éxito	

Gestionar metodologías.

Agregar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_grupo_metodologia	Nombre_metodologia	Descripcion_metodologia	Imagen_metodologia					Éxito / Fracaso	Criticidad en caso Fracaso
1	Agregar metodología	5	Cascada	descripción	imagen			Id no existe	Id no existe	Éxito	Éxito
2	Agregar metodología	4	Cascada	descripción	-			Error! Debe ingresar imagen	Error!	Éxito	Éxito
3	Agregar metodología	4	Cascada	descripción	imagen			Metodología ingresada	Metodología ingresada	Éxito	Éxito

Modificar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_metodologia	Id_grupo_metodologia	Nombre_metodologia	Descripcion_metodologia	Imagen_metodologia				Éxito / Fracaso	Criticidad en caso Fracaso
1	Modificar metodología	1	5	Cascada	descripción	imagen		Id no existe	Id no existe	Éxito	
2	Modificar metodología	1	4	Cascada	Descripción2	-		Error! Debe ingresar imagen	Error!	Éxito	
3	Modificar metodología	1	4	Cascada	Descripción2	imagen		Metodología actualizada	Metodología actualizada	Éxito	

Visualizar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre Metodología								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar Metodología	Cascada						Mostrar Metodología	Mostrar Metodología	Éxito	
2	Visualizar Metodología	Cascasda						Metodología no existe	Metodología no existe	Éxito	

Eliminar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_Metodología								Éxito / Fracaso	Criticidad en caso Fracaso
1	Eliminar Metodología	1						Metodología eliminada	Metodología eliminada	Éxito	

Gestionar prácticas ágiles.

Agregar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre_practica_agil	Descripcion_practica_agil	Imagen_practica_agil						Éxito / Fracaso	Criticidad en caso Fracaso
1	Agregar práctica ágil	Reuniones diarias	descripción	-				Error! Debe ingresar imagen	Error!	Éxito	
2	Agregar práctica ágil	Reuniones diarias	descripción	imagen				Práctica ágil ingresada	Práctica ágil ingresada	Éxito	

Modificar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_practica_agil	Nombre_practica_agil	Descripcion_practica_agil	Imagen_practica_agil					Éxito / Fracaso	Criticidad en caso Fracaso
1	Modificar práctica ágil	1	Reuniones diarias	Descripción2	-			Error! Debe ingresar imagen	Error!	Éxito	
2	Modificar práctica ágil	1	Reuniones diarias	Descripción2	imagen			Metodología actualizada	Metodología actualizada	Éxito	

Visualizar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre practica ágil								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar práctica ágil	Reuniones diarias						Mostrar práctica ágil	Mostrar práctica ágil	Éxito	
2	Visualizar práctica ágil	Dgasdg						práctica ágil no existe	práctica ágil no existe	Éxito	

Eliminar

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_practicaagil								Éxito / Fracaso	Criticidad en caso Fracaso
1	Eliminar práctica ágil	1						práctica ágil eliminada	práctica ágil eliminada	Éxito	

Visualizar sugerencias, registros y calificaciones.

Sugerencias

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		id_sugerencia								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar sugerencia	3						Mostrar sugerencia	Mostrar sugerencia	Éxito	
2	Visualizar sugerencia	6756da75						sugerencia no existe	sugerencia no existe	Éxito	

Registro y Calificaciones

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Idregistro								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar registro	4						Mostrar registro	Mostrar registro	Éxito	
2	Visualizar registro	123jm						Registro no existe	Registro no existe	Éxito	

Ingresar al perfil usuario.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre usuario	Contraseña							Éxito / Fracaso	Criticidad en caso Fracaso
1	Autenticar usuario	user1232	*******					Usuario Incorrecto	Usuario Incorrecto	Éxito	
2	Autenticar usuario	User	********					Contraseña Incorrecta	Contraseña Incorrecta	Éxito	
3	Autenticar usuario	User2122	********					Usuario y/o Contraseña Incorrectas	Usuario y/o Contraseña Incorrectas	Éxito	
4	Autenticar usuario	User	*******					Correcto	Correcto	Éxito	

Visualizar grupos de metodologías.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		nombre_grupode_metodologia								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar grupo de metodologías	Metodologías Lineales						Mostrar grupo de metodología	Mostrar grupo de metodología	Éxito	
2	Visualizar grupo de metodologías	6756da75						grupo de metodología no existe	grupo de metodología no existe	Éxito	

Visualizar metodologías.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre Metodología								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar Metodología	Cascada						Mostrar Metodología	Mostrar Metodología	Éxito	
2	Visualizar Metodología	Cascasda						Metodología no existe	Metodología no existe	Éxito	

Visualizar prácticas ágiles.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre practica ágil								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar práctica ágil	Reuniones diarias						Mostrar práctica ágil	Mostrar práctica ágil	Éxito	
2	Visualizar práctica ágil	Dgasdg						práctica ágil no existe	práctica ágil no existe	Éxito	

Responder cuestionario y ver recomendación.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Respuestas								Éxito / Fracaso	Criticidad en caso Fracaso
1	cuestionario	respuestas						Mostrar recomendación	Mostrar recomendación	Éxito	

Ver registro y calificar.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Id_registro								Éxito / Fracaso	Criticidad en caso Fracaso
1	Visualizar registro	4						Mostrar registro	Mostrar registro	Éxito	

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Calificación								Éxito / Fracaso	Criticidad en caso Fracaso
1	Calificar	bueno						Guardar calificación	Guardar calificación	Éxito	

Enviar sugerencia.

Id	Descripción Requerimiento Funcional	Entrada						Salida esperada	Salida Obtenida	Evaluación	
		Nombre usuario (opcional)	Sugerencia							Éxito / Fracaso	Criticidad en caso Fracaso
1	Enviar sugerencia		Sugerencia …					Sugerencia registrada	Sugerencia registrada	Éxito	
2	Enviar sugerencia	Nombre	Sugerencia …					Sugerencia registrada	Sugerencia registrada	Éxito	

10.6 Conclusiones de Prueba

Finalizando la realización de las pruebas al sistema se realizaron de forma favorable con éxito en todos los módulos asociados.

Se validan los datos y consistencia de los mismos con la base de datos planteada y se considera abarcados los requisitos funcionales del sistema.

Se obtuvo un resultado inesperado al actualizar y/o crear elementos que contemplan imágenes como las metodologías y prácticas ágiles, las cuales siempre deben ser agregadas, para qué la ruta sea actualizada en la base de datos, ya que al actualizar nuevamente o crear sin ingresar una imagen, el sistema ingresa una ruta vacía.

11 PLAN DEIMPLANTACIÓN Y PUESTA EN MARCHA

Nuestro sistema fue desarrollado usando el framework de software libre llamado Yii 2.0 basado en componentes PHP y la herramienta XAMPP versión 5.6 también de software libre que permite tener un servidor web locas Apache, y motor de base de datos MySQL, siendo además interprete de PHP.

Para la implantación y puesta en marcha del sistema se solicitará un servidor compatible con las herramientas que utilizadas en el desarrollo de éste, y que tenga soporte para Windows 7 o superior y/o Linux.

Una vez contando con los accesos al servidor que otorgara la Universidad del Bío-Bío se realizaran pruebas de conexión, para luego implementar finalmente el sistema completo para su posterior uso, de momento el sistema solo está disponible para el uso de forma local.

Luego se dejara en funcionamiento, para realizar pruebas y tendrá un uso real y práctico durante el semestre para la clase de Metodologías de Desarrollo de Software por lo que permanecerá en funcionamiento en la universidad mientras lo estimen necesario.

Estas pruebas permitirán poner a prueba el resultado del estudio, así como también tener una retroalimentación para mejorar el sistema en el futuro.

12 CONCLUSION

El proyecto realizado presento un gran y novedoso desafío, como investigación y desarrollo del sistema, que permitió utilizar gran parte de los conocimientos adquiridos durante el transcurso de la carrera. Se recolecto información relevante sobre cuáles son las metodologías más utilizadas y en qué tipo de proyectos eran implementadas, lo que permiten inferir que en otras empresas e instituciones de Chile la situación debe ser similar.

Se logró establecer qué criterios o factores afectaban más dentro del uso de las metodologías, lo que permitió establecer reglas y filtros para el desarrollo de un sistema de recomendación de grupos de metodologías. A su vez, al no existir una investigación similar, la muestra de los resultados obtenidos de la investigación es un gran aporte para considerar la realidad del uso de las metodologías de desarrollo de software en Chile junto a la zona cercana a la ciudad de Concepción. Por otro lado se obtuvieron conocimientos de los sistemas de recomendación y el funcionamiento de los algoritmos detrás de estos, lo cual, permitió desarrollar de mejor forma el sistema que es resultado de la investigación realizada.
De igual modo, el desarrollo de este proyecto de título permitió mejorar y aplicar los conocimientos de programación y aplicación de la estructura modelo-vista-controlador y el manejo de frameworks como herramientas de desarrollo de software.

Una de las limitaciones de esta investigación corresponde al pequeño tamaño de la muestra de empresas que participaron en el estudio. Esto origino que no se pudiera desarrollar el sistema de recomendación en base a metodologías de desarrollo especificas, por lo cual, se tuvo que considerar grupos de metodologías para la recomendación.

Futuros estudios pueden realizar una investigación más exhaustiva con una muestra mayor de empresas y más recursos para realizar un sistema que contenga un mayor porcentaje de asertividad en base a los filtros que pudiesen ser aplicados.

13 BIBLIOGRAFÍA

- Ahimbisibwe, Cavana&Daellenbach (2015). A contingency fit model of critical success factors for software development projects- A comparison of agile and traditional plan-based methodologies. Journal of Enterprise Information Management.
- Boehm B,(1988) "A Spiral Model of Software Development and Enhancement".
- DeGrace, Peter, y Stahl, Leslei (1990). Problemas perversos, soluciones justas: un católogo de paradigmas de la ingeniería moderna.
- Eyssautier De La Mora(2006), Metodología de la investigación.
- Felfernig, Friedrich, Jannach, and Zanker (2015). Constraint-Based Recommender Systems.
- Germain&Robillard. (2005). Engineering-based process and agile methodologies for software development: a comparative case study. The Journal of Systems and Software.
- Goodland, Riha (1999). «History of SSADM».
- McConnell, Steve (1996). Desarrollorápido: Domando Wild Software Schedules.
- Ozturk (2013). Selection of appropriate software development life cycle using fuzzy logic. Journal of Intelligent&FuzzySystems.
- Poppendieck, M., Poppendieck, T., Lean software development: an agile toolkit for software development managers.
- Pressman R.S., (2010) Ingeniería de Software, Un enfoque práctico, editorial: McGrawHill.: México.
- Rising, L., Janoff, N.S. (2000). The Scrum Software Development Process for Small Teams
- Shigeo (1989). A Study of the Toyota Production System from an Industrial Engineering
- Sommerville Ian., (2005). «Entrega Incremental». Ingeniería del Software

14 LINKOGRAFIA

- Definiciones
  - Software: http://tecnologiaedu.us.es/cuestionario/bibliovir/paz10.pdf
  - Metodología: https://es.oxforddictionaries.com/definicion/metodologia
  - Metodología de Desarrollo de Software**:** https://www.cms.gov/Research-Statistics-Data-and-Systems/CMS-Information-Technology/XLC/Downloads/SelectingDevelopmentApproach.pdf
  - Desarrollador de software: https://es.wikipedia.org/wiki/Desarrollador_de_software
  - Presupuesto: https://es.oxforddictionaries.com/definicion/presupuesto
  - Requerimientos: http://www.alegsa.com.ar/Dic/requerimientos.php
  - Adaptabilidad a cambios: http://ddp.usach.cl/sites/ddp/files/documentos/diccionario_de_competencias_0.pdf
  - Riesgo: https://es.oxforddictionaries.com/definicion/riesgo

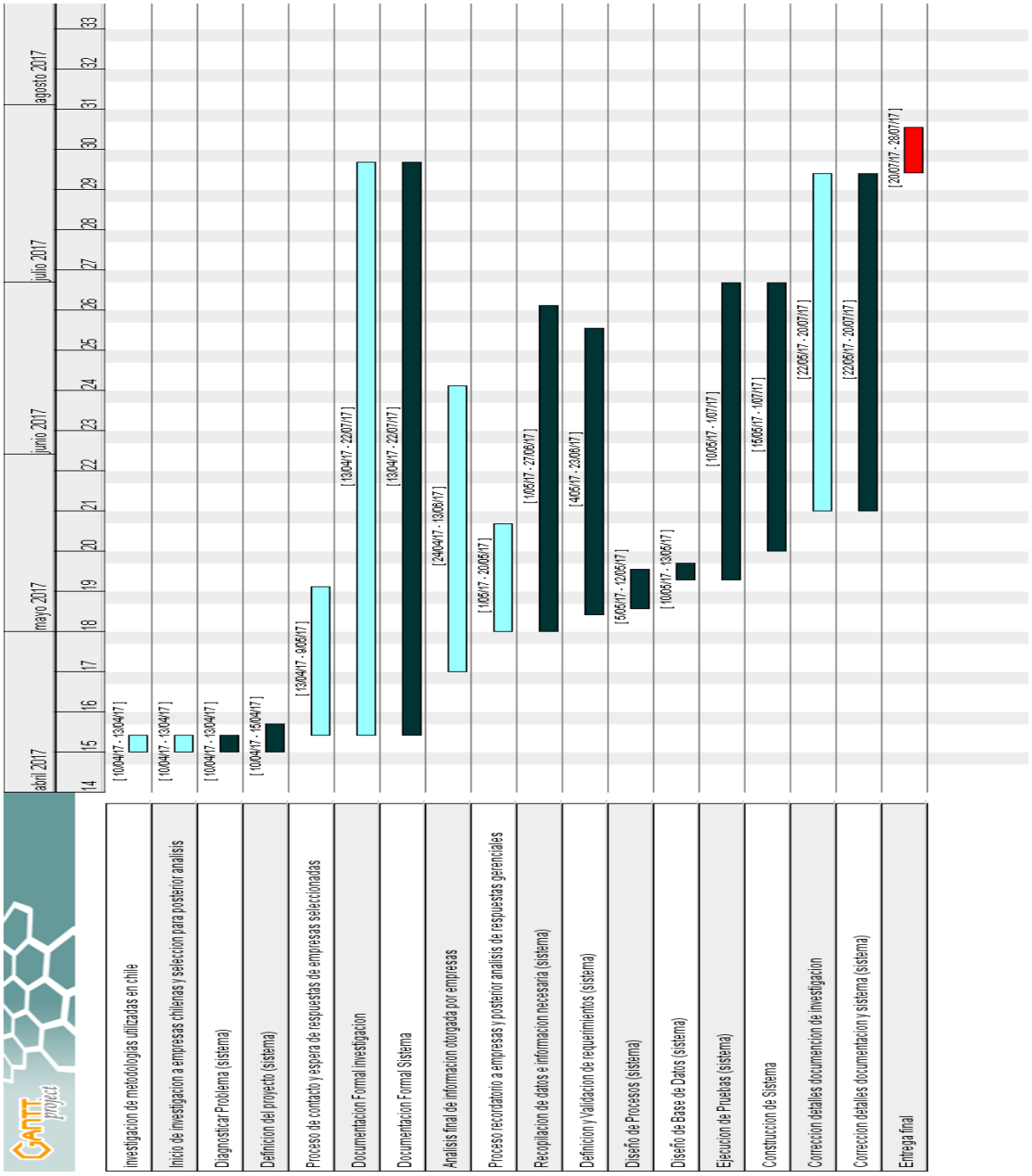
- Metodologías agiles
  - http://ingenieriadesoftware.mex.tl/52666_Presentacion.html
  - https://sites.google.com/site/ingsoportelogico/home

- Programación
  - https://www.youtube.com/user/DoingITeasyChannel
  - http://www.yiiframework.com/doc-2.0/guide-index.html
  - http://www.yiiframework.com/wiki/806/render-form-in-popup-via-ajax-create-and-update-with-ajax-validation-also-load-any-page-via-ajax-yii-2-0-2-3/

- Sistemas de recomendación
  - https://www.upf.edu/hipertextnet/numero-2/recomendacion.html
  - http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf
  - https://repositorio.uam.es/bitstream/handle/10486/660903/marina_pepa_sofia_tfg.pdf?sequence=1

15 ANEXO: PLANIFICACION INICIAL DEL PROYECTO

Nuestra planificación inicial tenia fecha de término el 28/07/2017. Por problemas de obtención de datos de las diferentes empresas pertenecientes al directorio de ejecutivos de Chile 2015 nuestro proyecto tardo más de lo planificado, ya que al no obtener una muestra aceptable de respuestas de dicho directorio se opto por realizar un estudio de campo para la obtención de nuevas empresas, lo cual no estaba contemplado en la planificación inicial.



16 ANEXO: METODOLOGÍA DE DESARROLLO DE SOFTWARE

16.1 Clasificación de grupos de metodología

A continuación se presentan los cuatro grandes grupos de metodologías de desarrollo de software (lineales, evolutivas, incrementales y ágiles) de los cuales se detallan las metodologías o modelos pertenecientes a dichos grupos, además de características propias de cada uno de estas.

16.1.1 Grupo de Metodologías Lineales

Dentro de este grupo se encuentra una serie de metodologías que en cierta forma, comparten algunas características. Dentro de las que se discutirán en este informe se encuentran el modelo cascada, cascada con iteración, cascada con sub-proyectos, structured system analysis and desingn method (SSADM) y Case de Oracle.

16.1.1.1 Modelo Cascada

Según Pressman (2010) y D'Onofrio (2010) el modelo en cascada es un proceso de desarrollo secuencial, en el que el desarrollo de software se concibe como un conjunto de etapas que se ejecutan una tras otra. Se le denomina así por las posiciones que ocupan las diferentes fases que componen el proyecto, colocadas una encima de otra, y siguiendo un flujo de ejecución de arriba hacia abajo, como una cascada. La siguiente figura muestra las fases del modelo.

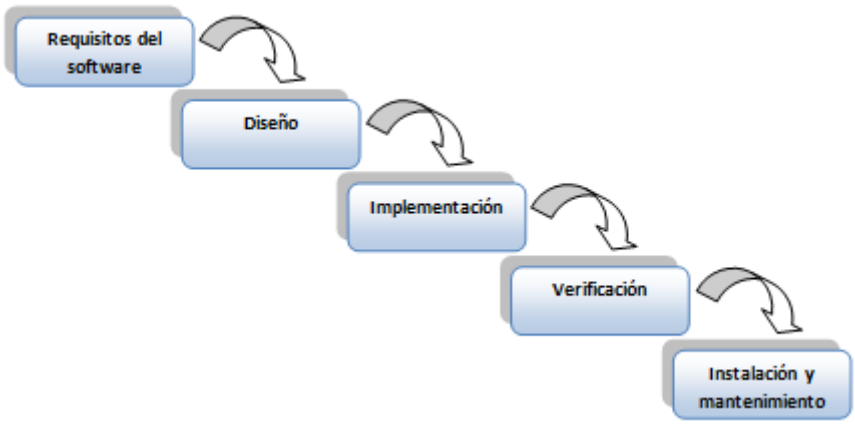


FIGURA N° 84: Modelo Cascada
Fuente: Elaboración propia a partir de Pressman (2010)

Fases del modelo

El modelo de desarrollo en cascada sigue una serie de etapas de forma sucesiva, la etapa siguiente empieza cuando termina la etapa anterior.

a) Requisitos del software

En esta fase se hace un análisis de las necesidades del cliente para determinar las características del software a desarrollar, y se especifica todo lo que debe hacer el sistema sin entrar en detalles técnicos. Hay que ser especialmente cuidadoso en esta primera fase, ya que en este modelo no se pueden añadir nuevos requisitos en mitad del proceso de desarrollo.

Por lo tanto, esta es la etapa en la que se lleva a cabo una descripción de los requisitos del software, y se acuerda entre el cliente y la empresa desarrolladora lo que el producto deberá hacer. Disponer de una especificación de los requisitos permite estimar de forma rigurosa las necesidades del software antes de su diseño. Además, permite tener una base a partir de la cual estimar el coste del producto, los riesgos y los plazos.

En el documento en el que se especifican los requisitos, se establece una lista de los requerimientos acordados. Los desarrolladores deben comprender de forma clara el producto que van a desarrollar. Esto se consigue teniendo una lista detallada de los requisitos, y con una comunicación fluida con el cliente hasta que termine el tiempo de desarrollo.

b) Diseño

En esta etapa se describe la estructura interna del software, y las relaciones entre las entidades que lo componen. Descompone y organiza el sistema en elementos que puedan elaborarse por separado, aprovechando las ventajas del desarrollo en equipo. Como resultado surge el SDD (Documento de Diseño del Software), que contiene la descripción de la estructura relacional global del sistema y la especificación de lo que debe hacer cada una de sus partes, así como la manera en que se combinan unas con otras.

Es conveniente distinguir entre diseño de alto nivel o arquitectónico y diseño detallado. El primero de ellos tiene como objetivo definir la estructura de la solución (una vez que la fase de análisis ha descrito el problema) identificando grandes módulos (conjuntos de funciones que van a estar asociadas) y sus relaciones. Con ello se define la arquitectura de la solución elegida. El segundo define los algoritmos empleados y la organización del código para comenzar la implementación.

c) Implementación

En esta fase se programan los requisitos especificados haciendo uso de las estructuras de datos diseñadas en la fase anterior. La programación es el proceso que lleva de la formulación de un problema de computación, a un programa que se ejecute produciendo los pasos necesarios para resolver dicho problema.

Al programar, tenemos que realizar actividades como el análisis de las condiciones, la creación de algoritmos, y la implementación de éstos en un lenguaje de programación específico.

d) Verificación

Como su propio nombre indica, una vez se termina la fase de implementación se verifica que todos los componentes del sistema funcionen correctamente y cumplen con los requisitos.

El objetivo de las pruebas es el de obtener información de la calidad del software, y sirven para encontrar defectos o bugs, aumentar la calidad del software, refinar el código previamente escrito sin miedo a romperlo o introducir nuevos bugs, etc.

e) Instalación y mantenimiento

Una vez se han desarrollado todas las funcionalidades del software y se ha comprobado que funcionan correctamente, se inicia la fase de instalación y mantenimiento. Se instala la aplicación en el sistema y se comprueba que funcione correctamente en el entorno en que se va a utilizar.

A partir de ahora hay que asegurarse de que el software funcione y hay que destinar recursos a mantenerlo. El mantenimiento del software consiste en la modificación del producto después de haber sido entregado al cliente, ya sea para corregir errores o para mejorar el rendimiento o las características. Para llevar a cabo correctamente la fase de mantenimiento, se necesita trazar un plan de antemano que nos prepare para todos los escenarios que puedan producirse durante esta fase. Para evitar futuros conflictos con el cliente, en el plan hay que especificar cómo los usuarios solicitarán las modificaciones o la corrección de errores, hacer una estimación del coste de la modificación de funcionalidades o corrección de errores, quién se encargará del mantenimiento, durante cuánto tiempo se dará soporte al software, etc.

Ventajas

Las principales ventajas de utilizar cascada en el proceso de desarrollo son:

- El tiempo que se pasa en diseñar el producto en las primeras fases del proceso puede evitar problemas que serían más costosos cuando el proyecto ya estuviese en fase de desarrollo.
- La documentación es muy exhaustiva y si se une al equipo un nuevo desarrollador, podrá comprender el proyecto leyendo la documentación.
- Al ser un proyecto muy estructurado, con fases bien definidas, es fácil entender el proyecto.
- Ideal para proyectos estables, donde los requisitos son claros y no van a cambiar a lo largo del proceso de desarrollo.

Desventajas

Las principales desventajas de utilizar cascada en el proceso de desarrollo son:

- En muchas ocasiones, los clientes no saben bien los requisitos que necesitarán antes de ver una primera versión del software en funcionamiento. Entonces, cambiarán muchos requisitos y añadirán otros nuevos, lo que supondrá volver a realizar fases ya superadas y provocará un incremento del coste.
- No se va mostrando al cliente el producto a medida que se va desarrollando, si no que se ve el resultado una vez ha terminado todo el proceso. Esto cual provoca inseguridad por parte del cliente que quiere ir viendo los avances en el producto. Los diseñadores pueden no tener en cuenta todas las dificultades que se encontrarán cuando estén diseñando un software, lo que conllevará rediseñar el proyecto para solventar el problema.
- Para proyectos a largo plazo, este modelo puede suponer un problema al cambiar las necesidades del usuario a lo largo del tiempo. Si por ejemplo, tenemos un proyecto que va a durar 5 años, es muy probable que los requisitos necesiten adaptarse a los gustos y del mercado.

16.1.2 Cascada con Iteración

Para Sommerville (2005) cascada con iteración es un modelo derivado del ciclo de vida en cascada. Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de recogida de requisitos. En qué consiste el modelo de cascada con iteración, consiste en la iteración de varios ciclos de vida en cascada. Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto.

Principales características

La retroalimentación forma parte importante en este modelo ya que reduce el riesgo a un fallo, de esta forma el sistema se adapta a las necesidades del cliente, con esto se asegura que la entrega sea lo que el cliente realmente necesita. La siguiente figura muestra las etapas asociadas a esta metodología.

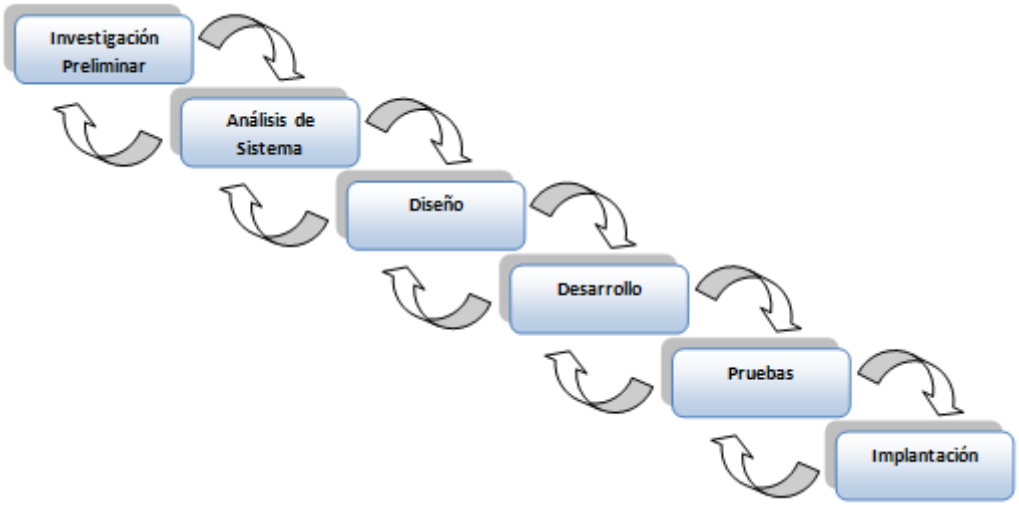


FIGURA N° 85: Cascada con Iteración
Fuente: Elaboración propia a partir de Sommerville (2005)

Fases del modelo

El modelo de desarrollo en cascada con iteración sigue una serie de iteraciones por etapas con fin de reducir los riesgos asociados, las etapas son las siguientes.

a) Investigación preliminar

En esta fase la investigación preliminar a su vez se descompone en dos sub-fases: comunicación y planeación.

Comunicación

Dentro de la sub- fase comunicación se detallan aspectos claves a considerar. Entre ellas se encuentran

- Aceptación proyecto, asignación fecha inicio y fecha arranque proyecto.
- Recursos del proyecto: cliente y proveedor
- Áreas clave y designación
- Planificación y calendario reuniones con responsables proyecto
- Identificación requerimientos a cubrir
- Comunicación al cliente metodología del proyecto
- Aprobación planificación y reuniones Confirmación cronogramas de reuniones.

Planeación

Todas las actividades orientadas a reconocer funcionalidades necesarias en el nuevo sistema. Entre ellas se encuentran:

- Diseño funcional del sistema.
- Ampliación al detalle del análisis requerimientos inicial.
- Definición procesos objetivo y análisis de posibles mejoras.
- Análisis diferencial entre la herramienta y los procesos objetivo (Análisis GAP).
- Funcionalidades ajenas al core o disponible mediante extensiones o desarrollos.
- Diseño interfaces con otras herramientas.
- Estrategia migración datos.
- Aprobación diseño funcional.

b) Análisis y diseño del sistema

En la fase de análisis y diseño del sistema se considera ajustes, parametrización y customización incremental.

Esta fase incluye además las siguientes sub-fases:

- Instalación entorno de desarrollo.
- Configuración y parametrización módulos y extensiones.
- Diseño y desarrollo funcionalidad adicional.
- Adaptación de informes.
- Desarrollo interfaces con otras herramientas
- Realización de pruebas y feedback con implantador y/o fabricante.

- Instalación entorno de producción.
- Diseño, desarrollo y adaptación procesos migración de datos.
- Validación por parte del cliente. Vuelta al principio y nueva iteración.

c) Desarrollo y pruebas

En la fase de desarrollo y pruebas del sistema se incluyen las siguientes sub-fases:

- Código y prueba.
- Preparación entornos de prueba y producción.
- Certificación del sistema y su integración con otras herramientas.
- Pruebas de rendimiento.
- Configuración de seguridad.
- Aceptación del sistema. Validación y aceptación del proyecto.

d) Implantación

En la fase de implementación y posterior puesta en marcha se consideran relevantes las siguientes sub-fases:

- Entrega y retroalimentación.
- Puesta en producción del sistema. Arranque del nuevo sistema.
- Corrección de incidencias.
- Soporte a usuarios en operativa diaria.
- Activación contrato soporte, mantenimiento y cierre del proyecto.
- Labores de mantenimiento y actualización del sistema.

Ventajas

Sus principales ventajas al utilizar cascada con iteración son:

- No hace falta que los requisitos estén totalmente definidos al inicio del desarrollo.
- Igual que otros modelos similares tiene las ventajas propias de realizar el desarrollo en pequeños ciclos.
- Los productos desarrollados con este modelo tienen una menor probabilidad de fallar.
- Se obtiene un aprendizaje en cada iteración.

Desventajas

Sus principales desventajas al utilizar cascada con iteración son:

- Al no ser necesario tener los requisitos definidos desde el principio, puede verse también como un inconveniente ya que pueden surgir problemas relacionados con la arquitectura.
- Al tener que entregar avances de los módulos esto puede provocar que el tiempo de desarrollo se aplace.

Cuando es apropiado utilizar este modelo

Se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario, suele ser utilizado en iteraciones cortas de 2 a 4 semanas incrementa la productividad del proyecto.

El equipo aprende a calcular la velocidad de desarrollo mientras el cliente proyecta cuantas iteraciones se necesitan para tener cada entrega, en función de la velocidad de desarrollo del equipo y tomar decisiones al respecto.

Permite gestionar y sincronizar de manera sencilla las necesidades del proyecto con respecto a las de otros proyectos.

Las iteraciones coincidiendo con meses naturales permiten sincronizar el trabajo del equipo con el de otros departamentos y con el resto de la organización.

16.1.3 Cascada con sub-proyectos

Según Pressman (2010) y Sommerville (2005) la metodología cascada con sub-proyectos se origina en la década de 1960 con el fin de desarrollar un sistema de negocio eficiente en una época de conglomerados empresariales.

La idea fue siempre hacerlo en una forma estructurada y metódica, reiterando cada una de las etapas del ciclo de vida del software.

Se entiende como una variación sobre el ciclo de vida en Cascada del software, denominada Cascada con Sub-proyectos, porque permite la ejecución de algunas de las tareas de la cascada en paralelo, siempre que se haya realizado una cuidadosa planificación. La Figura 86 muestra el diagrama que representa la metodología.

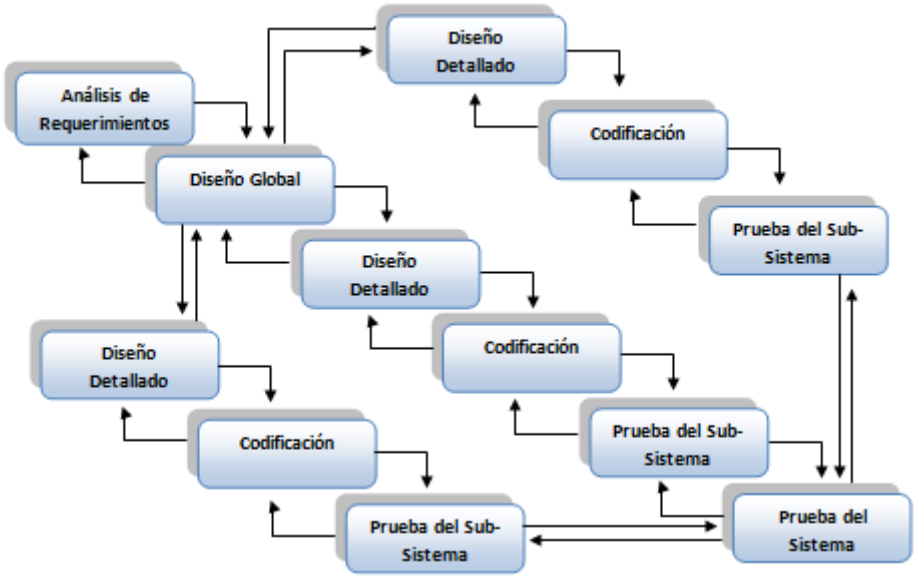


FIGURA N° 86: Cascada con sub-proyectos
Fuente: elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Fases del modelo

El modelo de desarrollo en cascada con sub-proyectos sigue una serie de iteraciones en paralelo con el fin de llegar a la etapa final de prueba del sistema, las etapas son las siguientes.

a) Análisis de requerimientos

En esta fase se analizan las necesidades de los usuarios finales del software para determinar qué objetivos debe cubrir, de esta fase surge un documento llamado SRD (Documento de especificación de requisitos).

También se debe consensuar todo lo que se requiere del sistema y será aquello lo que seguirá en las siguientes etapas, no pudiéndose requerir nuevos resultados a mitad del proceso de elaboración del software.

b) Diseño global (Diseño del Sistema)

Durante el proceso de diseño del sistema se distinguen cuáles son los requerimientos de software y cuáles los de hardware, después se establece una arquitectura completa del sistema.

Durante el diseño del software se identifican los subsistemas que componen el sistema y se describe cómo funciona cada uno y las relaciones entre éstos.

c) Diseño detallado (Diseño del programa)

En la fase de diseño del programa se realiza la creación de algoritmos para el funcionamiento del programa y se análisis de herramientas de programación

d) Codificación

En esta fase se inicia la codificación e implementación del código.

e) Pruebas del subsistema

En esta fase se inician ensayos y pruebas de errores, además de creación de bibliotecas.

f) Pruebas del sistema

Unión de los elementos programados para la composición del programa final

Comprobación de funcionamiento. Una vez realizadas las pruebas finales del sistema se puede dar paso a la implementación considerando los siguientes aspectos:

- Ejecución del programa.
- Mantención
- Se destina la mayor parte del presupuesto en esta parte

Ventajas

Los principales beneficios al desarrollar utilizando cascada con sub-proyectos son:

- Genera signos visibles de progreso, que se utilizan cuando existe una demanda en la velocidad del desarrollo.
- Se puede tener más gente trabajando al mismo tiempo.
- Permite entender bien el problema antes de la implementación final.

Desventajas

Las principales desventajas de utilizar cascada con sub-proyectos en el proceso de desarrollo son:

- El cliente puede quedar convencido con las primeras versiones y probablemente no vea la necesidad de completar el sistema o re-diseñarlo con la calidad necesaria.
- Pueden surgir dependencias entre los distintos sub-proyectos pero solo se tendrían que administrar los tiempos.
- Requiere trabajo del cliente para evaluar los distintos prototipos y traducirlo en nuevos requisitos.

- Requiere un tiempo adicional para el desarrollo del sistema.
- A la hora de implementación, no se sabe exactamente cuánto será el tiempo de desarrollo ni cuantos prototipos se tienen que desarrollar el proyecto final.

Cuando es apropiado utilizar este modelo

La metodología de cascada con sub-proyectos se utiliza por el principio de agilizar las partes menos complejas del proyecto sin tener que esperar a que finalice una parte más compleja, dividiendo el proyecto completo en sub-proyectos que mantendrán el método cascada para su desarrollo.

Por lo tanto, el modelo en cascada sólo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema (Sommerville, 2005).

Aunque la metodología en cascada se utiliza en un alto porcentaje de los proyectos actuales, es aconsejable que el método de sub-proyectos se aplique a proyectos complejos con una etapa de desarrollo de entre 1 a 3 años.

16.1.4 Structured Systems Analysis and Desing Method (SSADM)

Según Goodland and Riha (1999) y la publicación de Company Model Systems Ltd. (2002) el modelo SSADM es un enfoque de sistemas para el análisis y diseño de sistemas de información. SSADM es un método de cascada para el análisis y diseño de sistemas de información. Se considera que SSADM representa el pináculo del enfoque riguroso en la documentación hacia el diseño del sistema que contrasta con métodos ágiles como DSDM o SCRUM.

SSADM es una aplicación en particular y se basa en el trabajo de las diferentes escuelas de análisis estructurados métodos y desarrollo, Metodología blanda de sistemas, diseño estructurado, Método estructurado, Programación Estructurada, y análisis estructurado.

Técnicas

Las tres técnicas más importantes que se utilizan en SSADM son las siguientes:

Modelado de datos lógicos

El proceso de identificación, modelado y documentación de los requisitos de datos del sistema que está siendo diseñado. El resultado es un modelo de datos que contiene las entidades (cosas de las que una empresa necesita para registrar la información), atributos (datos sobre las entidades) y relaciones (asociaciones entre las entidades).

Modelado de flujo de datos

El proceso de identificar, modelar y documentar cómo los datos se mueven alrededor de un sistema de información. El Modelado de flujo de datos examina los procesos (actividades que transforman los datos de una forma a otra), almacenes de datos (las zonas de espera de los datos), entidades externas (lo que envía los datos a un sistema o recibe datos de un sistema), y los flujos de datos (rutas por el cual los datos pueden fluir).

Modelado Entidad Evento

Es un proceso de dos hebras: Behavior Modeling Entidad, identificar, modelar y documentar los eventos que afectan a cada entidad y la secuencia (o historia de vida) en el que se producen estos eventos, y Modelado de eventos, diseñando para cada caso el proceso para coordinar las historias de vida entidad.

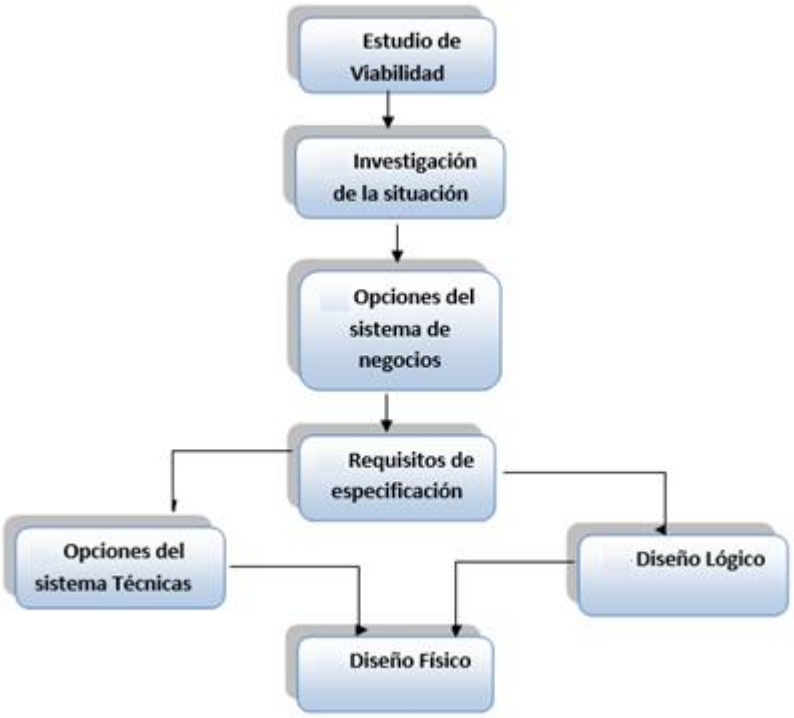


FIGURA N° 87: SSADM
Fuente: Elaboración propia a partir Goodland and Riha (1999)

Fases del modelo

El método SSADM implica la aplicación de una secuencia de tareas de análisis, documentación y diseño. La Figura 87 muestra el resumen de las etapas consideradas en esta metodología.

a) Estudio de viabilidad

Con el fin de determinar si es o no viable un determinado proyecto, tiene que haber algún tipo de investigación sobre los objetivos y las implicaciones del proyecto. Para los proyectos de muy pequeña escala esto puede no ser necesario en absoluto ya que el alcance del proyecto es fácil de entender. En proyectos de mayor envergadura, la viabilidad se puede hacer, pero en un sentido informal, ya sea porque no hay tiempo para un estudio formal o porque el proyecto es un "must-have", y tendrá que ser hecho de una manera u otra. Cuando un estudio de viabilidad se lleva a cabo, hay cuatro áreas principales de consideración:

- Técnica - ¿es el proyecto técnicamente posible?
- financiera - ¿puede permitirse el negocio para llevar a cabo el proyecto?
- Organizacional - ¿será el nuevo sistema sea compatible con las prácticas existentes?
- Ético - ¿es el impacto del nuevo sistema socialmente aceptable?

Para responder a estas preguntas, el estudio de viabilidad es efectivamente una versión condensada de un análisis de sistemas totalmente soplado y diseño. Los requisitos y los usuarios se analizan en cierta medida, algunas opciones de negocio son elaboradas e incluso algunos detalles de la ejecución técnica. El producto de esta etapa es un documento formal del estudio de factibilidad. SSADM especifica las secciones que el estudio debe contener incluyendo cualquier modelos preliminares que se han construido y también los detalles de las opciones de excluidos y los motivos de su rechazo.

b) Investigación de la situación actual

Esta es una de las etapas más importantes de SSADM. Los desarrolladores de SSADM entendieron que en casi todos los casos hay algún tipo de sistema de corriente incluso si está compuesta en su totalidad de las personas y de papel. A través de una combinación de entrevistar a los empleados, cuestionarios, observaciones de circulación y documentación existente, el analista llega a la comprensión completa del sistema, ya que se encuentra al principio del proyecto. Esto sirve para muchos propósitos:

- El analista aprende la terminología de la empresa, lo que los usuarios hacen y cómo lo hacen.
- El viejo sistema proporciona los requisitos básicos para el nuevo sistema.
- Fallas, errores y áreas de ineficiencia se resaltan y sus correcciones se añaden a los requisitos.
- El modelo de datos se puede construir.
- Los usuarios se involucran y aprenden las técnicas y modelos del analista.
- Los límites del sistema se pueden definir.

Los productos de esta fase son:

- Catálogo de Usuarios describe todos los usuarios del sistema y cómo interactuar con él.
- Catálogo de Necesidades detalla todos los requisitos del nuevo sistema.
- Servicios actuales Descripción compuso más de
- Entorno actual lógica de datos Modelo
- Diagrama de Contexto (DFD)
- Conjunto nivelado de DFD para la corriente sistema lógico
- Diccionario de datos completo incluyendo la relación entre los almacenes de datos y entidades
- Para producir los modelos, el analista trabaja a través de la construcción de los modelos que hemos descrito. Sin embargo, el primer conjunto de diagramas de flujo de datos (DFD) son el modelo físico actual, es decir, con todos los detalles de cómo se implementa el sistema antiguo. La versión final es el modelo lógico actual que es esencialmente la misma que la corriente física pero con toda referencia a la aplicación omitida junto con las redundancias como la repetición de la información que compone los usuarios y los requisitos catálogos.

c) Opciones del sistema de negocios

Tras investigar el sistema actual, el analista debe decidir sobre el diseño general del nuevo sistema. Para hacer esto, él o ella deben usar las salidas de la etapa anterior, se desarrolla un conjunto de opciones de negocios del sistema. Estas son diferentes formas en que el nuevo sistema podría ser producido variando de no hacer nada para tirar el viejo sistema en su totalidad y la construcción de

uno totalmente nuevo. El analista puede realizar una sesión de lluvia de ideas para que se generen tantas y diversas ideas como sea posible.

Las ideas se recogen entonces para formar un conjunto de dos o tres opciones diferentes que se presentan al usuario. Las opciones en cuenta lo siguiente:

- El grado de automatización
- El límite entre el sistema y los usuarios
- La distribución del sistema, por ejemplo, ¿es centralizada a una oficina o hacia fuera a través de varios?
- Costo / beneficio
- Impacto del nuevo sistema

Cuando sea necesario, la opción será documentada con una estructura de datos lógica y un diagrama de flujo de datos de nivel 1.

Los usuarios y analista juntos escogen una opción de negocio único. Esta puede ser una de las ya definidas o puede ser una síntesis de los diferentes aspectos de las opciones existentes. La salida de esta etapa es la opción seleccionada de negocios única, junto con todas las salidas de la etapa de factibilidad.

d) Requisitos de especificación

Esta es probablemente la etapa más compleja en SSADM. Usando los requisitos desarrollados en el estudio de viabilidad y trabajando en el marco de la opción de negocio seleccionado, el analista debe desarrollar una especificación lógica completa de lo que el nuevo sistema debe hacer. La especificación debe estar libre de error, ambigüedad e inconsistencia. Por lógica, nos referimos a que la especificación no dice cómo se implementará el sistema, sino que describe lo que el sistema va a hacer.

Para producir la especificación lógica, el analista construye los modelos lógicos necesarios tanto para los diagramas de flujo de datos (DFDs) y el modelo de datos lógicos (LDM), que consiste en la estructura lógica de datos (contemplados en otros métodos como diagramas entidad relación) y una descripción completa de los datos y sus relaciones. Estos se utilizan para producir la definición de funciones de todas las funciones que los usuarios requieren del sistema, una entidad de vida-Historias que describen todos los acontecimientos a través de la vida de una entidad, y el efecto de Correspondencia Diagramas que describen cómo interactúa cada uno de los eventos con todas las entidades pertinentes. Estos son continuamente comparan con los requisitos y en caso necesario,

se añaden los requisitos para y completados. El producto de esta etapa es un documento completo con la especificación de requisitos que se compone de:

- El catálogo de datos actualizada
- El catálogo de requisitos actualizado
- La especificación de procesamiento que a su vez se compone de
- Rol de usuario matriz de funciones
- Definiciones de funciones
- Modelo lógico de datos requerido
- Historias de vida entidad
- Diagramas efecto correspondencia
- Aunque algunos de estos artículos pueden ser desconocidos para usted, está más allá del alcance de esta unidad para entrar en ellos con gran detalle.

e) Opciones del sistema técnicas

Esta primera etapa es una implementación física del nuevo sistema. Al igual que las opciones del sistema de negocio, en esta etapa se generan un gran número de opciones para la aplicación del nuevo sistema. Esto se perfeccionó hasta dos o tres usuario para presentar desde que se elige la opción o sintetizado final. Sin embargo, las consideraciones son seres muy diferentes:

- Las arquitecturas de hardware
- El software a utilizar
- E costo de la implementación
- La dotación de personal necesaria
- Las limitaciones físicas, tales como un espacio ocupado por el sistema
- La distribución incluidas las redes que pueden requerir
- El formato general de la interfaz ofrecida a los usuarios
- Todos estos aspectos deben también ajustarse a las restricciones impuestas por la empresa, como el dinero y la estandarización de hardware y software disponibles.
- La salida de esta etapa es una opción de sistema técnico elegido.

f) Diseño lógico

Aunque el nivel anterior especifica los detalles de la ejecución, los resultados de esta etapa son independientes de la implementación y se concentran en los requisitos de la interfaz de la computadora humana. El diseño lógico especifica los principales métodos de interacción en términos de estructuras de menús y estructuras de mando.

Un área de actividad es la definición de los diálogos de usuario. Estas son las principales interfaces con que los usuarios podrán interactuar en el sistema. Otras actividades están relacionadas con el análisis de los efectos de actualización del sistema tanto de los acontecimientos en la necesidad de hacer consultas sobre los datos en el sistema. Ambos utilizan los eventos, descripciones de las funciones y diagramas efecto correspondencia producidos en el punto c (opciones del sistema de negocios), para determinar con precisión cómo actualizar y leer datos de una manera consistente y segura.

El producto de esta etapa es el diseño lógico que se compone de:

- Catálogo de datos
- Estructura de datos lógica requerida
- Modelo de proceso lógico - incluye diálogos y modelo para los procesos de actualización y consulta
- El estrés y momentos de flexión.

g) Diseño físico

Esta es la etapa final en la que todas las especificaciones lógicas del sistema se convierten en las descripciones del sistema en términos de hardware y software real. Esta es una etapa muy técnica y un simple resumen se presenta aquí.

La estructura lógica de los datos se convierte en una arquitectura física en términos de estructuras de base de datos. Se especifica la estructura exacta de las funciones y la forma en que se implementan. La estructura de datos física se optimiza cuando sea necesario para satisfacer los requisitos de tamaño y rendimiento.

El producto es un diseño físico completo que indica a los ingenieros de software la manera de construir el sistema en detalles específicos de hardware y software y para los estándares apropiados.

Ventajas

Un enfoque metodológico del estudio de una empresa (o un área de una empresa) a partir de un número de diferentes perspectivas es más probable que proporcione una comprensión más completa de la empresa, sus procesos y datos, que los enfoques "ad-hoc" que se utilizaron previamente. Esto a su vez debería (se esperaba) conducen a los sistemas que son más completos y correctos.

Desventajas

El enfoque SSADM de tener que completar una fase antes de comenzar la siguiente etapa que lleve a algunos proyectos en lo que se conoce como "parálisis de análisis". ¿Qué se quiere decir con esto? es que debido a que una empresa y sus procesos nunca permanecen igual por mucho tiempo, el equipo de sistemas continuamente tendría que revisar el análisis y diseño de productos para su modificación, causando (a veces muy largo) las demoras en llegar a las fases de la programación y entrega del sistema. En reconocimiento de esto, las versiones posteriores de la Metodología introdujeron un enfoque más opcional / dinámica al proceso.

También hay un coste en la formación de las personas a utilizar las técnicas. La curva de aprendizaje puede ser considerable si se utiliza el método de integración global, ya que no sólo hay varias técnicas de modelado para llegar a un acuerdo con, pero también hay una gran cantidad de normas para la preparación y presentación de documentos.
En resumen, el uso de esta metodología implica una tarea significativa que puede no ser adecuado para todos los proyectos.

16.1.5 Case de Oracle

Según Kress (2014) el modelo Case de Oracle se define como una metodología para el diseño de sistemas, cuya función principal es apoyar el trabajo manual y/o repetitivo, mediante automatización de tareas. El Case Method de Oracle ha pasado por varias transformaciones. Actualmente se le conoce como CDM (Custom Development Methodology) y su adaptación para java es JCDM.

CDM es un método intensivo en documentación con estándares predefinidos.
La documentación es mucha y requiere validaciones permanentes por parte de los clientes, con el fin de garantizar que los requerimientos se están cumpliendo adecuadamente hasta llegar a la aplicación final.

El volumen de documentación variará dependiendo del tamaño y la complejidad de la aplicación en desarrollo.

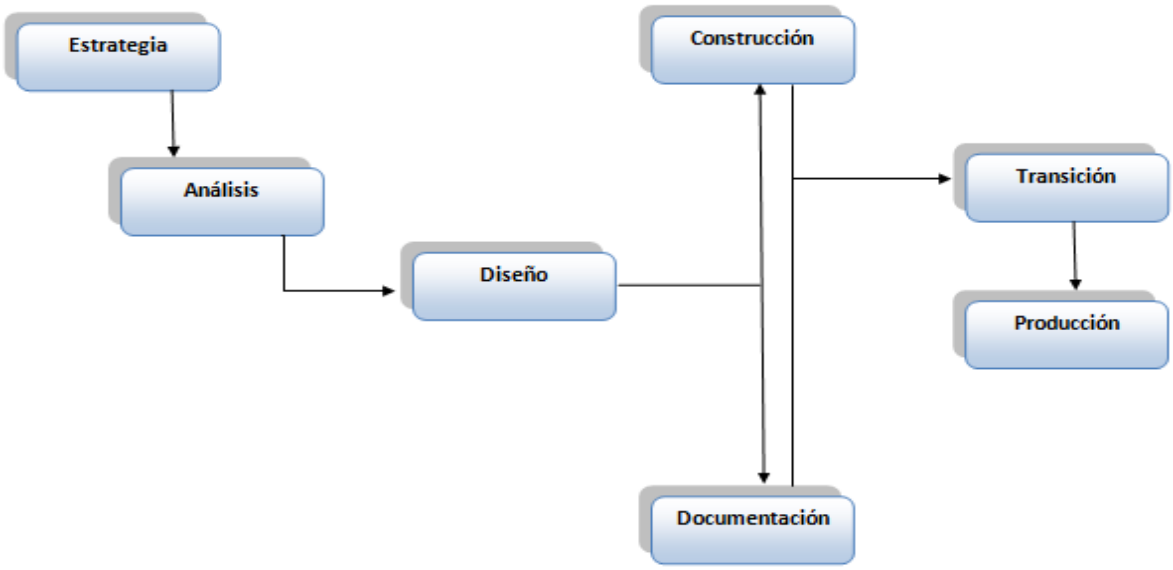


FIGURA N° 88: Case de Oracle
Fuente: Elaboración propia a partir Kress (2014)

Fases del modelo

El modelo Case de Oracle implica una secuencia de tareas de estrategia, análisis, diseño, construcción, documentación, transición y producción. La Figura 88 muestra el resumen de las etapas consideradas en este modelo.

a) Estrategia: En esta fase se definen los requisitos del sistema.

b) Análisis: En esta fase se formulan los requisitos de aplicación.

c) Diseño: En esta fase se inicia la conversión de los requisitos en especificaciones detalladas del sistema.

d) Construcción: En esta fase se escriben y prueban las aplicaciones. Las fases de construcción y documentación son procesos que se realizan en el mismo periodo de tiempo.

e) Documentación: En esta fase se inicia la documentación de aplicación y modelos.

f) Transición: En esta fase se inician los preparativos para el inicio de la operación (instalación del sistema de aplicación).

g) Producción: En esta fase el producto queda a disposición del usuario. Ya sea empaquetado para la venta o disponible en internet para descargar o instalar en el entorno del usuario, brindado apoyo y control a la aplicación y planificando las futuras ampliaciones funcionales.

Ventajas

Las principales ventajas de utilizar el método case de oracle son:

- Facilita una mejor comprensión del sistema.
- Procura desambiguar los requerimientos del cliente, mejorando la comunicación con éste.
- El sistema es descrito de manera más precisa.
- Aumenta la productividad reduciendo la incertidumbre.
- El sistema se asegura matemáticamente que es correcto según las especificaciones.
- Asegura un producto final de mayor calidad al cumplir de mejor manera los requerimientos.

Desventajas

Principales desventajas de utilizar case de oracle en del desarrollo son:

- El desarrollo de herramientas que apoyen el uso de métodos formales es complicado, y las aplicaciones resultantes son incómodas para los usuarios.
- Los investigadores por lo general desconocen la realidad de la industria. Carecen de inmersión sobre el cómo se desarrolla el actual proceso o producto.
- La aplicación de métodos formales encarece los productos y ralentiza su desarrollo.

Cuando es apropiado utilizar el modelo

Primeramente el proyecto debe de ser mediano o grande, para justificar a la excesiva documentación. Permite generar un seguimiento intensivo de las diferentes etapas o fases de desarrollo, para ello, se realizan un conjunto de tareas que se agrupan en procesos. Cada proceso hace parte de cada una de las fases de desarrollo y se reporta mediante un documento denominado Entregable.

16.1.6 Grupo de Metodologías Evolutivas

Dentro del grupo de metodologías evolutivas, se han considerado las siguientes: prototipo evolutivo, entrega por etapas, modelo espiral, cascada con reducción de riesgo.

16.1.7 Prototipo Evolutivo

Según Pressman (2010) y Sommerville (2005) el modelo de desarrollo Prototipo Evolutivo construye una serie de grandes versiones sucesivas de un producto. El proceso comienza con una serie de requisitos, a partir de los cuales se desarrollan una serie de prototipos para aclarar aspectos particulares de los requerimientos del usuario, se exponen a este y se van refinando paso a paso.

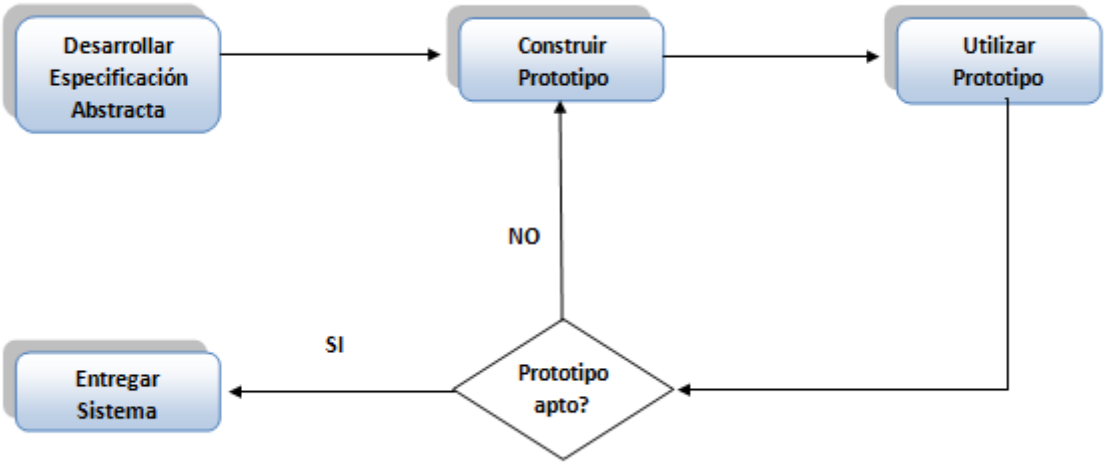


FIGURA N° 89: Prototipo Evolutivo
Fuente: Elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Como se hace un prototipo

En la Figura 89 resume las fases de un prototipo evolutivo, las que se especifican a continuación.

a) Desarrollar especificación abstracta: Se definen objetivos generales del software, identifica requerimientos que conozca y detecta áreas que requieren mayor atención.

b) Construir prototipo: Se planea rápidamente una iteración para hacer el prototipo.
Se lleva a cabo el modelado centrándose en aspectos del software que serán visibles al usuario.
Se construye el prototipo.

c) Utilizar prototipo: Se entrega y es evaluado por los participantes que dan retroalimentación para mejorar los requerimientos.

d) Entregar sistema: Una vez que el prototipo es aceptado se realiza la entrega de este.

Cuando un prototipo se desarrolla con el sólo propósito de precisar y entender mejor y las necesidades del cliente y después no se va a aprovechar ni total ni parcialmente en la implementación del sistema final, se habla de un prototipo desechable.

Ventajas

Entre las ventajas de desarrollar prototipos se encuentran:

- Es ideal para sistemas que no tiene bien definidos sus requerimientos.
- La especificación se puede mostrar de forma creciente.
- Cuando el cliente no puede especificar el conjunto total de los requerimientos.
- Cuando no se logra identificar de forma apropiada el área de aplicación.
- Cuando los desarrolladores no están seguros de la arquitectura o los algoritmos adecuados a utilizar.
- Se puede utilizar en distintos tipos de metodologías. Por ejemplo, en el modelo cascada puede ser usado para administrar cada esfuerzo de desarrollo.

Desventajas

Las desventajas de desarrollar prototipos:

- Este modelo es que está enfocado a la producción de prototipos.
- La estructura es más deficiente (a menudo).
- El progreso no es visible.
- Existe una imposibilidad de conocer al inicio del proyecto lo que se tardará en crear un producto aceptable.
- Esta aproximación puede convertirse fácilmente en una excusa para realizar el desarrollo con el modelo de codificar y corregir.

Cuando es apropiado es utilizar este modelo

Se suele utilizar en proyectos en los que los requisitos no están claros por parte del usuario. También es conveniente para sistemas pequeños y de tamaño medio (500.000 líneas de código).Ya que para sistemas más grandes el proceso evolutivo resulta agudo y complejo.

16.1.8 Entrega Evolutiva

Para Pressman (2010) y Sommerville (2005) en una Entrega Evolutiva se desarrolla una versión de producto, se muestra al cliente y se refina el producto en función de la realimentación del cliente. Es un modelo que se encuentra entre el prototipado evolutivo y la entrega por etapas, puesto que se desarrolla una versión del producto, se muestra al cliente, se refina el producto en función de los comentarios del cliente. El parecido entre ambos modelos depende de hasta qué punto se lleva a cabo una planificación para adaptarse a las solicitudes de los clientes.

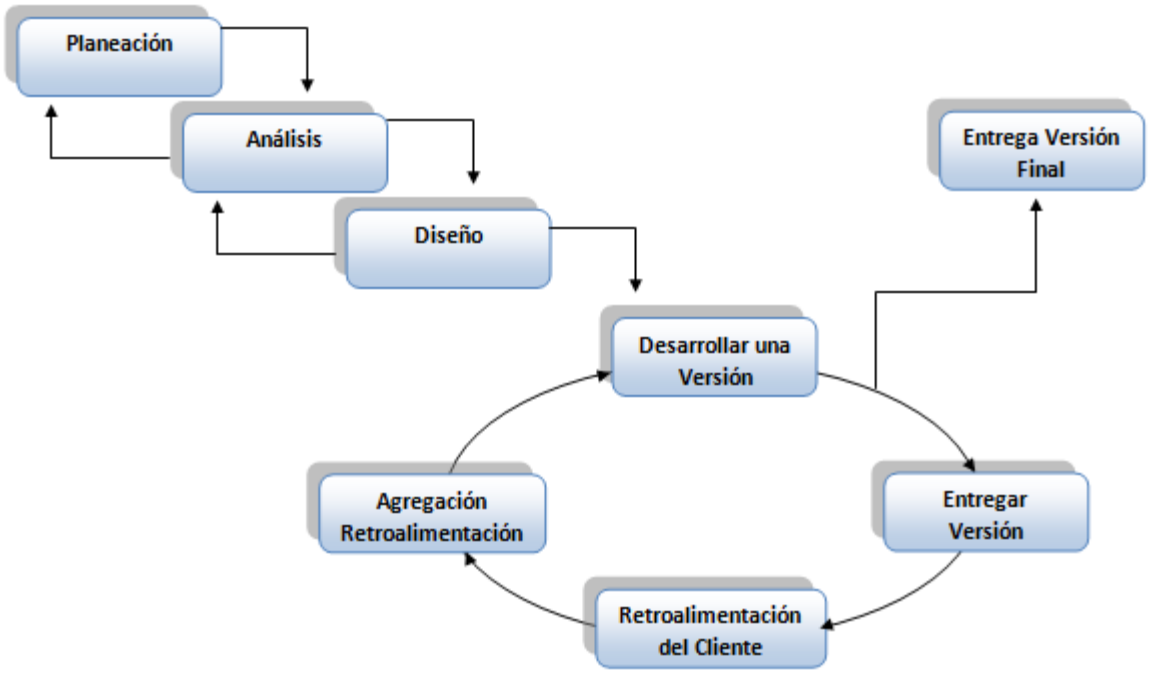


FIGURA N° 90: Entrega Evolutiva
Fuente: Elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Fases del modelo

En la Figura 90 se resumen las fases de entrega evolutiva, de las cuales se detallan planeación, análisis, diseño, desarrollar versión, entregar versión, retroalimentación del cliente, agregación retroalimentación, entrega versión final.

a) Planeación: Propuesta inicial de las necesidades que el cliente busca solucionar a través de un proyecto informático. En esta etapa se define los objetivos globales del proyecto.

b) Análisis: Recoger los requerimientos o necesidades del cliente y determinar los requisitos del producto/servicio así como los recursos (materiales, financieros, humanos), equipo y herramientas que se utilizaran a lo largo del desarrollo del software.

c) Diseño: primer bosquejo del software que busca satisfacer los requerimientos funcionales y no funcionales entregados por el cliente.

d) Desarrollar una versión: etapa centrada en la programación de las distintas funcionalidades requeridas para el software.

e) Entregar versión: adjuntar documentación del software junto con el prototipo desarrollado en la etapa anterior para ser evaluada por el cliente.

f) Retroalimentación del cliente: el cliente realiza pruebas funcionales con usuarios para verificar el correcto manejo de la aplicación concluyendo con una evaluación con modificaciones que espera del producto para satisfacer de mejor manera los requerimientos o bien, agregar requerimientos para mejorar la experiencia para el usuario final al utilizar el software a través de un informe.

g) Agregación retroalimentación: el equipo recibe la retroalimentación del cliente para mejorar las funcionalidades diseñadas o crear nuevas, con el objetivo de satisfacer requerimientos nuevos propuesto por los usuarios.

h) Entrega versión final: última etapa en donde se realiza la entrega del software al cliente además de la documentación relativa a este.

Ventajas

Ventajas de desarrollar utilizando entregas evolutivas:

- Es iterativo, en cada ciclo se entrega al cliente un producto operacional para ser evaluado.
- Gestión de riesgos técnicos, por ejemplo: disponibilidad de hardware específico Permite variar el personal asignado en cada ciclo.
- No tiene tiempo límite, se trabaja hasta cumplir con el presupuesto.
- La especificación puede desarrollarse de forma creciente. Los usuarios y desarrolladores logran un mejor entendimiento del sistema, esto se refleja en una mejora de la calidad del software.
- Es más efectivo que el modelo de cascada, ya que cumple con las necesidades inmediatas del cliente.

Desventajas

Desventajas de desarrollar utilizando entregas evolutivas:

- La primera iteración puede plantear los mismos problemas que un modelo lineal secuencial.
- Cada entrega implica un aumento en código, documentación y manuales de uso.
- La cantidad de iteraciones está directamente relacionada con el presupuesto, por lo cual puede que se entregue una versión inconclusa del proyecto.
- La mala documentación puede provocar retardos en las iteraciones, debido a las rotaciones del personal.

Cuando es apropiado utilizar este modelo

Se recomienda utilizar este modelo cuando se utiliza el prototipado evolutivo y se necesita un poco más de libertad en la decisiones o cuando se ocupa entrega por etapas y quieren darle más poder al usuario. Se sugiere usar el modelo cuando los requerimientos entregados por el cliente no están claramente definidos o bien son sujetos a cambios y se dispone de tiempo de sobra para cumplirlos.

16.1.9 Modelo espiral

Según Boehm (1987) el modelo espiral es un modelo de tipo evolutivo, ya que combina el modelo clásico con el diseño de prototipos, donde el desarrollo es iterativo, es decir, el software se desarrolla en una serie de versiones incrementales. Durante las primeras iteraciones la versión incremental suele ser un modelo en papel o un prototipo, luego de avanzar el proyecto, en las últimas iteraciones, se producen versiones cada vez más completas y al final, el sistema ya queda totalmente funcional.

Un modelo espiral se divide en fases o regiones, estas comienzan con la determinación de objetivos tanto funcionales como de rendimiento. Luego se enumeran algunas formas posibles de alcanzar estos objetivos identificando las fuentes de riesgos posibles para luego dar paso a la resolución de estos riesgos y comenzar con las actividades de desarrollo, para finalizar el ciclo con la planificación del siguiente ciclo de la espiral, repitiendo estos pasos hasta que el sistema esté completamente funcional.

Modelo original de Boehm: No posee un número definido de iteraciones, ya que estas son definidas por el equipo de desarrollo del proyecto.

FIGURA N° 91: Modelo Espiral
Fuente: Elaboración propia a partir de Boehm (1987)

Fases del modelo

En la Figura 91 se definen las iteraciones del modelo espiral las que están divididas en 4 etapas: planificación, análisis de riesgo, ingeniería, evaluación.

a) Planificación: consiste en la determinación de los objetivos, alternativas y restricciones.

b) Análisis del riesgo: análisis de alternativas e identificación/resolución de riesgos.

c) Ingeniería: consiste en el desarrollar y probar el producto para llevarlo al siguiente nivel, que puede ser el final.

d) Evaluación: es la valoración de los resultados obtenidos por parte del cliente.

Ventajas

Dentro de las ventajas de utilizar modelo espiral tenemos:

- Este método puede adaptarse y aplicarse a lo largo del software.
- El desarrollador y cliente pueden comprender y reaccionar de mejor forma ante los riesgos en cada una de las etapas.
- En la utilización de grandes sistemas ha doblado su productividad.
- Sufrir retrasos corre un riesgo menor en la entrega del proyecto, porque se comprueban los conflictos tempranamente y existe la forma de corregir a tiempo.

Desventajas

Dentro de las desventajas de utilizar modelo espiral tenemos:

- Debido a su elevada complejidad no se aconseja utilizarlo en pequeñas empresas.
- Requiere experiencia en la identificación de riesgos.
- Es un Método costoso.
- Genera mucho tiempo en la elaboración del sistema.
- Requiere la participación continua del cliente.

Cuando es apropiado utilizar este modelo

Este modelo es el indicado para desarrollar software con diferentes versiones actualizadas como se hace con los programas modernos de PC´s.

La ingeniería puede desarrollarse a través del ciclo de vida clásico o el de construcción de prototipos. Este es el enfoque más realista recientemente.

Existe una gran combinación de factores que imposibilitan realizar una verificación minuciosa de todas las posibles situaciones de ejecución que se puedan presentar. Poniendo como ejemplo la creación de un sistema operativo, esto es una tarea que requiere proyecto, gestión, numerosos recursos y todo un equipo disciplinado de trabajo.

Actualmente el modelo estudiado y descrito previamente es utilizado en proyectos grandes, como por ejemplo:

- Futuros sistemas de combate en el ámbito militar
- Software para gestión de las subvenciones agrarias de un país.
- Software para la gestión de la actividad de negocio de una empresa (nóminas, facturación, etc.)
- El Modelo Espiral es particularmente apto para el desarrollo de Sistemas Operativos (complejos).
- También es apto en sistemas de altos riesgos o críticos (Ej. navegadores y controladores aeronáuticos) y en todos aquellos en que sea necesaria una fuerte gestión del proyecto y sus riesgos, técnicos o de gestión.

16.1.10 Cascada con Reducción de Riesgo

Según Pressman (2010) uno de los problemas del ciclo cascada es que si se entienden mal los requisitos llevara a un problema, esto solo se descubrirá cuando se entregue el producto final. Para evitar este problema se puede hacer un desarrollo iterativo durante las fases detalladas en la Figura 92 que resumen el modelo cascada con reducción de riesgo.

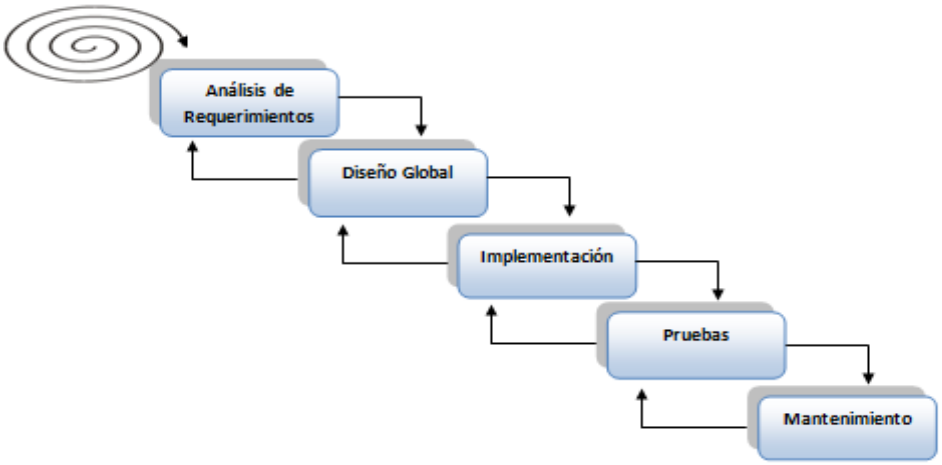


FIGURA N° 92: Cascada con Reducción de Riesgo
Fuente: Elaboración propia a partir Pressman (2010)

Fases del modelo

Para facilitar el entendimiento del cascada con reducción de riesgo se realizará una comparación pon cascada original por cada una de las fases definidas como: análisis de requerimientos, diseño global, implementación, pruebas, mantenimiento.

a) Análisis de requerimientos

Cascada original: Se analiza las necesidades de los usuarios finales para determinar qué objetivos debe cumplir. En este punto surge el documento de especificación de requerimientos. En esta etapa se debe saber todo lo que realizara el sistema, todo aquello será lo que se desarrollara en las siguientes etapas. En este caso no se pueden cambiar los requerimientos hasta que todo esté terminado.

Cascada con reducción de riesgo: Aquí también se analiza las necesidades de usuarios finales y también surge el documento de especificación de requerimientos, la diferencia que existe es que se ocupa un proceso iterativo para poder analizar los requerimientos.

b) Diseño Global

Cascada original: En esta etapa el software se centra en cuatro etapas diferentes:

- Estructura de los datos
- Arquitectura de software
- Detalle procedimental
- Caracterización de la interfaz

Cascada con reducción de riesgo: En el caso de la cascada con reducción de riesgos, también se utilizan las mismas etapas que la cascada original a diferencia que se utilizan unos pasos iterativos para crear un prototipito del sistema para saber si los requisitos solicitados por el usuario cumplen con ellos, los pasos son:

- Preguntar al usuario
- Hacer el diseño global que se desprende del primer punto
- Hacer un prototipito de la interfaz, entrevista con el usuario y volver al punto uno para identificar más requisitos y corregir malos entendiste entre el diseñador y el usuario.
- Cabe destacar que el proceso de estos pasos se produce entre la etapa de análisis y la etapa de diseño global.

c) Implementación

Cascada original: La etapa del diseño se lleva a cabo en lenguajes de programación para las distintas funciones, cada una es su lenguaje específico.

Cascada con reducción de riesgo: Después de que estén todos los requisitos y el usuario este conforme con el prototipito, se lleva a cabo la implementación con los respectivos lenguajes de programación.

d) Pruebas

Cascada original: Aquí, se juntan los diferentes programas que fueron desarrollados en sus respectivos lenguajes y se prueban como un sistema completo. Después de que la etapa de prueba haya sido exitosa se le entrega al cliente.

Cascada con reducción de riesgo: La etapa de prueba en la cascada con reducción de riesgo es exactamente igual que en la cascada original, se hacen las pruebas pertinentes y finalmente se le entrega al usuario final.

e) Mantenimiento

Cascada original: Se instala el sistema y se coloca en funcionamiento corrigiendo todos los problemas o errores que aparecieron ahora y no en la etapa de prueba, además si el usuario quisiera agregar algún requerimiento se puede implementar.

Cascada con reducción de riesgo: Ocurre lo mismo que en la cascada original, se instala el sistema y se coloca en funcionamiento corrigiendo errores.

Ventajas

Las ventajas de utilizar cascada con reducción de riesgo son:

- Se controla el riesgo en la fase de requerimientos
- Puede desarrollarse un prototipo de interfaz de usuario para la identificación de requerimientos
- El preámbulo no está limitado a los requerimientos.

Desventajas

Las desventajas de utilizar cascada con reducción de riesgo son las siguientes:

- Las desventajas son las mismas que cascada pura, exceptuando que se trata de evitar el riesgo de volver atrás en una etapa avanzada.
- No se puede retroceder
- Los errores que surgen en el programa completo causan mucho problema
- Recomendado para proyectos grandes

Cuando es apropiado utilizar este modelo

Es recomendable usar esta metodología para proyectos que utilicen tecnología ya conocida, puesto que aceleran el trabajo y mejoran el rendimiento y reducen el riesgo.

16.1.11 Cascada con Fases Solapadas

Winston W. Royce (1929-1995) en 1970 tomó las actividades fundamentales de las etapas de Especificación, Desarrollo, Validación y Evolución del Software, presentándolas por separado. Además de permitir la iteración de sus etapas de aplicación, con el fin de solucionar problemas de forma temprana y a un bajo costo, procura la modularización de las partes funcionales del sistema de información a desarrollar facilitando el prototipado. La Figura 93 resume las etapas del modelo cascada con fases solapadas.

FIGURA N° 93: Cascada con Fases Solapadas
Fuente: Elaboración propia a partir de Royce (1995)

Fases del modelo

Cascada con fases solapadas implica una secuencia de actividades tales como: concepto de software, análisis de requerimiento, diseño global, diseño detallado, codificación depuración, prueba del sistema. A continuación se detalla cada una de las actividades mencionadas.

a) Concepto de software: Es la construcción de la idea del proyecto que surge de las necesidades de nuestro entorno.

b) Análisis de requerimiento: Se analizan las necesidades de los usuarios finales del software para determinar el objetivo que éste debe cumplir.

c) Diseño Global: Se descompone y se organiza en elementos que puedan funcionar por separado.

d) Diseño Detallado: Se desarrolla el algoritmo necesario para el cumplimiento de los requerimientos del usuario.

e) Codificación Depuración: Se implementa el código fuente haciendo uso tanto de prototipos y prueba y error para corregir errores.

f) Prueba de sistema: Los elementos ya programados se ensamblan para componer el sistema. Se prueba que éste funcione correctamente y cumpla con los requisitos antes de ser instalado.

Ventajas

Algunas de las ventajas de desarrollar utilizando cascada con reducción de riesgo son:

- Reducción de tiempo
- Resulta más dinámico e integral
- Permite iterar problemáticas que surgen en el proceso
- La planificación es más sencilla.
- Este modelo está dirigido por documentos
- Ayuda a localizar errores en etapas tempranas del proyecto a un bajo costo
- Permite trabajar con poco personal cualificado.

Desventajas

Algunas de las desventajas de desarrollar utilizando cascada con reducción de riesgo son:

- Es más difícil de controlar el progreso del proyecto debido a que los finales de fase ya no son un punto de referencia claro.
- Difícil para identificar el inicio y el final de cada etapa.
- Difícil de reconocer todos los requerimientos desde un inicio.
- No refleja realmente el proceso de desarrollo del software.
- Se tarda mucho tiempo en pasar por todo el ciclo
- Perpetúa el fracaso de la industria del software en su comunicación con el usuario final.
- El mantenimiento se realiza en el código fuente
- Las revisiones de proyectos de gran complejidad son muy difíciles.

Cuando es apropiado utilizar este modelo

Será utilizado mayoritariamente en el desarrollo de productos de software con innovaciones importantes, o en el uso de tecnologías nuevas o poco probadas según las exigencias del mercado en las cuales se tengan medianamente claros los requisitos para así aprovechar la posibilidad de corregir errores en la superposición de fases.

16.2 Grupo de Metodologías Incrementales

Dentro de este grupo se encuentra una serie de metodologías que comparten algunas características. Dentro de las que se encuentran: modelo incremental, rational unified process (RUP), entrega por etapas y diseño por planificación.

16.2.1 Modelo incremental

Según Pressman (2010) y Sommerville (2005) el modelo incremental es una forma de reducir la repetición del trabajo en el proceso de desarrollo y dar la oportunidad de retrasar la toma de decisiones en los requisitos hasta adquirir experiencia con el sistema.

El Modelo Incremental combina elementos del método cascada con la filosofía interactiva de construcción de prototipos. Este modelo aplica secuencias lineales de forma escalonada mientras progresa el tiempo. Cada secuencia lineal produce un incremento del software. El primer incremento generalmente es un producto esencial denominado núcleo.

El modelo es de naturaleza interactiva, brindando al final de cada incremento la entrega de un producto completamente operacional.

Los primeros pasos los pueden realizar un grupo reducido de personas y en cada incremento se puede ir añadiendo personal si es necesario. Por otro lado los incrementos se pueden planear para gestionar riesgos técnicos.

Durante el proceso se trata de llevar a cabo al proyecto en diferentes partes que al final terminará siendo la solución completa requerida por el cliente, pero éstas partes no se pueden realizar en cualquier orden, sino que dependen de lo que el cliente esté necesitando con más urgencia.

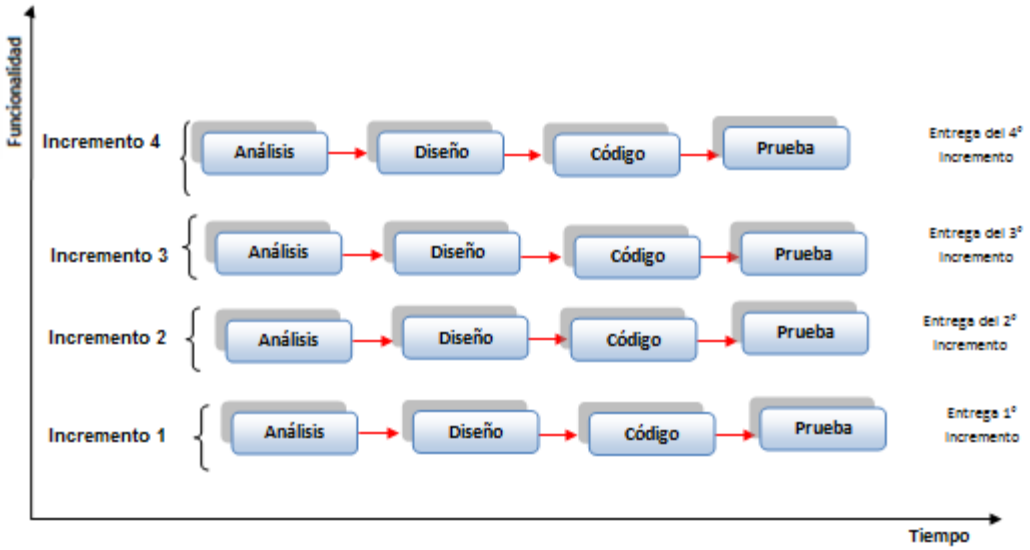


FIGURA N° 94: Modelo Incremental
Fuente: Elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Fases del modelo

El modelo consta de cuatro etapas, las cuales son: Análisis, Diseño, Codificación, Prueba. Cada una de estas se realiza en cada iteración del proceso. La Figura 94 resume el modelo incremental.

a) Análisis: Se identifican los requerimientos del sistema y que es lo que se desea entregar, se eligen las tecnologías a usar, se definen las tareas y los plazos para cada integrante del grupo de desarrollo y se identifican los posibles riesgos.

b) Diseño: Se realiza un diseño del sistema a desarrollar.

c) Código: Consiste en la programación del sistema diseñado anteriormente con las especificaciones técnicas definidas en el análisis**.**

d) Pruebas: Consiste en la presentación del primer prototipo al cliente, una vez probado y que este quede en funcionamiento, el cliente genera nuevos comentarios los cuales serán tomados por el grupo desarrollador y serán añadidos en el nuevo análisis del sistema.

Ventajas

Las ventajas de desarrollar utilizando modelo incremental se detallan a continuación.

- Los clientes no tienen que esperar hasta que el sistema se entregue completamente para comenzar a hacer uso de él.
- Los clientes pueden usar los incrementos iníciales como prototipo del software.
- Provee un impacto ventajoso frente al cliente, que es la entrega temprana de partes operativas del Software.
- Minimización del riesgo de falla en el proyecto porque los errores se van corrigiendo progresivamente.
- El modelo proporciona todas las ventajas del modelo en cascada realimentado, reduciendo sus desventajas sólo al ámbito de cada incremento.
- Permite entregar al cliente un producto más rápido en comparación del modelo de cascada.
- Resulta más sencillo acomodar cambios al acotar el tamaño de los incrementos

Desventajas

Las desventajas de utilizar el modelo incremental en el desarrollo son las siguientes:

- Difícil de aplicar a sistemas transaccionales que tienden a ser integrados y a operar como un todo.
- Riesgos largos y complejos.
- Pueden aumentar el coste debido a las pruebas.
- Los errores en los requisitos se detectan tarde.
- Requiere de mucha planeación, tanto administrativa como técnica.
- Requiere de metas claras para conocer el estado del proyecto.

Cuando es apropiado utilizar este modelo

Es útil cuando el proyecto tiene un plazo concreto. Este modelo se utiliza cuando no se conoce si el producto se tendrá para la última entrega. A diferencia del modelo de entrega por etapas, estas están ordenadas por orden de prioridad, así que la fecha tope aunque no hayamos terminado el proyecto estamos seguros de haber cubierto las funcionalidades más importantes.

16.2.2 Rational Unified Process (RUP)

Para Jacobson, Booch y Rumbaugh (1999) RUP puede tener definiciones diferentes dependiendo del contexto en el que se desee aplicar este modelo. Es por esto que se dice que RUP no es solo una metodología sino que un conjunto de metodologías adaptables al contexto y necesidades de cada organización.

Las definiciones de RUP pueden ser las siguientes:

- Es una metodología del desarrollo de software iterativa y que es guiada por casos de uso.
- Es un proceso de desarrollo de software bien definido y estructurado.
- Es un conjunto de herramientas y prácticas que ayudan al desarrollo de software.

FIGURA N° 95: RUP
Fuente: Elaboración propia a partir de Booch y Rumbaugh (1999)

Fases del modelo

RUP se desarrolla en forma secuencial, al concluir cada fase se verifica con una evaluación si se cumplieron los objetivos de la misma, o no. Este modelo se divide en cuatro fases en las que a su vez se realizan iteraciones según el proyecto que sea. La Figura 95 resume las etapas del modelo RUP.

a) Inicio: Consiste en especificar y delimitar los objetivos del proyecto y su alcance con las partes interesadas, describir los riesgos relacionados al mismo y asegurar que el proyecto sea viable, dando un enfoque general de la arquitectura de software.

b) Elaboración: Se establece la arquitectura base del sistema para brindar una plataforma segura, se definen los casos de uso escogidos para ello, teniendo en consideración los aspectos de mayor relevancia y se realiza una evaluación de riesgo.

c) Construcción: La finalidad de esta fase es culminar con la funcionalidad del sistema, esclareciendo las dudas que puedan existir, verificando que se cumplan los requerimientos pendientes, todo en función de la arquitectura base definida previamente.

d) Transición o Cierre: El propósito de esta fase es garantizar la disponibilidad del software para los usuarios finales, hacer cambios menores solicitados por el usuario, depurar el producto en relación a los errores encontrados en las pruebas, brindar la capacitación concerniente a los usuarios y verificar que el producto final cumpla con los requerimientos entregados por las partes interesadas.

Ventajas

Esta es una metodología completa en sí misma, con énfasis en la documentación precisa, las ventajas de desarrollar utilizando RUP serian las siguientes:

- Mitigación temprana de posibles riesgos altos.
- Progreso visible en las etapas tempranas.
- El tiempo de desarrollo requerido es menor debido a la reutilización de los componentes.
- Los usuarios están involucrados continuamente.

Desventajas

Algunas de las desventajas de desarrollar utilizando el modelo RUP son:

- Los miembros del equipo tienen que ser expertos en su campo para desarrollar un software bajo esta metodología.
- El proceso de desarrollo es demasiado complejo y desorganizado.

- En proyectos de vanguardia que utilizan la nueva tecnología, la reutilización de componentes no será posible. Por lo tanto el ahorro de tiempo se podría haber hecho será imposible de cumplir.
- Integración en todo el proceso de desarrollo de software, en teoría suena una buena cosa. Pero sobre todo grandes proyectos con el desarrollo de múltiples flujos que sólo se sumará a la confusión y causar más problemas durante las etapas de prueba.

Cuando es apropiado utilizar este modelo

Requiere una definición formal previa del alcance, hitos más importantes y fechas específicas.

RUP es recomendado para proyectos grandes, de larga duración, proyectos a nivel de empresa de complejidad media o alta.

RUP no es un proceso adecuado para todos los tipos de desarrollo, por ejemplo, para desarrollo de software embebido.

16.2.3 Entrega por Etapas

Para Pressman (2010) y Sommerville (2005) el modelo de desarrollo por Etapas es similar al Modelo de Prototipos con la diferencia de que, en este, se conoce exactamente lo que se va a construir. La principal característica de este modelo es que, el software no se entrega al cliente de una sola vez ni al final del proyecto, sino que se entrega a través de una serie de etapas a lo largo del proyecto. Permitiendo una correcta retroalimentación de parte del cliente.

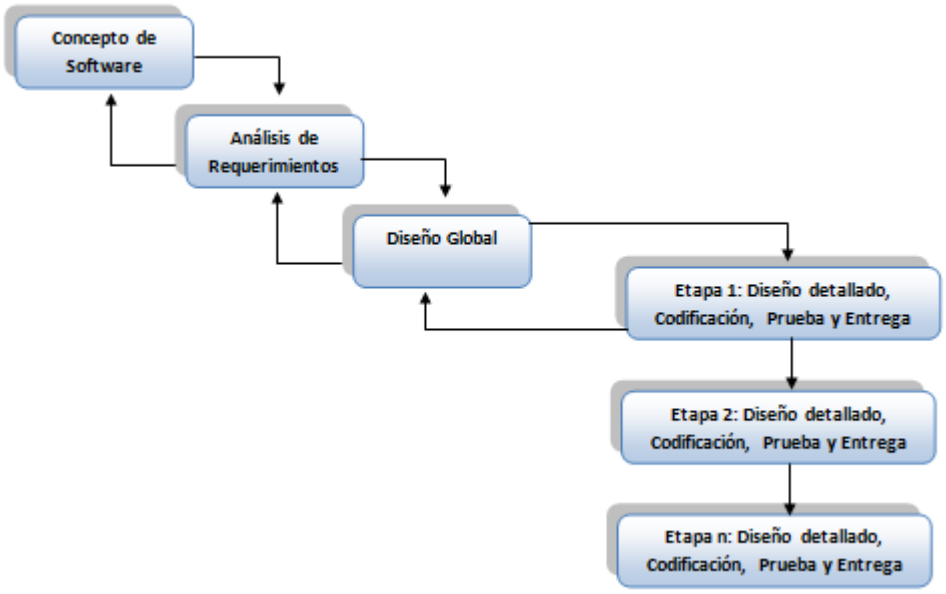


FIGURA N° 96: Entrega por Etapas
Fuente: Elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Fases del modelo

El modelo consta de etapas, las cuales son: concepto de software, análisis de requerimientos, diseño global y etapas de 1 a n. La Figura 96 resume el modelo de entrega por etapas.

a) Concepto de software: Es una situación que debe tener una solución o que debe ser mejorado. Teniendo en cuenta que el problema es un asunto del que se espera una rápida y efectiva solución.

b) Análisis de Requerimientos: Lo que se busca lograr en esta etapa es identificar lo que desea el usuario y la forma en que se le dará una solución a esto.

- Identificar casos de usos
- Dar detalle a los casos de uso en mención
- Definir una interfaz inicial
- Desarrollo del modelo
- Validar los modelos

c) Diseño Global: Se busca una solución en cualquier campo, teniendo en cuenta las restricciones y análisis de requerimientos que se hayan encontrado en el desarrollo de un resultado propicio al problema.

d) Etapa 1 a n: En esta fase se muestra el software al cliente en etapas cada vez más refinadas en las cuales se debe considerar lo siguiente:

- Diseño detallado
- Codificación
- Prueba
- Entrega

Ventajas

Las principales ventajas de desarrollar utilizando entrega por etapas son:

- Permite hacer modificaciones mientras se realiza el proyecto
- Requiere de un poco tiempo de gestión
- Genera un sistema Altamente fiable y con amplio desarrollo
- Permite proporcionar una funcionalidad útil en manos del cliente, sin la necesidad de tener la aplicación finalizada
- Proporciona signos tangibles de progreso

Desventajas

Las desventajas al desarrollar utilizando entrega por etapas son:

- No es viable sin una planificación adecuada, ya que se debe estar sometido a una planificación predefinida.
- No es recomendable para casos de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido y/o de alto índice de riesgos.
- Los errores en los requisitos se detectan tarde.
- Cada fase de una iteración es rígida y no se superponen con otras.
- Trabaja con poca comprensión sobre la arquitectura.
- Trabaja con poca identificación de los requerimientos de diseño.

Cuando es apropiado utilizar este modelo

En proyectos largos, en los cueles se debe forzar la visibilidad.

Cuando se está seguro de las prestaciones que debe tener el producto. Si tiene dudas sobre algunos aspectos significativos, no utilice este modelo.

Solo funciona bien en sistemas en los que se puede desarrollar independientemente subconjuntos útiles del producto.

16.2.4 Diseño por planificación

Para Pressman (2010) y Sommerville (2005) diseño por planificación es similar al modelo de entrega por etapas, pero se diferencia en que no siempre se conoce al principio si se tendrá el producto para la última entrega. Se pueden tener cinco etapas planificadas, pero sólo se llega a la tercera etapa debido a que se tiene una fecha límite que no se puede cambiar. Uno de los elementos críticos de este modelo es priorizar los requerimientos y planificar sus etapas de tal manera que las primeras contengan los requerimientos de mayor prioridad, los requerimientos de baja prioridad se dejan para más tarde.

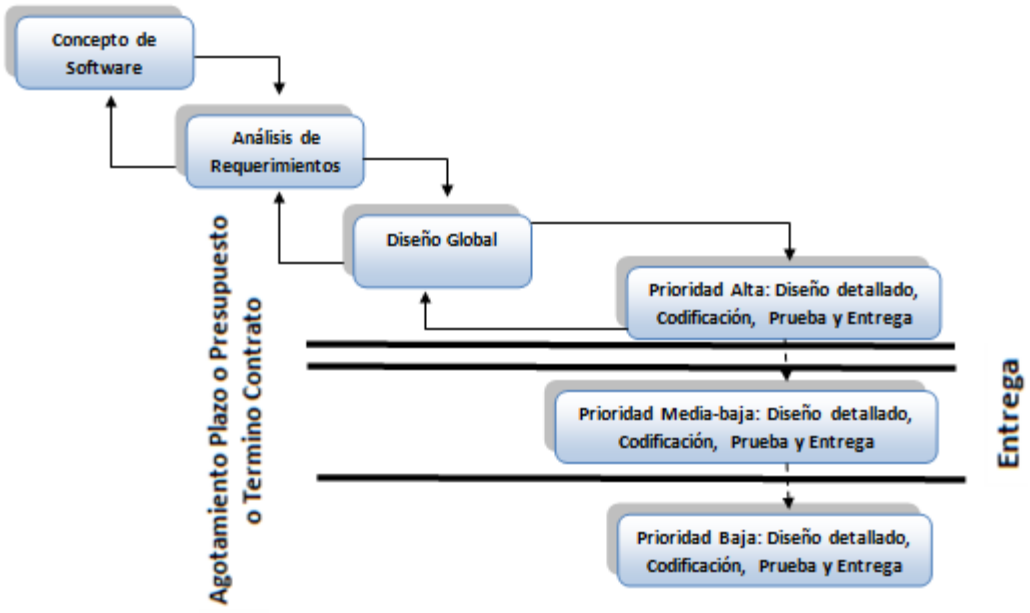


FIGURA N° 97: Diseño por planificación
Fuente: elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Fases del modelo

En la Figura 97 se resumen las fases del modelo por planificación, de las cuales se detallan concepto de software, análisis de requerimientos, diseño global y entregas las que se definen por su prioridad.

a) Concepto de software: Es una situación que debe tener una solución o que debe ser mejorado. Teniendo en cuenta que el problema es un asunto del que se espera una rápida y efectiva solución.

b) Análisis de Requerimientos: Lo que se busca lograr en esta etapa es identificar lo que desea el usuario y la forma en que se le dará una solución a esto.

- Identificar casos de usos
- Dar detalle a los casos de uso en mención
- Definir una interfaz inicial
- Desarrollo del modelo
- Validar los modelos

c) Diseño Global: Se busca una solución en cualquier campo, teniendo en cuenta las restricciones y análisis de requerimientos que se hayan encontrado en el desarrollo de un resultado propicio al problema.

d) Prioridad (Alta, Media - Baja, Baja): En esta fase de prioridad la que encontramos en 3 niveles se realizan las entregas considerando los siguientes aspectos:

- Diseño detallado
- Codificación
- Prueba
- Entrega

Ventajas

Puede ser una estrategia válida para asegurar que se tiene un producto listo en una fecha determinada.

Esta estrategia es particularmente útil para las partes del producto que no se quieren realizar en el camino crítico

Desventajas

Si no se completan todas las etapas, se desperdiciará tiempo en la especificación, arquitectura y diseños de prestaciones que no se van a entregar.

Si se ha gastado tiempo en una gran cantidad de requerimientos incompletos que no se van a entregar, se debería tener tiempo para resumir en uno o dos requerimientos más completos.

Cuando es apropiado utilizar este modelo

Es útil cuando el proyecto tiene un plazo concreto. Este modelo se utiliza cuando no se conoce si el producto se tendrá para la última entrega. A diferencia del modelo de entrega por etapas, estas están ordenadas por orden de prioridad, así que la fecha tope aunque no hayamos terminado el proyecto estamos seguros de haber cubierto las funcionalidades más importantes.

16.2.5 Grupo de Metodologías Ágiles

Dentro del grupo de metodologías ágiles, se han considerado las siguientes: scrum, extreme programming (XP), dynamic systems development method (DSMD), adaptive software development (ASD), crystal clear, feature driven development (FDD), lean software development (LSD), kanban, open up.

16.2.6 SCRUM

Para Schwaber, Rising, Janoff (2000) Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo**,** y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. La Figura 98 muestra una referencia del modelo scrum.

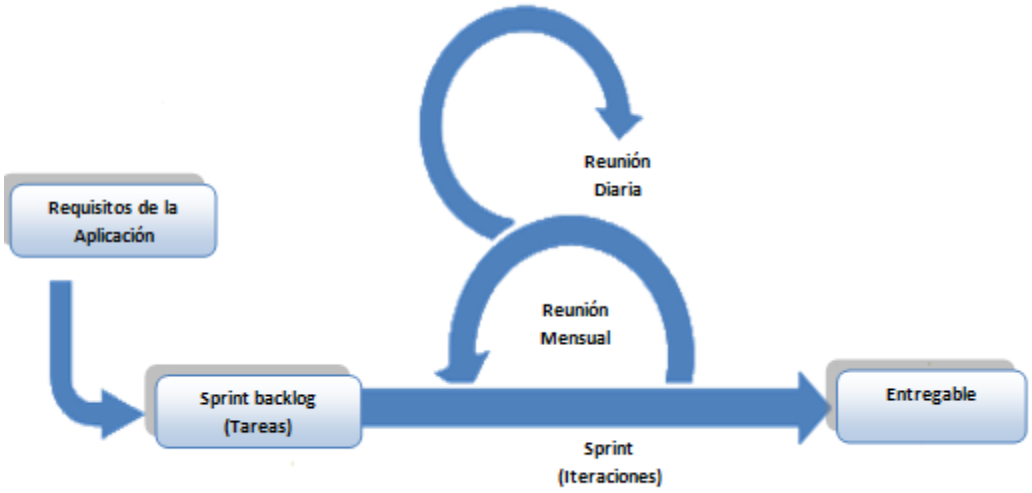


FIGURA N° 98: SCRUM
Fuente: Elaboración propia a partir de Schwaber, Rising, Janoff (2000)

Los valores de Scrum son:

• Equipos auto-dirigidos y auto-organizados. No hay manager que decida, ni otros títulos que "miembros del equipo"; la excepción es el Scrum Master que debe ser 50% programador y que resuelve problemas, pero no manda. Los observadores externos pueden observar, pero no interferir ni opinar.

• Una vez elegida una tarea, no se agrega trabajo extra. En caso que se agregue algo, se recomienda quitar alguna otra cosa.

• Encuentros diarios se realizan siempre en el mismo lugar, en círculo. El encuentro diario impide caer en el dilema señalado por Fred Brooks: "¿Cómo es que un proyecto puede atrasarse un año?: Un día a la vez".

• Iteraciones de treinta días; se admite que sean más frecuentes.

• Demostración a participantes externos al fin de cada iteración.

• Al principio de cada iteración, planeamiento adaptativo guiado por el cliente.

Scrum define roles:

a) El Scrum Master

Interactúa con el cliente y el equipo. Es responsable de asegurarse que el proyecto se lleve a cabo de acuerdo con las prácticas, valores y reglas de Scrum y que progrese según lo previsto. Coordina los encuentros diarios, formula lastres preguntas canónicas y se encarga de eliminar eventuales obstáculos. Debe ser miembro del equipo y trabajar a la par.

b) Propietario del Proyecto

Es el responsable oficial del proyecto, gestión, control y visibilidad de la lista de acumulación o lista de retraso del producto (product backlog). Es elegido por el Scrum Master, el cliente y los ejecutivos a cargo. Toma las decisiones finales de las tareas asignadas al registro y convierte sus elementos en rasgos a desarrollar.

c) Equipo de Scrum

Tiene autoridad para reorganizarse y definir las acciones necesarias o sugerir remoción de impedimentos. El equipo posee la misma estructura del "equipo quirúrgico" desarrollado por IBM.

d) Cliente

Participa en las tareas relacionadas con los ítems del registro.

e) Management

Está a cargo de las decisiones fundamentales y participa en la definición de los objetivos y requerimientos. Por ejemplo, selecciona al Dueño del Producto, evalúa el progreso y reduce el registro de acumulación junto con el Scrum Master.

f) Usuario

La dimensión del equipo total de Scrum no debería ser superior a diez ingenieros. El número ideal es siete, más o menos dos, una cifra canónica en ciencia cognitiva. Si hay más, lo más recomendable es formar varios equipos. No hay una técnica oficial para coordinar equipos múltiples, pero se han documentado experiencias de hasta 800miembros, divididos en Scrums de Scrum, definiendo un equipo central que se encarga de la coordinación, las pruebas cruzadas y la rotación de los miembros. El texto que relata esa experiencia es Agile Software Development Ecosystems, de Jim High Smith.

El ciclo de vida de Scrum es el siguiente:

a) Pre-Juego: Planeamiento

El propósito es establecer la visión, definir expectativas y asegurarse la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso (backlog) del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos. El registro de acumulación es de alto nivel de abstracción.

b) Pre-Juego: Montaje (Staging)

El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.

c) Juego o Desarrollo

El propósito es implementar un sistema listo para entrega en una serie de iteraciones de treinta días llamadas "corridas" (sprints). Las actividades son un encuentro de planeamiento de corridas en cada iteración, la definición del registro de acumulación de corridas y los estimados, y encuentros diarios de Scrum.

d) Pos-Juego: Liberación

El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta. Usualmente los registros de acumulación se llevan en planillas de cálculo comunes, antes que en una herramienta sofisticada de gestión de proyectos. Los elementos del registro pueden ser prestaciones del software, funciones, corrección de bugs, mejoras requeridas y actualizaciones de tecnología. Hay un registro total del producto y otro específico para cada corrida de 30 días. En la jerga de Scrum se llaman "paquetes" a los objetos o componentes que necesitan cambiarse en la siguiente iteración.

16.2.7 eXtreme Programming (XP)

Según literatura de Pressman (2010) y Sommerville (2005), la Programación Extrema es sin duda alguna el método ágil que primero viene a la mente cuando se habla de modelos no tradicionales y el más transgresor entre ellos.

XP se funda en cuatro valores: comunicación, simplicidad, feedback y coraje. Pero tan conocidos como sus valores son sus prácticas. En la Figura 99 se presenta un diagrama de XP.

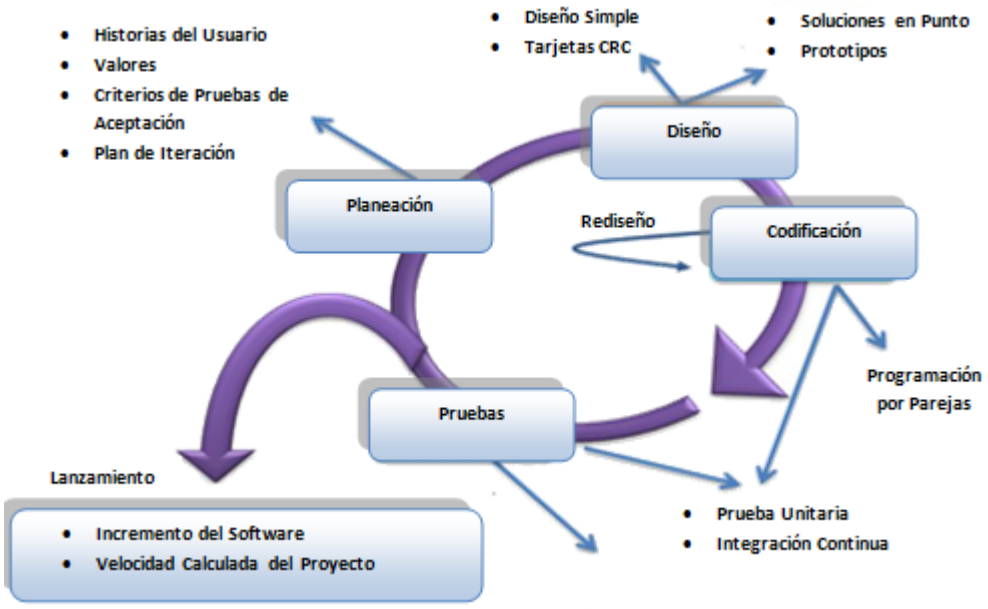


FIGURA N° 99: XP
Fuente: elaboración propia a partir de Pressman (2010) y Sommerville (2005)

Gestación de XP

a) Juego de Planeamiento

Busca determinar rápidamente el alcance de la versión siguiente, combinando prioridades de negocio definidas por el cliente con las estimaciones técnicas de los programadores. Éstos estiman el esfuerzo necesario para implementar las historias del cliente y éste decide sobre el alcance y la agenda de las entregas. Las historias se escriben en pequeñas fichas, que algunas veces se tiran. Cuando esto sucede, lo único restante que se parece a un requerimiento es una multitud de pruebas automatizadas, las pruebas de aceptación.

b) Entregas pequeñas y frecuentes

Se "produce" un pequeño sistema rápidamente, al menos uno cada dos o tres meses. Pueden liberarse nuevas versiones diariamente, pero al menos se debe liberar una cada mes. Se agregan pocos rasgos cada vez.

c) Metáforas del sistema

El sistema se define a través de una metáfora o un conjunto de metáforas, una "historia compartida" por clientes, managers y programadores que orienta todo el sistema describiendo como funciona. Una metáfora puede interpretarse como una arquitectura simplificada. La concepción de metáfora que se aplica en XP deriva de los estudios de La koff y Johnson, bien conocidos en lingüística y psicología cognitiva.

d) Diseño simple

El énfasis se deposita en diseñar la solución más simple susceptible de implementarse en el momento. Se eliminan complejidades innecesarias y código extra, y se define la menor cantidad de clases posible. No debe duplicarse código. En un oxímoron deliberado, se urge a "decir todo una vez y una sola vez". Nadie en XP llega a prescribir que no haya diseño concreto, pero el diseño se limita a algunas tarjetas elaboradas en sesiones de diseño de 10 a 30 minutos. Esta es la práctica donde se impone el minimalismo de YAGNI: no implementar nada que no se necesite ahora; o bien, nunca implementar algo que vaya a necesitarse más adelante; minimizar diagramas y documentos.

e) Prueba continua

El desarrollo está orientado por las pruebas. Los clientes ayudan a escribir las pruebas funcionales antes que se escriba el código. Esto es test-driven development. El propósito del código real no es cumplir un requerimiento, sino pasarlas pruebas. Las pruebas y el código son escritas por el mismo programador, pero la prueba debería realizarse sin intervención humana, y es a todo o nada. Hay dos clases de prueba: la prueba unitaria, que verifica una sola clase, o un pequeño conjunto de clases; la prueba de aceptación verifica todo el sistema, o una gran parte.

f) Refactorización continúa

Se refactoriza el sistema eliminando duplicación, mejorando la comunicación y agregando flexibilidad sin cambiar la funcionalidad. El proceso consiste en una serie de pequeñas transformaciones que modifican la estructura interna preservando su conducta aparente. La práctica también se conoce como Mejora Continua de Código o Refactorización implacable. Se lo ha parafraseado diciendo: "Si funciona bien, arréglelo de todos modos". Se recomiendan herramientas automáticas. En sus comentarios a [Hig00b] Ken Orr recomienda GeneXus de ARTech, Uruguay, virtuoso ejecutor de las mejores promesas incumplidas de las herramientas CASE.

g) Programación en pares

Todo el código está escrito por pares de programadores. Dos personas escriben código en una computadora, turnándose en el uso del ratón y el teclado. El que no está escribiendo, piensa desde un punto de vista más estratégico y realiza lo que podría llamarse revisión de código en tiempo real. Los roles pueden cambiarse varias veces al día. Esta práctica no es en absoluto nueva.

h) Propiedad colectiva del código

Cualquiera puede cambiar cualquier parte del código en cualquier momento, siempre que escriba antes la prueba correspondiente.

i) Integración continúa

Cada pieza se integra a la base de código apenas está lista, varias veces al día. Debe correrse la prueba antes y después de la integración. Hay una máquina (solamente) dedicada a este propósito.

j) Ritmo sostenible

Trabajando un máximo de 8 horas por día, antes se llamaba a esta práctica Semana de 40 horas. En XP todo el mundo debe irse a casa a las cinco de la tarde. Dado que el desarrollo de software se considera un ejercicio creativo, se estima que hay que estar fresco y descansado para hacerlo eficientemente; con ello se motiva a los participantes, se evita la rotación del personal y se mejora la calidad del producto. Aunque podrían admitirse excepciones, no se permiten dos semanas seguidas de tiempo adicional. Si esto sucede, se lo trata como problema a resolver.

k) Todo el equipo en el mismo lugar

El cliente debe estar presente y disponible a tiempo completo para el equipo. También se llama El Cliente en el Sitio. Como esto parecía no cumplirse, se especificó que el representante del cliente debe ser preferentemente un analista.

l) Estándares de codificación

Se deben seguir reglas de codificación y comunicarse a través del código. Otros la resuelven poniéndose de acuerdo en estilos de notación, identación y nomenclatura, así como en un valor apreciado en la práctica, el llamado "código revelador de intenciones". Como en XP rige un cierto purismo de codificación, los comentarios no son bien vistos. Si el código es tan oscuro que necesita comentario, se lo debe reescribir o refactorizar.

m) Espacio abierto

Es preferible una sala grande con pequeños cubículos o, mejor todavía, sin divisiones. Los pares de programadores deben estar en el centro. En la periferia se ubican las máquinas privadas. En un encuentro de espacio abierto la agenda no se establece verticalmente.

n) Reglas justas

El equipo tiene sus propias reglas a seguir, pero se pueden cambiar en cualquier momento. En XP se piensa que no existe un proceso que sirva para todos los proyectos; lo que se hace habitualmente es adaptar un conjunto de prácticas simples a las características de cada proyecto. Las prácticas se han ido modificando con el tiempo. Las versiones más recientes enumeran diecinueve prácticas, agrupadas en cuatro clases.

o) Prácticas conjuntas

- Iteraciones
- Vocabulario Común – Reemplaza a Metáforas
- Espacio de trabajo abierto
- Retrospectivas
- Prácticas de Programador
- Desarrollo orientado a pruebas
- Programación en pares
- Refactorización
- Propiedad colectiva
- Integración continua
- YAGNI– Equivale a Diseño Simple

p) Prácticas de Management

- Responsabilidad aceptada
- Cobertura aérea para el equipo
- Revisión trimestral
- Espejo – El manager debe comunicar un fiel reflejo del estado de cosas
- Ritmo sostenible

q) Prácticas de Cliente

- Narración de historias
- Planeamiento de entrega
- Prueba de aceptación
- Entregas frecuentes

Las prácticas también pueden adaptarse a las características de los sistemas particulares. Pero no todo es negociable y la integridad del método se sabe frágil. La particularidad de XP radica en no requerir ninguna herramienta fuera del ambiente de programación y prueba. Al contrario de otros métodos, que permiten modelado, XP demanda comunicación oral tanto para los requerimientos como para el diseño.

Ciclo de vida de XP

Algunos autores sugieren implementar spikes (o sea "púas" o "astillas") para estimar la duración y dificultad de una tarea inmediata. Un spike es un experimento dinámico de código, realizado para determinar cómo podría resolverse un problema. Es la versión ágil de la idea de prototipo. Se lo llama así porque "va de punta a punta, pero es muy fino" y porque en el recorrido de un árbol de opciones implementaría una opción de búsqueda depth-first. Entre los artefactos que se utilizan en XP vale la pena mencionar las tarjetas de historias (story cards); son tarjetas comunes de papel en que se escriben breves requerimientos de un rasgo, jamás casos de uso. Las tarjetas tienen una granularidad de diez o veinte días. Las tarjetas se usan para estimar prioridades, alcance y tiempo de realización; en caso de discrepancia, gana la estimación más optimista. Otros productos son listas de tareas en papel o en una pizarra (jamás en computadora) y gráficos visibles pegados en la pared. Los roles de XP son pocos. Un cliente que escribe las historias y las pruebas de aceptación; programadores en pares; verificadores (que ayudan al cliente a desarrollar las pruebas); consultores técnicos; y, como parte del management, un coach o consejero que es la conciencia del grupo, interviene y enseña, y un seguidor de rastros (tracker) que colecta las métricas y avisa cuando hay una estimación alarmante, además de un Gran Jefe. El management es la parte menos satisfactoriamente caracterizada en la bibliografía, como si fuera un mal necesario.

Los equipos de XP son típicamente pequeños, de tres a veinte personas. Los obstáculos más comunes surgidos en proyectos XP son la "fantasía" de pretender que el cliente se quede en el sitio y la resistencia de muchos programadores atrabajar en pares. Craig Larman señala como factores

negativos la ausencia de énfasis en la arquitectura durante las primeras iteraciones (no hay arquitectos en XP) y la consiguiente falta de métodos de diseño arquitectónico.

16.2.8 Dynamic Systems Development Method (DSDM)

Según Coleman and Verbrunggen (1998) y Beynon (2003), el método de desarrollo de sistemas dinámicos (en inglés Dynamic Systems Development Method o DSDM) es un método que provee un framework para el desarrollo ágil de software, apoyado por su continua implicación del usuario en un desarrollo iterativo y creciente que sea sensible a los requerimientos cambiantes, para desarrollar un sistema que reúna las necesidades de la empresa en tiempo y presupuesto.

DSDM consiste en 3 fases:

- Fase del pre-proyecto
- Fase del ciclo de vida del proyecto

  La fase del ciclo de vida del proyecto se subdivide en 5 actividades:

  - Estudio de viabilidad,
  - Estudio de la empresa,
  - Iteración del modelo funcional,
  - Diseño e iteración de la estructura,
  - Implementación.

Los cinco procesos centrales se suelen representar con el siguiente gráfico, familiarmente denominado "de las 3 pizzas y el queso". Ver Figura 100.

- Fase del post-proyecto.

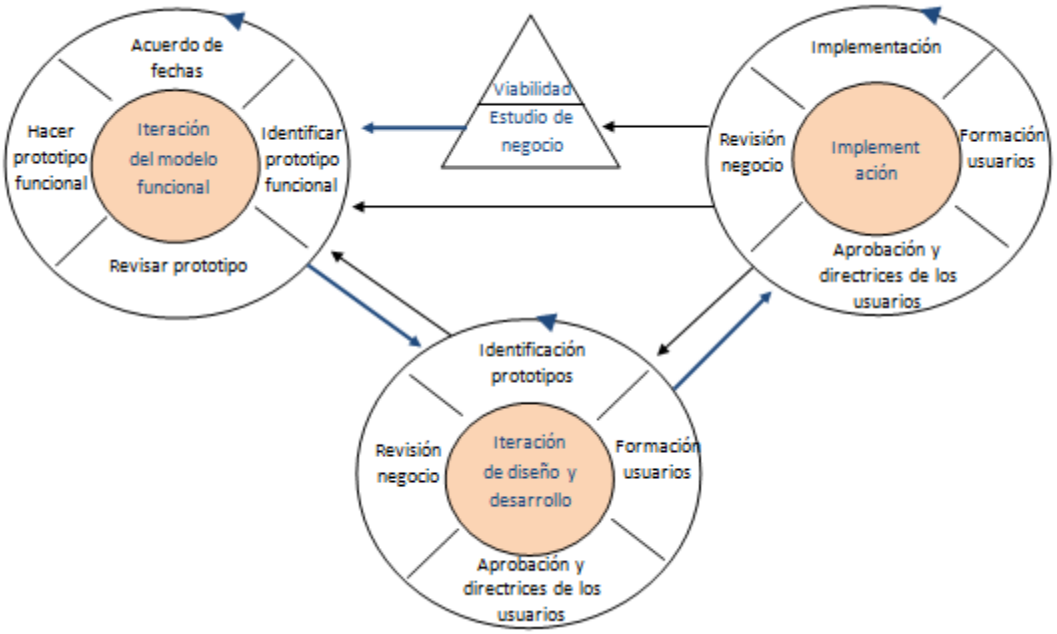


FIGURA N° 100: DSDM
Fuente: elaboración propia a partir de Coleman and Verbrunggen (1998)

DSDM reconoce que los proyectos son limitados por el tiempo y los recursos, y los planes acorde a las necesidades de la empresa. Para alcanzar estas metas, DSDM promueve el uso del RAD con el consecuente peligro que demasiadas esquinas estén cortadas. DSDM aplica algunos principios, roles, y técnicas. En algunas circunstancias, hay posibilidades para integrar contenido de otros métodos, tal como el Proceso Unificado Racional (RUP), Programación Extrema (XP), y Proyectos en ambientes controlados (PRINCE2), para complementar el DSDM en la realización de un proyecto. Otro método ágil que tiene semejanzas proceso y concepto con DSDM es Scrum.

Principios del DSDM

La versión actual en uso desde el 2006 presenta 9 principios subyacentes al DSDM consistentes en cuatro fundamentos y cinco puntos de partida para la estructura del método. Estos principios forman los pilares del desarrollo mediante DSDM.

Involucrar al cliente es la clave para llevar un proyecto eficiente y efectivo, donde ambos, cliente y desarrolladores, comparten un entorno de trabajo para que las decisiones puedan ser tomadas con precisión.

El equipo del proyecto debe tener el poder para tomar decisiones que son importantes para el progreso del proyecto, sin esperar aprobación de niveles superiores.

DSDM se centra en la entrega frecuente de productos, asumiendo que entregar algo temprano es siempre mejor que entregar todo al final. Al entregar el producto frecuentemente desde una etapa temprana del proyecto, el producto puede ser verificado y revisado allí donde la documentación de registro y revisión puede ser tenida en cuenta en la siguiente fase o iteración.
El principal criterio de aceptación de entregables en DSDM reside en entregar un sistema que satisface las actuales necesidades de negocio. No está dirigida tanto a proporcionar un sistema perfecto que resuelva todas las necesidades posibles del negocio, si no que centra sus esfuerzos en aquellas funcionalidades críticas para alcanzar las metas establecidas en el proyecto/negocio.
El desarrollo es iterativo e incremental, guiado por la realimentación de los usuarios para converger en una solución de negocio precisa.

Todos los cambios durante el desarrollo son reversibles.
El alcance de alto nivel y los requerimientos deberían ser base-lined antes de que comience el proyecto.
Las pruebas son realizadas durante todo el ciclo vital del proyecto. Esto tiene que hacerse para evitar un caro coste extraordinario en arreglos y mantenimiento del sistema después de la entrega.
La comunicación y cooperación entre todas las partes interesadas en el proyecto es un prerrequisito importante para llevar un proyecto efectivo y eficiente.

DSDM también se apoya en otros principios.
Ningún sistema es construido a la perfección en el primer intento.
La entrega del proyecto debería ser a tiempo, respetando presupuestos y con buena calidad.
Ambas técnicas de Desarrollo y Gestión del proyecto están incluidas en DSDM.
La Evaluación de riesgos debiera centrarse en entregar función de negocio, no en el proceso de construcción.
La gestión recompensa la entrega de productos más que la consecución de tareas.
La Estimación debería estar basada en la funcionalidad del negocio en lugar de líneas de código.

16.2.9 Adaptive Software Development (ASD)

Según Barreto (2003) Adaptive Software Development es el modelo de implementación de patrones ágiles para desarrollo de software, diseñado por Jim High Smith, que materializa las fases de la gestión ágil de la siguiente forma: Ver Figura 101 representaciones de fases de ASD.

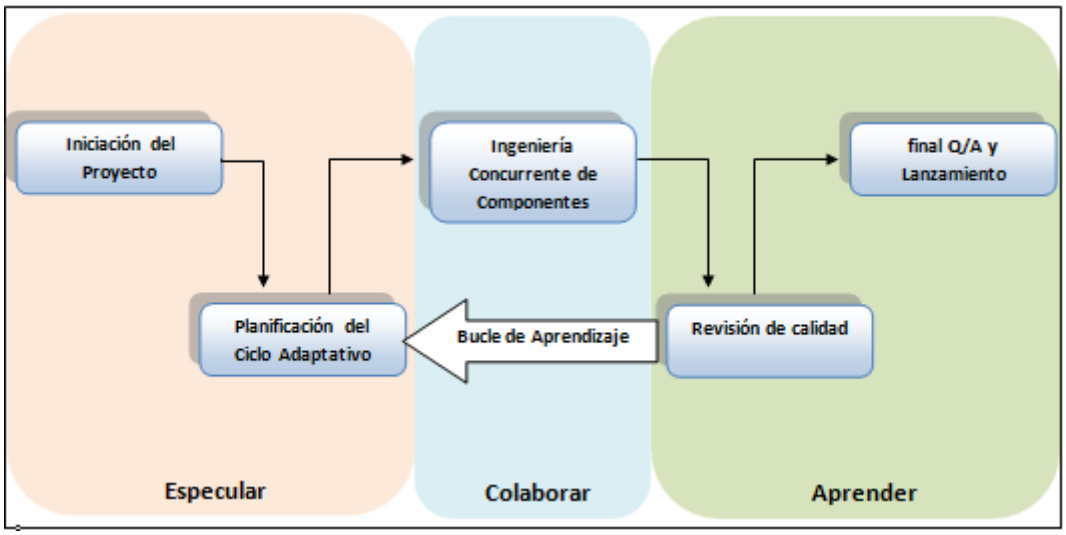


FIGURA N° 101: ASD
Fuente: elaboración propia a partir de Barreto (2003)

Fases del modelo

A continuación se detallan las etapas de modelo ASD las que consisten en: especulación, colaboración y aprendizaje detallados a continuación.

a) Especulación

- Inicio para determinar la misión del proyecto.
- Fijación del marco temporal del proyecto.
- Definición del número de iteraciones y duración de cada una.
- Asignación de funcionalidad a cada iteración.

b) Colaboración

- Desarrollo concurrente del trabajo de construcción y gestión del producto

c) Aprendizaje

- En cada iteración se revisa:
- Calidad, con criterios de cliente.
- Calidad, con criterios técnicos.
- Funcionalidad desarrollada
- Estado del proyecto

Características básicas de ASD

- Trabajo orientado y guiado por la misión del proyecto.
- Basado en la funcionalidad
- Desarrollo iterativo
- Desarrollo acotado temporalmente
- Guiado por los riesgos
- Trabajo tolerante al cambio.

16.2.10 Crystal Clear

Según Barreto (2003) la familia Crystal dispone un código de color para marcar la complejidad de una metodología: cuanto más oscuro un color, más "pesado" es el método. Cuanto más crítico es un sistema, más rigor se requiere. El código cromático se aplica a una forma tabular elaborada por Cockburn que se usa en muchos más para situar el rango de complejidad al cual se aplica una metodología. La Figura 102 representa las actividades realizadas en Crystal Clear.

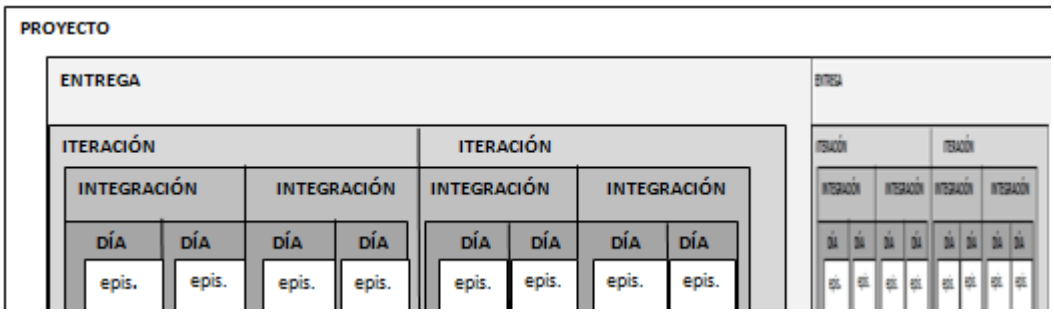


FIGURA N° 102: Crystal Clear
Fuente: elaboración propia a partir de Barreto (2003)

Propiedades de Crystal Clear

A continuación se detallan las principales propiedades de Crystal Clear.

a) Entrega frecuente

Consiste en entregar software a los clientes con frecuencia, no solamente en compilar el código. La frecuencia dependerá del proyecto, pero puede ser diaria, semanal, mensual o lo que fuere. La entrega puede hacerse sin despliegue, si es que se consigue algún usuario cortés o curioso que suministre feedback.

b) Comunicación osmótica

Todos juntos en el mismo cuarto. Una variante especial es disponer en la sala de un diseñador sénior; eso se llama Experto al Alcance de la Oreja. Una reunión separada para que los concurrentes se concentren mejor es descripta como El Cono del Silencio.

c) Mejora reflexiva.

Tomarse un pequeño tiempo (unas pocas horas ciertas semanas o una vez al mes) para pensar bien qué se está haciendo, cotejar notas, reflexionar, discutir.

d) Seguridad personal

Hablar cuando algo molesta: decirle amigablemente al manager que la agenda no es realista, o a un colega que su código necesita mejorarse. Esto es importante porque el equipo puede descubrir y reparar sus debilidades. No es provechoso encubrir los desacuerdos con gentileza y conciliación. Técnicamente, estas cuestiones se han caracterizado como una importante variable de confianza y han sido estudiadas con seriedad en la literatura.

e) Foco

Saber lo que se está haciendo y tener la tranquilidad y el tiempo para hacerlo. Lo primero debe venir de la comunicación sobre dirección y prioridades, típicamente con el Patrocinador Ejecutivo. Lo segundo, de un ambiente en que la gente no se vea compelida a hacer otras cosas incompatibles.

f) Fácil acceso a usuarios expertos

Una comunicación de Keil a la ACM demostró hace tiempo, sobre una amplia muestra estadística, la importancia del contacto directo con expertos en el desarrollo de un proyecto. Un encuentro

semanal o semi-semanal con llamados telefónicos adicionales parece ser una buena pauta. Otra variante es que los programadores se entrenen para ser usuarios durante un tiempo. El equipo de desarrollo, de todas maneras, incluye un Experto en Negocios.

Ambiente técnico con prueba automatizada, management de configuración e integración frecuente

Microsoft estableció la idea de los builds cotidianos. Muchos equipos ágiles compilan e integran varias veces al día. Crystal Clear no requiere ninguna estrategia o técnica, pero conviene tener unas cuantas a mano para empezar.

Las siguientes técnicas favorecen el desarrollo de la metodología:

a) Entrevistas de proyectos

Se suele entrevistar a más de un responsable para tener visiones más ricas. La idea es averiguar cuáles son las prioridades, obtener una lista de rasgos deseados, saber cuáles son los requerimientos más críticos y cuáles los más negociables. Si se trata de una actualización o corrección, saber cuáles son las cosas que se hicieron bien y merecen preservarse y los errores que no se quieren repetir.

b) Talleres de reflexión

El equipo debe detenerse treinta minutos o una hora para reflexionar sobre sus convenciones de trabajo, discutir inconvenientes y mejoras y planear para el período siguiente. De aquí puede salir material para poner en un poster como Radiador de Información.

c) Planeamiento Blitz

Una técnica puede ser el Juego de Planeamiento de XP. En este juego, se ponen tarjetas indexadas en una mesa, con una historia de usuario o función visible en cada una. El grupo finge que no hay dependencias entre tarjetas, y las alinea en secuencias de desarrollo preferidas. Los programadores escriben en cada tarjeta el tiempo estimado para desarrollar cada función. El patrocinador o embajador del usuario escribe la secuencia de prioridades, teniendo en cuenta los tiempos referidos y el valor de negocio de cada función. Las tarjetas se agrupan en períodos de tres semanas llamados iteraciones que se agrupan en entregas, usualmente no más largas de tres meses.

d) Estimación Delphi con estimaciones de pericia

La técnica se llama así por analogía con el oráculo de Delfos; se la describió por primera vez en el clásico Surviving Object-Oriented Projects de Cockburn, reputado como uno de los mejores libros sobre el paradigma de objetos. En el proceso Delphi se reúnen los expertos responsables y proceden como en un remate para proponer el tamaño del sistema, su tiempo de ejecución, la fecha de las entregas según dependencias técnicas y de negocios y para equilibrar las entregas en paquetes de igual tamaño.

e) Encuentros diarios de pie

La palabra clave es "brevedad", cinco a diez minutos como máximo. No se trata de discutir problemas, sino de identificarlos. Los problemas sólo se discuten en otros encuentros posteriores, con la gente que tiene que ver en ellos. La técnica se origina en Scrum. Se deben hacer de pie para que la gente no escriba en sus notebooks, garabatee papeles o se quede dormida.

f) Miniatura de procesos

La "Hora Extrema" fue inventada por Peter Merel para introducir a la gente en XP en 60 minutos y proporciona lineamientos canónicos que pueden usarse para articular esta práctica. Una forma de presentar Crystal Clear puede insumir entre 90 minutos y un día. La idea es que la gente pueda "degustar" la nueva metodología.

g) Gráficos de quemado

Su nombre viene de los gráficos de quemado de calorías de los regímenes dietéticos; se usan también en Scrum. Se trata de una técnica de graficación para descubrir demoras y problemas tempranamente en el proceso, evitando que se descubra demasiado tarde que todavía no se sabe cuánto falta. Para ello se hace una estimación del tiempo faltante para programar lo que resta al ritmo actual, lo cual sirve para tener dominio de proyectos en los cuales las prioridad es cambian bruscamente y con frecuencia.

h) Programación lado a lado

Mucha gente siente que la programación en pares de XP involucra una presión excesiva; la versión de Crystal Clear establece proximidad, pero cada quien se aboca a su trabajo asignado, prestando un ojo a lo que hace su compañero, quien tiene su propia máquina. Esta es una ampliación de la Comunicación Osmótica al contexto de la programación.

Existen ocho roles nominados en Crystal Clear: Patrocinador, Usuario Experto, Diseñador Principal, Diseñador-Programador, Experto en Negocios, Coordinador, Verificador, Escritor. A continuación se describen en bastardilla los artefactos de los que son responsables los roles de Crystal Clear, detalladamente descriptos en la documentación.

- **Patrocinador**

Produce la Declaración de Misión con Prioridades de Compromiso (Trade off). Consigue los recursos y define la totalidad del proyecto.

- **Usuario Experto**

Junto con el Experto en Negocios produce la Lista de Actores-Objetivos y el Archivo de Casos de Uso y Requerimientos. Debe familiarizarse con el uso del sistema, sugerir atajos de teclado, modos de operación, información a visualizar simultáneamente, navegación, etcétera.

- **Diseñador Principal**

Produce la Descripción Arquitectónica. Se supone que debe ser al menos un profesional de Nivel 3.

El Diseñador Principal tiene roles de coordinador, arquitecto, mentor y programador más experto.

- **Diseñador-Programador**

Produce, junto con el Diseñador Principal, los Borradores de Pantallas, el Modelo Común de Dominio, las Notas y Diagramas de Diseño, el Código Fuente, el Código de Migración, las Pruebas y el Sistema Empaquetado. Un programa en Crystal Clear es “diseño y programa”; sus programadores son diseñadores-programadores. En CC un diseñador que no programe no tiene cabida.

- **Experto en Negocios**

Junto con el Usuario Experto produce la Lista de Actores-Objetivos y el Archivo de Casos de Uso y Requerimientos. Debe conocer las reglas y políticas del negocio.

- **Coordinador**

Con la ayuda del equipo, produce el Mapa de Proyecto, el Plan de Entrega, el Estado del Proyecto, la Lista de Riesgos, el Plan y Estado de Iteración y la Agenda de Visualización.

- **Verificador**

Produce el Reporte de Bugs. Puede ser un programador en tiempo parcial, o un equipo de varias personas.

- **Escritor**

Produce el Manual de Usuario

El Equipo como Grupo es responsable de producir la Estructura y Convenciones del Equipo y los Resultados del Taller de Reflexión. A pesar que no contempla el desarrollo de software propiamente dicho, Crystal Clear involucra unos veinte productos de trabajo o artefactos. Mencionamos los más importantes:

Declaración de la misión

Documento de un párrafo a una página, describiendo el propósito.

Estructura del equipo

Lista de equipos y miembros.

Metodología

Comprende roles, estructura, proceso, productos de trabajo que mantienen, métodos de revisión.

Secuencia de entrega

Declaración o diagrama de dependencia; muestra el orden de las entregas y lo que hay en cada una.

Cronograma de visualización y entrega

Lista, planilla de hoja de cálculo o herramienta de gestión de proyectos.

Lista de riesgos

Descripción de riesgos por orden descendente de prioridad.

Estatus del proyecto

Lista hitos, fecha prevista, fecha efectiva y comentarios.

Lista de actores-objetivos

Lista de dos columnas, planilla de hoja de cálculo, diagrama de caso de uso o similar.

Casos de uso anotados

Requerimientos funcionales.

Archivo de requerimientos

Colección de información indicando qué se debe construir, quiénes han de utilizarlo, de qué manera proporciona valor y qué restricciones afectan al diseño.
Los métodos Crystal no prescriben las prácticas de desarrollo, las herramientas o los productos que pueden usarse, pudiendo combinarse con otros métodos como SCRUM, XP y Microsoft Solutions Framework.

16.2.11 Feature Driven Development (FDD)

Según Stephen R. Palmer, John M. (2002) FDD fue desarrollada por Jeff De Luca y Peter Coad a mediados de los años 90. Esta metodología se enfoca en iteraciones cortas, que permiten entregas tangibles del producto en un periodo corto de tiempo, de como máximo dos semanas.
FDD consiste en cinco procesos secuenciales durante los cuales se diseña y construye el sistema. La parte iterativa soporta desarrollo ágil con rápidas adaptaciones a cambios en requerimientos y necesidades del negocio. Cada fase del proceso tiene un criterio de entrada, tareas, pruebas y un criterio de salida. Típicamente, la iteración de un rasgo tiene un tiempo determinado de una a tres semanas.

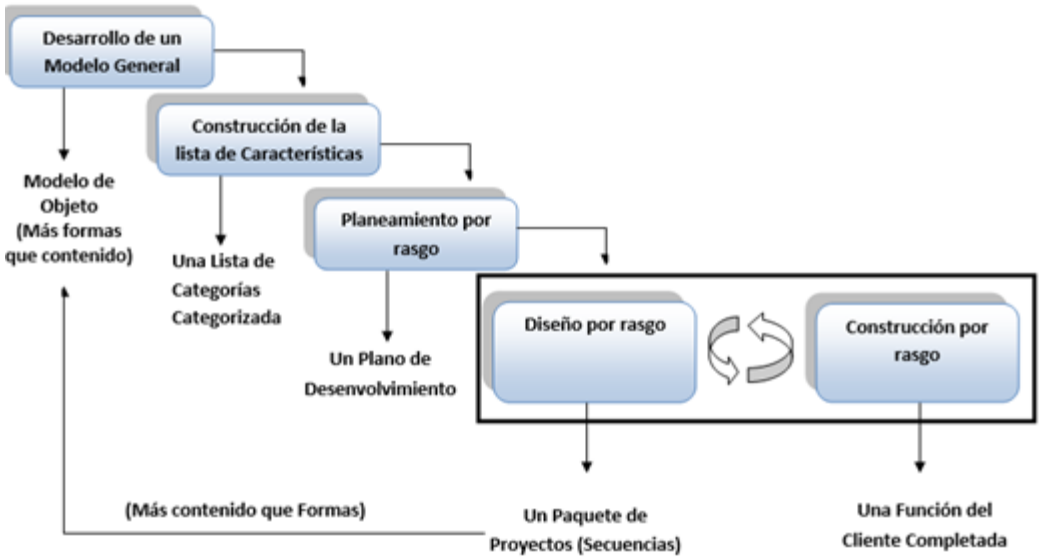


FIGURA N° 103: FDD
Fuente: elaboración propia a partir de Stephen R. Palmer, John M. (2002)

fases del modelo

La Figura 103 detalla las fases de FDD, las que se detallaran a continuación.

a) Desarrollo de un modelo general

Cuando comienza este desarrollo, los expertos de dominio ya están al tanto de la visión, el contexto y los requerimientos del sistema a construir. A esta altura se espera que existan requerimientos tales como casos de uso o especificaciones funcionales. FDD, sin embargo, no cubre este aspecto. Los expertos de dominio presentan un ensayo (walkthrough) en el que los miembros del equipo y el arquitecto principal se informan de la descripción de alto nivel del sistema. El dominio general se subdivide en áreas más específicas y se define un ensayo más detallado para cada uno de los miembros del dominio. Luego de cada ensayo, un equipo de desarrollo trabaja en pequeños grupos para producir modelos de objeto de cada área de dominio. Simultáneamente, se construye un gran modelo general para todo el sistema.

b) Construcción de la lista de características

Los ensayos, modelos de objeto y documentación de requerimientos proporcionan la base para construir una amplia lista de rasgos. Los rasgos son pequeños ítems útiles a los ojos del cliente. Son similares a las tarjetas de historias de XP y se escriben en un lenguaje que todas las partes puedan entender. Las funciones se agrupan conforme a diversas actividades en áreas de dominio

específicas. La lista de rasgos es revisada por los usuarios y patrocinador es para asegurar su validez y exhaustividad. Los rasgos que requieran más de diez días se descomponen en otros más pequeños.

c) Planeamiento por rasgo

Incluye la creación de un plan de alto nivel, en el que los conjuntos de rasgos se ponen en secuencia conforme a su prioridad y dependencia, y se asigna a los programadores jefes. Las listas se priorizan en secciones que se llaman paquetes de diseño. Luego se asignan las clases definidas en la selección del modelo general a programadores individuales, o sea propietarios de clases. Se pone fecha para los conjuntos de rasgos.

d) Diseño por rasgo y Construcción por rasgo

Se selecciona un pequeño conjunto de rasgos del conjunto y los propietarios de clases seleccionan los correspondientes equipos dispuestos por rasgos. Se procede luego iterativamente hasta que se producen los rasgos seleccionados. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. Puede haber varios grupos trabajando en paralelo. El proceso iterativo incluye inspección de diseño, codificación, prueba de unidad, integración e inspección de código. Luego de una iteración exitosa, los rasgos completos se promueven al build principal. Este proceso puede demorar una o dos semanas en implementarse.

FDD consiste en un conjunto de "mejores prácticas" que distan de ser nuevas pero se combinan de manera original. Las prácticas canónicas son:

•Modelado de objetos del dominio, resultante en un framework cuando se agregan los rasgos. Esta forma de modelado descompone un problema mayor en otros menores; el diseño y la implementación de cada clase u objeto es un problema pequeño a resolver. Cuando se combinan las clases completas, constituyen la solución al problema mayor.

•Una forma particular de la técnica es el modelado en colores, que agrega una dimensión adicional de visualización. Si bien se puede modelar en blanco y negro, en FDD el modelado basado en objetos es imperativo.

• Desarrollo por rasgo. Hacer simplemente que las clases y objetos funcionen no refleja lo que el cliente pide. El seguimiento del progreso se realiza mediante examen de pequeñas funcionalidades

descompuestas y funciones valoradas por el cliente. Un rasgo en FDD es una función pequeña expresada en la forma <acción><resultado>
<por | para | de | a><objeto> con los operadores adecuados entre los términos. Por ejemplo, calcular el importe total de una venta; determinar la última operación de un cajero; validar la contraseña de un usuario.

• Propiedad individual de clases (código). Cada clase tiene una sola persona nominada como responsable por su consistencia, performance e integridad conceptual.

• Equipos de Rasgos, pequeños y dinámicamente formados. La existencia de un equipo garantiza que un conjunto de mentes se apliquen a cada decisión y se tomen en cuenta múltiples alternativas.

• Inspección. Se refiere al uso de los mejores mecanismos de detección conocidos.

• Builds regulares. Siempre se tiene un sistema disponible. Los builds forman la base a partir de la cual se van agregando nuevos rasgos.

• Administración de configuración. Permite realizar seguimiento histórico de las últimas versiones completas de código fuente.

• Reporte de progreso. Se comunica a todos los niveles organizacionales necesarios.
FDD suministra un rico conjunto de artefactos para la planificación y control de los proyectos.

Los principios de FDD son pocos y simples:

- Se requiere un sistema para construir sistemas si se pretende escalar a proyectos grandes.
- Un proceso simple y bien definido trabaja mejor.
- Los pasos de un proceso deben ser lógicos y su mérito inmediatamente obvio para cada miembro del equipo.
- Vanagloriarse del proceso puede impedir el trabajo real.
- Los buenos procesos van hasta el fondo del asunto, de modo que los miembros del equipo se puedan concentrar en los resultados.
- Los ciclos cortos, iterativos, orientados por rasgos (features) son mejores.

Un miembro de un equipo puede tener otros roles a cargo, y un solo rol puede ser compartido por varias personas.

Hay tres categorías de rol en FDD:

- Roles claves
- Roles de soporte
- Roles adicionales.

Roles claves de un proyecto

- Administrador del proyecto, quien tiene la última palabra en materia de visión, cronograma y asignación del personal;
- Arquitecto jefe(puede dividirse en arquitecto de dominio y arquitecto técnico);
- Manager de desarrollo, que puede combinarse con arquitecto jefe o manager de proyecto;
- Programador jefe, que participa en el análisis del requerimiento y selecciona rasgos del conjunto a desarrollar en la siguiente iteración;
- Propietarios de clases, que trabajan bajo la guía del programador jefe en diseño, codificación, prueba y documentación, repartidos por rasgos y
- Experto de dominio, que puede ser un cliente, patrocinador, analista de negocios o una mezcla de todo eso.

Roles de soporte

- Administrador de entrega, que controla el progreso del proceso revisando los reportes del programador jefe y manteniendo reuniones breves con él; reporta al manager del proyecto;
- Abogado/gurú de lenguaje, que conoce a la perfección el lenguaje y la tecnología;
- Ingeniero de construcción, que se encarga del control de versiones de los builds y publica la documentación;
- Herramientista (toolsmith), que construye herramientas ad hoc o mantiene bases de datos y sitios Web y
- Administrador del sistema, que controla el ambiente de trabajo y el sistema cuando se lo entrega.

Roles adicionales

- Verificadores: verifica si el sistema que se construye satisfará necesidades del cliente.
- Encargados del despliegue: Responsables de convertir datos existentes al formato requerido por el nuevo sistema y participar en el desarrollo de nuevos lanzamientos.
- Escritores técnicos: responsable de la documentación para el usuario. (Manuales, guías, etc.)

16.2.12 Lean Software Development (LSD)

Poppendieck (2003) define la Figura 104 como los principios LSD.

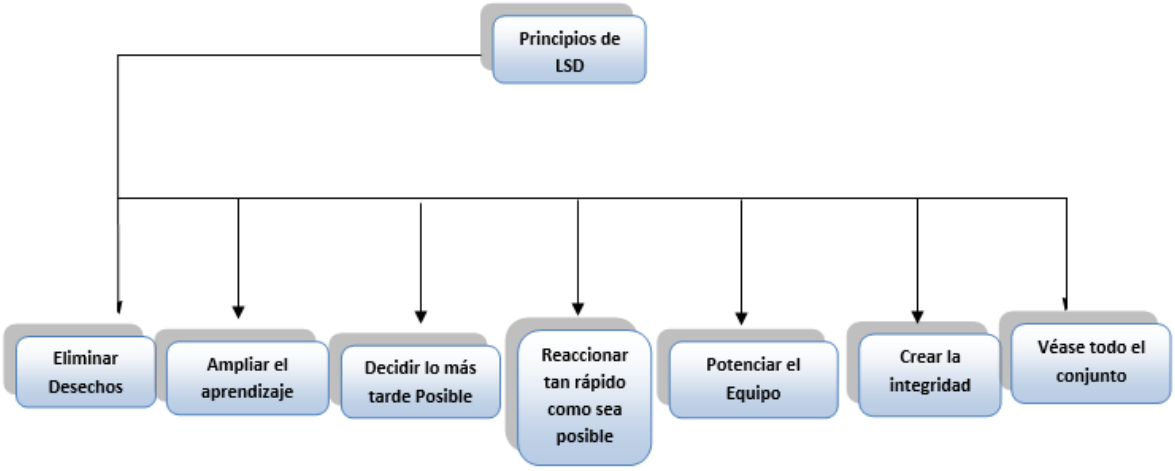


FIGURA N° 104: LSD
Fuente: elaboración propia a partir de Poppendieck (2003)

Fases del modelo

El desarrollo Lean puede resumirse en siete principios, similares a los principios fabricación Lean. A continuación de definen las fases del desarrollo lean.

a) Eliminar Desechos

Todo lo que no añade valor al cliente se considera un desperdicio: -código y funcionalidades innecesarias -retraso en el proceso de desarrollo de software -requisitos poco claros -burocracia -comunicación interna lenta

Con el fin de poder eliminar los desperdicios deberíamos ser capaces de reconocerlos y encontrarlos. Si alguna actividad podría ser excluida o el mismo resultado podría ser logrado sin ella, esta actividad es considerada un desperdicio. Los procesos y funcionalidades extra que no son

usados por el cliente son desperdicios. Las esperas ocasionadas por otras actividades, equipos o procesos son desperdicio. Los defectos y la baja calidad son los desperdicios. Los gastos de gestión que no producen valor real son desperdicios. Se utiliza una técnica llamada value streammapping (o mapa de flujo de valor) para distinguir y reconocer los desperdicios. El segundo paso consiste en señalar las fuentes de los desperdicios y eliminarlos. Lo mismo debe hacerse iterativamente hasta que incluso los procesos y procedimientos que parecían esenciales sean eliminados.

b) Ampliar el aprendizaje

El desarrollo de software es un proceso de aprendizaje continuo, a ello se le suman los retos de los equipos de desarrollo y el tamaño del producto final. El mejor enfoque para encarar una mejora en el ambiente de desarrollo de software es ampliar el aprendizaje. La acumulación de defectos debe evitarse ejecutando las pruebas tan pronto como el código está escrito en lugar de añadir más documentación o planificación detallada. Las distintas ideas podrían ser probadas escribiendo código e integrándolo. El proceso de recopilación de requisitos de usuarios podría simplificarse mediante la presentación de las pantallas de los usuarios finales para que estos puedan hacer sus aportes. El proceso de aprendizaje es acelerado con el uso iteraciones cortas cada una de ellas acompañada de refactorización y sus pruebas de integración.

Incrementando la retroalimentación mediante reuniones cortas con los clientes se ayuda a determinar la fase actual de desarrollo y se ajustan los esfuerzos para introducir mejoras en el futuro. Durante las reuniones, tanto los clientes como el equipo de desarrollo, logran aprender sobre el alcance del problema y buscan posibles soluciones para un mejor desarrollo. Por lo tanto, los clientes comprenden mejor sus necesidades basándose en el resultado de los esfuerzos del desarrollo y los desarrolladores aprenden a satisfacer mejor estas necesidades.

Otra idea para ampliar el aprendizaje es a través de la integración del cliente en el ambiente de desarrollo para concentrar la comunicación en las soluciones futuras y no en las soluciones posibles, promoviendo así el nacimiento de la solución a través del diálogo con el cliente.

c) Decidir lo más tarde posible

El desarrollo de software está siempre asociado con cierto grado de incertidumbre, los mejores resultados se alcanzan con un enfoque basado en opciones por lo que se pueden retrasar las decisiones tanto como sea posible hasta que éstas se basen en hechos y no en suposiciones y pronósticos inciertos. Cuanto más complejo es un proyecto, más capacidad para el cambio debe

incluirse en éste, así que debe permitirse el retraso de los compromisos importantes y cruciales. El enfoque iterativo promueve este principio: la capacidad de adaptarse a los cambios y corregir los errores, ya que un error podría ser muy costoso si se descubre después de la liberación del sistema.
Un enfoque de desarrollo de software ágil puede llevarles opciones rápidamente a los clientes, lo que implica, retrasar algunas decisiones cruciales hasta que los clientes hayan reconocido mejor sus necesidades. Esto también permite la adaptación tardía a los cambios y previene las costosas decisiones delimitadas por la tecnología. Esto no significa que no haya planificación involucrada en el proceso, por el contrario, las actividades de planificación deben centrarse en las diferentes opciones y se les adapta a la situación actual; así como, se deben clarificar las situaciones confusas estableciendo las pautas para una acción rápida. Evaluar las diferentes opciones es eficaz tan pronto como queda claro que ellos no son libres, pero proporcionando la flexibilidad necesaria para una tardía toma de decisiones.

d) Reaccionar tan rápido como sea posible

En la era de la rápida evolución de tecnológica, no es el más grande quien sobrevive, sino el más rápido. Cuanto antes se entrega el producto final sin defectos considerables más pronto se pueden recibir comentarios y se incorporan en la siguiente iteración. Cuanto más cortas sean las iteraciones, mejor es el aprendizaje y la comunicación dentro del equipo. Sin velocidad, las decisiones no pueden ser postergadas. La velocidad asegura el cumplimiento de las necesidades actuales del cliente y no lo que éste requería para ayer. Esto les da la oportunidad de demorarse pensando lo que realmente necesitan, hasta que adquieran un mejor conocimiento. Los clientes valoran la entrega rápida de un producto de calidad.
La ideología de producción Just In Time podría aplicarse a programas de desarrollo, reconociendo sus necesidades específicas y el ambiente. Lo anterior se logra mediante la presentación de resultados, la necesidad de dejar que el equipo se organizarse y dividiendo las tareas para lograr el resultado necesario para una iteración específica.
Al principio, el cliente dispone los requisitos necesarios. Esto podría ser simplemente presentar los requisitos en pequeñas fichas o historias y los desarrolladores estimarán el tiempo necesario para la aplicación de cada tarjeta. Así, la organización del trabajo cambia en sistema autopropulsado: cada mañana durante una reunión inicial cada miembro del equipo evalúa lo que se ha hecho ayer, lo que hay que hacer hoy y mañana y pregunta por cualquier nueva entrada necesaria de parte de sus colegas o del cliente. Esto requiere la transparencia del proceso, que es también beneficioso para la comunicación del equipo.

e) Potenciar el equipo

Ha habido una creencia tradicional en la mayoría de las empresas acerca de la toma de decisiones en la organización: los administradores no pueden decirle a los trabajadores cómo hacer su propio trabajo. En una técnica llamada Work-Out, los roles cambian: a los directivos se les enseña a escuchar a los desarrolladores, de manera que éstos puedan explicar mejor qué acciones podrían tomarse, así como ofrecer sugerencias para mejoras. Los directores de proyecto más experimentados simplemente han declarado la clave de éxito de los proyectos: "Buscar la buena gente y dejarles hacer su propio trabajo".

Otra creencia errónea ha sido considerar a las personas como recursos. Las personas podrían ser los recursos desde el punto de vista de una hoja de datos estadísticos, pero en el desarrollo de software, así como cualquier organización de negocios, las personas necesitan algo más que la lista de tareas y la seguridad de que no será alterada durante la realización de las tareas. Las personas necesitan motivación y un propósito superior para el cual trabajar: un objetivo alcanzable dentro de la realidad con la garantía de que el equipo puede elegir sus propios compromisos. Los desarrolladores deberían tener acceso a los clientes; el jefe de equipo debe proporcionar apoyo y ayuda en situaciones difíciles, así como asegurarse de que el escepticismo no arruine el espíritu de equipo.

f) Crear la integridad

El siempre exigente cliente debe tener una experiencia general del sistema. A esto se le llama percepción de integridad: ¿cómo es publicitado, entregado, implementado y accedido? ¿Qué tan intuitivo es su uso? ¿Precio? ¿Qué tan bien resuelve los problemas?. Integridad Conceptual significa que los componentes separados del sistema funcionan bien juntos, como en un todo, logrando equilibrio entre la flexibilidad, mantenibilidad, eficiencia y capacidad de respuesta. Esto podría lograrse mediante la comprensión del dominio del problema y resolviéndolo al mismo tiempo, no secuencialmente. La información necesaria es recibida por los pequeños lotes, no en una vasta cantidad y con una preferible comunicación cara a cara, sin ninguna documentación por escrito. El flujo de información debe ser constante en ambas direcciones, a partir del cliente a los desarrolladores y viceversa, evitando así la gran y estresante cantidad de información después de un largo periodo de desarrollo en el aislamiento.

Una de las maneras más saludables hacia una arquitectura integrante es la refactorización. Cuantas más funcionalidades se añaden a las del sistema, más se pierde del código base para futuras mejoras. Así como en la Programación extrema (XP), la refactorización es mantener la sencillez, la

claridad, la cantidad mínima de funcionalidades en el código. Las repeticiones en el código son signo de un mal diseño de código y deben evitarse. El completo y automatizado proceso de construcción debe ir acompañada de una suite completa y automatizada de pruebas, tanto para desarrolladores y clientes que tengan la misma versión, sincronización y semántica que el sistema actual. Al final, la integridad debe ser verificada con una prueba global, garantizando que el sistema hace lo que el cliente espera que haga. Las pruebas automatizadas también son consideradas como parte del proceso de producción y, por tanto, si no agregan valor deben considerarse residuos. Las pruebas automatizadas no deberían ser un objetivo, sino, un medio para un fin; específicamente para la reducción de defectos.

g) Véase todo el conjunto

Los sistemas de software hoy en día no son simplemente la suma de sus partes, sino también el producto de sus interacciones. Los defectos en el software tienden a acumularse durante el proceso de desarrollo por medio de la descomposición de las grandes tareas en pequeñas tareas y de la normalización de las diferentes etapas de desarrollo. Las causas reales de los defectos deben ser encontradas y eliminadas. Cuanto más grande sea el sistema, más serán las organizaciones que participan en su desarrollo y más partes son las desarrolladas por diferentes equipos y mayor es la importancia de tener bien definidas las relaciones entre los diferentes proveedores con el fin de producir una buena interacción entre los componentes del sistema.

La manera de pensar ofrecida Lean tiene que ser bien entendida por todos los miembros de un proyecto antes de aplicarlo de manera concreta en una situación de la vida real.

"Piensa en grande, actúa en pequeño, equivócate rápido; aprende con rapidez" estas consignas resumen la importancia de comprender el terreno y la idoneidad de implementar los principios Lean lo largo del proceso de desarrollo de software. Sólo cuando todos los principios de Lean se aplican al mismo tiempo, combinado con un fuerte "sentido común" en relación con el ambiente de trabajo, hay una base para el éxito en el desarrollo de software.

16.2.13 Kanban

Según Shingo (1989) Kanban está basado en la simple idea de que el Trabajo en Progreso debe ser limitado, y que algo nuevo solo debería empezarse recién cuando una pieza existente de trabajo es entregada. El Kanban implica una señal visual de que un nuevo fragmento de trabajo puede ser comenzado debido a que la cantidad de trabajo en progreso está por debajo del límite acordado. Podemos decir que Kanban es una herramienta que nos sirve para ordenar la forma en la que trabajamos, que sin dudas irá mejor para algunos escenarios, y peor para otros.

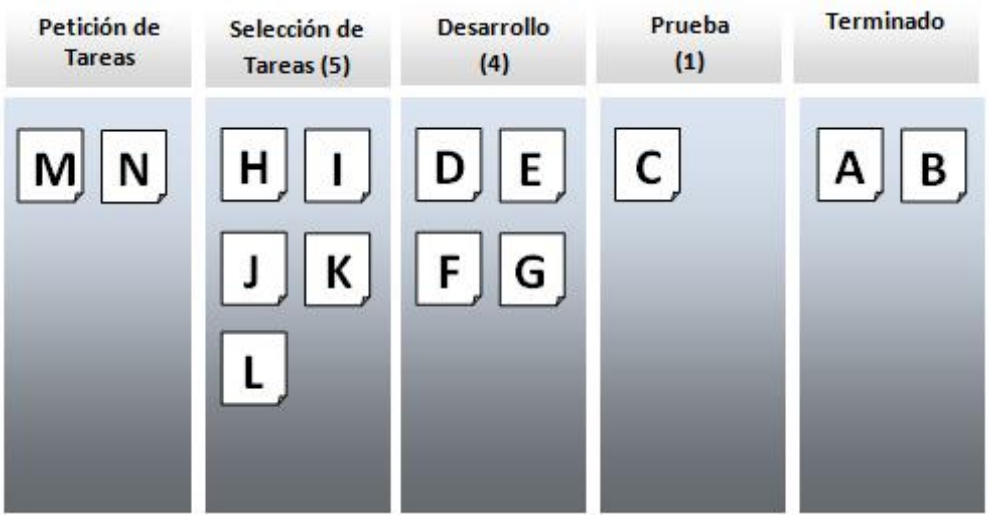


FIGURA N° 105: Kanban
Fuente: elaboración propia a partir de Shingo (1989)

Principios de Kanban

Visualizar el flujo de trabajo

Consiste en la visualización de todo el proceso de desarrollo, mediante un tablero físico, generalmente, públicamente asequible. El objetivo de mostrar el proceso, consiste en:

Entender mejor el proceso de trabajo actual;

Conocer los problemas que puedan surgir y tomar decisiones;

Mejorar la comunicación entre todos los interesados/participantes del proyecto;

Hacer los futuros procesos más predecibles.

Limitar la cantidad de trabajo en progreso

Una de las principales ideas del Kanban es que el trabajo en curso (work in progress) debería estar limitado, es decir, que el número máximo de tareas que se pueden realizar en cada fase debe ser conocido.

En Kanban se debe definir cuantas tareas como máximo puede realizarse en cada fase del ciclo de trabajo, a ese número de tareas se le llama límite del "work in progress". A esto se añade otra idea tan razonable como que para empezar con una nueva tarea alguna otra tarea previa debe haber finalizado.

> **Backlog:** Un conjunto priorizado de tareas a hacer por el equipo. ¿Quién prioriza las tareas? el jefe del departamento, el miembro del equipo que conozca más a fondo el dominio, etc. Kanban no impone un rol para esto.
>
> **In Progress:** Con las tareas actualmente siendo trabajadas.
>
> **Done:** Los ítems que fueron terminados.
>
> **Etcétera:** El tablero Kanban puede contar con más columnas, dependiendo del flujo de trabajo en particular. Por ejemplo, si la validación del código es realizada por una porción del equipo, puede haber otra columna llamada "In Review / QA".

Ventajas

Ventajas de desarrollar utilizando kanban:

- Reducción del trabajo en progreso (limitación del WIP).
- Control del flujo de trabajo (fácil localización de cuellos de botella, y control sencillo del estado de desarrollo del producto).
- Fácil de aplicar (apenas existen reglas y se puede llevar a cabo de forma artesanal, no requiere software adicional).
- Promueve el trabajo en equipo (los desarrolladores se comunican entre sí y se ayudan cuando alguna actividad da problemas).
- Reducción del tiempo perdido (todo el desarrollo se hace bajo demanda, por lo que no hay partes que sobren, además si alguien acaba su trabajo ayuda al resto del equipo).
- Facilidad para añadir mejoras (el tablero permite que todo el equipo de desarrollo pueda ver cómo se está haciendo el producto desde una visión global, lo que permite que puedan reflejar a sus compañeros diversas mejoras sobre el mismo).

Desventajas

Desventajas de desarrollar utilizando kanban:

- Dificultad de realizar las entregas a tiempo en grandes proyectos (dado que no hay un control específico del tiempo empleado en cada actividad, en grandes proyectos pueden acumularse un gran número de pequeños retrasos que provocarían la demora en la entrega del producto final).
- Falta de reglas (aunque la existencia de pocas reglas es una ventaja, puede convertirse en un problema cuando el desarrollador es inexperto y necesita una guía para realizar su trabajo. Por ello, se aconseja hacer uso de Kanban tras haber ganado experiencia con otras metodologías, ya que de este modo habrá reglas que se habrán interiorizado).
- Dificultad a la hora de prever posibles problemas (aunque la localización y solución de problemas es sencilla en Kanban, su previsión antes de que ocurran no lo es).

Cuando es apropiado utilizarlo este modelo

Kanban es una metodología efectiva en proyectos de corta-media duración en los que el equipo de desarrollo tiene experiencia en otras metodologías de desarrollo, dado que con el uso de Kanban se ahorran un gran número de reglas innecesarias, se promueve el trabajo en equipo y se reduce el tiempo perdido.

16.2.14 Open Up

Según información oficial de empresa Eclipse dedicada al desarrollo, Open Up se define de la siguiente manera.

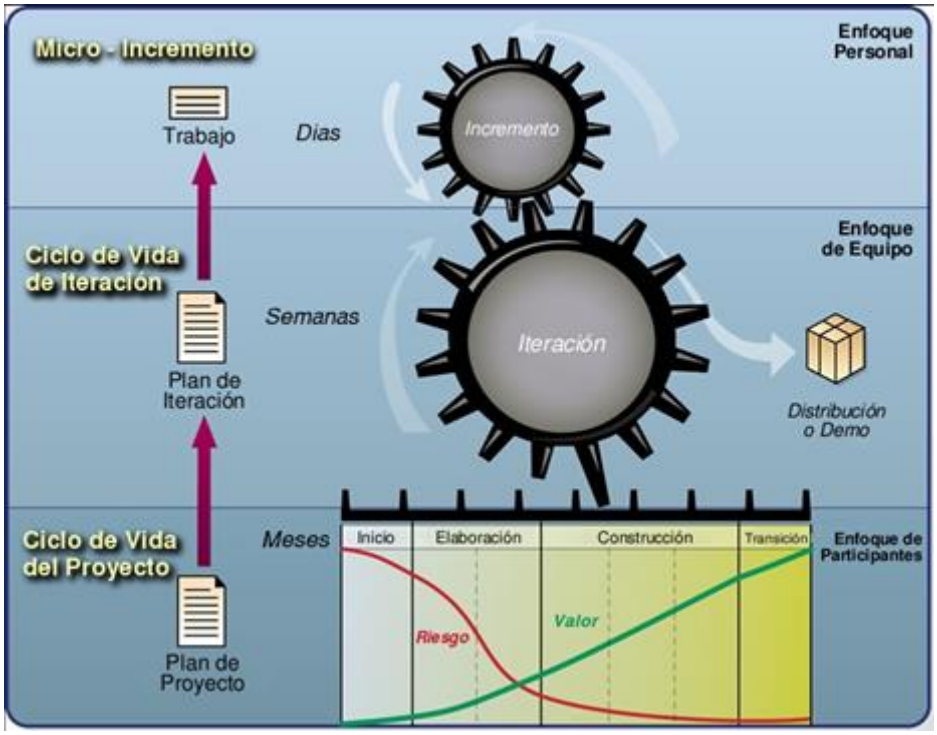


FIGURA N° 106: Open Up
Fuente: EPL (2008)

Open Up es un método y un proceso de desarrollo de software propuesto por un conjunto de empresas de tecnología, quienes lo donaron en el año 2007 a la Fundación Eclipse. La fundación lo ha publicado bajo una licencia libre y lo mantiene como método de ejemplo dentro del proyecto Eclipse Process Framework.

Descripción

El Open Up es un proceso mínimo y suficiente, lo que significa que solo el contenido fundamental y necesario es 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. La mayoría de los elementos de Open Up están declarados para fomentar el intercambio de información entre los equipos de desarrollo y mantener un entendimiento compartido del proyecto, sus objetivos, alcance y avances.

Principios del Open Up

Colaborar para sincronizar intereses y compartir conocimiento. Este principio promueve prácticas que impulsan un ambiente de equipo saludable, facilitan la colaboración y desarrollan un conocimiento compartido del proyecto.

Equilibrar las prioridades para maximizar el beneficio obtenido por los interesados en el proyecto. Este principio promueve prácticas que permiten a los participantes de los proyectos desarrollar una solución que maximice los beneficios obtenidos por los participantes y que cumple con los requisitos y restricciones del proyecto.

Centrarse en la arquitectura de forma temprana para minimizar el riesgo y organizar el desarrollo. Desarrollo evolutivo para obtener retroalimentación y mejoramiento continuo. Este principio promueve prácticas que permiten a los equipos de desarrollo obtener retroalimentación temprana y continua de los participantes del proyecto, permitiendo demostrarles incrementos progresivos en la funcionalidad.

Organización de los componentes del Open Up

El Open Up está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso. El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método.

Los elementos del Open Up dirigen la organización del trabajo en los niveles personal, de equipo y de interesados.

A nivel personal, los integrantes de un proyecto contribuyen con su trabajo con pequeños incrementos en funcionalidad, denominados micro-incrementos, los cuales representan los resultados obtenidos en pocas horas o pocos días de trabajo. La solución evoluciona basada en dichos micro-incrementos de tal forma que el progreso puede ser visualizado efectivamente cada día. Los integrantes del equipo de desarrollo de forma abierta comparten su progreso diario el cual incrementa la visibilidad en el trabajo, la confianza y el trabajo en equipo.

El proyecto en general se divide en iteraciones, las cuales son planificadas en un intervalo definido de tiempo que no superan las pocas semanas. El Open Up tiene elementos que ayudan a los equipos de trabajo a enfocar los esfuerzos a través del ciclo de vida de cada iteración de tal forma que se

puedan distribuir funcionalidades incrementales de una manera predecible, una versión totalmente probada y funcional al final de cada iteración.

El Open Up estructura el ciclo de vida de un proyecto en cuatro fases: concepción, elaboración, construcción y transición. El ciclo de vida del proyecto provee a los interesados un mecanismo de supervisión y dirección para controlar los fundamentos del proyecto, su ámbito, la exposición a los riesgos, el aumento de valor y otros aspectos.

16.3 Justificación de selección de criterios

La literatura indica una serie de criterios a considerar al momento de seleccionar una metodología de desarrollo. A continuación se describe los principales criterios incorporados en este estudio:

a) Presupuesto apropiado: Pressman (2010) indica que el presupuesto debe estar acorde al proyecto evitando el parar el desarrollo retrasando y aumentar plazos establecidos.

b) Requerimientos específicos: Pressman (2010) indica que contar con requerimientos específicos al inicio del proyecto permite definir tareas, alcance y la naturaleza del problema que se va a resolver, mientras se define lo que se requiere en el proceso de elaboración donde se refina y modifican los requerimientos.

c) Variabilidad de requerimientos: Pressman (2010) afirma que si bien los cambios son inevitables estos deben tomarse en cuenta y con mucha importancia, ya que a medida que avanza el tiempo, el cambio puede afectar en mayor medida al desarrollo del proyecto, sobre todo en metodologías tradicionales como es cascada.

d) Participación del cliente: En el estudio de Ahimbisibwe, Cavana&Daellenbach (2015) se recolectaron diversos factores críticos de éxito, y entre los factores que competen al cliente se agregó la participación de este.

e) Adaptabilidad a cambios: Pressman (2010) indica que los cambios suceden, pero es relevante tener esto en cuenta ya que los cambios al transcurrir el tiempo de desarrollo afectan en mayor medida al progreso.

Es relativamente fácil efectuar un cambio cuando el equipo de software reúne los requerimientos (al principio de un proyecto). El escenario de uso tal vez tenga que modificarse, la lista de funciones puede aumentar, o editarse una especificación escrita. Los costos de hacer que esto funcione son mínimos, y el tiempo requerido no perjudica el resultado del proyecto. Pero, ¿qué pasa una vez transcurridos algunos meses? El equipo está a la mitad de las pruebas de validación (algo que ocurre cuando el proyecto está relativamente avanzado) y un participante de importancia solicita que se haga un cambio funcional grande. El cambio requiere modificar el diseño de la arquitectura del software, el diseño y construcción de tres componentes nuevos, hacer cambios en otros cinco componentes, diseñar nuevas pruebas, etc. Los costos aumentan con rapidez, y no son pocos el tiempo y el dinero requeridos para asegurar que se haga el cambio sin efectos colaterales no intencionados.

f) Tiempo desarrollo: Basado en el estudio de Ahimbisibwe, Cavana&Daellenbach (2015) señala que la calendarización del proyecto se vuelve un punto importante al momento de querer reducir el tiempo de desarrollo del proyecto.

g) Tamaño proyecto: Pressman (2010) indica el tamaño del proyecto es otro factor importante que puede afectar la precisión y la eficacia de las estimaciones.

h) Experiencia desarrolladores - Problemática similar - Requerimientos similares: Germain & Robillard (2005) hacen notar que la experiencia de los equipos en el uso de los lenguajes de programación, problemática y requerimientos similares son de alta relevancia, ya que el desconocimiento puede causar el retraso del desarrollo de un sistema. Dentro de la experiencia de los equipos, se considera también la experiencia del jefe de proyecto.

i) Riesgos del proyecto: Pressman (2010) indica que los riesgos del proyecto amenazan el plan del proyecto, es decir, si los riesgos del proyecto se vuelven reales, es probable que el calendario del proyecto se deslice y que los costos aumenten. Los riesgos del proyecto identifican potenciales problemas de presupuesto, calendario, personal (tanto técnico como en la organización), recursos, participantes y requisitos, así como su impacto sobre un proyecto de software. En el capítulo 26, la complejidad, el tamaño y el grado de incertidumbre estructural del proyecto también se definieron como factores de riesgos para el proyecto (y la estimación).

j) Complejidad de desarrollo: Basado en Ozturk (2013) un estudio relacionado a la selección con la lógica difusa, la complejidad del desarrollo se seleccionó como importante al momento de escoger un tipo de metodología. También están dentro de los criterios que usaron para generar una tabla para comparar las diversas metodologías.

k) Documentación de proyecto: Pressman (2010) indica que comparan la documentación generada por los métodos tradicionales con los métodos ágiles, diciéndonos que métodos tradicionales se enfocan en documentación exacta sin errores, que un software funcional, por otro lado los métodos ágiles se enfocan en entregar un producto funcional dejando en segundo plano la documentación exhaustiva.

l) Tamaño del equipo desarrollo: Basado en el estudio de Ahimbisibwe, Cavana & Daellenbach (2015) dentro de sus parámetros se toma en cuenta el tamaño de los equipos, ya que como conclusión en la revisión de documentos que realizaron establecieron de acuerdo a la teoría que en el desarrollo tradicional se usan grandes equipos, así como en el desarrollo ágil los equipos que trabajan son más pequeños, convirtiendo este criterio en un punto importante a la hora de seleccionar un tipo de metodología.

En base a la teoría se generó un instrumento para medir la importancia que dan las empresas dedicadas al desarrollo de software a los diferentes criterios seleccionados (Anexo Encuesta).

17 METODOLOGÍA DELA INVESTIGACIÓN

17.1 Recolección de datos

Para la obtención de datos empresariales se analizó una base de datos perteneciente al directorio de empresas y ejecutivos de Chile del año 2015, en la que se encontraban más de 20.000 ejecutivos de diferentes áreas. Luego del análisis se encontraron alrededor de 170 ejecutivos de diferentes empresas Chilenas pertenecientes a áreas de informática.

En nuestro estudio piloto se realizó un análisis de la validez de contenido de la encuesta, de la cual participaron cuatro expertos en el área de metodologías y desarrollo de software.

Para recolectar información necesaria, se examinan diversas alternativas de software y herramientas online para crear una encuesta que pueda ser usada en la recolección de datos, tales como; Survio, Typeform, ZohoSurvey, Lime Survey y formularios de Google drive. Se opta por Google Drive ya que ofrece las mismas características de los otros software de encuestas pero de forma gratuita y sin restricciones.

Una vez enviadas las encuestas por correo a las empresas Chilenas que cuentan con área o departamento de informática se determinó una espera de una semana para obtener las respuestas. Pasado el plazo de espera se realiza recordatorios por correos y llamados telefónicos. Además se realizó un estudio de campo en las empresas más cercanas de la zona de Concepción dentro de las cuales se visitó empresas de desarrollo de software, de áreas de salud, educación y comercio automotriz. Finalmente se obtienen respuesta de 35 empresas de las cuales se obtuvieron 54 proyectos diferentes.

18 ANEXO: ENCUESTA

UNIVERSIDAD DEL BÍO-BÍO

Sistema de Recomendación de Metodologías de Desarrollo de Software

El objetivo de este proyecto es desarrollar un sistema de recomendación piloto para apoyar a los encargados de tomar la decisión de optar por una metodología de desarrollo de sistemas. Para esto es importante contar con su opinión respecto a los factores que podrían influir en el éxito o fracaso de un proyecto de desarrollo de software basado en una metodología específica. Se le solicitará contestar algunas preguntas para identificar los factores de un proyecto exitoso o fracasado. En el caso que quisiera informar más de un proyecto puede responder esta encuesta nuevamente.

Se define Sistema de Recomendación como una herramienta que genera sugerencias sobre un determinado objeto de estudio a partir de las preferencias y opiniones dadas por usuarios. Por otro lado, se define Metodología de Desarrollo de Software como un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.

Todas sus respuestas son completamente confidenciales y su uso será sólo para fines académicos. Por favor, ingrese el mismo correo electrónico al que se le envió la encuesta con el fin de evitar se le envíen nuevas solicitudes.
Esta encuesta no le debería tomar más de 10 minutos en contestar.

Como un incentivo a su participación, al final del proyecto se le enviara un link en el cual podrá hacer uso del sistema de recomendación resultante de esta investigación en sus futuros proyectos.

Dirección de correo electrónico: .....

18.1 I) Información demográfica

Género:	__ Masculino	__ Femenino
Edad:	__ Menor de 30 años __ Entre 40 y 50 años	__ Entre 30 y 40 años __ Mayor de 50 años
Educación:	__ Técnico Profesional __ Maestría/MBA __ Otro:_____	__ Universitario __ Doctorado
¿Cuántos años lleva en su cargo?		
¿Cuántos años lleva trabajando en su empresa?		
¿Cuánto de su opinión personal influye en la decisión final de adoptar una metodología de desarrollo de software? indique su respuesta en %.		
Si su respuesta anterior fue menor de 100%, indique quienes toman la decisión de adoptar una metodología de desarrollo.		

18.2 II) Información de las Metodologías de Desarrollo

1. A continuación se presentan 4 grupos de metodologías de desarrollo de las cuales se da una pequeña reseña. Indique cuál o cuáles de estas metodologías utiliza más comúnmente en el desarrollo de proyectos de software (la respuesta de esta pregunta es la base para contestar las siguientes).

__ Metodologías Lineales (p.ej. Cascada, Cascada con Iteración, Cascada con Sub-proyectos, SSADM, CASE de Oracle, otros.) El proyecto progresa a través de una secuencia ordenada de pasos partiendo desde el concepto inicial del software hasta la prueba del sistema, revisando cada etapa antes de pasar a la siguiente.

Especifique su selección:_____

__ Metodologías Evolutivas (p.ej. Prototipo Evolutivo, Entrega Evolutiva, Modelo Espiral, Cascada con reducción de riesgos, Cascada con fases solapadas, otros.) En el desarrollo del proyecto es frecuente que los requerimientos cambien conforme avance el proyecto. A medida que progresa el proyecto el desarrollador y el usuario comprenden y reaccionan mejor ante riesgos en cada uno de los niveles evolutivos (las etapas del desarrollo no se dan por terminadas). Se caracteriza por la manera en la que permiten desarrollar versiones cada vez más completas del software conforme avance el tiempo.

Especifique su selección:_____

__ Metodologías Incrementales (p.ej. Modelo Incremental, RUP, Entrega por etapas, Diseño por planificación, otros.) Este modelo entrega el software en pequeñas partes utilizables llamadas incrementos. Los primeros pasos los puede realizar un grupo reducido de personas y en cada incremento se puede añadir personal. Por otro lado los incrementos se pueden planear para gestionar riesgos técnicos.

Especifique su selección:_____

__ Metodologías Ágiles (p.ej. SCRUM, XP, DSDM, ASD, Crystal Clear, FDD, LSD, Kanban, Open Up, otros.) Estas metodologías iterativas e incrementales buscan la satisfacción del cliente y la entrega rápida de software, los equipos suelen ser pequeños para efectuar el proyecto, su comunicación es activa y continua entre desarrolladores y usuario.

Especifique su selección:_____

__ Prácticas Ágiles. Se refiere a las prácticas utilizadas en las metodologías agiles, pero no la implementación completa de una metodología (p.ej. Sprints, reuniones diarias, programación en parejas, historias de usuario, entre otras.)

Especifique su selección:_____

__ No Utiliza Metodología

Indique porque:_____

__ Otra:

Especifique:_____

18.3 III) Considere un proyecto que haya realizado.

Por favor, considere un proyecto que haya realizado, que sea representativo del uso de la(s) metodología(s) seleccionada(s) anteriormente.

2. De acuerdo a la(s) metodología(s) utilizada(s), ¿Consideraría un éxito o fracaso el desarrollo de su proyecto? (Considerando como éxito factores como la entrega a tiempo, costos programados, cumplimiento de objetivos, mejoramiento de procesos, entre otros). Justifique su respuesta.

_____
_____

3. ¿En qué ámbito fue enfocado el proyecto?

__ Área administrativa o gerencial
__ Área de operaciones
__ Área de desarrollo
__ Área de soporte
__ Otro._____

4. Tamaño del proyecto realizado

__ Pequeño
__ Mediano
__Grande

5. Indique la variable que consideró para determinar la respuesta anterior. (ejemplo, tamaño del equipo, funcionalidades del software, líneas de código, etc.)

_____
_____

6. ¿Cómo clasificaría el tiempo involucrado en el desarrollo del proyecto?

__ Corto (menos de 6 meses)
__ Medio (entre 6 meses y 1 año)
__ Largo (más de 1 años)

7. ¿Cómo fue el tiempo de desarrollo real, respecto al estimado? ¿Se cumplieron los plazos establecidos?

_____

8. La experiencia del equipo de desarrollo en el uso de la metodología era:

__ Nula (la mayoría de equipo no conocía la metodología utilizada)
__ Mínima (la mayoría del equipo conocía a grandes rasgos la metodología, pero no lo suficiente)
__ Media (la mayoría del equipo ya habían trabajado con la pero no son expertos)
__ Alta (la mayoría del equipo conocía todos los puntos de la metodología y saben aplicarla de manera óptima)

9. ¿Se consideró la gestión del riesgo en el desarrollo del proyecto?

__ Si
__ No

10. Indique el tamaño del equipo de desarrolladores que participó en el proyecto anterior.

__ Menos de 4 personas
__ Entre 5 y 9 personas
__ Más de 9 personas

11. ¿Considera que el presupuesto disponible para desarrollar el proyecto fue holgado o estrecho?

__Holgado

__Estrecho

18.4 IV) Información de la importancia de ciertos criterios.

Por favor, seleccione la opción que mejor represente su opinión respecto de la importancia que atribuye a cada uno de los siguientes factores, en el resultado que tuvo el proyecto considerado:

¿Qué tan importante fue	Nada Importante	Poco Importante	Medianamente Importante	Importante	Muy Importante
haber enfrentado un problema similar anteriormente?	1	2	3	4	5
haber trabajado en requerimientos similares?	1	2	3	4	5
que los requerimientos sean bien especificados?	1	2	3	4	5
la variabilidad de los requerimientos?	1	2	3	4	5
el aporte que pueda entregar el usuario durante el desarrollo del proyecto?	1	2	3	4	5
Contar con certidumbre en la estimación del tiempo de desarrollo del proyecto?	1	2	3	4	5
el tamaño de					

proyecto?	1	2	3	4	5
experiencia del jefe de proyecto?	1	2	3	4	5
la experiencia de los participantes del equipo en el desarrollo del software?	1	2	3	4	5
considerar el riesgo explícitamente asociado al proyecto?(tiempo, costo, etc.)	1	2	3	4	5
la complejidad de desarrollo del proyecto?	1	2	3	4	5
documentar el proyecto durante el desarrollo?	1	2	3	4	5
contar con un presupuesto apropiado (holgado) para el proyecto?	1	2	3	4	5

¿Considera algún otro factor como importante al momento de decidir por una metodología de desarrollo de software cuando realice un próximo proyecto?

__ Si

__ No

Si su respuesta es Sí, indique cuál _____

Muchas gracias por su participación!

19 ANEXO: DICCIONARIO DE DATOS DEL MODELO DE DATOS

Entidad/Tabla: Usuario

Descripción: Tabla en la que se almacenan datos asociados a los usuarios (usuarios comunes y administrador) del sistema.

Atributo	Descripción	Tipo Dato	Dominio
Idusuario	Identificador único.	Integer	Números naturales
Nombreusuario	Nombre del usuario o administrador del sistema.	Variable characters	N/A
Email	Correo electrónico del usuario del sistema.	Variable characters	N/A
Contraseña	Contraseña de la cuanta de acceso al sistema.	Variable characters	N/A
Rol	Número 1 o 2 el que indica si es usuario (1) o administrador (2).	Integer	Números naturales
Activo	Número 0 o 1 el que indica si es un usuario con cuenta activada (1), la cual se puede desactivar (0).	Integer	Números naturales

Entidad/Tabla: Grupo de metodologías

Descripción: Tabla en la que se almacenan datos asociados a los diferentes grupos de metodologías.

Atributo	Descripción	Tipo Dato	Dominio
id_grupo_metodologia	Identificador único.	Integer	Números naturales
nombre_grupo_metodologia	Nombre representativo del grupo de metodologías.	Variable characters	N/A
descripcion_del_grupo_de_metodologia	Descripción del grupo de metodologías.	Long variable characters	N/A

Entidad/Tabla: Metodologías

Descripción: Tabla en la que se almacenan datos asociados a las diferentes metodologías pertenecientes a los diferentes grupos de metodologías.

Atributo	Descripción	Tipo Dato	Dominio
id_metodologia	Identificador único.	Integer	Números naturales
id_grupo_metodologia	Clave foránea.	Integer	Números naturales
nombre_de_metodologia	Nombre representativo de una metodología.	Variable characters	N/A
descripcion_de_metodologia	Descripción de la metodología.	Long variable characters	N/A
imagen_de_metodologia	Imagen representativa de la metodología.	Image	N/A

Entidad/Tabla: Prácticas ágiles

Descripción: Tabla en la que se almacenan datos asociados a las diferentes prácticas ágiles.

Atributo	Descripción	Tipo Dato	Dominio
Id_practica_agil	Identificador único.	Integer	Números naturales
nombre_practica_agil	Nombre representativo de la práctica ágil.	Variable characters	N/A
descripcion_practica_agil	Descripción de la práctica ágil.	Long variable characters	N/A
imagen_practica_agil	Imagen representativa de la práctica ágil.	Image	N/A

Entidad/Tabla: Acerca de

Descripción: Tabla en la que se almacenan datos asociados a acerca de, en la cual se encontrara información del sistema, estudio, etc.

Atributo	Descripción	Tipo Dato	Dominio
id_acercade	Identificador único.	Integer	Números naturales
Descripción	Descripción de acerca de (que es el sistema de recomendación, propósito, etc.).	Long variable characters	N/A

Entidad/Tabla: Cuestionario

Descripción: Tabla en la que se almacenan datos asociados al cuestionario, pregunta - respuesta.

Atributo	Descripción	Tipo Dato	Dominio
idPregunta	Identificador único.	Integer	Números naturales
Pregunta	Preguntas del cuestionario.	Variable characters	N/A
Respuesta	Respuesta asociada a cada pregunta del cuestionario.	Variable characters	N/A

Entidad/Tabla: Respuestas

Descripción: Tabla en la que se almacenan datos asociados a las posibles respuestas que pudiera tener cada una de las preguntas del cuestionario.

Atributo	Descripción	Tipo Dato	Dominio
idPropia	Identificador único.	Integer	Números naturales
idPregunta	Clave foránea.	Integer	Números naturales
posiblerespuesta	Posibles respuestas asociadas a cada pregunta del cuestionario.	Variable characters	N/A

Entidad/Tabla: Registro de respuestas

Descripción: Tabla en la que se almacenan datos asociados a las respuestas ya ingresadas al sistema por medio del cuestionario, para su calificación.

Atributo	Descripción	Tipo Dato	Dominio
Idregistro	Identificador único.	Integer	Números naturales
Idusuario	Clave foránea.	Integer	Números naturales
Fecharegistro	Fecha en la cual se respondió cuestionario y generó una recomendación.	Date	Fechas validas
pregunta1	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta2	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta3	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta4	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta5	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta6	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta7	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta8	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta9	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta10	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta11	Pregunta con su respuesta almacenada.	Variable characters	N/A
pregunta12	Pregunta con su respuesta almacenada.	Variable	N/A

		characters	
pregunta13	Pregunta con su respuesta almacenada.	Variable characters	N/A
Calificacion	Calificación de la recomendación generada por el sistema según respuestas ingresadas.	Variable characters	N/A

Entidad/Tabla: Sugerencias

Descripción: Tabla en la que se almacenan datos asociados a las posibles sugerencias ingresadas por usuario del sistema.

Atributo	**Descripción**	**Tipo Dato**	**Dominio**
Idsugerencia	Identificador único.	Integer	Números naturales
Nombre	Nombre de usuario que ingresa una sugerencia (opcional).	Variable characters	N/A
Sugerencia	Sugerencia ingresa por usuario respecto al funcionamiento del sistema o recomendación generada.	Variable characters	N/A