

**Software Project Management Plan**

**(SPMP)**

**Presentado a:**

Ing. Jaime Pavlich PhD.

Ing. Miguel Torres MSc.

Luisa Alvarez

Stiven Ávila

David Calle

Fabián Merchán

Luis Montaño

Alfredo Santamaría

1. Historial de cambios

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VERSIÓN | FECHA | SECCIÓN DEL DOCUMENTO | DESCRIPCIÓN DEL CAMBIO | RESPONSABLE |
| 0.1 | 1/03/15 | 10.2 | Monitoreo u control de progreso | SoS Master |
| 0.2 | 07/03/15 | 1, 12.5, 6.4, 8.1, 8. 4, 2, 9.3.1, 9.3.2 | Se añade el prefacio, control de calidad, entregables, ciclo de vida, interfaces externas, actividades calendario | SoS Master, Scrum Master de Calidad y Administración |
| 0.3 | 08/03/15 | 9.3.4, 12.2, 6.5 | Se añade la asignación de presupuesto y el plan de análisis y administración de riesgos, resumen de calendarización y presupuesto. | SoS Master, Equipo área de calidad y administración de la configuración |
| 0.4 | 10/03/2015 | 6.6 | Se añade evolución del plan | SoS Master |
| 0.5 | 10/03/2015 | 8.4.2 | Se añade la sección 8.4.3 | SoS Master |
| 0.6 | 08/04/2015 | 10.1 | Se añade gestión de requerimientos | Equipo área desarrollo |

1. Prefacio

Este documento presenta el desarrollo del Software Project Management Plan aplicado a la creación de una aplicación denominada RAWR, la cual pretende ser una red social de mascotas. Fue desarrollado de manera conjunta por Stiven Ávila, Luisa Álvarez, David Calle, Fabián Merchán, Luis Montaño y Alfredo Santamaría. Todos estudiantes de la Pontificia Universidad Javeriana, inscritos en el programa de Ingeniería de Sistemas.

Entre los temas que se abordaran encontramos: la vista general del proyecto en donde se definirán alcances y objetivos. Posteriormente se analizará el contexto del proyecto, en esta parte se definirán lenguajes, herramientas y el modelo del ciclo de vida a seguir. Más adelante se mostrarán los planes para la administración del proyecto en donde encontramos la descomposición del mismo en actividades. Seguidamente trataremos el tema de monitoreo, control del proyecto y para finalizar se determinarán los procesos de soporte necesarios, para esto se crearan planes de control de riesgo y de control de calidad.

Este trabajo se ha realizado con el propósito de aplicar la Ingeniería de Software al momento de crear un producto, en nuestro caso se quiere diseñar una aplicación llamada RAWR cuyo objetivo es facilitar la interacción social entre las mascotas y sus dueños. Por medio de RAWR las personas podrán crear cuentas de usuario, inscribir sus mascotas, y las entidades comerciales podrán ofrecer sus productos y servicios.

El documento está dirigido a Miguel Torres y Jaime Pavlich, quienes figuran como asesores en el organigrama de RAWR. Además es importante que todos los integrantes tengan acceso al documento y completo conocimiento de lo que se encuentra en cada una de las secciones.

1. Tabla de contenidos

Contenido

[1. Historial de cambios 2](#_Toc413829862)

[2. Prefacio 3](#_Toc413829863)

[3. Tabla de contenidos 4](#_Toc413829864)

[4. Lista de figuras 6](#_Toc413829865)

[5. Lista de Tablas 7](#_Toc413829866)

[6. Vista general del proyecto 8](#_Toc413829867)

[6.1 Visión del producto 9](#_Toc413829868)

[6.1.1 Perspectiva 9](#_Toc413829869)

[6.2.1 Propósito 9](#_Toc413829870)

[6.2.2 Alcance 9](#_Toc413829871)

[6.2.3 Objetivos 10](#_Toc413829872)

[6.3 Supuestos y restricciones 10](#_Toc413829873)

[6.3.1 Supuestos 10](#_Toc413829874)

[6.3.2 Restricciones 10](#_Toc413829875)

[6.4 Entregables 11](#_Toc413829876)

[6.5 Resumen Calendarización y Presupuesto 14](#_Toc413829877)

[6.5.1 Objetivos 14](#_Toc413829878)

[6.5.2 Responsables 14](#_Toc413829879)

[6.5.3 Recursos Requeridos: 14](#_Toc413829880)

[6.5.4 Resumen 14](#_Toc413829881)

[6.6 Evolución del plan 16](#_Toc413829882)

[6.5.1 Objetivos 16](#_Toc413829883)

[6.5.2 Responsables 16](#_Toc413829884)

[6.5.3 Recursos Requeridos: 16](#_Toc413829885)

[6.5.4 Puesto en Marcha 16](#_Toc413829886)

[8. Contexto del Proyecto 17](#_Toc413829887)

[8.1 Ciclo de Vida 17](#_Toc413829888)

[RUP 17](#_Toc413829889)

[SCRUM 19](#_Toc413829890)

[Extreme Programming 20](#_Toc413829891)

[8.1.1 Análisis de Alternativas y Justificación 20](#_Toc413829892)

[8.2 Lenguajes y Herramientas 23](#_Toc413829893)

[8.2.1 Metodología de Desarrollo 23](#_Toc413829894)

[8.2.2 Lenguaje de Programación 23](#_Toc413829895)

[8.2.3 Herramientas 24](#_Toc413829896)

[8.3 Plan de Aceptación del Producto 27](#_Toc413829897)

[8.3.1 Objetivos 27](#_Toc413829898)

[8.3.2 Responsables 27](#_Toc413829899)

[8.3.3 Recursos Requeridos 27](#_Toc413829900)

[8.3.4 Puesta en marcha 28](#_Toc413829901)

[8.3.3 Riesgos 29](#_Toc413829902)

[8.4 Organización del Proyecto 29](#_Toc413829903)

[8.4.1 Interfaces Externas 29](#_Toc413829904)

[8.4.2 Organigrama y Descripción de Roles 31](#_Toc413829905)

[9. Administración del Proyecto 33](#_Toc413829906)

[9.1 Métodos y Herramientas de Estimación 33](#_Toc413829907)

[9.2 Inicio del Proyecto 33](#_Toc413829908)

[9.2.2 Infraestructura 33](#_Toc413829909)

[9.3 Planes de Trabajo del Proyecto 34](#_Toc413829910)

[9.3.1 Descomposición de Actividades 34](#_Toc413829911)

[9.3.2 Calendarización 35](#_Toc413829912)

[*9.3.2.2 Responsables:* 35](#_Toc413829913)

[9.3.3 Asignación de recursos 36](#_Toc413829914)

[9.3.4 Asignación de presupuesto y justificación 36](#_Toc413829915)

[10 Monitoreo y Control del Proyecto 37](#_Toc413829916)

[10.2 Monitoreo y control de progreso 37](#_Toc413829917)

[10.2.1 Objetivos 37](#_Toc413829918)

[10.2.2 Responsables 37](#_Toc413829919)

[10.2.3 Recursos 37](#_Toc413829920)

[10.2.3 Puesta en marcha 37](#_Toc413829921)

[10.2.3.3 Reporte de progreso individual 38](#_Toc413829922)

[10.3 Cierre del Proyecto 38](#_Toc413829923)

[11 Entrega del Producto 39](#_Toc413829924)

[12.1 Ambiente de Trabajo 39](#_Toc413829925)

[12.1.1 Reglamento 39](#_Toc413829926)

[12.2.2 Penalizaciones 40](#_Toc413829927)

[12.2.3 Multas y sanciones 41](#_Toc413829928)

[12.2 Análisis y administración de riesgos 41](#_Toc413829929)

[12.2.1 Objetivos 41](#_Toc413829930)

[12.2.2 Responsables 41](#_Toc413829931)

[12.2.3 Recursos 41](#_Toc413829932)

[12.2.4 Puesta en Marcha 41](#_Toc413829933)

[12.3 Plan de administración de configuración y documentación. 44](#_Toc413829934)

[12.3.1 Objetivos 44](#_Toc413829935)

[12.3.2 Responsables 44](#_Toc413829936)

[12.3.3 Recursos requeridos 44](#_Toc413829937)

[12.3.4 Puesta en marcha 44](#_Toc413829938)

[12.3.5 Supervisión y control 45](#_Toc413829939)

[12.4 Métricas y Proceso de Medición 46](#_Toc413829940)

[12.5 Control de Calidad 46](#_Toc413829941)

[12.5.1 Plan de control de la calidad del proyecto 46](#_Toc413829942)

1. Lista de figuras

[Ilustración 1: Resumen completo de las actividades más relevantes del Proyecto 15](#_Toc413746029)

[Ilustración 2: Modelo de ciclo de vida RUP 18](#_Toc413746030)

[Ilustración 3: Actividades a realizar en una iteración 19](#_Toc413746031)

[Ilustración 4: Modelo en cascada 21](#_Toc413746032)

[Ilustración 5: Ingeniería De Software basada en componentes 21](#_Toc413746033)

[Ilustración 6: Proceso De Revisión De Documentos 41](#_Toc413746034)

[Ilustración 7: Proceso De Revisión De Código 42](#_Toc413746035)

[Ilustración 8: Proceso De Revisión De La Evolución Del proyecto 43](#_Toc413746036)

[Ilustración 9: Proceso De Revisión De Requerimientos 44](#_Toc413746037)

[Ilustración 10: proceso de revisión de la aplicación 45](#_Toc413746038)

1. Lista de Tablas

[Tabla 1: Entregable a asesores 12](#_Toc413912138)

[Tabla 2: Entregables manejados por RAWR 14](#_Toc413912139)

[Tabla 3: Resumen de las actividades más relevantes del Proyecto con sus costos en el presupuesto 16](#_Toc413912140)

[Tabla 4. Criterios de aceptación para prototipos. 29](#_Toc413912141)

[Tabla 5 :Entregables y prototipos 29](#_Toc413912142)

[Tabla 6: Personas externas y su relación con el proyecto RAWR. 30](#_Toc413912143)

[Tabla 7: Entidades externas y su relación con el proyecto RAWR. 30](#_Toc413912144)

[Tabla 8: Maquinas personales de los integrantes del grupo 35](#_Toc413912145)

[Tabla 9: Matriz de Tolerancia de riesgos [42]. 44](#_Toc413912146)

[Tabla 10: Información De Los Procesos 55](#_Toc413912147)

1. Vista general del proyecto

# 6.1 Visión del producto

## 6.1.1 Perspectiva

Cada día son más las personas que consideran tener un animal acompañante en su hogar. RAWR surge de un proyecto universitario para la asignatura “Ingeniería de software” dictada por el ingeniero Jaime Pavlich PhD. y el ingeniero Miguel Torres MSc.

RAWR, es una aplicación enfocada a la interacción social entre las mascotas y sus dueños. En ella es posible interactuar con los diferentes usuarios por medio de publicaciones, navegación en mapas, comercio de productos, o promoción de servicios [1].

## 6.2.1 Propósito

Durante el proyecto se espera que los integrantes del grupo de trabajo apliquen las enseñanzas adquiridas en la asignatura en una aproximación a lo que sería realmente un proceso de desarrollo de software al realizar un producto (la red social de mascotas RAWR en este caso) utilizando una estructura de desarrollo de software adecuada. Realizar la aplicación por medio de un proceso de software espera demostrar la importancia de aplicar los estándares diseñados por las diferentes entidades dedicadas a ello (IEEE, ACM, ente otros.) en pro de desarrollar software de manera organizada, entregando un producto de calidad [2].

Al final del proyecto RAWR como aplicación para dispositivos móviles tiene como objetivo facilitar la interacción social entre los dueños de mascotas y sus mascotas.

## 6.2.2 Alcance

La red social para mascotas que RAWR implementará cuenta con las siguientes características [1]:

* Espacio para crear una cuenta como [usuario dueño](file:///C:\Users\Fabián%20Merchán\Google%20Drive\Proyecto%20Ingesoft\Primera%20entrega\Glosario%20del%20documento.docx).
* Un usuario dueño podrá inscribir cuantas mascotas desee.
* Posibilidad de añadir o modificar sus datos personales así como cambiar su foto de perfil o la de sus mascotas.
* Posibilidad de crear un [usuario empresa](file:///C:\Users\Luis-PC\Glosario%20del%20documento.docx#usuarioEmpresa) con características y restricciones especiales
* Los usuarios se mostrarán ante los otros usuarios a través de la mascota seleccionada.
* Posibilidad de comunicación entre usuarios por medio de mensajes.
* Un usuario podrá hacer y recibir ofertas de diferentes usuarios propietarios de mascotas como de usuarios propietarios de negocios por medio de una sección de clasificados.
* Posibilidad de ubicación de usuarios o empresas amigas.
* El usuario podrá reportar la pérdida de su mascota e informar si tiene información de alguna mascota desparecida.
* El usuario podrá enviar solicitudes de amistad a otras mascotas.
* Cada mascota tendrá sus propios amigos.
* El usuario puede afiliarse a las publicaciones de un usuario empresa.
* Un usuario puede calificar la satisfacción y calidad de los productos o servicios de un usuario empresa.
* Un usuario dueño o comerciante podrá crear publicaciones de libre expresión, pueden incluir imágenes.
* Un usuario dueño o empresa puede comentar publicaciones de otros usuarios.

Puede encontrar más información en los casos de uso o consultar funcionalidades más detalladas en el SRS [1].

## 6.2.3 Objetivos

### 6.1.4.1 General

* Elaborar una red social para mascotas ejecutando un plan de desarrollo de software diseñado con base en estándares y metodologías adecuadas, además de los requerimientos y cronogramas establecidos por los clientes [1].

### 6.1.4.2 Específicos

* Realizar un plan de desarrollo de software usando las diferentes normas puestas a disposición por la IEEE u otra entidad dedicada a estandarizar planes de desarrollo de software, de modo que se entregue un producto que además de cumplir con las exigencias del cliente sea de calidad.
* Ejecutar el plan de desarrollo de software aplicando todos los procesos, planes y buenas prácticas definidas.
* Fortalecer los lazos comerciales y sociales entre proveedores de productos y servicios de mascotas y los dueños de mascotas por medio de una red social.

# 6.3 Supuestos y restricciones

## 6.3.1 Supuestos

La aplicación se desarrolla bajo las siguientes suposiciones:

* Cada uno de los usuarios de la aplicación móvil tienen al menos una mascota o cuentan con alguna empresa/negocio cuyo objeto social está relacionado con mascotas.

## 6.3.2 Restricciones

* La aplicación será construida para dispositivos móviles LG Nexus 4 con sistema operativo Android, cuya versión mínima debe ser 2.3 (Gingerbread).
* Se debe cumplir a cabalidad con las exigencias de los clientes. Esto incluye requerimientos, exigencias y fechas establecidas de entrega.
* No se cuenta con inversión de capital en el proyecto y la compañía no registra ingresos. Por lo tanto, los egresos deben ser mínimos. Sin embargo, en caso de ser necesario el registro de algún gasto su costo debe ser repartido entre todos los integrantes del equipo de trabajo.
* La aplicación móvil solo se realizará hasta la línea base, el alcance del curso no abarca plan de mantenimiento.

# 6.4 Entregables

Los entregables que recibirán los asesores son:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre del entregable | Cantidad | Descripción | Fecha | Dirigido a |
| SPMP | No aplica | Plan de gestión de proyectos de software, que se realiza con el propósito de que un proyecto pueda concluir con éxito [3]. Esta entrega comprende:   * SPMP. * Plan de estimación. * Casos de uso refinados (al menos un 40%). * Presentación. * Reporte gerencial. | Jueves 12 de Marzo de 2015 | Asesores  Y RAWR |
| SRS | No aplica | Especificación de los procesos necesarios que deben ser implementados para la ingeniería de requerimientos de un sistema o producto [4]. Esta entrega comprende:   * SPMP corregido. * SRS. * Prototipo. * Implementación al menos del caso de uso más difícil. * Reporte gerencial. * Presentación. | Jueves 23 de Abril de 2015 | Asesores  Y RAWR |
| SDD | No aplica | Documento de diseño de software, especifica la manera en que está organizada la arquitectura del sistema [5]. Esta entrega comprende:   * SDD. * SPMP corregido. * SRS corregido. * Prototipo. * Implementación con al menos el 70% de la aplicación funcionando. * Métricas resultantes de las pruebas. * Manuales. * Reporte gerencial. * Presentación. | Viernes 29 de Mayo de 2015. | Asesores  Y RAWR |

Tabla 1: Entregable a asesores

**Nota:** Esta tabla ha sido elaborada por el grupo RAWR con base en el calendario fijado.

Entre los entregables para los clientes encontramos:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del entregable** | **Cantidad** | **Descripción** | **Fecha** | **Dirigido a** |
| Aplicación | No aplica | Se entregaran versiones beta de la aplicación, con el fin de adquirir retroalimentación de los clientes. | Jueves 23 de Abril de 2015 Y viernes 29 de Mayo de 2015. | Clientes |
| Manuales | 1 | Documento que contiene la descripción del manual de usuario. | Jueves 23 de Abril de 2015 Y viernes 29 de Mayo de 2015. | Clientes |

Entre los entregables manejados al interior del grupo podemos encontrar:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Nombre del entregable** | **Cantidad** | **Descripción** | **Fecha** | **Dirigido a** |
| Políticas del grupo | No aplica | Documento que especifica las reglas que los integrantes del grupo deben seguir. Comprende:   * Compromisos. * Reglas. * Sanciones. | Martes 10 de Febrero de 2015. | RAWR |
| Nombre y logotipo de la aplicación. | No aplica | Definición de la imagen del grupo y el nombre. Este entregable contiene:   * Imagen del grupo. * Nombre del grupo. | Martes 10 de Febrero de 2015. | RAWR |
| Evaluación de riesgos | No aplica | Documento mediante el cual se hace una exploración de riesgos de manera general. Contiene:   * Lista de riesgos | Martes 17 de Febrero de 2015. | RAWR |
| Estudio de mercado | No aplica | Documento que tiene los resultados de las entrevistas y encuestas realizadas a veterinarias y propietarios de mascotas. Incluye:   * Resultado de encuestas realizadas a dueños de mascotas. * Entrevistas a clínica raza, veterinaria santa Isabel, veterinaria Luis Alfonso espejo y Clinipet. | Miércoles 18 de Febrero de 2015. | RAWR |
| Definición del Ciclo de vida | No aplica | Documento en donde se especifica el modelo de ciclo de vida a utilizar durante el desarrollo del proyecto. Contiene:   * Definición del modelo. * Justificación. | Lunes 23 de Febrero 23 De 2015 | RAWR |
| Calendario | No aplica | Definición del calendario y de cada una de las actividades que se van a desarrollar en el proyecto. Este entregable contiene:   * Carta de Gantt. * WBS. | Febrero 23 De 2015 | RAWR |
| Organigrama | No aplica | Definición del organigrama del grupo. Contiene:   * División del equipo en Scrums. * Imagen de relación entre los componentes. | Jueves 3 de Marzo de 2015. | RAWR |
| Referencias | No aplica | Documento que contiene las referencias que se han utilizado en el proyecto son manejados mediante:   * Microsoft Word. * Zotero. | Jueves 12 de Marzo de 2015 | RAWR |
| Lista de requerimientos | No aplica | Documento que contiene la lista de requerimientos. Elaborado a partir de:   * Estudio de mercado * Entrevistas y encuestas. * Restricciones fijadas por la clase. | Martes 17 de Marzo de 2015. | RAWR |
| Diseño y desarrollo del prototipo | No aplica | Documento que especifica el diseño del prototipo. Contiene:   * Diseño del prototipo. * Casos de prueba. | Viernes 17 de abril de 2015 | RAWR |
| Manuales | 2 | Documento que contiene los manuales de usuario y administrador. | Miércoles 27 de Mayo de 2015. | RAWR |

Tabla 2: Entregables manejados por RAWR

**Nota:** Esta tabla ha sido elaborada por el grupo RAWR con base en el calendario fijado.

Puede encontrar más información acerca de los entregables en la sección de calendarización **citar sección.**

# 6.5 Resumen Calendarización y Presupuesto

## 6.5.1 Objetivos

* Describir procesos y actividades que se van a realizar en determinadas fechas.
* Describir procesos y actividades para garantizar la realización de las diferentes fases del proyecto en las entregas correspondientes.
* Describir y conocer el costo total del proyecto y su correspondiente presupuesto.

## 6.5.2 Responsables

* SoS Master
* El cumplimiento de las fechas, procesos, actividades y tareas es responsabilidad del todos los miembros de RAWR.

## 6.5.3 Recursos Requeridos:

* Microsoft Excel.

## 6.5.4 Resumen

El proyecto está compuesto por tres fases principales que contienen las actividades a realizar en cada una de ellas. La ilustración de resumen de actividades muestra las actividades enmarcadas en los plazos de entrega.

Ilustración : Resumen completo de las actividades más relevantes del Proyecto

**Nota:** Esta Ilustración fue realizada por el grupo RAWR

|  |  |  |
| --- | --- | --- |
| **Primera Entrega** | | |
| **Actividades** | **Fechas** | **Costos $** |
| Definición Reglas de Trabajo | 10/02/15 | 156.000 |
| Definir nombre aplicación y logo | 10/02/15 | 156.000 |
| Definir Repositorio | 12/02/15 | 26.000 |
| Análisis del proyecto | 17/02/15-24/02/15 | 2’080.000 |
| Fase Concepción SPMP | 26/02/15-12/03/15 | 3’432.000 |
| Fase Concepción Documento de casos de uso | 26/02/15-12/03/15 | 858.000 |
| Analizar Riesgos | 05/03/15-10/03/15 | 624.000 |
| Correcciones pre-entrega | 11/03/15-12/03/15 | 702.000 |
| Elaboración presentación | 11/03/15-12/03/15 | 52.000 |
| **Segunda Entrega** | | |
| **Actividades** | **Fechas** | **Costos $** |
| Revisión entregables anteriores | 14/03/15-22/03/15 | 1’560.000 |
| Análisis e investigación de requerimientos | 15/03/15-17/03/15 | 234.000 |
| Planeación del proyecto | 14/03/15-19/03/15 | 286.000 |
| Levantamiento de requerimientos | 07/03/15-07/03/15 | 78.000 |
| Fase de elaboración SRS | 19/03/15-22/04/15 | 8,970 |
| Diseño y justificación del prototipo | 08/04/15-17/04/15 | 702.000 |
| Desarrollo del prototipo | 13/04/15-17/04/15 | 390.000 |
| Correcciones pre-entrega | 22/04/15 | 338.000 |
| Elaboración presentación | 22/04/15-23/04/15 | 39.000 |
| **Tercera Entrega** | | |
| **Actividades** | **Fechas** | **Costos $** |
| Revisiones entregables anteriores. | 24/04/15-04/05/15 | 1’794.000 |
| Análisis e investigación de requerimientos | 24/04/15-04/04/15 | 546.000 |
| Fase de construcción | 11/05/15-27/05/15 | 1’560.000 |
| Plan de pruebas | 14/05/15-27/05/15 | 780.000 |
| Plan de manuales | 14/05/15-27/05/15 | 780.000 |
| Métricas | 14/05/15-27/05/15 | 260.000 |
| Correcciones pre-entrega | 28/05/15 | 156.000 |
| Elaboración presentación | 28/05/15 | 26.000 |

Tabla 3: Resumen de las actividades más relevantes del Proyecto con sus costos en el presupuesto

**Nota:** Esta Ilustración fue realizada por el grupo RAWR

El presupuesto asignado para el proyecto tomando en cuenta un salario por persona de $13,000 COP/hora y un estimado de 2451 horas de elaboración da un costo total de $31’863.000 COP. Para ver una descripción detallada de costos por actividad [**ver Anexo Estimación Costo Proyecto**](file:///C:\Users\Estudiante\Downloads\Anexos\Estimación%20costo%20proyecto.xlsx).

# 6.6 Evolución del plan

## 6.6.1 Objetivos

* Establecer métodos por los cuales se garantice la actualización del presente documento (SPMP).
* Describir procesos y actividades para garantizar la realización de los diferentes cambios propuestos.

## 6.6.2 Responsables

* SoS Master
* Scrum Master de Calidad y administración de configuración [6].
* Las correcciones y cambios del SPMP son responsabilidad de todos.

## 6.6.3 Recursos Requeridos:

* Microsoft Word.

## 6.6.4 Puesto en Marcha

Para la realización de un cambio al SPMP es requerida la aprobación por parte del integrante asignado a la revisión cruzada de la semana. Adicionalmente, el control de cambios se maneja a través del [formato de control de cambios](file:///C:\Users\Estudiante\Downloads\Anexos\Formato%20de%20control%20de%20cambios.xlsx), el cual debe estar disponible al inicio de cada documento y contener la información de quién lo realizó, quién lo corrigió, número de errores, sugerencias, tiempo de elaboración y corrección. Para más información referirse al [plan de administración de la Configuración y Documentación](#_12.3_Plan_de)**.**

8. Contexto del Proyecto

# 8.1 Ciclo de Vida

RAWR considera apropiado para el proyecto utilizar un ciclo de vida basado en los modelos iterativos incrementales tomando RUP como base y complementándolo con buenas prácticas de SCRUM y XP. Se usará UML como herramienta de modelado con el fin de formular ideas complejas de forma resumida y precisa. De esta manera se intenta que todos los integrantes del grupo entiendan los modelos funcionales y de objetos del sistema [7].

## 8.1.1 RUP

RUP es un modelo de ciclo de vida iterativo incremental que se presenta como un enfoque disciplinado para asignar tareas y responsabilidades dentro de una organización con el fin de lograr la producción de software de alta calidad. Una de las desventajas de RUP es que para equipos pequeños la documentación que se necesita hacer puede abrumar al equipo de trabajo y no tener suficiente tiempo para trabar en el desarrollo de software [8], además de tener una lista larga de roles con actividades previamente definidas. Para evitar estos problemas en RAWR es de vital importancia adaptar el modelo de ciclo de vida con el fin de suplir nuestras necesidades utilizando buenas prácticas como la simplificación de roles de SCRUM [9].

Las mejores prácticas de RUP son el desarrollo de software iterativo, manejo de requerimientos, usar una arquitectura basada en componentes, modelar visualmente el software y verificar su calidad [10].

El proceso de RUP se puede describir en dos dimensiones por medio de la Ilustración 2.

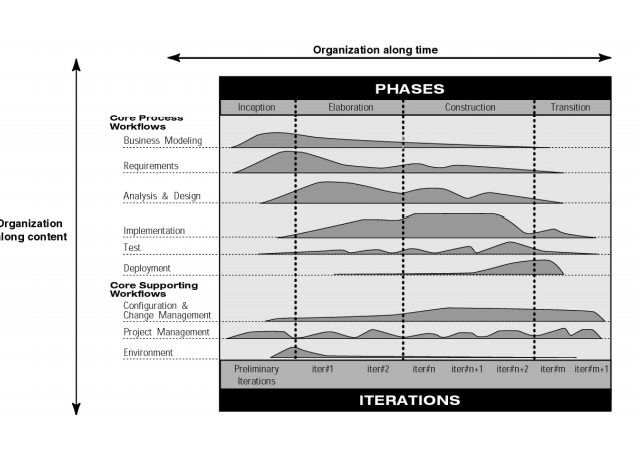


Ilustración : Modelo de ciclo de vida RUP

**Nota:** Esta ilustración ha sido extraída de [5].

El ciclo de vida del proyecto se divide en fases que se pueden llevar a cabo en una o más iteraciones. Cada iteración es un ciclo completo de desarrollo que resulta en la terminación de un producto ejecutable o de un subconjunto del producto final. Adicionalmente, cada fase tiene un proceso de retroalimentación en su cierre. Las fases con su conjunto de tareas traducidas al proyecto a realizar son:

* Fase de concepción
  + SPMP
* Fase de elaboración
  + SRS
    - Entendimiento de los requerimientos.
    - Prototipo de Arquitectura.
  + Inicio del Desarrollo, prototipo Inicial
  + SDD
* Fase de construcción
  + Prototipo funcional con al menos el 70% de los requerimientos implementados junto con su descripción.
  + Manuales de usuario [11].

Para el caso particular del proyecto no hay una fase de transición definida dentro de los objetivos. Sin embargo, al finalizar el proyecto los integrantes del grupo evaluarán si quieren pasar el producto a producción. Para el proyecto RAWR, cada una de las fases definirá un hito para el proyecto.

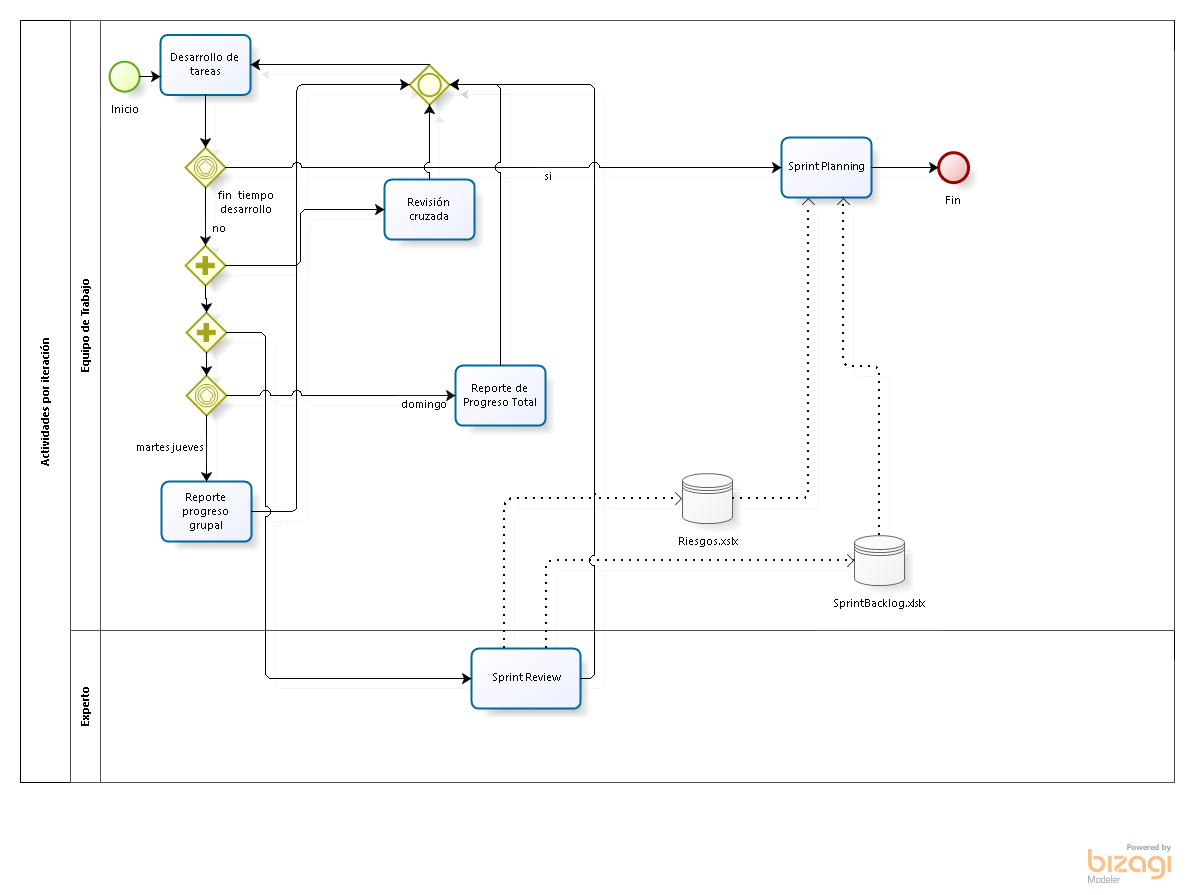


Ilustración 3: Actividades a realizar en una iteración.

**Nota**: El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

## 8.1.2 SCRUM

SCRUM es un marco de trabajo ágil y un proceso de desarrollo iterativo incremental [9] basado en la experiencia y en la adaptación, el cual permite que en cada iteración se pueda mejorar la forma en la que se resuelve el problema. Debido a que todo el modelo se basa en enfatizar la colaboración se debe tener buena comunicación y apropiada documentación. Cabe recalcar que se recomienda que el número de personas sea pequeño (entre cinco y nueve participantes) para poder seguir el esquema adecuadamente, lo cual encaja bien en nuestro de grupo de 6 personas.

Buenas Prácticas con SCRUM a realizar durante el proyecto:

* Sprint review.
* Sprint backlog.
* Daily Scrum, llamado [reporte de progreso grupal por área.](#_10.2.3.1_Reporte_de)
* Scrum of Scrum [13].
* SoS meeting, llamado [reporte de progreso grupal total](#_10.2.3.2_Reporte_de).
* Simplificación de Roles

## 8.1.3 Extreme Programming

Es un método utilizado cuando se desarrolla software que presente requerimientos cambiantes o vagos [14]. Cuenta con dos principios importantes:

1. Pruebas automáticas.
2. Programación en parejas.

Buenas prácticas a realizar durante el proyecto:

* Pruebas: Se debe diseñar una prueba y realizar el diseño del software con solo lo suficiente para pasarla [15].
* Programación de parejas: Todo el código es construido por dos programadores en un lugar de trabajo con un sólo computador, se presenta un continuo diálogo entre personas [15].
* Refactoring: Mejorar el código sin que se afecte el comportamiento del sistema, sirve para eliminar código redundante,  mejorar tiempos de respuesta de la aplicación y eliminar errores [15].

## 8.1.4 Análisis de Alternativas y Justificación

Las alternativas de modelos que se tuvieron en cuenta fueron:

* Cascada
* Modelo en v
* Modelo basado en componentes.

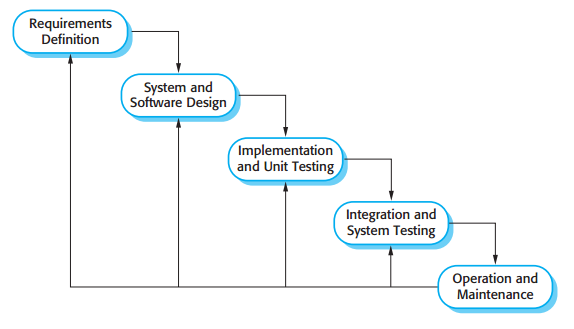


Ilustración : Modelo en cascada

**Nota:** Esta ilustración ha sido extraída del libro S*oftware Engineering* de Ian Sommerville [16]

El modelo en cascada a pesar de que sigue siendo ampliamente utilizado tiene muchas limitaciones; entre ellas están la falta de gestión de riegos temprana, poca reutilización de módulos y una baja orientación a las actividades [9]. Esto es causado porque las actividades son secuenciales. El objetivo de cascada es nunca volver a una actividad una vez que se termina; lo que causa que no se puedan introducir requerimientos en medio del proyecto [7]. Para RAWR estas limitaciones generan un gran problema ya que no se tienen los requerimientos claros desde el inicio del proyecto y la retroalimentación constante de los clientes no será aprovechada correctamente [7].

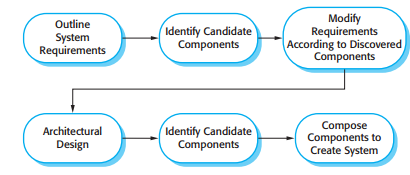


Ilustración 5: Ingeniería De Software basada en componentes.

**Nota**: Esta ilustración ha sido extraída del libro Software Engineering de Ian Sommerville [16]

Por otro lado, el modelo basado en componentes no se puede usar debido a que no se han hecho proyectos propios para reutilizar algún componente.

Finalmente, el modelo en V resuelve algunos de los problemas que tiene el modelo en cascada ya que se reconoce la relación entre cada una de las actividades de desarrollo y la verificación [7]. Sin embargo, el problema con este modelo es que sobre simplifica la verificación; es decir se debe esperar hasta el final de una fase para poder validarla [17]. Por lo tanto, en el caso de que se esperase y la retroalimentación de los clientes hiciese que fuera necesario cambiar muchos aspectos del sistema generaría que se incremente el esfuerzo y los recursos necesarios; esto debido a que es posible que se necesite hacer modificaciones y agregar componentes a la implementación que se había realizado previamente.

Después de analizar estos modelos se estudiaron los modelos de ciclo de vida iterativos incremental. Estos modelos ofrecen muchas ventajas que no tenían los mencionados previamente, tales como la gestión de riesgos en etapas tempranas [8] aprovechando las reuniones semanales con nuestros clientes y asesores quienes nos brindan retroalimentación constante que nos permiten poder priorizar riesgos y actividades. Además, los modelos iterativos ayudan a tener una administración de proyecto más simplificada ya que se asignan unidades de trabajo más pequeñas que pueden ser mejor controladas [8]. Otra ventaja que tienen los modelos de ciclo de vida iterativos es que la calidad general del software es mayor debido a las revisiones que hay en cada una de las iteraciones. Por último, estos modelos permiten agregar nuevos requerimientos a medida que se avanza en el proyecto.

Teniendo esto en cuenta, se escogió un modelo iterativo incremental que se complementara con buenas prácticas de XP y Scrum. Como base del modelo de ciclo de vida se utilizará el RUP, el cual encaja muy bien con las tres entregas del proyecto y los planes que se van a ejecutar durante él. Se va a hacer modelado, análisis, diseño e implementación además de pruebas en cada una de las iteraciones a realizar; teniendo siempre unos flujos de trabajo de soporte que son: el la administración de proyecto, gestión de la configuración, gestión de riesgos y calidad [11]. Finalmente, se utilizará UML como lenguaje de modelado con el fin de ver más fácilmente el comportamiento de la arquitectura y los componentes del sistema [11]. Las buenas prácticas de [XP](#_Extreme_Programming) y [Scrum](#_SCRUM) aplicadas en el proyecto se describieron anteriormente de manera general.

Scrum como metodología se basa en la experiencia y adaptación. De este ciclo de vida se rescatarán buenas prácticas que ayudarán al equipo a realizar un mejor trabajo como las reuniones diarias que fomentan la comunicación entre todos los miembros del equipo y permiten conocer el avance de cada tarea asignada con el objetivo de encontrar retrasos y priorizar la realización de actividades [18]; sin embargo, por motivos de disponibilidad horaria de los estudiantes se tienen que restringir a dos reuniones semanales que tienen una duración máxima de 1 hora cada una. Adicionalmente, se utilizarán listas para facilitar la identificación de actividades a realizar en el futuro sin tener que ir al detalle de los diagramas de actividad [9].

Por otro lado, ya que la metodología Scrum no reconoce sub-equipos dentro de los equipos de trabajo [19], se decide realizar un Scrum of Scrums el cual consiste básicamente en tener varios equipos Scrum [20] trabajando en un mismo proyecto. Con el fin de evitar redundancia de trabajo y problemas con dependencias de tareas se adopta el SoS meeting explicado en [13]. Una descripción más detallada de la estructura de los equipos Scrum puede verse en el [organigrama](#_8.4.2.1_Organigrama).

Algunas de las buenas prácticas que usaremos de XP son: programación en parejas, la cual permite que haya una revisión constante del trabajo realizado por cada integrante. En sentido general, integrantes experimentados pueden emparejarse con personas con relativamente poca experiencia para ayudarlos en el proceso de formación, transmitiéndoles el conocimiento [21], en el caso particular de RAWR no se hará para transmitir experiencia sino para compartir las pocas experiencias de cada uno para resolver problemas más fácilmente. Sin embargo, por motivos de disponibilidad horaria no se puede aplicar la programación en parejas, por lo que se hará un trabajo en parejas con una constante retroalimentación entre cada uno de los participantes para lograr una revisión cruzada. La segunda buena práctica de XP que se utilizará son las pruebas automáticas. Se realizaran con el propósito de evitar que se propaguen errores en la rama principal del proyecto [21].

Cabe recalcar que se están uniendo la buena administración del proyecto de SCRUM con las buenas prácticas de desarrollo que ofrece XP y el modelado visual con UML para facilitar el entendimiento del software; de esta manera se compensan las debilidades de cada uno de estos modelos de ciclo de vida.

# 8.2 Lenguajes y Herramientas

## 8.2.1 Metodología de Desarrollo

El proyecto RAWR será una aplicación para dispositivos móviles con sistema operativo Android. Por esta razón, se utilizará el framework de este Sistema Operativo que permite programar en JAVA por medio del Android SDK (Software Development Kit); el cual compila el código en un paquete Android o APK que instala la aplicación en dispositivo móvil [22].

Este sistema operativo fue escogido por las siguientes razones:

* Las aplicaciones son varios componentes los cuales son independientes.
* Permite ser programado para una gran variedad de dispositivos móviles.
* El framework divide los componentes en tipos diferentes, cada uno con una funcionalidad especifica lo que permite organización y consistencia en el código.
* Se separa la presentación de la lógica de negoció utilizando archivos XML para interfaces de usuario y recursos y clases Java que extienden componentes para manejar la lógica [22].

## 8.2.2 Lenguaje de Programación

El lenguaje que se utilizará para desarrollar el proyecto RAWR es JAVA, el cual es un lenguaje orientado a objetos que por medio del Android SDK cumple con las necesidades del proyecto pues permite desarrollar una aplicación para dispositivos móviles que tengan sistema operativo Android [22]. Adicionalmente, se utilizará PHP para hacer por medio de scripts las consultas de la base de datos.

## 8.2.3 Herramientas

Las herramientas[[1]](#footnote-1) que se usarán durante el transcurso del proyecto para permitir un desarrollo que cumpla con los objetivos planeados están listadas a continuación con una breve descripción y organizadas por su funcionalidad:

### 8.2.3.1 Comunicación

Las Herramientas que se usarán para la comunicación interna del grupo de trabajo del proyecto son:

* Slack:
  + Aplicación web diseñada para la comunicación de equipos de una forma organizada y facilidades para búsqueda eficiente y en contexto.
  + Página web: [Slack](https://slack.com/)
* Google Hangouts:
  + Aplicación web para comunicación por medio de video-llamadas que permite realizar reuniones del grupo de forma virtual.
  + [plug-in Hangouts](https://www.google.com/tools/dlpage/hangoutplugin)

### 8.2.3.2 Organización

Las Herramientas que se utilizarán para intercambiar y manejar la planeación del desarrollo del proyecto son:

* Google Drive:
  + Aplicación para el manejo de documentos de forma simultánea.
  + [Página para descargar Google Drive para escritorio](https://tools.google.com/dlpage/drive/index.html?hl=en)
* Microsoft Project:
  + Herramienta para el manejo de recursos y tareas en el grupo de trabajo.
* Microsoft Word:
  + Creación y manejo de documentos a lo largo del proyecto con facilidades para agregar comentarios.
* Microsoft Excel:
  + Herramienta para el manejo de hojas de cálculo para registrar datos en el transcurso del proyecto y analizar para toma de decisiones así como manejo de estimaciones y métricas.
* Zotero:
  + Herramienta para mantener referencias y bibliografía compartida entre el grupo de trabajo y facilitar la unión de documentos en cuanto a sus referencias, cuenta con un plug-in para Microsoft Word.
  + [Descarga y plug-in para Word](https://www.zotero.org/support/word_processor_integration)
* Bizagi:
  + Herramienta para manejo de procesos por medio de diagramas BPMN (Business Process Model and Notation)
  + [Link de Descarga Bizagi](http://download.cnet.com/Bizagi-Modeler/3000-2064_4-10829436.html)

### 8.2.3.2 Diseño

* Adobe Illustrator:
  + Herramienta de diseño visual que trabaja con vectores para manejo de logos e interfaz gráfica [23].
* Adobe Photoshop:
  + Herramienta para manejo y edición de imágenes, trabaja con pixeles (*raster*)*.* También se usará para el manejo de logos e interfaz gráfica del sistema [24] [25].
* Pencil:
  + Herramienta open-source para diseño y diagramación de interfaces gráficas (GUI) para cualquier tipo de plataforma.
  + [Página para descarga de Pencil](http://pencil.evolus.vn/)
* StarUML:
  + Herramienta para el modelado de diseño de la estructura, arquitectura y comportamiento de la aplicación en UML.
  + [Página de descarga de StarUML](http://staruml.io/download)

### 8.2.3.3 Desarrollo

Las siguientes son las herramientas que se utilizarán para desarrollar el código de la aplicación, las pruebas y el control de versionamiento del código:

* Android Studio:
  + IDE diseñado exclusivamente para desarrollo de aplicaciones en Android basado en IntelliJ IDE[[2]](#footnote-2).
  + [Link para descargar Android Studio](http://developer.android.com/sdk/index.html)
* Git:
  + Sistema de versionamiento distribuido utilizado para controlar le versionamiento del código de la aplicación.
  + Permite hacer seguimiento a las versiones del código fuente de una forma ordenada, rápida y distribuida[[3]](#footnote-3) [26] [27].
  + [Link para descargar Git](http://git-scm.com/downloads)
* Github:
  + Servicio de hospedaje de repositorios Git en la web.
  + Este permite gestionar el código fuente del proyecto con herramientas para colaboración, revisión, seguimiento de errores, hitos entre otros.
  + Página web: [github](https://github.com/)
* JUnit:
  + Framework para realizar pruebas unitarias para el código desarrollado en lenguaje JAVA.
  + Está integrado en el ambiente de desarrollo para Android como una extensión que provee componentes específicos para crear clases de prueba y objetos y métodos mock[[4]](#footnote-4) [22] [28].
* DigitalOcean:
  + Servicio de Hosting en la nube basado en discos de estado sólido que provee un servicio de 1Gb/s con un API simple para controlar los servidores.
  + Página Web: [DigitalOcean](https://www.digitalocean.com/)
* PHP:
  + Lenguaje de programación que se usará, por medio de scripts, para hacer la conexión de la base de datos con la aplicación Android.
  + [Página para descargar PHP](http://php.net/downloads.php)
* Sublime Text 3:
  + Editor de texto que se utilizará para realizar los scripts necesarios en el lenguaje PHP
  + [Página de descarga Sublime Text 3](http://www.sublimetext.com/3)
* MySQL:
  + Sistema de manejo de bases de datos relacionales open-source que maneja el lenguaje SQL. Se usará para manejar la base de datos de los usuarios de RAWR.

## 8.2.4 Análisis de Alternativas y Justificación

**Herramientas de organización**: Se contempló inicialmente hacer uso de los documentos de Google los cuales permiten editar en simultáneo en internet y hacer comentarios. Sin embargo, el equipo de trabajo decidió utilizar Microsoft Word para la creación y revisión de documentos pues facilita el uso de hipervínculos entre documentos y el manejo de referencias por medio de zotero.

Antes de comenzar a utilizar la Herramienta Slack el grupo intentó comunicarse por medio de un grupo de Facebook; lo cual no resultó útil para las necesidades que se presentaban pues era difícil la búsqueda de información y la organización de temas.

**Diseño Gráfico**: Además de Illustrator y Photoshop también se contemplaron las opciones de Corel Draw que al igual que Illustrator es un programa de diseño que trabaja sobre vectores; sin embargo, fueron descartadas debido a que 2 integrantes del grupo ya habían trabajado con los dos programas seleccionados y tenían en sus máquinas instalados dichos programas.

**Versionamiento:** Las herramientas contempladas para manejar el progreso y revisión de código en el proyecto fueron TortoiseSVN [29] (Apache Subversion), Mercurial y Monotone. Inicialmente se decidió que se quería trabajar con un sistema de versionamiento que no fuera centralizado [30] pues se consideró que esto tenía unas limitaciones para el trabajo simultaneo pues solo existe un repositorio central de donde los programadores deben bajar únicamente la parte de código que están modificando lo cual también impide trabajar cuando no halla conexión a internet. Es por esta razón que se descartaron todas las opciones de *Subversion.*

Se decidió que el sistema de versionamiento que se utilizaría para manejar las versiones del proyecto RAWR sería distribuido, es decir que permitiera bajar el repositorio localmente y así poder trabajar de forma simultánea y sin necesidad de conexión a internet de forma permanente.

Mercurial y Monotone, al igual que git, son sistemas de versionamiento distribuido que tiene muchas cosas en común y se diferencian por pequeños detalles que muchas veces no son perceptibles para el usuario. Monotone [31] tiene un diseño orientado a la integridad comprometiendo desempeño, esto hace que sea lento en muchos casos ya que hace muchas revisiones para validar la integridad y aunque ya se han hecho algunas optimizaciones para mejorar esto, sigue siendo más lento a comparación de Git y Mercurial. En cuanto a Mercurial [32] se encontró que el manejo de *branches* es más complicado que en Git ya que en Mercurial una rama (*branch*) está embebida dentro de un *commit* mientras que en Git es únicamente un apuntador al commit lo cual hace más fácil su manejo.

Por último se decidió usar Git por 2 razones principales: primero dos integrantes del grupo ya tenían algunos conocimientos previos sobre este sistema y segundo por la facilidad que provee Github [26], herramienta web que tiene varias características para que el flujo de trabajo en equipo sea colaborativo y permita revisiones cruzadas como los pull request y la capacidad de hacer comentarios antes de unir los cambios a la rama principal (*master branch*).

**Desarrollo:** La otra alternativa contemplada para desarrollar la aplicación móvil RAWR para Android era el IDE Eclipse para programar con el lenguaje Java y Android ADT (plugin para Eclipse). Pero aunque todo el grupo está familiarizado con el entorno de desarrollo Eclipse se decidió utilizar Android Studio pues es una herramienta diseñada específicamente para el desarrollo móvil en Android y toma algunos conceptos de diseño utilizados en Eclipse pero mejorando las capacidades que se relacionan con el desarrollo móvil, como por ejemplo el entorno de diseño de interfaz gráfica. Además Android Studio viene con un soporte completo de JUnit para realizar pruebas unitarias [22] [33]. Aunque es posible programar en otro lenguaje diferente a JAVA para estos dispositivos por medio del Android NDK, estos se usan para situaciones muy específicas donde el uso de código nativo como C/C++ es necesario para garantizar un alto rendimiento cuando el uso de CPU es intenso tales como simulaciones, procesamiento de señales o motores de juego.

**Servidor y base de Datos:** Se escogió DigitalOcean [34] como servidor del proyecto RAWR por encima de hacer uso del servido del departamento de Ingeniera de Sistemas de la Pontifica Universidad Javeriana pues, por experiencia, se sabe que este servidor tiene problemas de desempeño cuando muchos usuarios lo acceden al tiempo y DigitalOcean permite tener un servidor gratuito a estudiantes. En cuanto a la base de datos se utilizará MySQL porque es una herramienta open source que provee un servicio gratuito de manejo de bases de datos relacionales, también se consideró la base de datos ofrecida por Oracle, sin embargo, por la necesidad de pedir permisos de uso a la facultad de Ingeniería de Sistemas y por el costo de este producto se decidió descartar.

# 8.3 Plan de Aceptación del Producto

El plan define la forma en que se va a evaluar el producto para que de forma objetiva se acepte o rechace según los requerimientos que el producto debe cumplir [35], [16].

## 8.3.1 Objetivos

* Definir los criterios que se tienen en cuenta en el momento de aceptar o rechazar los prototipos del producto.
* Definir las técnicas y herramientas utilizadas para la revisión y aceptación del producto
* Listar los hitos que debe cumplir el producto para ser considerado aceptado.

## 8.3.2 Responsables

* Scrum Master de Calidad
* Equipo del área de calidad

## 8.3.3 Recursos Requeridos

Para este plan se necesitan las herramientas que se listaron en la sección de lenguajes y herramientas de este documento ([ver sección 8.2 Lenguajes y Herramientas](#_8.2_Lenguajes_y)).

## 8.3.4 Puesta en marcha

El plan de aceptación del producto se divide en 2 partes que son Los criterios de aceptación del software y de los documentos del producto. Este plan está fuertemente ligado al control de calidad (ver sección 12.5 Control de Calidad) y con los casos de uso definidos para el proyecto.

Para el plan de aceptación de los prototipos se relaciona con el proceso de revisión de código ([ver sección 12.5.1.5.2 Proceso de revisión de código](#_12.5.1.5.2_Proceso_de)) y se definieron los siguientes puntos críticos para la aceptación del producto [36], [37] :

* Desempeño: Hace referencia a la velocidad de respuesta que espera el cliente que presenten las diferentes funcionalidades de la aplicación.
* Mantenibilidad: En este ítem se revisa la relación que tiene el código con el diseño planteado así como la documentación, organización y las pruebas unitarias realizadas para los módulos que hacen parte de la aplicación.
* Funcionalidad: Este punto está ligado a los requerimientos y casos de uso definidos en el diseño del problema, así por cada prototipo entregado se debe revisar si cumple con los requerimientos que se implementarían en la iteración dada.

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo | Técnicas | Criterios y Herramientas | Acciones Correctivas |
| Desempeño | Se revisará el tiempo de respuesta de las funcionalidades implementadas en el sistema | Por lo menos el 80% de las funcionalidades deben tener un tiempo de respuesta menor a 3 segundos. El otro 20% de las funcionalidades podrán tener un tiempo de respuesta de hasta 5 segundos. | Se optimizará o buscara una solución diferente para el porcentaje necesario de funcionalidades que no cumplen con los criterios. |
| Mantenibilidad | Se hará una revisión cruzada por parte de los programadores. | Se tendrá en cuenta la presentación y documentación del código antes de ser aceptado en el repositorio | Si el revisor considera que el código no cumple con los criterios de mantenibilidad se hará un comentario para que se corrija antes de unir el código a la línea principal del proyecto (ver sección 12.3 Configuración y Mantenimiento). |
| Funcionalidad | Por cada caso de uso implementado se debe revisar que cumplan la funcionalidad requerida | Este criterio será probado por medio de las pruebas unitarias medidas | Se deberá modificar el código de tal forma que pase las pruebas unitarias. |

Tabla 4. Criterios de aceptación para prototipos.

Para el caso de los documentos entregables se seguirá el proceso de revisión de documentos ([ver sección 12.5.1.5.1 Proceso de revisión de documentos](#_12.5.1.5.1_Proceso_de)), y se definieron los siguientes criterios para la aceptación de los documentos los cuales fueron definidos por los profesores en las rúbricas del proyecto:

* Consistencia
* Completitud
* Investigación
* Forma
* Proceso
* Notación

Estos criterios serán medidos con una calificación cuantitativa de 0 a 5 donde 5 significa excelente por parte de los clientes que en este caso son los profesores de la asignatura.

Entregables y Prototipos:

Dado el ciclo de vida elegido para RAWR, ([ver sección 8.1 Ciclo de Vida](#_8.1_Ciclo_de)) se entregarán prototipos que incrementan la funcionalidad del producto. Estos serán los entregables y prototipos a entregar [35] [36]:

|  |  |  |
| --- | --- | --- |
| Entregable | Prototipo | Fecha |
| SPMP y Casos de Uso |  | 12 de Marzo |
| SRS | Primer Prototipo | 23 de Abril |
| SDD | Segundo Prototipo | 29 de Mayo |

Tabla 5 : Entregables y prototipos

# 8.4 Organización del Proyecto

## 8.4.1 Interfaces Externas

A continuación se presentan las interfaces externas con las que, durante el desarrollo del proyecto RAWR, interactuaremos. Están divididos en dos tablas, una de personas y otra de entidades donde se describen la relación que estas tiene con el proyecto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Descripción | Rol | Comunicación | Datos |
| Jaime Pavlich | Profesor Asociado de planta del departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana. Profesor de la Materia Ingeniería de Software[38]. | Cliente y Asesor: Representa uno de los clientes dando visto bueno de aceptación y correcciones de los documentos y producto entregado en cada iteración. Adicionalmente será en cierto grado asesor durante el desarrollo del producto. | La comunicación se hará realizando reuniones periódicas para revisar el proceso y avances del proyecto y vía email. | **E-mail:** jpavlich@javeriana.edu.co **Tel:** (+57 1) 320 8320 Ext. 4731[38]  **Oficina:** Edificio 67 - 300 |
| Miguel Torres | Profesor Asociado de planta del departamento de Ingeniería de Sistemas de la Pontificia Universidad Javeriana. Profesor de la Materia Ingeniería de Software[38]. | Cliente y Asesor: Representa uno de los clientes dando visto bueno de aceptación y correcciones de los documentos y producto entregado en cada iteración. Adicionalmente será asesor durante el proceso de documentación. | La comunicación se hará realizando reuniones periódicas para revisar el proceso y avances del proyecto y vía email. | **E-mail:** metorres@javeriana.edu.co **Tel:** (+57 1) 320 8320 Ext. 5316[38]  **Oficina:** Edificio 9, 3er piso, Facultad de Ingeniería de Sistemas |

Tabla 6: Personas externas y su relación con el proyecto RAWR.

|  |  |
| --- | --- |
| Nombre Entidad | Descripción y Relación |
| Biblioteca Pontificia Universidad Javeriana | Recurso principal de consulta de referencias tanto de libros físicos como documentos digitales. |
| Salón 2 – 205 Edificio Barón  Pontificia Universidad Javeriana | Lugar donde se realizarán la mayoría de reuniones del grupo para comunicación durante el desarrollo del proyecto RAWR. |

Tabla 7: Entidades externas y su relación con el proyecto RAWR.

## 8.4.2 Organigrama y Descripción de Roles

### 8.4.2.1 Organigrama

RAWR adopta el modelo de Scrum of Scrums explicado en [20] en el cual hay varios equipos Scrum [19] dentro de uno más grande. Los Scrum master y equipos de cada área fueron escogidos de acuerdo a sus aptitudes y preferencias. Las relaciones entre los roles y el flujo de comunicación de la organización se muestra en la siguiente ilustración.

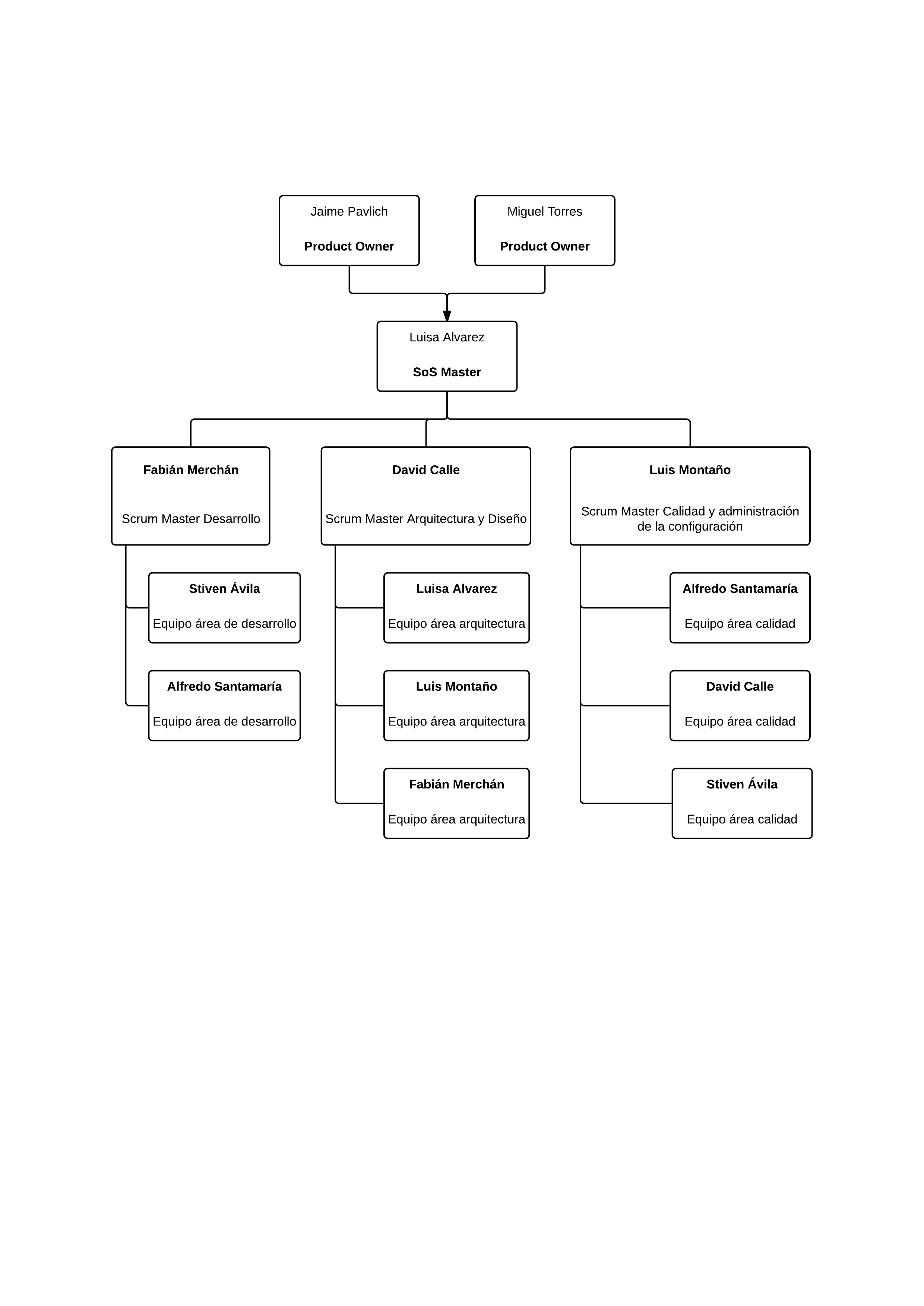


Ilustración 1: Organigrama RAWR.

**Nota:** El anterior organigrama fue realizado por RAWR en la herramienta Lucidchart.

### 8.4.2.2 Descripción de roles

La especificación de las responsabilidades específicas de cada rol junto a sus aptitudes y planes asociados se encuentra en el archivo anexo Descripción Roles ([Ver anexo Descripción Roles](file:///C:\Users\Estudiante\Downloads\Anexos\Descripción%20Roles.xlsx)). Adicionalmente, todos los Scrum Master tienen las siguientes responsabilidades:

* Guiar al equipo en ser autoorganizado y multifuncional [19].
* Ayudar al equipo a crear productos de valor [19].
* Eliminar impedimentos para el progreso del equipo [19].

RAWR añade al SoS Master algunas de las responsabilidades del rol gerente.

### 8.4.2.3 Flujo de información y decisión

El proceso de toma de decisiones se realizará por consenso siguiendo la metodología de [40]. El proceso consta de tres pasos fundamentales:Una propuesta, una discusión grupal y un conjunto de decisiones individuales. Cada decisión tiene un propósito:

* Bloquear.
* Apartarse.
* Consentir.

La decisión de bloqueo no puede ser por razones personales. Es deber del integrante explicar porque lo hace y del grupo determinar la validez del bloqueo. El grupo llegara a una decisión final cuando ningún integrante tenga una decisión de bloqueo valida.

En caso de que una decisión afecte a varias áreas de la organización se realizará de nuevo el proceso de toma de decisión en el SoS meeting.

El canal de comunicación entre áreas de trabajo son los Scrum Master y se hará por medio de un SoS meeting explicado más en detalle en el [Reporte de progreso grupal total](#_10.2.3.2_Reporte_de).

9. Administración del Proyecto

# 9.1 Métodos y Herramientas de Estimación

## 9.1.1 Objetivo General

El objetivo de este plan es definir los métodos y herramientas que se utilizarán para estimar tanto el costo como el tiempo que tomará realizar el proyecto RAWR de, tal forma que sea posible adaptar tales estimaciones a los cambios que se presenten durante el desarrollo del proyecto.

## 9.1.2 Responsables

El Master de calidad y su equipo serán los responsables de ejecutar el plan de estimación así como hacer modificaciones, adaptando estas estimaciones al entorno cambiante del proyecto haciendo uso del método seleccionado de estimación para luego comparar las mediciones tomadas con lo esperado.

## 9.1.3 Recursos Requeridos

Para realizar la estimación del proyecto se hará uso de los conocimientos y apreciaciones de cada integrante del grupo y de Microsoft Excel para consignar los datos y unidades estimadas para cada parte específica del proyecto y así hallar un estimado del costo y el tiempo del proyecto RAWR.

## 9.1.4 Puesta en Marcha

Para realizar la estimación del proyecto, el grupo se basó en los métodos de estimación agiles Wide-Band Delphi y Planning Poker (Scrum Poker) para hacer un proceso intuitivo donde todos los integrantes del grupo estiman las partes del proyecto que van a realizar dependiendo de sus roles y realizando una retroalimentación para buscar corregir errores de estimación [1].

Wide-Band Delphi [2] el cual se deriva del método Delphi y ha sido usado en diferentes tipos de industrias. Este proceso ha sido exitoso pues se escogen algunas personas del equipo los cuales representan ramas importantes del proyecto y se reúnen para hacer correcciones entre los integrantes y así se tiene retroalimentación temprana que permite identificar errores y errores de estimación.

Este método es útil para este proyecto pues es un grupo pequeño de 6 personas las cuales tienen asignadas roles que representan las partes importantes del proyecto ([ver sección 8.4.2 Organigrama y Descripción de Roles](#_8.4.2_Organigrama_y)), así en este caso todos los integrantes del grupo aportan para crear la estimación del proyecto. Luego de discutir los estimativos de cada experto se integran y se hace retro alimentación para mejorar puntos específicos y corregir errores. Este método se divide en los siguientes pasos:

* Elección de equipo
* Reunión inicial
* Preparación individual
* Reunión de estimación
* Reunir Tareas
* Revisar Resultados

Planning Poker [3] [4] es un método de estimación donde se utilizan unas cartas que se les dan a cada persona con una secuencia determinada que representa una unidad con la que el equipo va a realizar la estimación. Se discuten las características del proyecto con el dueño del producto y cada persona elige una carta para indicar la estimación que cree que se le debe asignar a dicha característica. Si todos eligen el mismo valor este es el que se le asigna, si esto no ocurre las personas discuten este problema y se vuelven a seleccionar las cartas.

Teniendo en cuenta estas dos metodologías para la estimación en proyectos agiles, el grupo decidió tomar algunos conceptos que son útiles para este caso en específico.

En la estimación del proyecto todos los integrantes del grupo intervendrán para dar sus opiniones sobre unas tareas que definirán como principales en la realización del producto y su documentación, luego se dan estimativos teniendo en cuenta a la persona encargada según los roles asignados y los demás integrantes darán retroalimentación a este estimativo para que se acerque lo más posible a la realidad.

Para tomar la dicción de esta estimación de horas por tarea se tendrá en cuenta la complejidad de dicha tarea así como el tamaño que se espera.

El estimativo de cada tarea será consignado en unidades de horas de trabajo por integrante que representan el tiempo necesario para cumplir esta tarea de forma exitosa y se asignara un valor por hora de trabajo. Inicialmente se hará una división por cada entrega y cada tarea estará subdividida en secciones que explican a qué parte del desarrollo del proyecto pertenecen.

Todas estas horas son sumadas para finalmente dar un tiempo estimado para la finalización del proyecto y con esto conseguir una idea del costo de todo el proyecto. ([ver anexo Estimación de costo del proyecto](file:///C:\Users\Estudiante\Downloads\Anexos\Estimación%20costo%20proyecto.xlsx))

# 9.2 Inicio del Proyecto

## 9.2.1 Plan de entrenamiento del personal

### 9.2.1.1 Objetivos

* Establecer un proceso de inclusión de herramientas de aprendizaje.
* Controlar y verificar la utilidad de las herramientas de aprendizaje utilizadas.

### 9.2.1.2 Responsables

* Equipo del área de desarrollo.
* Equipo del área de arquitectura y diseño

### 9.2.1.3 Recursos requeridos

* Microsoft Office Excel[[5]](#footnote-5)

### 9.2.1.4 Puesta en marcha

Para considerar la inclusión de una nueva herramienta de aprendizaje se debe remitir con alguno de los integrantes del equipo de desarrollo del área de desarrollo o del área de arquitectura y diseño según el tipo de herramienta[[6]](#footnote-6), el cual debe completar la sección de control en el [formato de control y supervisión de herramientas](file:///C:\Users\Fabián%20Merchán\Google%20Drive\Proyecto%20Ingesoft\Primera%20entrega\SPMP\Formatos\Formato%20de%20control%20y%20supervisión%20de%20herramientas.xlsx) en un plazo no mayor de dos días.

### 9.2.1.4.1 Supervisión y control

El equipo de desarrollo del área de desarrollo junto al de arquitectura y diseño deberá verificar la eficiencia del proceso de aprendizaje de la herramienta, preguntando a los integrantes acerca de la experiencia de aprendizaje y luego consignando los resultados en la sección de supervisión en el [formato de control y supervisión de herramientas](file:///C:\Users\Fabián%20Merchán\Google%20Drive\Proyecto%20Ingesoft\Primera%20entrega\SPMP\Formatos\Formato%20de%20control%20y%20supervisión%20de%20herramientas.xlsx) máximo una semana después de la inclusión de la herramienta.

### 9.2.1.4.2 Acciones correctivas

Si una herramienta por algún motivo, dejase de ser funcional[[7]](#footnote-7) debe ser retirada del plan de entrenamiento del personal, se debe por la persona que aprobó el recurso de aprendizaje.

## 9.2.2 Infraestructura

Dentro de la infraestructura que soporta el proyecto RAWR se encuentran las instalaciones físicas, herramientas de software definidas anteriormente ([ver sección 8.2 Lenguajes y Herramientas](#_8.2_Lenguajes_y)) e infraestructura de hardware necesaria para desarrollar de forma exitosa el producto.

### 9.2.2.1 Instalaciones

El trabajo se realizará en su mayoría en las casas de los integrantes del grupo para el desarrollo tanto de documentos como de código de la aplicación. Las instalaciones de la Pontificia Universidad Javeriana serán usadas para realizar reuniones presenciales donde se hablará del avance del proyecto y también para las reuniones con los clientes y asesores, es decir los profesores de la materia Ingeniería de Software. También se usará la instalación de la biblioteca general de la Pontificia Universidad Javeriana para realizar consultas y por ultimo las salas de Sistemas de la Facultad de Ingeniería esporádicamente para tener encuentros presenciales donde se tratarán temas técnicos y es necesario tener un computador.

### 9.2.2.2 Herramientas de software

##### 9.2.2.2.1 Herramientas para desarrollo software

Las herramientas y lenguajes utilizados para el desarrollo de software ([ver sección 8.2 Lenguajes y Herramientas](#_8.2_Lenguajes_y)) deben ser instaladas en las maquinas personales de los integrantes del área de desarrollo al principio del proyecto, así se podrá trabajar desde las casas de cada integrante y se hará el seguimiento e integración a través de GitHub. Adicionalmente, los integrantes del área de desarrollo deberán crear el servidor en DigitalOcean e instalar la base de datos MySql antes de comenzar con el desarrollo de la aplicación.

#### 9.2.2.2.2 Herramientas para organización y comunicación

Las herramientas definidas para manejar la organización del proyecto y comunicación ([ver sección 8.2 Lenguajes y Herramientas](#_8.2_Lenguajes_y)), google Drive, Microsoft Word, Microsoft Excel, Zotero, Google Hangouts y Slack deben ser instaladas correctamente según corresponda en los equipos personales de todos los integrantes del grupo. Microsoft Project deberá ser instalada por el SoS Master pues esta herramienta será utilizada en diversas tareas para este rol. Bizagi deberá ser instalada por el SoS Master y Master de calidad pues esta herramienta será utilizada en diversas tareas para estos roles.

#### 9.2.2.2.3 Herramientas de diseño

Las herramientas definidas para el diseño gráfico de la aplicación ([ver sección 8.2 Lenguajes y Herramientas](#_8.2_Lenguajes_y)), deberán ser instaladas o preparadas correctamente por los integrantes del área de diseño gráfico. La herramienta starUML deberá ser instalada por el Master de arquitectura y diseño, y las personas que pertenezcan a esta área para realizar sus tareas de forma correcta.

##### 9.2.2.3 Herramientas de hardware

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nombre De La persona | Sistema Operativo | Procesador | Modelo | Tipo De Sistema | RAM | Tamaño De Disco Duro |
| Luis | OS X Yosemite | Intel Core i5 | Intel Core i5 dual Core de 2.5 GHz |  | 4,00 GB | 500 GB |
| Luis | Windows 8 | Intel (R) Core(TM) i7 | Intel (R) Core(TM) i7-3630QM CPU @ 2.40GHz | 64 bits | 6,00 GB | 698 GB |
| David | Windows 8.1 Pro / Ubuntu 14.10 | Inter (R) Core(TM) i5 | Intel (R) Core(TM) i5-3210M @2.5 GHz | 64 bits | 6,00 GB | 1 TB |
| Luisa | Windows 8.1 | Intel(R) Celeron(R) | Intel(R) Celeron(R) CPU N2830 2.16GHz | 64 bits | 1.89Gb | 500GB |
| Alfredo | Windows 8.1 | Intel (R) Core(TM) i7 | Intel (R) Core(TM) i7 CPU Q820 @ 1.73GHz | 64 bits | 4,00 GB | 260GB |
| Stiven | Windows 7 | Intel (R) Core(TM) i5 | Intel (R) Core(TM) i5 CPU 4570 @ 3.2GHz | 64 bits | 8,00 GB | 500 GB |
| Fabián | Windows 7 | Intel (R) Core 2Dduo | Intel (R) Core(TM) 2Duo CPU T5450@ 1,66GHz | 32 bits | 2,00 Gb | 140 GB |

Tabla 8: Maquinas personales de los integrantes del grupo

Adicionalmente se estableció que para el desarrollo del producto se trabajará sobre un dispositivo móvil: Google Nexus 4.

# 9.3 Planes de Trabajo del Proyecto

## 9.3.1 Descomposición de Actividades

El proyecto se compone de cuatro fases distintas, en un inicio el grupo no se encontraba muy organizado y por lo tanto no tenía muy claro qué realizar. Es por ello que la primera categoría es tareas iniciales, puesto que era trabajo a realizar, pero no teníamos muy claro qué desarrollar primero ni qué medidas tomar. Posteriormente organizamos el desarrollo del SPMP por Sprints ye entregas, en los cuales se debían de realizar ciertos componentes de los documentos para finalmente tener una revisión cruzada, reunión con el profesor y reunión grupal en la cual se discute cómo va el proyecto, hacia donde va, qué hay que mejorar y qué componentes ya se cumplieron. A continuación se muestran las principales actividades del proyecto con su descomposición. [**WBS-RAWR.docx**](file:///C:\Users\Estudiante\Downloads\Anexos\WBS-RAWR.docx)

## 9.3.2 Calendarización

### 9.3.2.1 Objetivos específicos:

* Describir procesos y actividades para garantizar que estas se realicen en los tiempo estipulados por los roles adecuados.
* Describir procesos y actividades para garantizar la realización de las diferentes fases del proyecto en las entregas correspondientes.
* Realizar un diagrama de Gantt, que muestre y clarifique los tiempo, fechas de entrega para desarrollar las diferentes actividades.

#### **9.3.2.2 Responsables:**

* SoS Master
* El cumplimiento de las fechas, procesos, actividades y tareas es responsabilidad del todos los miembros del grupo.

### 9.3.2.3 Recursos Requeridos:

Microsoft Excel.

### 9.3.2.4 Puesta En Marcha:

Dependiendo de cada entregable, acorde a las fechas asignadas por el cronograma de la materia de Ingeniería de Software, el calendario se debe actualizar, revisar, mejorar y hasta se puede requerir reasignar tareas acorde a lo sucedido en los entregables precedentes. Sean componentes como la calidad, métricas o riesgos que posteriormente deben ser estudiados para la toma de decisiones en el proyecto.

### 9.3.2.5 Calendario

A continuación se encuentran diferentes links con los calendarios de las correspondientes entregas, adicionalmente un diagrama conjunto de Gantt con las actividades, los roles que las deben realizar, el tiempo mostrado en el ‘eje x’ y las relaciones de precedencia se indican con líneas entre actividades.

Diagrama de Gantt completo de Inicio a fin del Proyecto [**Diagrama Gantt Completo.pdf**](file:///C:\Users\Estudiante\Downloads\Anexos\Diagrama%20de%20Gantt-Completo.pdf)[**(Diagrama de Gant.mmp**](file:///C:\Users\Estudiante\Downloads\Anexos\Diagrama%20de%20Gantt%20Completo.mpp)**)**

[Tabla resumen entrega uno](file:///C:\Users\Estudiante\Downloads\Anexos\PRIMERA%20ENTREGA-V0.3.docx)

[Tabla resumen entrega dos](file:///C:\Users\Estudiante\Downloads\Anexos\SEGUNDA%20ENTREGA-V0.3.docx)

[Tabla resumen entrega tres](file:///C:\Users\Estudiante\Downloads\Anexos\TERCERA%20ENTREGA%20%20v03.docx)

Adicionalmente es necesario detallar las actividades que se realicen en momentos específicos puesto que son las mismas actividades por iteración, por lo tanto en la Ilustración 2 se especifica el detalle para una sola iteración.

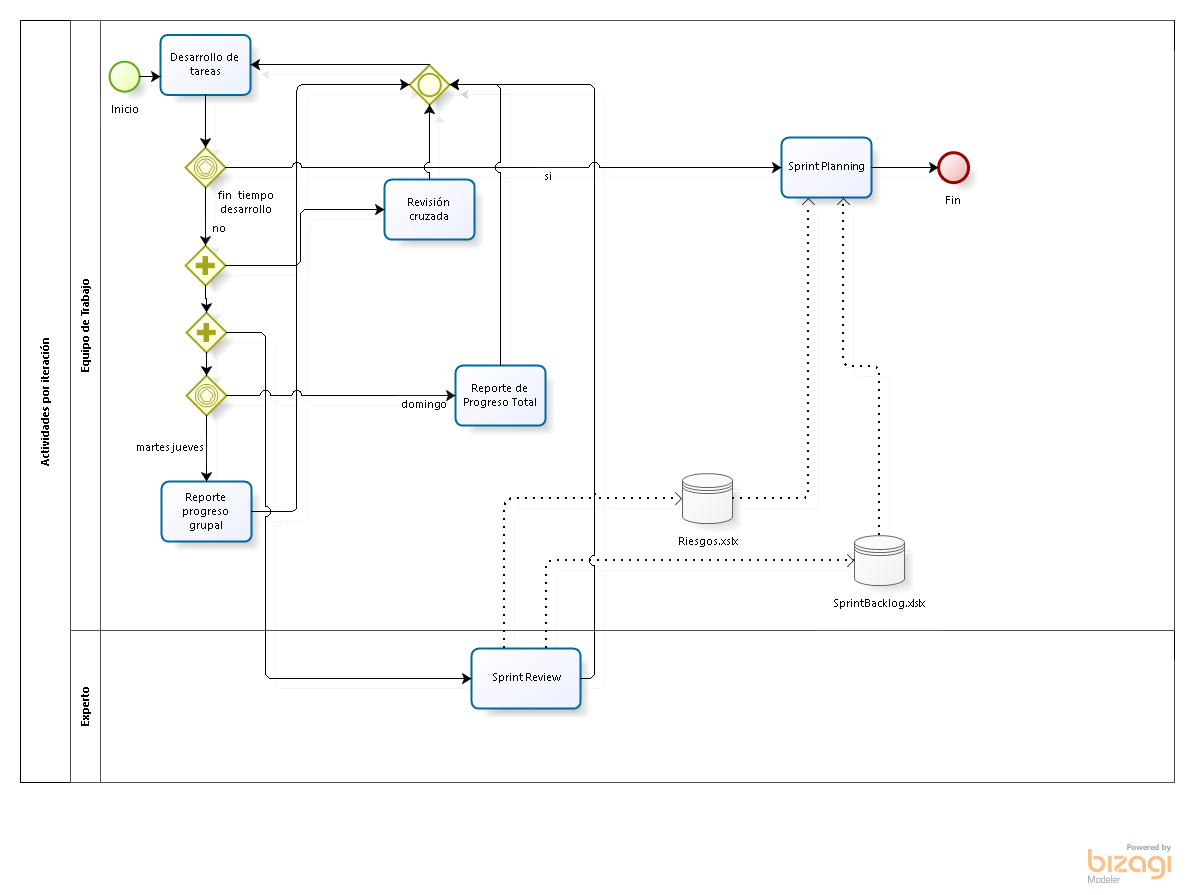


Ilustración 2: Actividades a realizar en una iteración

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler.

En cada iteración se empezará a desarrollar lo que se planee en el sprint planning de la iteración pasada, después de esta actividad se ejecutan dos actividades en paralelo.

* Sprint review
* Revisión cruzada

Adicional a esto los martes y jueves se harán los reportes de progreso grupal y los domingos reportes de progreso total. Una vez terminadas o cumplido el tiempo máximo de de desarrollo, se hace el sprint planning con el fin de definir las tareas que se van a hacer en la siguiente iteración.

## 9.3.3 Asignación de recursos

La asignación de recursos se hace con el propósito de conocer los insumos que están ligados a la ejecución de cada actividad. Para la asignación se deben tener en cuenta los recursos humanos y materiales. La información acerca de los recursos necesarios y roles requeridos la puede encontrar en la sección de calendarización. **Insertar nota de sección.**

## 9.3.4 Asignación de presupuesto y justificación

El presupuesto para el desarrollo de la aplicación RAWR se calcula a partir de la cantidad de horas y los recursos humanos necesarios para realizar cada tarea existente en el [cronograma](#_9.3.2.5_Calendario) del proyecto.

La hora que se paga a cada recurso humano es la misma sin importar su rol y se establece en $13.000 COP el cual es el salario promedio de un practicante de ingeniería de sistemas en la Pontificia Universidad Javeriana [Luisa].

El costo total del proyecto así como la especificación de costos por tarea se encuentra en el anexo de presupuesto ([ver Anexo Estimación costo proyecto](file:///C:\Users\Estudiante\Downloads\Anexos\Estimación%20costo%20proyecto.xlsx)).

10 Monitoreo y Control del Proyecto

# 10.1 Gestión de requerimientos

## 10.1.1 Objetivos

* Definir actividades para la correcta administración de requerimientos.
* Definir actividades para el control de cambios al SRS.

## 10.1.2 Responsables

* Equipo área de arquitectura y diseño.
* Scrum Master área de desarrollo.
* Scrum Master área de arquitectura y diseño.
* SoS Master.

## 10.1.3 Recursos

* Microsoft Excel y Microsoft Word.

## 10.1.4 Puesta en marcha

Tomando como base la metodología descrita en [1] se definen 5 actividades que permiten identificar, especificar, modelar, priorizar, analizar y validar los requerimientos del proyecto.

### 10.1.4.1 Elicitación

Con el fin de entender las necesidades de los usuarios y los stakeholders [1] se usan 3 actividades:

* **Entendimiento del dominio del problema:** El propósito de esta actividad es investigar y examinar en detalle la situación en la cual el sistema funcionará [1], para esto se realizará una encuesta a los potenciales usuarios dueños de mascotas y entrevistas a los potenciales usuarios dueños de empresas.
* **Identificación de las fuentes de requerimientos:** Las fuentes de información se dividieron según su rol en el sistema. Para el rol de usuario dueño no se tiene una fuente especifica de información por lo que se realizan técnicas de recolección de información por medio de encuestas[2]. Por otro lado, para el rol de usuario empresa se recurre a veterinarias y tiendas de productos para mascotas. Finalmente, como guía para el diseño y concepción de la red social se utiliza como base Facebook Inc.[[8]](#footnote-8)
* **Análisis de stakeholders:** La descripción de los stakeholders puede ser encontrada en la Tabla 6.

### 10.1.4.2 Especificación y modelado

Con el fin de representar los requerimientos funcionales del sistema de manera gráfica y mostrar las interfaces y funciones se utilizarán DFDs [3]. Por otro lado, para realizar la especificación detallada de los requerimientos se utilizara una plantilla de especificación basada en la que provee Volere [4]. La plantilla empleada cuenta con los siguientes campos:

**ID requerimiento:** Identificador único que facilita la trazabilidad del requerimiento.

**Versión:** Indica la versión actual de la especificación. Comienza en 1 y va aumentando en decimales de .1. En caso de que un requerimiento no se encuentre en su primera versión la información de los cambios hechos se verá reflejada en una plantilla de control de cambio.

**Tipo:** Indica si el requerimiento es funcional, no funcional o una restricción.

**Fecha creación:** Fecha en la que se escribió la versión inicial de la especificación del requerimiento

**Fecha modificación:** Fecha en la que se escribió la versión actual de la especificación del requerimiento.

**Nombre:** Conjunto de máximo 4 palabras que describen el requerimiento.

**Descripción:** Frase corta que describe el requerimiento en lenguaje natural.

**Criterio de culminación:** Frase corta que describe como validar si hay una funcionalidad implementada que satisfaga el requerimiento.

**Restricciones:** Lista de limitaciones en campos o restricciones para cumplir un requerimiento.

**Creador:** Lista de personas que concibieron el requerimiento.

**Valor prioridad:** Valor final obtenido de la fase de priorización.

**Estado:** Indica si el requerimiento esta propuesto, aprobado, en implementación, en validación o terminado.

**Requerimientos asociados:** Lista de identificadores de requerimientos que tienen una asociación directa de dependencia.

**Casos de uso asociados:** Lista de identificadores de casos de uso que dieron origen al requerimiento.

**Comentarios:** Espacio para escribir notas adicionales.

### 10.1.4.3Priorización

Con el fin de realizar una priorización cuantitativa se utiliza la metodología descrita en la sección 2.1 del SRS.

### 10.1.4.4 Trazabilidad

Con el fin de llevar la traza y conocer las dependencias de cada requerimiento se utiliza la metodología descrita en la sección 2.2 del SRS.

### 10.1.4.5 Calidad

Para asegurar buenos atributos en la especificación de los requerimientos se utiliza la metodología descrita en la [sección 12.5.1.5.4](#_12.5.1.5.4_Proceso_de).

## 10.1.5 Control de cambios

RAWR define un proceso formal para realizar cambios a los requerimientos una vez estén aprobados por los productowners.

### 10.1.5.1 Petición de cambio

En caso de que se quiera realizar un cambio a la especificación de un requerimiento o al conjunto de requerimientos en general el individuo que lo proponga deberá llenar una fila en el formato de petición de cambio de requerimiento y coordinar una reunión en no más de una semana con el Scrum master del área de desarrollo, el Scrum master del área de arquitectura y diseño y el SoS master con el fin de evaluar el impacto que tendrá el cambio en el proyecto y analizar la factibilidad de realizarlo de manera subjetiva.

El SoS Master debe registrar las conclusiones de la reunión en la fila del formato de petición de cambio de requerimiento correspondiente. Cada fila del formato cuenta con los siguientes campos:

* Nombre solicitante: Nombre de quien realiza la petición.
* Id requerimiento: Identificador único del requerimiento.
* Tipo de petición: Determina si es una petición para agregar, eliminar o modificar un requerimiento.
* Fecha de petición: Indica la fecha en la que se realizó la petición.
* Razón petición: Breve información de por qué se realiza la petición.
* Aprobado: Se debe escribir SI o NO Indicando si la petición fue aprobada.
* Notas: Descripción que sustenta la decisión tomada.

El proceso de petición de cambio se puede observar gráficamente en el BPMN mostrado en la Ilustración 1.

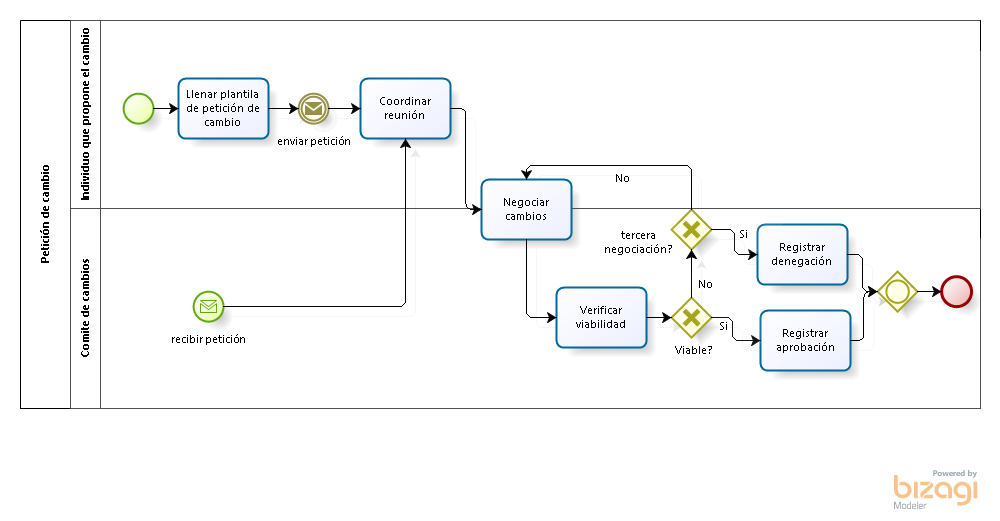


Ilustración 1: Diagrama BPMN de petición de cambio

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler

# 10.2 Monitoreo y control de progreso

## 10.2.1 Objetivos

* Describir actividades para reportar el progreso individual y grupal.
* Describir actividades para el control de cronograma.

## 10.2.2 Responsables

* SoS Master
* Scrum Masters.

## 10.2.3 Recursos

Microsoft Project y Microsoft Excel.

## 10.2.3 Puesta en marcha

RAWR define cuatro actividades principales que permiten reportar el progreso individual, reportar el avance del proyecto y controlar el cronograma.

### 10.2.3.1 Reporte de progreso grupal por área

Con el fin de informar al Scrum master del área respectiva el estado de la porción de proyecto que se está elaborando se realizará una modificación al Scrum diario explicado en [19] realizándolo los días martes y jueves. En el Scrum cada miembro del equipo documentará por escrito en una plantilla de avance por área (Ver anexo Plantilla Avance Área) y explicará verbalmente lo siguiente:

* ¿Qué hizo desde el último Scrum?
* ¿Qué hará hasta el próximo Scrum?
* ¿Hay algún impedimento que evite que el equipo o el miembro no cumpla con el objetivo del sprint?

### 10.2.3.2 Reporte de progreso grupal total

Se hará un “Scrum-of-Scrums meeting” [41] virtual todos los domingos con el fin de informar al SoS Master el estado de todo el proyecto. Un miembro de cada Scrum en rol de representante participará en el SoS meeting. El representante es escogido por el equipo y no necesariamente debe ser el mismo para todas las reuniones [41]. En el SoS meeting cada representante documentara por escrito en la plantilla de avance total (Ver anexo Plantilla Avance Total) y explicará verbalmente lo siguiente:

* ¿Qué hizo su equipo desde el último Scrum?
* ¿Qué hará su equipo hasta el próximo Scrum?
* ¿Hay algún impedimento que evite que el equipo no cumpla con el objetivo del sprint?

### 10.2.3.3 Reporte de progreso individual

Con el propósito de llevar un registro estadístico del desempeño de cada miembro se utilizará un Sprint Backlog [19] (Ver anexo Plantilla Tareas) en el que después de una revisión cruzada (Ver sección revisiones cruzadas) el evaluador y el evaluado en conjunto acuerdo estimarán (Ver sección 9.1) y documentarán las horas faltantes para terminar las actividades revisadas.

### 10.2.3.4 Sprint Review

El SoS Master tendrá una reunión con un Product Owner cada semana con el fin de revisar el sprint backlog, analizar que tareas se han terminado, discutir problemas ocurridos durante el sprint y realizar retroalimentación de calidad.

### 10.2.3.5 Sprint Planning meeting

Con el fin de planificar el trabajo que se realizará en el próximo Sprint toda la organización se reunirá el martes anterior a la terminación del Sprint activo a responder las siguientes preguntas:

* ¿Qué puede ser terminado en el siguiente Sprint?
* ¿Cómo se conseguirá completar el trabajo seleccionado?

El SoS Master se encargará de crear un Sprint Backlog de acuerdo a las conclusiones de la discusión.

### 10.2.3.6 Monitoreo y acciones correctivas

Con el fin de realizar un seguimiento al [cronograma](#_9.3.2.5_Calendario) el SoS Master revisará cada 2 días el progreso del proyecto real y lo contrastará con el estimado. En caso de que hubiera una diferencia considerable después de 3 revisiones consecutivas se utilizará un espacio del siguiente Scrum para realizar [reenvíos de tareas](#_10.2.3.4.1_Reenvío_de), [estudio de desempeño individual](#_10.2.3.4.2_Estudio_de) a los responsables de las tareas atrasadas, verificación del [reglamento](#_12.1.1_Reglamento) y ejecución de planes de mitigación y contingencia de riesgos (ver anexo Riesgos).

#### 10.2.3.6.1 Reenvío de tareas

En caso de que las horas estimadas para acabar una tarea superen la fecha de entrega o el desempeño individual de un integrante no sea el adecuado el Scrum Master del área respectiva procederá a reenviar la tarea al integrante o a los integrantes que tengan menos carga en ese momento. En caso de que todos los integrantes del área estén sobrecargados se procederá a reenviarla a quienes tengan menos trabajo asignado de toda la organización.

##### 10.2.3.6.2 Estudio de desempeño individual

Cuando un sancionado pasa por un estudio de desempeño individual estará a prueba por una semana. Pasada esa semana se utilizará un espacio del siguiente Scrum para revisar el estado de las tareas que tenga asignadas. En caso de que las tareas presenten un retraso el integrante pasará por juicio de grupo. Si la decisión del juicio es de permanencia se volverá a inicial el estudio de desempeño individual. Las iteraciones continuaran hasta que la decisión del juicio sea de expulsión o las tareas del integrante no tengan ningún retraso.

# 10.3 Cierre del Proyecto

## 10.3.1 Objetivo General

Explicar conjunto de actividades a realizar para el cierre del proyecto.

## 10.3.2 Objetivos Específicos

Describir el conjunto de actividades a realizar por cada rol junto con los recursos necesarios para llevarlo a cabo.

## 10.3.3 Responsables

* Sos Master.
* Sos Master calidad y administración de configuración.

## 10.3.4 Recursos requeridos

* 2 discos compactos
* 2 Computadores portátiles

## 10.3.5 Puesta en Marcha

Las actividades relacionadas a la entrega del proyecto son:

* Reunión de retroalimentación
* Elaboración del reporte gerencial por parte del SoS master
* Revisión contendido de las entregas

La revisión del contenido de las entregas consistirá en que previo a cada entrega el SoS master y el master de configuración quemarán en un disco compacto el contenido del proyecto el día previo a cada una de las entregas. Una vez ser verifique que todo el contenido quede guardado en el CD, traerán el disco al lugar de la entrega, después cuando esté el grupo reunido, una hora previa a la clase, se hará una última prueba en un computador, que traerá el master de arquitectura, en donde se escogerá el CD que tenga la mejor condición visual (rayones, manchones) y de contenido.

11 Entrega del Producto

Antes de hacer una entrega formal al cliente hay que ejecutar [el plan de aceptación](#_8.3_Plan_de). El Responsable de la entrega del producto final será el SoS master y se llevará a cabo el 19 de mayo de 2015 en el que el grupo de trabajo sustentará con una presentación donde se muestra a los clientes el producto terminado. La entrega se hará en formato DVD y contendrá los siguientes entregables:

* SPMP y refinación de casos de uso
* SRS y SDD
* Prototipo Final

12 Procesos de Soporte

# 12.1 Ambiente de Trabajo

Con el fin de asegurar un ambiente de trabajo sano. El equipo debe atenerse a las reglas aquí descritas y someterse a las sanciones en caso de incumplir alguna de ellas. En caso de que en el transcurso del proyecto se vea la necesidad de agregar o suprimir una regla se procederá a una votación grupal para tomar una decisión.

## 12.1.1 Reglamento

* Todo integrante debe asistir a las reuniones acordadas en el canal correspondiente de Slack.
* Todo integrante debe completar las plantillas solicitadas por el gerente en los tiempos acordados.
* Todo integrante debe referenciar en formato IEEE la información utilizada al elaborar un documento.
* Todo integrante debe terminar sus tareas en los tiempos establecidos y cumpliendo con los parámetros de calidad acordados en el [plan de control de calidad](#_12.5.1_Plan_de).
* Todo integrante debe revisar Slack por lo menos una vez al día.
* Todo integrante debe cumplir con sus tareas asignadas en los tiempos establecidos.
* Está prohibida la agresión verbal y física.
* Está prohibido ingerir bebidas alcohólicas o estar bajo sus efectos al reunirse con el grupo de trabajo.
* Cuando se realice una votación virtual cada integrante tendrá un plazo de 24 horas para dar su voto. Si pasado el plazo no contesta se asumirá que está de acuerdo con la decisión.
* El registro de las sanciones, multas y penalizaciones se realizara en el formato de control de reglamento.
* El documento de control es de acceso público pero solo el SoS master tiene derecho a modificarlo.

## 12.2.2 Penalizaciones

Penalizaciones leves:

* Llegar hasta 15 minutos tarde a una reunión sin previo aviso.
* Entregar una tarea con un retraso de hasta 3 horas.
* Referenciar con una estructura inapropiada.
* Se penalizara al creador y al evaluador si luego de la aprobación de un documento faltan hasta 3 ítems que debería tener según la plantilla de SPMP provista por el cliente.

Penalizaciones Moderadas:

* Llegar entre 15 y 30 minutos tarde a una reunión sin previo aviso.
* Entregar una tarea con un retraso de 3 a 5 horas.
* Se penalizara al creador y al evaluador si luego de la aprobación de un documento faltan de 4 a 6 cosas de las que debería tener según la plantilla de SPMP provista por el cliente.

Penalizaciones Graves:

* No asistir a una reunión sin excusa valida.
* Entregar una tarea con un retraso de más de 5 horas.
* No referenciar información externa utilizada en los documentos.
* Se penalizara al creador y al evaluador si luego de la aprobación de un documento faltan más de 6 cosas de las que debería tener según la plantilla de SPMP provista por el cliente.

Juicio de grupo:

* Incumplir una tarea asignada sin previo aviso.

Acumulación de penalizaciones:

* 3 penalizaciones leves conllevan a una penalización moderada.
* 3 penalizaciones moderadas conllevan a una penalización grave.
* 3 penalizaciones graves conllevan a pasar por juicio de grupo.

## 12.2.3 Multas y sanciones

* Por cada penalización leve el integrante deberá pagar a RAWR una suma de $5.000 COP.
* Por cada penalización moderada el integrante deberá pagar a RAWR una suma de $10.000 COP.
* Por cada penalización grave el integrante deberá pagar a RAWR una suma de $20.000 COP y se procederá a hacer reasignación de tareas.
* En caso de que un integrante deba pasar por juicio de grupo los demás integrantes procederán a realizar una votación anónima para decidir su permanencia en el grupo.
* En caso de que un integrante pase por juicio de grupo y el veredicto sea de permanencia los demás integrantes procederán a decidir la multa que le será asignada.
* En caso de ocurrir una agresión verbal o física se expulsará del grupo al agresor.

# 12.2 Análisis y administración de riesgos

## 12.2.1 Objetivos

* Describir actividades para la identificación, clasificación y priorización de riesgos.
* Describir actividades para mitigación y contingencia de riesgos.

## 12.2.2 Responsables

* SoS Master
* Equipo del área de calidad y administración de la configuración

## 12.2.3 Recursos

Microsoft Office Excel.

## 12.2.4 Puesta en Marcha

RAWR utiliza el modelo de administración de riesgos continuo explicado en [42] adaptado para seguir dos pasos fundamentales: la evaluación de riesgos y el control de riesgos. Divididos a su vez en 3 pasos cada uno: por un lado la identificación, análisis y priorización de riesgos y por otro el plan de administración, la resolución y el monitoreo de los mismos [43].

Las actividades descritas anteriormente se realizaran en el primer reporte de progreso grupal de cada Sprint de la siguiente manera:

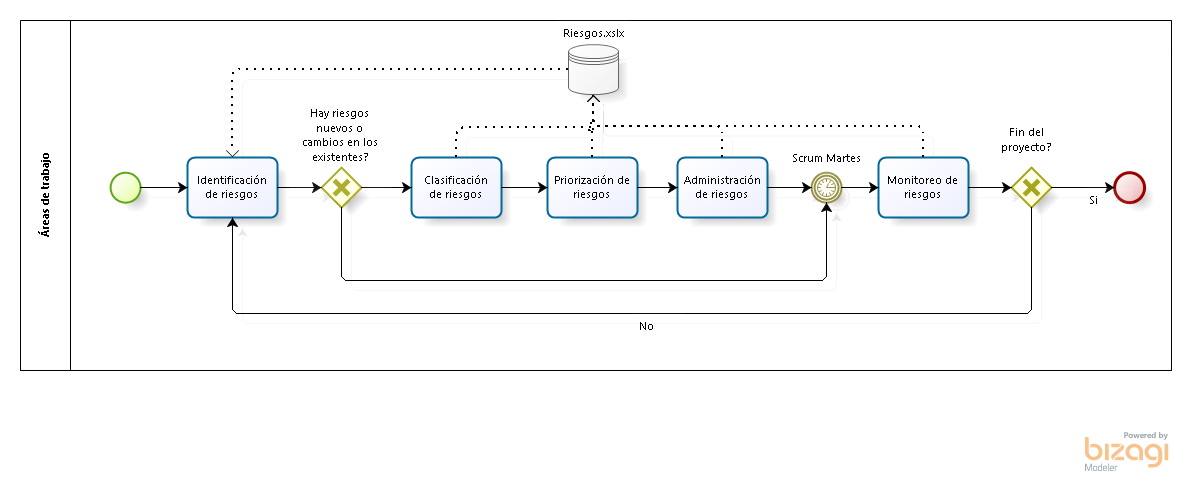


Ilustración 1: BPMN Análisis y administración de riesgos

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos [12].

### 12.2.4.1 Identificación de riesgos

Con el fin de generar una lista de riesgos específicos se dedicará un tiempo de la revisión del Sprint para que los integrantes de RAWR identifiquen y documenten los posibles riesgos de la próxima iteración en la pestaña “Riesgos” de la plantilla de administración de riesgos (ver anexo Riesgos). En la plantilla se deberán escribir las condiciones para que un riesgo ocurra y las consecuencias que podría causar [44].

Para hacer más eficiente la identificación de riesgos el equipo analizará las listas de riesgos más frecuentes [45] y responderá las preguntas provistas en [46] con el fin de generar ideas.

Los tipos de riesgos que serán tomados en cuenta serán de usuario (U), requerimientos (R), complejidad del proyecto g(PC), equipo (T), ambiente organizacional (OE) y planeación y control (P&C) [45].

### 12.2.4.2 Clasificación de riesgos

Con el fin de cuantificar los riesgos se documentará en la pestaña “Clasificación” de la plantilla de administración de riesgos (Ver anexo Riesgos) los resultados del análisis de probabilidad de ocurrencia y valoración de impacto descrito en [42]. La cuantificación se realizará de la siguiente manera:

Probabilidad de ocurrencia:

* Raro: menor al 20%.
* Improbable: Entre 20% y 40%.
* Posible: Entre 40% y 60%.
* Probable: Entre 60% y 80%.
* Muy Probable: Mayor a 80%.

Valoración de impacto:

* Insignificante.
* Menor.
* Moderado.
* Crítico.
* Catastrófico.

Una vez que cada riesgo tenga una probabilidad de ocurrencia y una valoración de impacto definida se procederá a clasificarlo según la Tabla 1. Los riesgos que tengan una tolerancia elevada pasaran a la etapa de priorización.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tolerancia** | | **Impacto** | | | | |
| **Insignificante** | **Menor** | **Moderado** | **Crítico** | **Catastrófico** |
| **Probabilidad** | **Raro** | Elevada | Elevada | Elevada | Elevada | Media |
| **Improbable** | Elevada | Elevada | Media | Media | Media |
| **Posible** | Elevada | Elevada | Media | Media | Baja |
| **Probable** | Elevada | Media | Media | Baja | Baja |
| **Muy probable** | Elevada | Media | Baja | Baja | Baja |

Tabla 9: Matriz de Tolerancia de riesgos [42].

#### 12.2.4.3 Priorización de riesgos

El siguiente paso es ordenar los riesgos que tienen una tolerancia baja por medio de la Ecuación 1, en donde P es la probabilidad de ocurrencia y C es el costo para el proyecto si el riesgo ocurre [47]. Posteriormente el gerente documentará en la pestaña “Top 10” de la plantilla de riesgos (ver anexo Riesgos) los 10 primeros. En caso de que dos riesgos o más tengan el mismo puntaje se realizará una votación entre todos los integrantes del equipo para decidir sus prioridades [43].

Ecuación 1. Exposición al riesgo global.

### 12.2.4.4 Administración de riesgos.

Una vez que se hayan escogido los 10 riesgos a los que se les hará un análisis más detallado el SoS Master procederá a liderar una discusión entre el equipo con el fin de decidir el plan de mitigación, el plan de contingencia [48] y la frecuencia con la que se monitoreará cada riesgo. Posteriormente, el equipo documentará las decisiones en las pestañas “Riesgo x” (donde x es la prioridad del riesgo) en la plantilla de riesgos (Ver anexo Riesgos).

### 12.2.4.5 Monitoreo de riesgos.

Todas las semanas en el Scrum del martes el equipo analizará si hay riesgos nuevos que hayan surgido en el seguimiento del plan y de ser necesario actualizará la plantilla de riesgos (ver anexo Riesgos). Adicionalmente, el equipo reevaluará los riesgos ya existentes con el fin de actualizar su información en caso de que sus características hayan cambiado.

# 12.3 Plan de administración de configuración y documentación.

Esta sección describe en detalle el enfoque de gestión de configuración para el proyecto. Se explica en con detalle herramientas, procesos y ejecutores [49],[50],[6],[51],[52].

### 12.3.1 Objetivos

* Establecer las normas para manejar el repositorio del proyecto
* Establecer un control de versionamiento.
* Asegurar el control y la verificación de cambios.

### 12.3.2 Responsables

* Equipo del área de desarrollo
* Equipo del área de calidad y administración de la configuración

### 12.3.3 Recursos requeridos

* Github
* Microsoft Word
* Microsoft Excel[[9]](#footnote-9)

## 12.3.4 Puesta en marcha

### 12.3.4.1 Formato de versionamiento.

El siguiente es el formato para la numeración de versiones para código y documentación del proyecto

A.B.C

Los tres son números enteros no negativos dónde:

**A:** Es la versión principal. Cada vez que se termina un hito se aumenta esta cifra.

**B:** Número designado para indicar la incorporación de una nueva funcionalidad o parte de un hito.

**C:** Numero designado para control de errores. Aumenta según la cantidad de errores encontrados en la versión actual.

### 12.3.4.2 Control de versiones de documentos

Comprende todos los documentos tales como planes, manuales, SPMP, SRS y SDD.

#### 12.3.4.1.1 Antes de la línea base.

Al inicio del proyecto el control de cambios es informal[[10]](#footnote-10). Sin embargo, es requerida una aprobación por parte del integrante asignado a la [revisión cruzada de la semana](file:///C:\Users\Estudiante\Downloads\Formatos\Revisión%20cruzada%20de%20la%20semana.xlsx). Adicionalmente, se deben seguir las siguientes normas [53],[54]:

* El control de cambios se maneja a través del [formato de control de versiones](file:///C:\Users\Estudiante\Downloads\Formatos\Control%20de%20versiones%20para%20documentos.xlsx).
* Debe ser documentado en un formato de control de versiones que debe estar disponible al inicio de cada documento.
* Las versiones anteriores deben guardarse con el nombre del documento y su respectiva versión en una carpeta de versiones pasadas incluida en el mismo directorio del documento actual.

#### 12.3.4.1.2 Después de la línea base.

Luego de se apruebe la línea base cada cambio debe hacerse formal por medio del [formato de control de cambios](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20de%20control%20de%20cambios.xlsx). Debe existir un formato de control de cambios para cada documento que sea línea base. Los encargados de evaluar una petición de cambio son los integrantes del equipo del área de calidad y administración de la configuración quienes deben evaluarlo en un plazo no mayor a dos días [55],[56].

### 12.3.4.3 Control de versiones de software y código fuente

En [*Github*](file:///C:\Users\Estudiante\Proyecto%20Ingesoft\Primera%20entrega\SPMP\8.2LenguajesyHerramientasV.0.3.docx#Herramientas) el SoS Master del área de desarrollo creará una rama principal o [*master branch*](file:///C:\Users\Estudiante\Proyecto%20Ingesoft\Primera%20entrega\Glosario%20del%20documento.docx#masterBranch) del proyecto. Los cambios solo se deben realizar por medio de ramas o [branch](file:///C:\Users\Estudiante\Proyecto%20Ingesoft\Primera%20entrega\Glosario%20del%20documento.docx#branch) de la siguiente forma [26],[57]:

1. Escribir un breve resumen del cambio realizado en el [formato de control de cambios](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20de%20control%20de%20cambios.xlsx) donde se indiquen las características y cambios de la nueva versión.
2. Los integrantes que deseen incorporar sus avances o colaboraciones deben crear una rama con la copia de la rama principal del proyecto y trabajar en ella.
3. Parámetros de nombramiento de ramas secundarias.

<[Versión esperada](file:///C:\Users\Estudiante\Downloads\SPMP.docx#FormatoDeVersionamiento)> <Problema relacionado>[[11]](#footnote-11) <Descripción del cambio>

La descripción del cambio no debe superar las 20 palabras.

1. Para integrar una modificación al *master branch* se debe hacer una petición de aceptación[[12]](#footnote-12).
2. Cualquier integrante del equipo del área de desarrollo puede aprobar o rechazar una petición.
3. La persona que apruebe la petición de cambio debe consignar su nombre en el formato de control de cambios y en el repositorio[[13]](#footnote-13).

## 12.3.5 Supervisión y control

Se realiza por medio del [formato de supervisión de cambios](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20de%20supervisión%20de%20cambios.xlsx). Debe ser completado por cualquier miembro del equipo del área de calidad y administración de la configuración en un plazo no mayor a dos días desde la presentación del cambio.

Adicionalmente se deben seguir las siguientes normas:

* Si al finalizar la fecha límite de cambios consignada en el [formato de control de cambios](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20de%20control%20de%20cambios.xlsx) no se ha realizado el cambio se debe proceder a [sancionar al solicitante](file:///C:\Users\Estudiante\Downloads\SPMP.docx#12.1 Ambiente de Trabajo).
* Si en la fecha de revisión no se cumple el objetivo o el cambio afecta negativamente, inmediatamente el solicitante del cambio debe revertir los cambios hasta la versión anterior de la solicitud.[[14]](#footnote-14)
* En caso de que un integrante no siga el formato de versiones será [sancionado](file:///C:\Users\Estudiante\Downloads\SPMP.docx#12.1 Ambiente de Trabajo) y deberá corregir el error en un plazo no mayor a un día.

# 12.4 Métricas y Proceso de Medición

La medición de objetivos en RAWR se hará mediante el método de medición GQM (goal, question, metric) [58].

## 12.4.1 Objetivos

* Establecer patrones de medición.

## 12.4.2 Responsables

* Equipo de calidad y configuración.

## 12.4.3 Recursos requeridos

* Android Studio
* Microsoft Office Word
* Microsoft Office Excel

## 12.4.4 Puesta en marcha

### 12.4.4.1 Medición de un documento

Se debe evaluar un documento a través del [formato de evaluación de documentos](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20para%20la%20medición%20de%20documentos.xlsx) [59]. A pesar que la evaluación es subjetiva al criterio del lector, se considera que la evaluación de cualquier miembro del equipo de calidad y configuración es válida.

En la bibliografía del documento se encuentran diferentes documentos guía con consejos y buenas prácticas para la escritura de documentos[58] [60].

### 12.4.4.2 Medición de desarrollo y programación.

#### 12.4.4.2.1 Densidad de errores

Ecuación que indica que la densidad de errores es proporcional a la cantidad de errores (cantidad de procedimientos o desarrollos incorrectos) e inversamente proporcional al tamaño del código (cantidad total de líneas de código) [61].

#### 12.4.4.2.2 Productividad

Donde se indica que la productividad es proporcional a la cantidad de líneas de código producida e inversamente proporcional al esfuerzo (tiempo requerido para producir las líneas de código realizadas).

Los números obtenidos con las ecuaciones definidas en los numerales anteriores, no pueden representar un máximo o un mínimo absoluto, sin embargo son datos susceptibles a comparar con los demás desarrolladores, por lo cual el [plan de control de calidad](file:///C:\Users\Estudiante\Downloads\SPMP.docx#12.5 Control de Calidad) modera estos resultados.

### 12.4.4.3 Medición de requerimientos

Que indica que la ausencia de ambigüedad de los requerimientos es proporcional a la cantidad de requerimientos en las que los revisores tuvieron interpretaciones idénticas y es inversamente proporcional a la cantidad de requerimientos totales [56].

# 12.5 Control de Calidad

La gestión de calidad debe describir procesos y actividades que determinen responsabilidades, objetivos y políticas de calidad para que el proyecto cumpla con los requerimientos especificados por el cliente [62]. Con el propósito de garantizar un buen nivel de calidad en cada documento, código, diagrama, aplicación y otros archivos que deben ser entregados a los clientes y manejados en el interior del grupo se crea el siguiente plan de control de la calidad del proyecto.

## 12.5.1 Plan de control de la calidad del proyecto

### 12.5.1.1 Objetivo general:

* Asegurar que los documentos y archivos generados durante el desarrollo del proyecto posean un buen nivel de calidad por medio de la ejecución de actividades.

### 12.5.1.2 Objetivos específicos:

* Describir procesos y actividades para garantizar la calidad en la elaboración de documentos.
* Describir procesos y actividades para garantizar la calidad en la creación de código
* Describir procesos y actividades para garantizar la calidad en la elaboración de diagramas.

### 12.5.1.3 Responsables:

* Scrum Master de Calidad.
* Todos los miembros del grupo[62].

### 12.5.1.4 Recursos Requeridos:

* Microsoft Excel.

### 12.5.1.5 Puesta En Marcha:

* Dependiendo de cada entregable se establecerá un proceso para determinar la calidad, además se pueden incluir métricas que posteriormente deben ser estudiadas para la toma de decisiones en el proyecto.

Todos los documentos, códigos, diagramas y en general los entregables, contaran con revisión cruzada con el fin de garantizar un buen nivel en la calidad de los mismos. Las personas encargadas de realizar la revisión cruzada se establecerán al inicio de cada semana por medio del siguiente formato [revisión cruzada de la semana](file:///C:\Users\Estudiante\Downloads\Formatos\Revisión%20cruzada%20de%20la%20semana.xlsx).

#### 12.5.1.5.1 Proceso de revisión de documentos:

La revisión de los documentos se debe hacer siguiendo las actividades:

* Revisar si el documento cumple con las plantillas establecidas.
* Revisar ortografía.
* Revisar la coherencia.
* Revisar concisión [63].
* Revisar la estructura de las referencias.
* Revisar las fuentes bibliográficas.
* Hacer una retroalimentación [64].
* Registrar errores en la plantilla.
* Corregir errores.
* Aprobar el documento.

Si el documento es SPMP, SRS o SDD se evaluara con las plantillas que se encuentran en uvirtual, si es un entregable que se maneja al interior del grupo se deben seguir los formatos: [formato para documentos](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20-%20Realizar%20Documento.docx), [formato para tablas](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20-%20Tablas.docx), [formato para figuras](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20-%20Figuras.docx).

Los resultados de la ejecución de las anteriores actividades se deben almacenar en el [control de versiones para documentos](file:///C:\Users\Estudiante\Downloads\Formatos\Control%20de%20versiones%20para%20documentos.xlsx) y el [formato para la medición de documentos](file:///C:\Users\Estudiante\Downloads\Formatos\Formato%20para%20la%20medición%20de%20documentos.xlsx), tal como se había estipulado en la [sección 12.4.4.1](#_12.4.4.1_Medición_de).

Las actividades anteriormente mencionadas se encuentran interrelacionadas y se deben realizar de acuerdo al siguiente diagrama BPMN:

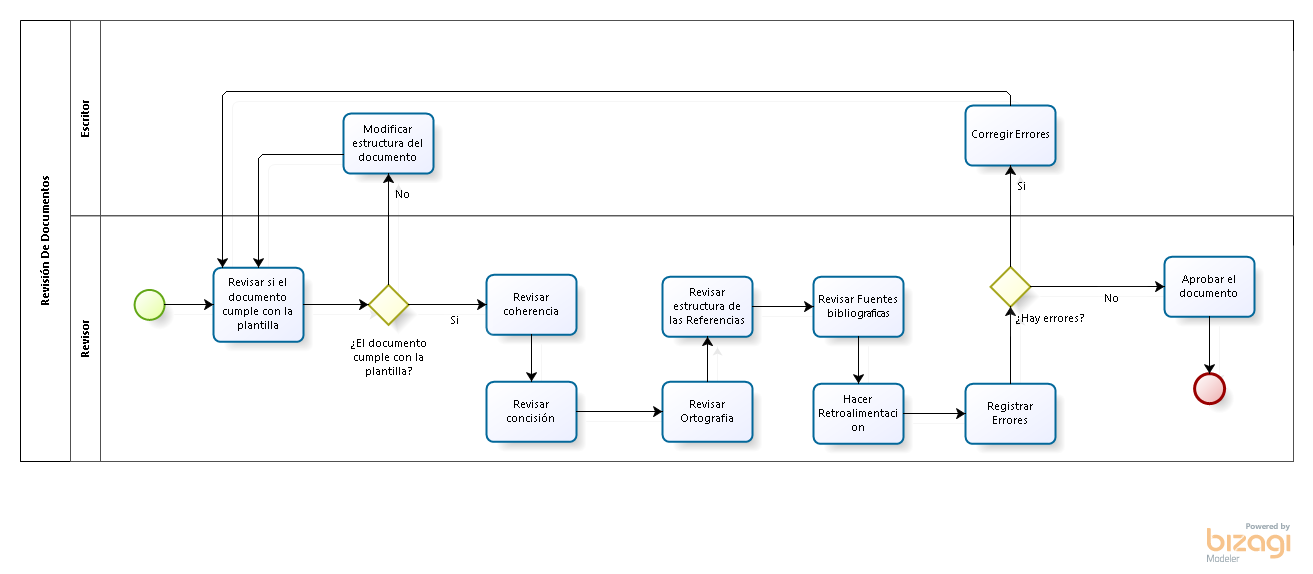


Ilustración : Proceso De Revisión De Documentos

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

En la ilustración se puede observar el proceso que se debe llevar a cabo al momento de realizar la revisión de un documento. Es necesario hacer correcciones hasta que la persona encargada de revisar el entregable lo apruebe. Además se debe registrar el número de errores con el fin de medir a la persona que realizó el archivo y poder determinar si hay una mejora en la elaboración de escritos posteriores. En este caso la persona encargada de revisar el documento es la persona asignada para realizar la revisión cruzada.

#### 12.5.1.5.2 Proceso de revisión del código.

La revisión del código se debe hacer siguiendo las actividades:

* Revisar el diseño del sistema.
* Revisar la documentación del programa.
* Revisar la coherencia entre el diseño y el código.
* Verificar que se siguieron buenas prácticas de programación.
* Crear casos de prueba.
* Probar la aplicación.
* Registrar de errores en la plantilla.
* Corregir Errores.
* Aprobar el código.

Además de lo anterior, la revisión de la calidad del código se hará en base a lo descrito en la [sección 12.4.4.2](#_12.4.4.2_Medición_de)

Las actividades anteriormente mencionadas se encuentran interrelacionadas y se realizaran de acuerdo al siguiente diagrama BPMN:

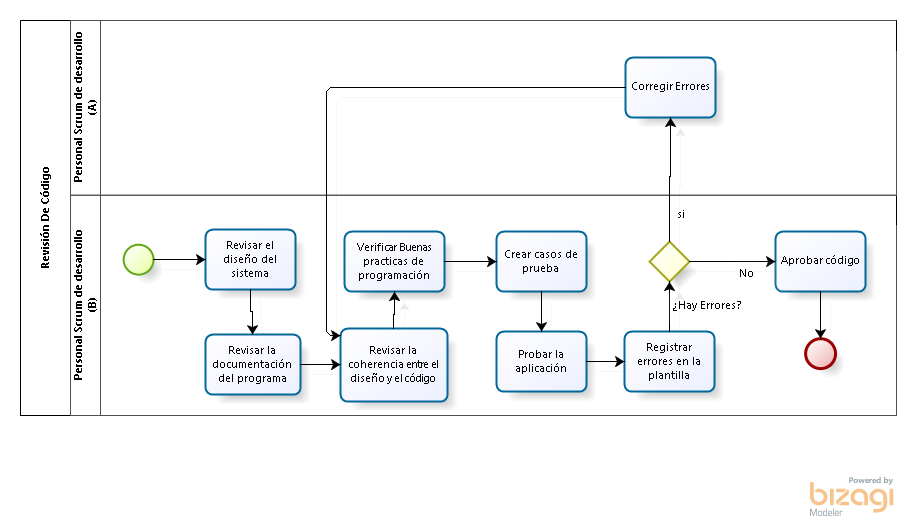


Ilustración : Proceso De Revisión De Código

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

En la ilustración podemos observar el proceso que se debe ejecutar con el fin de revisar el código. Este proceso se debe llevar a cabo por el personal del área de desarrollo, la persona A es quien desarrolló el código y la persona B es la persona encargada de realizar pruebas. La persona B es responsable de ejecutar las actividades descritas anteriormente y sugerir correcciones a los desarrolladores, el código debe ser corregido hasta que esté aprobado. El Scrum Master de desarrollo es el encargado de administrar el personal y escoger a las personas que van a desarrollar el código y a las que van a realizarle pruebas.

#### 12.5.1.5.3 Proceso de revisión de la evolución del proyecto

La revisión de la evolución del proyecto se debe hacer siguiendo las actividades:

* Crear un calendario.
* Realizar reuniones al interior del grupo.
* Asignar tareas y responsabilidades.
* Añadir tareas al calendario.
* Hacer seguimiento a las tareas por medio de un calendario.
* Reformular tiempos y asignación de tareas.
* Aprobar calendarización.

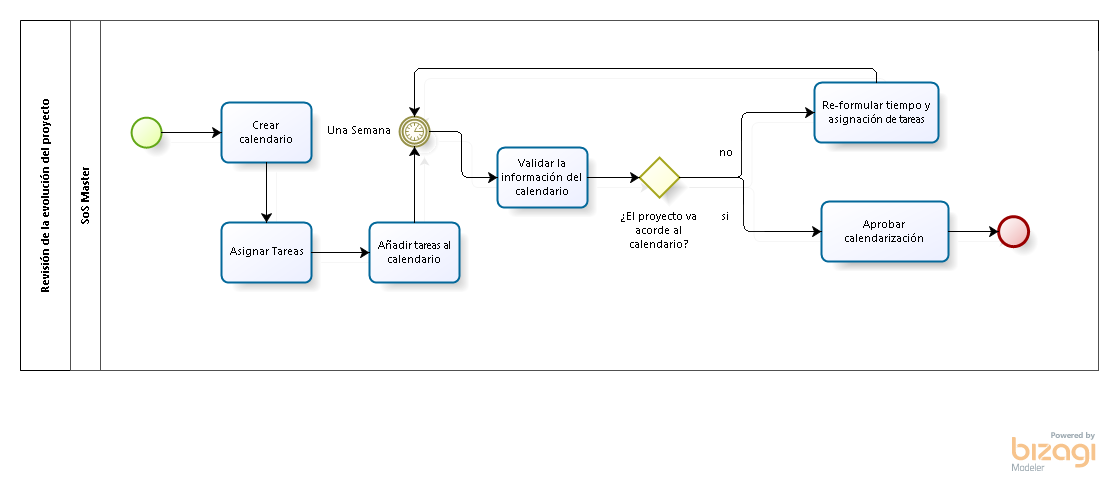


Ilustración : Proceso De Revisión De La Evolución Del proyecto

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

Las actividades del proyecto las puede encontrar en el [Diagrama De Gantt](file:///C:\Users\Estudiante\Downloads\Anexos\Diagrama%20de%20Gantt-Completo.pdf).

La evolución del proyecto se mide a partir de lo planeado y lo ejecutado. La planecación puede afectar el éxito del proyecto. La ilustración número tres nos muestra el proceso que se debe llevar a cabo para revisar la evolución del proyecto, este proceso está enfocado a determinar los cambios necesarios que se deben hacer en el calendario con el fin de cumplir con los tiempos de entrega estipulados. El responsable de ejecutar este proceso es el SoS Master.

#### 12.5.1.5.4 Proceso de revisión de los requerimientos

La revisión de los requerimientos se debe hacer siguiendo las actividades:

* La revisión de la evolución de los requerimientos se debe hacer siguiendo las actividades:
* Revisar que los requerimientos sean atómicos.
* Revisar que no haya ambigüedades [65].
* Revisar se hayan identificado los casos de uso relacionados con cada requerimiento.
* Revisar la especificación y documentación de los requerimientos [66].
* Verificar los atributos de calidad asociados a cada requerimiento.
* Registrar errores y dar retroalimentación.

Aprobar requerimientos.

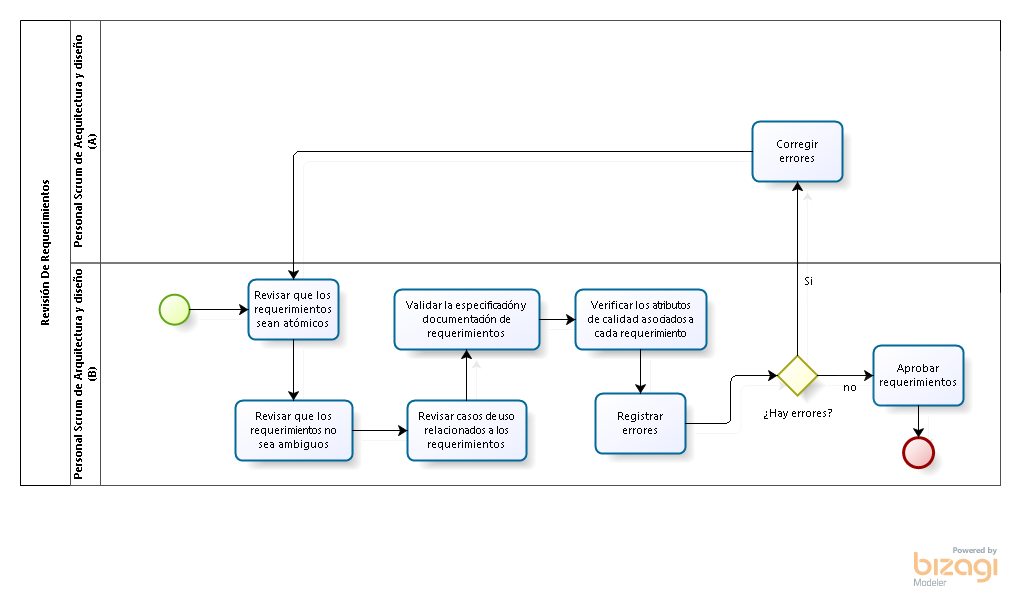


Ilustración : Proceso De Revisión De Requerimientos

**Nota:** El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

La evaluacion de la calidad de los requerimientos, se realizara en base a lo planteado en la [seccion 12.4.4.3.](#_12.4.4.3_Medición_de)

La anterior ilustración nos muestra las actividades que se deben desarrollar con el propósito de hacer una revisión de los requerimientos. Este proceso está orientado a revisar el análisis que se ha hecho con los requerimientos. El proceso debe desarrollarse por el personal del Scrum de Arquitectura y diseño, la persona A actúa similar a un analista o arquitecto, la persona B es quien revisa los requerimientos.

#### 12.5.1.5.5 Proceso de revisión de la aplicación:

* En el proceso de verificar la calidad de la aplicación muchas personas están involucradas y deben aprobarla antes de que llegue a manos del cliente, la siguiente imagen muestra de manera general el proceso que se lleva a cabo para revisar la aplicación.
* La aplicación debe ser examinada teniendo en cuenta los criterios de aceptación que se fijaron con el cliente en el cliente.
* La aplicación se evalúa gradualmente y al final se debe hacer una auditoria general.

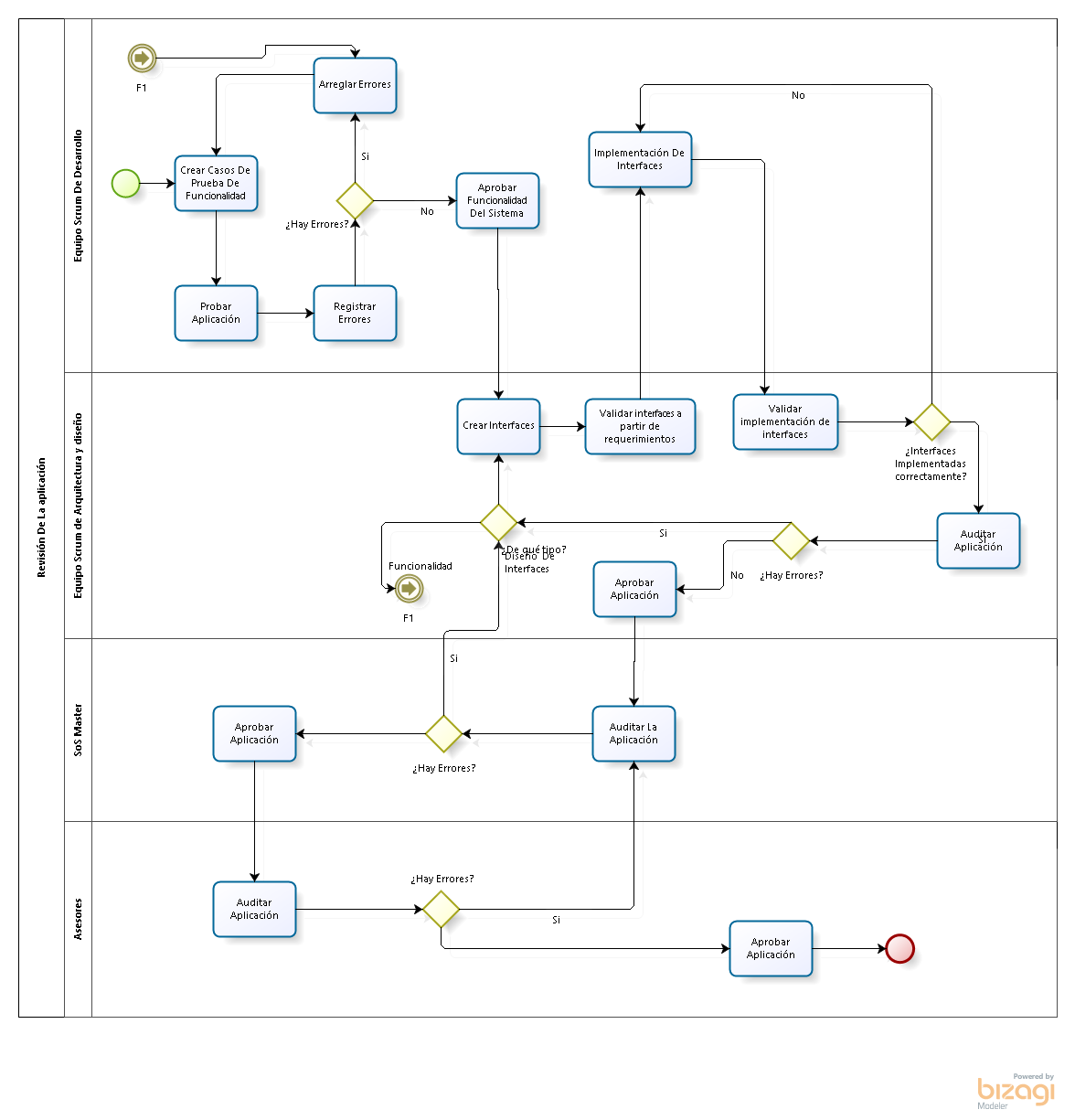


Ilustración : proceso de revisión de la aplicación

Nota: El anterior diagrama fue realizado por el grupo RAWR en la herramienta Bizagi Modeler siguiendo las buenas practicas que se describen en sus documentos[12].

Como se puede apreciar en la figura anterior el proceso de revisión de la aplicación es transversal a todas las partes del grupo, entre las cosas que se deben validar encontramos código, funcionalidades, interfaces, requerimientos y criterios de aceptación, por lo que la anterior imagen sólo muestra un esquema generalizado cuyo propósito es exponer los actores que intervienen en el proceso de auditar la aplicación y algunas de las actividades que se deben realizar. Cada actividad de aprobación se debe hacer teniendo en cuenta requerimientos funcionales, no funcionales y de calidad.

#### 12.5.1.5.6 Datos de los procesos:

La siguiente tabla muestra información importante de cada uno de los procesos definidos anteriormente.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Nombre Del Proceso | Momentos del proyecto en que ocurren | Responsables | Herramientas | Entradas | Salidas |
| Revisión De Documentos | Se realiza cada vez que se deba generar un documento, por lo tanto es proceso transversal en todo el proyecto. | En cada sprint se asignan parejas para la revisión de los documentos.  Todos los miembros del equipo. | Plantilla elaborada al interior del grupo. | Documento. | Documento válido. |
| Revisión De Código | Se realiza en cada iteración en la fase de desarrollo y pruebas. | Equipo del área de Desarrollo. | Pruebas Unitarias [67].  JUnit.  Javadoc. | Código. | Código valido. |
| Revisión De La Evolución Del Proyecto | Se realiza en cada iteración, fase, sprint. | SoS master y todos los miembros del equipo. | Actas.  Diagrama de Gantt.  Microsoft Project. | Calendario actual. | Calendario válido. |
| Revisión De Requerimientos | Se realiza principalmente en las etapas de requerimientos, análisis y diseño. | Equipo de arquitectura y desarrollo. | Herramientas Case[68]. | Requerimientos. | Requerimientos válidos. |
| Revisión De La Aplicación | Se realizan revisiones gradualmente y al finalizar se realiza una revisión general. | Participan todos los miembros del equipo y el cliente. | Pruebas Unitarias [67].  Plan de aceptación del producto. | Aplicación. | Aplicación final. |

Tabla 10: Información De Los Procesos

**Nota:** la anterior tabla fue realizada por el grupo RAWR

Referencias

[1] T. software development company Rational, “Rational Unified Process: Best Practices for Software Development Teams.” .

[2] R. Velasco P and R. Ruiz Méndez, “¿Cómo redactar propósitos?” .

[3] Institute of Electrical and Electronics Engineers, International Electrotechnical Commission, and International Organization for Standardization, *Systems and software engineering life cycle processes -- project management = Ingénierie du logiciel: processus de cycle de vie -- gestion de projet.* New York: Institute of Electrical and Electronics Engineers, 2009.

[4] International Organization for Standardization, International Electrotechnical Commission, Institute of Electrical and Electronics Engineers, and IEEE-SA Standards Board, *Systems and software engineering life cycle processes: requirements engineering = Ingénierie des systèmes et du logiciel : processus de cycle de vie : ingénierie des exigences.* Geneva; New York: ISO : IEC ; Institute of Electrical and Electronics Engineers, 2011.

[5] International Organization for Standardization, International Electrotechnical Commission, Institute of Electrical and Electronics Engineers, and IEEE-SA Standards Board, *Systems and software engineering architecture description = Ingeniérie des systèmes et des logiciels: description de l’architecture.* Geneva; New York: ISO : IEC ; Institute of Electrical and Electronics Engineers, 2011.

[6] IEEE, “IEEE Standard for Software Configuration Management Plans.” IEEE Computer Society, 1998.

[7] B. Bruegge and A. H. Dutoit, *Ingeniería de software orientado a objetos*. México: Pearson Education, 2002.

[8] J. A. Osorio, M. R. V. Chaudron, and W. Heijstek, “Moving from Waterfall to Iterative Development: An Empirical Evaluation of Advantages, Disadvantages and Risks of RUP,” 2011, pp. 453–460.

[9] H. Guang-yong, “Study and practice of import Scrum agile software development,” 2011, pp. 217–220.

[10] Fan Guo, Bainan Xia, and Fei Xue, “Analysis on software processes and enhancement for RUP,” 2011, pp. 295–298.

[11] “Rational Unified Process Best Practices for Software Development Teams.” Rational, IBM, 11/01.

[12] Bizagi, “Bizagi BPMN 2.0.” 2014.

[13] M. Paasivaara, C. Lassenius, and V. T. Heikkilä, “Inter-team coordination in large-scale globally distributed scrum: do scrum-of-scrums really work?,” 2012, p. 235.

[14] M. Torres, J. Rodrigues, and J. Pavlich-Mariscal, “Modelos de Proceso de Software. Modelos de Ciclos de Vida del Software.” 2015.

[15] D. Phillips, *The software project manager’s handbook: principles that work at work*, 2nd ed. Los Alamitos, CA: IEEE Computer Society, 2004.

[16] I. Sommervile, *Software Engineering*, 9th ed. Pearson Addison-Wesley, 2011.

[17] D. Firesmith, “Using V Models for Testing.” Software Engineering Institute.

[18] G. Rodriguez, A. Soria, and M. Campo, “Supporting Virtual Meetings in Distributed Scrum Teams,” *IEEE Lat. Am. Trans.*, vol. 10, no. 6, pp. 2316–2323, Dec. 2012.

[19] Ken Schwaber and Jeff Sutherland, “La Guía de Scrum,” Jul-2013. [Online]. Available: http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-ES.pdf#zoom=100. [Accessed: 27-Feb-2015].

[20] F. Leandro, “Scrum of Scrums: Running Agile on Large Projects.” 05-Jun-2013.

[21] J. Kivi, D. Haydon, J. Hayes, R. Schneider, and G. Succi, “Extreme programming: a university team design experience,” 2000, vol. 2, pp. 816–820.

[22] “Android Developers.” [Online]. Available: http://developer.android.com/index.html. [Accessed: 08-Mar-2015].

[23] S. Steuer, *The Adobe Illustrator CS2 wow! book tips, tricks and techniques from 100 top illustrator artists*. Adobe Press, 2006.

[24] A. Faulkner, *Adobe Photoshop CS3 classroom in a book; the official training workbook from Adobe Systems*. AdobePress, 2007.

[25] K. Tallon, *Ilustración digital de moda con Illustrator y Photoshop*, 1st ed. Parramón Ediciones, 2008.

[26] J. Loeliger and M. McCullough, *Version Control with Git*, 2nd ed. O’REILLY, 2012.

[27] “Git.” [Online]. Available: http://git-scm.com/. [Accessed: 08-Mar-2015].

[28] P. Tahchiev, F. Leme, V. Massol, and G. Gregory, *JUnit in Action*. Manning Publications, 2010.

[29] S. Küng, L. Onken, and S. Large, “TortoiseSVN A Subversion client for Windows Version 1.8.” 2013.

[30] C. M. Pilato, B. Collins-Sussman, and B. W. Fitzpatrick, *Version Control with Subversion*, 2nd ed. O’Reilly, 2008.

[31] “Monotone vs Git | Version Control Systems | Hammer Principle.” [Online]. Available: http://hammerprinciple.com/versioncontrol/items/monotone/git. [Accessed: 08-Mar-2015].

[32] B. O’Sullivan, *Mercurial: The Definitive Guide*, 1st ed. O’Reilly, 2009.

[33] R. St. John, “Android Studio vs. Eclipse: What You Need To Know,” *airpair*. .

[34] DigitalOceanDigitalOcean, Copyright © 2015, DigitalOcean TM Inc., and DigitalOceanDigitalOcean, “DigitalOcean Features | SSD Cloud Server, VPS Server, Simple Cloud Hosting.” [Online]. Available: https://www.digitalocean.com/features/technology/. [Accessed: 08-Mar-2015].

[35] R.-A. Collaris and E. Dekker, *ScrumUP A Visual Guide to Agile IT Improvement Using Scrum, XP and RUP*, Draft Edition. 2012.

[36] C. Ebert, R. Dumke, and A. Schmietendorf, *Best Practices in Software Measurement*. Springer, 2005.

[37] B. Agarwal, S. Tayal, and M. Gupta, *Software Engineering & Testing*. Jones and Barlett, 2010.

[38] “Departamento de Ingeniería de Sistemas | Pontificia Universidad Javeriana.” [Online]. Available: http://ingenieria.javeriana.edu.co/facultad/departamentos/profesores/ingenieria-sistemas. [Accessed: 07-Mar-2015].

[39] Sutherland J, “Future of scrum: parallel pipelining of sprints in complex projects,” presented at the Agile Conference, USA, 2005, pp. 90–99.

[40] Gaia Education, “Tres pasos en el proceso de toma de decisiones por consenso.” .

[41] B. Brett, “Software measurement tool.” [Online]. Available: http://www.brettboschma.com/portfolio/measurement.html. [Accessed: 08-Mar-2015].

[42] R. Pressman, *Ingeniería del Software un enfoque práctico*, 7th ed. McGraw Hill, 2010.

[43] R. Heller, “An Introduction to Function Point Analysis.” [Online]. Available: http://www.qpmg.com/fp-intro.htm. [Accessed: 08-Mar-2015].

[44] D. Longstreet, “Fundamentals of Function Point Analysis,” *Softw. Dev. Mag.*

[45] “Function Points project estimation.” [Online]. Available: http://www.functionpoints.org/. [Accessed: 08-Mar-2015].

[46] D. Longstreet and Longstreet Consulting Inc, “Function Points Analysis Training Course.” SoftwareMetrics.com.

[47] Durasiewicz S., Paasiraava M, and Lassenius C, “Distributed Agile Development: Using Scrum in a Large Project,” Bangalore, 2008, pp. 87–95.

[48] NASA, “NASA Systems Engineering Handbook.” 2007.

[49] Boehm Barry, “Software risk managment: principles and practices,” *Softw. IEEE*, vol. 8, no. 1, pp. 32–41, Jun. 1991.

[50] Zardari S., “Software Risk Management,” presented at the Information Management and Engineering, Kuala Lumpur, 2009, pp. 375–379.

[51] Tharwon Arnuphaptrairong, “Top Ten Lists of Software Project Risks : Evidence from the Literature Survey,” presented at the International Multiconference of engineers and computer scientist, Hong Kong, 2001, vol. 1.

[52] James Helm, “Software Risk Checklist - Taxonomy.” .

[53] Roger Pressman, “Software Engineering,” in *Software Engineering, a practitioner approach*, 7th ed., McGraw Hill, pp. 744–760.

[54] Avdoshin S., “Software Risk Management,” presented at the Software Engineering Conference in Russia, Moscow, 2011, pp. 1–6.

[55] A. Abran and J. W. Moore, *Software engineering body of knowledge SWEBOK*, vol. 1, 1 vols. IEEE Computer Society, 2004.

[56] R. Kennedy, “Software Development Best Practices: The 7 Attributes of a Good Software Configuration Management System,” presented at the Software Development Best Practices, 2006.

[57] IEEE, “IEEE Guide to Software Configuration Management.” IEEE Computer Society, 1987.

[58] Terasoft, “Software Project Management Plan (SPMP) for Nirvana National Bank ATM Software Project.” .

[59] Dharma Consulting, “Gestión de proyectos.” 2015.

[60] A. M. Jonassen Hass, *Configuration management principles and practice*. Pearson Education, 2003.

[61] EcuRed, “Control de cambios.” .

[62] Roger Pressman, “Software Engineering,” in *Software Engineering, a practitioner approach*, 7th ed., McGraw Hill, pp. 744–760.

[63] Blando, “Pull Request and Code Review in GitHub’s Shared Repository Model.” 2012.

[64] Project Management Institute, Inc., *FUNDAMENTOS PARA LA DIRECCIÓN DE PROYECTOS(GUÍA DEL PMBOK®)*, Cuarta edición. EE.UU, 2008.

[65] Centro de Escritura Javeriano, “Durante la escritura, ‘la escritura concisa,’” p. 4.

[66] Silvia Senz, “Procesos de control de calidad del texto,” p. 16, Diciembre De-2005.

[67] Jose Hernando Bahamon, “CONTROL DE CALIDAD EN EL SOFTWARE,” ICESI, 10.

[68] Sommerville, “Procesos de la ingenieria de requerimientos,” in *Ingenieria de software*, .

[69] IEEE Computer Society, Software Engineering Technical Committee, American National Standards Institute, IEEE Standards Board, and Institute of Electrical and Electronics Engineers, *IEEE standard for software unit testing*. New York, N.Y.: Institute of Electrical and Electronics Engineers, 1986.

[70] Andrea Alarcon, “Herramientas CASE para ingenieria de Requisitos.” Cultura Cientifica JDC, 2008.

1. Estas están sujetas a cambios durante el transcurso del proyecto. [↑](#footnote-ref-1)
2. IDE diseñado para manejar de forma sobresaliente desarrollo *Enterprise,* web y móvil (https://www.jetbrains.com/idea/) [↑](#footnote-ref-2)
3. Distribuido se refiere en este caso a que se hace una copia de todo el repositorio en la máquina del usuario, así cada usuario tiene un back-up completo del servidor principal [↑](#footnote-ref-3)
4. Mock se refiere a objetos y métodos simulados para crear un ambiente en donde se pueda probar de forma aislada algún componente de código [↑](#footnote-ref-4)
5. Más información en [lenguajes y herramientas](file:///C:\Users\Fabián%20Merchán\Google%20Drive\Proyecto%20Ingesoft\Primera%20entrega\SPMP\8.2%20Lenguajes%20y%20Herramientas%20V.0.5.docx) [↑](#footnote-ref-5)
6. Si es herramienta de código debe dirigirse con el área de desarrollo, de lo contrario se debe ser analizada por el área de arquitectura y diseño. [↑](#footnote-ref-6)
7. Esto es que falle el servidor que aloja el recurso, o no se consiga de nuevo el recurso de entrenamiento [↑](#footnote-ref-7)
8. www.facebook.com [↑](#footnote-ref-8)
9. Más información en [lenguajes y herramientas](#8.2.3.2 Organización). [↑](#footnote-ref-9)
10. No es necesaria una evidencia escrita de la petición de cambio. Puede ser verbal. [↑](#footnote-ref-10)
11. Si lo hay (Opcional). [↑](#footnote-ref-11)
12. Se conoce en *GitHub* como *pull request* [↑](#footnote-ref-12)
13. Más información del uso de *GitHub* en la sección de [entrenamiento del personal](file:///C:\Users\Estudiante\Downloads\9.2.1%20Plan%20de%20entrenamiento%20del%20personal%20%5bV0%5d.docx). [↑](#footnote-ref-13)
14. La reversión de cambios depende de la herramienta utilizada, puede encontrar información útil y tutoriales en la sección de [entrenamiento del personal](file:///C:\Users\Estudiante\Downloads\9.2.1%20Plan%20de%20entrenamiento%20del%20personal%20%5bV0%5d.docx). [↑](#footnote-ref-14)