**DESARROLLO DE UN MARCO DE TRABAJO ORIENTADO A PROCESOS DEVOPS SIGUIENDO EL ESTÁNDAR ISO/IEC 29110 PARA EL CASO DE ESTUDIO DE INTEGRACIÓN DE PROCESOS DEL PROYECTO SANAMBIENTE.**

AARÓN LEVI GRAJALES GÓMEZ

CRISTIAN FERNANDO BALANTA PAZÚ

PROYECTO DE GRADO PARA OBTENER EL TÍTULO DE INGENIERO DE SISTEMAS

Asesor(a): Beatriz Eugenia Marín, Ingeniera de sistemas

INSTITUCIÓN UNIVERSITARIA ANTONIO JOSÉ CAMACHO

FACULTAD DE INGENIERÍAS

INGENIERÍA DE SISTEMAS

ANTEPROYECTO

SANTIAGO DE CALI

2019

**Resumen**

El presente proyecto consiste en desarrollar un marco de trabajo donde los procesos están orientados a la ISO/IEC 29110, en la cual se tomará como base para llevar a cabo el desarrollo del este proyecto que nos permite tener un monitoreo continuo al desarrollo del proyecto, también se tomará en cuenta la metodología ICONIX que ayudará a complementar la ISO/IEC 29110 en relación a los entregables que se deben dar al usuario que los solicita, esto permitirá que los módulos de trabajo del proyecto (calidad y desarrollo) tengan un orden y estén al tanto de aquellas tareas que deben realizar y entregar, y de esta misma forma proporciona la integración de ellos debido que hay tareas que son complementos de otras y requiere la comunicación entre ellos para poder realizarse.

**Abstract**

The present project consists of developing a working framework where the processes are oriented to the ISO/IEC 29110, in which it will be taken as base to carry out the development of this project that allows us to have a continuous monitoring to the development of the project, also it will be taken into account the ICONIX methodology that will help to complement the ISO/IEC 29110 in relation to the deliverables that must be given to the user that requests them, this will allow the work modules of the project (quality and development) to have an order and be aware of those tasks they must perform and deliver, and in this same way provides the integration of them because there are tasks that are complementary to others and requires communication between them to be able to perform.

**Tabla de contenido**

[Introducción 1](#_Toc12704693)

[1. Problema de investigación 2](#_Toc12704694)

[1.1 Planteamiento del problema 2](#_Toc12704695)

[1.2 Formulación del problema 5](#_Toc12704696)

[1.3 Sistematización del problema 5](#_Toc12704697)

[2. Objetivos 6](#_Toc12704698)

[2.1 Objetivo general 6](#_Toc12704699)

[2.2 Objetivos específicos 6](#_Toc12704700)

[3. Justificación 7](#_Toc12704701)

[4. Marco de referencia 8](#_Toc12704702)

[4.1 Marco histórico o antecedentes 8](#_Toc12704703)

[4.2 Marco teórico 10](#_Toc12704704)

[4.2.1 Estándar ISO/IEC 29110 10](#_Toc12704705)

[4.2.2 DevOps 20](#_Toc12704706)

[4.2.3 Metodología ICONIX 26](#_Toc12704707)

[4.3 Marco conceptual 35](#_Toc12704708)

[4.3.1 Proceso de software 35](#_Toc12704709)

[4.3.2 Metodología de desarrollo de software 36](#_Toc12704710)

[4.3.3 UML (Unified Modeling Language) 38](#_Toc12704711)

[4.3.4 Calidad de software 39](#_Toc12704712)

[4.3.5 Modelo de calidad de software 40](#_Toc12704713)

[4.4 Marco legal 41](#_Toc12704714)

[5. Desarrollo 42](#_Toc12704715)

[6. Conclusiones 43](#_Toc12704716)

[7. Referencias 44](#_Toc12704717)

**Lista de tablas**

[Tabla 1. Diagnóstico y pronóstico del proyecto 4](#_Toc12704690)

[Tabla 2. Productos o artefactos según la ISO/IEC 29110 17](#_Toc12704691)

[Tabla 3. Diagramas de UML por categoría. 39](#_Toc12704692)

**Lista de figuras**

[Figura 1. Diagrama Causa y efecto 5](#_Toc12704675)

[Figura 2. Público objetivo ISO/IEC 29110 11](#_Toc12704676)

[Figura 3. Procesos básicos de la guía de perfiles. 13](#_Toc12704677)

[Figura 4. Diagrama del proceso de gestión del proyecto. 14](#_Toc12704678)

[Figura 5. Diagrama del proceso de implementación de software. 16](#_Toc12704679)

[Figura 6. Definición de DevOps. 21](#_Toc12704680)

[Figura 7. Ejemplo de una estructura organizacional en la que se visualiza erróneamente la instauración de un departamento de tipo DevOps. 23](#_Toc12704681)

[Figura 8. DevOps no es una suite de herramientas. 24](#_Toc12704682)

[Figura 9. Arquitectura de referencia de DevOps. 26](#_Toc12704683)

[Figura 10. Proceso de ICONIX. 27](#_Toc12704684)

[Figura 11. Ubicación del modelado de dominio en el proceso de ICONIX. 29](#_Toc12704685)

[Figura 12. Ejemplo de diagrama de caso de uso. 30](#_Toc12704686)

[Figura 13. Objetos del diagrama de robustez. 31](#_Toc12704687)

[Figura 14. Ejemplo de un diagrama de robustez. 31](#_Toc12704688)

[Figura 15. Elementos de un diagrama de secuencia. 33](#_Toc12704689)

# Introducción

El siguiente trabajo describe el cómo implementar un marco de trabajo siguiendo estándares y metodologías que brinda procesos o pasos a seguir para poder desarrollar y gestionar todos los procesos de la elaboración de software. El estándar ISO/IEC 29110 es el que tomará como base para llevar a cabo el desarrollo de este proyecto, éste se divide en dos procesos, el primero es la gestión de proyectos y el segundo es la implementación de software. Éste estándar se enfoca en dar lineamientos para empresas pequeñas puesto que dichas empresas están empezando a surgir y no tienen una guía de apoyo para competir de manera equilibrada en el mercado con las empresas macro.

Una de las metodologías que ayudará a complementar la ISO/IEC es la “metodología ICONIX” ya que es un modelo de trabajo basado en procesos que permite una mayor velocidad en el desarrollo del proyecto, con la seguridad y solidez que da al utilizarlo de manera adecuada, esta metodología permite tener proyectos de alta calidad, en un tiempo corto.

Para poder usar estos modelos mencionados anteriormente, también hará complemento el movimiento DevOps, el cual tiene como función mejorar la agilidad en la prestación de servicios, fomentando una mayor comunicación y colaboración entre los equipos de desarrollo y operaciones, de manera que no se produzca una desarticulación al momento de llevar a cabo el desarrollo del proyecto.

# Problema de investigación

## Planteamiento del problema

Sanambiente S.A.S es una empresa especializada en ofrecer soluciones integrales en productos y/o servicios que aportan al cuidado y mejoramiento de las condiciones ambientales en un medio ambiente específico.

La estructura organizacional de la empresa se encuentra conformada brevemente de la siguiente manera: en el puesto más alto se encuentra la junta directiva y el gerente general, luego se encuentran las gerencias de ventas, estrategia, técnica y administrativa (ésta última posee las oficinas de tesorería, logística y contabilidad), seguidas cada una por su respectiva jefatura (ejemplo: la gerencia técnica posee jefe técnico, jefe de tecnología, coordinador de servicios integrales y coordinador de proyectos), y adicionalmente por un líder jurídico que poseen.

Los datos que maneja la empresa son de tipo ambiental, todos estos dependiendo del producto y/o servicio y la información específica que tiene como labor transferir de los datalogger de las estaciones concretas para brindar al cliente, entre los que se encuentran: datos de muestreo continuo de agua por diferentes metodologías, como son: muestreo continuo en intervalos de tiempo, muestreo continuo o simple y muestreo por eventos (caudal, calidad de agua, lluvia) ; medidas en forma instantánea de la humedad del suelo que se especifique, conductividad eléctrica/salinidad y temperatura; muestreo de aguas superficiales, agua potable, aguas residuales y no residuales con toma de muestra puntual, compuesta e integrada en cuerpo lótico (ecosistema de aguas fluyentes) y medición en sitio de caudal, pH, temperatura, conductividad, oxígeno disuelto; mediciones continuas de partículas PM10 o PM2.5 o PM10-2.5; medición del ruido ambiental (ruido que existe en un determinado lugar y en un momento dado); medición de la calidad del aire, particularmente, la medición de CO (monóxido de carbono) y O3 (ozono), pero éstas se llevan a cabo en un área delimitada, dependiendo en donde se encuentre ubicado el producto y/o servicio encargado de realizar esta labor; medición y transmisión de información meteorológica, nivel o hidrometeorológica gracias a la implementación de comunicación satelital GOES.

La empresa Sanambiente maneja actualmente un sistema que se encarga de realizar la transferencia de datos ambientales de los datalogger de las estaciones, esto es logrado gracias a los protocolos de comunicación que utilizan, que en este caso son FTP y Modbus.

Para realizar la transferencia de datos por el protocolo Modbus, el sistema manda una ID, el puerto, la IP y el mensaje solicitando los datos ambientales, de igual forma la estación donde se encuentra el datalogger responde con un mensaje al sistema, con los datos ambientales requeridos, luego, lo que hace el sistema es capturarlos y almacenarlos en la base de datos. Para realizar la transferencia de datos por el protocolo FTP, es un poco diferente, ya que inicialmente se debe configurar un pequeño servidor FTP que generalmente se configura con el Internet Information Services propio de Windows, en el cuál se encontrarán almacenados en una carpeta los datos ambientales transferidos del datalogger de la estación específica, luego, con otra aplicación llamada FTP Import/Export se leen estos datos de la carpeta, y la forma en que esta aplicación realiza la lectura es de acuerdo a una plantilla ya creada que está estructurada por canales (ej. SO2, NH3), luego de leerlos la aplicación procede a realizar el almacenamientos de los datos en la base de datos.

Para realizar el procesamiento de datos, el módulo de validación de datos ejerce la lectura de los criterios de validación por un umbral establecido para cada parámetro o dato ambiental, estructurados y generados por la aplicación llamada Setup. Este módulo, se encarga de ejecutar los criterios sobre los datos que se están transfiriendo del datalogger, generando validaciones de superación de umbral sobre cada uno de los datos, por ejemplo, si un dato (ej. PM10) ubicado en cierta casilla supera el número de umbral establecido, al lado de este dato se coloca una columna adicional llamada estado y le asigna el número 2 (2: dato inválido); si otro dato se encuentra en el umbral establecido y no lo ha superado, se le asigna el número 1 (1: dato válido).

El inconveniente que posee la empresa Sanambiente está relacionado con la desarticulación de aplicativos, que requiere intervención manual de los desarrolladores para que la ejecución de la transferencia y procesamiento de los datos ambientales se lleve a cabo sin problemas.

Para dar solución a este inconveniente se generará un aplicativo que integre los diferentes procesos de la gestión de datos. Para ello se han propuesto varios proyectos de grado que abordan diferentes perspectivas: calidad y desarrollo, pero se requiere un marco integrador de cada perspectiva del proyecto.

Tabla 1. Diagnóstico y pronóstico del proyecto

|  |  |  |  |
| --- | --- | --- | --- |
| **Síntoma** | **Causa** | **Efecto** | **Consecuencia** |
| Los módulos de calidad y desarrollo abordan el proyecto por separado. | No hay integración de los módulos. | Esto conlleva que el proyecto pierda la orientación. | Productos pequeños sin conexión. |
| Falta de comunicación de los módulos calidad y desarrollo. | Falta de coordinación con los integrantes de los módulos. | Que cada módulo realice sus objetivos por sus lados. | Falta de sincronización de todos los módulos y sus áreas de trabajos. |
| **Diagnóstico** | | **Pronóstico** | |

(Elaboración propia)

Imagen que contiene texto, captura de pantalla, mapa

Descripción generada automáticamente

Figura 1. Diagrama Causa y efecto

(Elaboración propia)

## Formulación del problema

¿Cómo integrar los módulos del proyecto de gestión de datos de Sanambiente para que cumplan con criterios de calidad y robustez hacia el futuro?

## Sistematización del problema

* ¿Cuáles elementos de la ISO/IEC 29110 se pueden ajustar a las características del proyecto Sanambiente?
* ¿Cuáles son las herramientas propuestas por la ISO/IEC 29110 que permitan la integración de los diferentes equipos de trabajo del proyecto Sanambiente?
* ¿Cómo asegurar que el proyecto Sanambiente cumple adecuadamente un marco de calidad?
* ¿Cómo asegurar que el marco propuesto cumple los estándares seleccionados para enmarcar el proyecto?

# Objetivos

## Objetivo general

Establecer un marco de trabajo orientado a procesos DevOps siguiendo los lineamientos dados por la ISO/IEC 29110 utilizando como caso de estudio la integración de procesos del proyecto Sanambiente.

## Objetivos específicos

* Identificar los procesos y medidas propuestas por la ISO/IEC 29110 que se adecúen a las características del gestor de datos ambientales de la empresa Sanambiente.
* Establecer la ruta de procesos y herramientas que permitan la integración de los módulos del proyecto de Sanambiente de acuerdo a la ISO/IEC 29110.
* Aplicar el marco de trabajo identificado en la ISO/IEC 29110 al proyecto Sanambiente.
* Evaluar los resultados de la aplicación del marco de trabajo siguiendo DevOps.

# Justificación

En el presente proyecto se realizará el aporte de un marco de trabajo para la integración de los módulos (calidad y desarrollo), el cual se llevará a cabo tomando como base la metodología DevOps, ya que brinda una comunicación eficiente entre los módulos, adicional a esto, la investigación otorgará unos conocimientos para ser empleados en el futuro, en un área de trabajo que requiera la aplicación de un marco con unos lineamientos a seguir para realizar la integración efectiva de todos los módulos implicados en un proyecto. Además, la integración de los módulos conllevará a tener una mejor comunicación donde el proceso de la realización del aplicativo tendrá un avance significativo ya que se cuenta con la participación de estos, donde los resultados que se obtendrán será el desarrollo de software, y un marco de trabajo que puede seguir como guía para otros proyectos.

El proyecto por realizar está vinculado particularmente al programa académico de Ingeniería de sistemas, por lo cual se afirma que, para llevarlo a cabo, se pondrá en práctica todo el conocimiento adquirido a lo largo del programa (ejemplo: análisis de requerimientos, programación, bases de datos, entre otros).

Este proyecto poseerá escalabilidad, esto quiere decir que seguirá en constante evolución a lo largo del tiempo, añadiendo nuevas funcionalidades y mejoras, sin degradar su desempeño

# Marco de referencia

## Marco histórico o antecedentes

Para el presente proyecto se analizaron diferentes proyectos que tuvieron el mismo o similar objeto de estudio que el presente proyecto tiene, estos fueron tomados como base para llevar a cabo el desarrollo de este proyecto, estos proyectos brindan una guía para saber cómo proceder y de qué forma iniciar la elaboración del proyecto, los pasos a seguir, cuáles pueden ser de utilidad para el presente proyecto y poderlos adecuar de acuerdo a las necesidades que éste tiene determinadas. La información que será mostrada en este marco es de aquellos proyectos que tienen relación con los temas que el presente proyecto aborda: ISO/IEC 29110, DevOps y Metodología ICONIX. El tipo de revisión que se hizo a estos proyectos es meramente enfocado en la parte donde tocan los temas mencionados anteriormente, qué uso hicieron de estos temas en sus proyectos, cómo lo ejecutaron y qué resultados les dieron. Para la realización de este marco se hizo el análisis y estudio de cinco documentos, tesis, concretamente; pero entre ellos, cuatro fueron los que más tenían mejor dominio del tema y mostraban de forma clara la aplicación de estos temas al detalle.

Según (Madruñero Padilla, 2018) cuyo proyecto denominado “Implementación del estándar ISO/IEC 29110 en el proceso de desarrollo de software de la dirección de desarrollo tecnológico e informático de la universidad técnica del norte”  este trabajo se manejó la ISO/IEC 29110 donde se llevó a cabo la gestión de procesos de desarrollo de software en la universidad técnica del norte, de la ciudad Ibarra-Ecuador, donde se buscaba la formalidad en todas las etapas del proceso de desarrollo mejorando la gestión del desarrollo permitiendo incorporar hitos de control en el proceso y su gestión para lograr una toma de decisiones oportuna en el proyecto.

Como resultado que se obtuvo en esa investigación fue la implementación del estándar ISO/IEC 29110 ya que pudo establecer una relación con la metodología SCRUM en la cual con llevo aportar a la institución una metodología para la elaboración del desarrollo de software.

En la tesis (Laporte, O´connor, & García, 2016) bajo el título general de “ THE IMPLEMENTATION OF ISO/IEC 29110 SOFTWARE ENGINEERING STANDARDS AND GUIDES IN VERY SMALL ENTITIES” se demuestra que el ISO/IEC 29110 tiene perfiles de ciclo de vida para empresas muy pequeñas (de desarrollo de software) que deben seguir para que puedan elaborar software con altos estándares de calidad, así mismo, este estándar permite que las empresas pequeñas puedan hacer gestión a sus proyectos que permite tener un seguimiento a los proceso que se realizan, dichos procesos ayudan al mejoramiento y el rendimiento de la empresa.

El resultado que se puede obtener por este proyecto es que las empresas de desarrollo de software pueden hacer software que cumplan con los estándares de calidad ya que la ISO/IEC 29110 ayuda a las pequeñas empresas poder desarrollar software de calidad, como resultado de la implementación de esta ISO a cualquier empresa pequeña contribuye a ser competitiva en el mercado nacional o internacional.

Según la tesis de (LEGARIA, 2018) con el título de “IMPLEMENTACION DE PROCESO ORGANIZACIONAL DE GESTION DE PROYECTOS EN DEVELOPIT” donde nos habla de cómo la ISO/IEC 29110 cumple con los proceso de la gestión de proyecto, DevelopIT como empresa necesita una norma que se adecue a la compañía para eso se utiliza la ISO/IEC 29110 ya que tiene menos procesos respecto a la gestión de proyectos, estos procesos que tiene la ISO/IEC 29110 se adaptara para los proyectos de infraestructura tecnológica que implementa DevelopIT esto permita a la organización pueda ejecutar proyectos considerados complejos y no complejos.

Se puede concluir que, para implementar un proyecto de gestión de proyectos en una pequeña empresa es necesario que esta sea adaptada a las necesidades de la organización, En ese sentido la norma ISO/IEC 29110 cumple con las características para ser adaptada a pequeñas organizaciones. También es posible que otras organizaciones con características similares a DevelopIT puedan hacer uso de este proceso de gestión de proyectos, ya que, al basarse en una norma, y al diseñarse con la intención de adaptarse a las características de pequeñas organizaciones, debiera ser por lo tanto útil y usable por organizaciones con similares características, es decir, pequeñas organizaciones, que no tienen personal especializado en gestión de proyectos, sus recursos son escasos y que los roles de los integrantes son multitarea.

De acuerdo (PORRAS, 2019) en el estudio “METODOLOGÍA ÁGIL ICONIX EN LA CALIDAD DEL PRODUCTO SOFTWARE, LIMA, 2017” tiene como objetivo de que la metodología ágil iconix brinda proceso para el desarrollo de software mediante la aplicación de técnicas probadas y aceptadas de ingeniería de software para empresas micro, pequeñas y medianas esto permitirá que este tipo de empresas tenga un mejoramiento de desarrollo de software, además puede tener productos aceptables y sostenible.

El resultado que se obtuvo de este proyecto fue la implementación de la metodología ICONIX que ayudo mejorar la calidad del desarrollo del software que se estaba buscando, gracias a esta metodología que brinda pasos a seguir donde se enfoca en cómo se debe desarrollar un sistema, Este proyecto ayudara a la empresa que brinda el servicio de desarrollo de software nuevas formas de hacer aplicaciones con agilidad que tiene beneficios como: disminuir el tiempo y los costos.

## Marco teórico

### Estándar ISO/IEC 29110

Es una norma ISO bajo el título general *Ingeniería de Software – Perfiles del ciclo de vida para entidades muy pequeñas* (Very Small Enterprises (VSEs)). La norma se basa en subconjuntos de elementos normativos apropiados, conocidos como perfiles VSE. El propósito de los perfiles VSE es definir un subconjunto de normas internacionales para el contexto de VSE (ISO/IEC, 2011, p. vi).

La norma ha sido desarrollada para mejorar la calidad de los productos y/o servicios y el rendimiento de los procesos en pequeñas entidades que en el momento se encuentran aisladas de algunas actividades económicas y de este modo contar con los mismos niveles de competitividad en el mercado que las grandes entidades. La ISO/IEC 29110 está dirigida al público objetivo que se encuentra divido en 5 partes (ver figura 2):



Figura 2. Público objetivo ISO/IEC 29110[[1]](#footnote-1)

En la parte 1 se definen los términos que serán usados a lo largo del conjunto de documentos que comprende la ISO/IEC 29110. Introduce conceptos de procesos, ciclo de vida y estandarización; igualmente introduce las características y requisitos que debe poseer una VSE para implementar el estándar.

En la parte 2 se introducen los conceptos para el perfil estandarizado de software para VSEs, además de los términos que serán usados a lo largo del conjunto de documentos que comprende el estándar. Los elementos comunes a todos los perfiles estandarizados como la estructura, conformidad y evaluación, igualmente son especificados en esta parte. Finalmente introduce la taxonomía de los perfiles del estándar.

En la parte 3 se definen las directrices de evaluación del proceso y los requisitos mínimos de cumplimiento que deben seguir las VSEs para satisfacer el objetivo de los perfiles VSE definidos en el estándar. Además, contiene información útil para los desarrolladores de métodos y herramientas de evaluación. Esta parte está dirigida a las personas que tiene relación directa con el proceso de evaluación.

En la parte 4 se proporciona las especificaciones para todos los perfiles del grupo de perfiles genéricos. Este grupo es aplicable a las VSEs que no desarrollan productos de software críticos. Así mismo los perfiles de las VSEs se aplican y son dirigidos a los autores/proveedores de guías, herramientas u otros materiales de apoyo.

En la parte 5 se proporciona una guía de gestión e ingeniería de implementación para el perfil básico del grupo de perfiles genéricos descrito en la parte 4 (Ramos & Mendoza, 2014, p. 23).

La norma proporciona procesos de gestión de proyectos e implementación de software, las prácticas que integran están basadas en la selección de la ISO/IEC 12207:2008, bajo el título general *Sistemas e ingeniería de software – Procesos del ciclo de vida del software* e ISO/IEC 15289:2006, bajo el título general *Sistemas e ingeniería de software – Contenido de los sistemas y productos de información del proceso del ciclo de vida del software (Documentación)* elementos estándares (ISO/IEC, 2011, p. 3)

De acuerdo con la norma ISO/IEC 29110 se definen dos procesos: Gestión del proyecto (Project Management) e Implementación de software (Software Implementation), los cuales están interrelacionados (ver figura 2) (Matla Cruz, 2014, p. 31).



Figura 3. Procesos básicos de la guía de perfiles[[2]](#footnote-2).

A continuación, se realizará una breve descripción de cada uno de los procesos y las actividades que conllevan.

* Proceso de gestión del proyecto (PM):

Tiene como propósito establecer y llevar a cabo de manera sistemática las *Tareas* del proyecto de implementación de software, lo que permite cumplir con los objetivos del proyecto en la calidad, tiempo y costos esperados. El proceso de gestión de proyectos tiene las siguientes actividades:

* + Planificación del proyecto.
  + Ejecución del proyecto.
  + Evaluación y control del proyecto.
  + Cierre del proyecto.

El siguiente diagrama muestra de manera clara el flujo de información entre las actividades del proceso de gestión del proyecto, incluyendo además los productos de trabajo más relevantes y cómo están relacionados (ver figura 4).



Figura 4. Diagrama del proceso de gestión del proyecto[[3]](#footnote-3).

* Proceso de implementación de software (SI):

Tiene como propósito la realización sistemática de las actividades de análisis, construcción, integración y pruebas de productos de software nuevos o modificados de acuerdo con los requisitos especificados (ISO/IEC, 2011, p. 16). El proceso de implementación de software tiene las siguientes actividades:

* Inicio de la implementación del software.
* Análisis de los requerimientos de software.
* Diseño arquitectónico y detallado de programas informáticos.
* Construcción de software.
* Integración de software y pruebas.
* Entrega de productos.

El siguiente diagrama muestra de manera clara el flujo de información entre las actividades del proceso de implementación de software, incluyendo además los productos de trabajo más relevantes y cómo están relacionados (ver figura 5).



Figura 5. Diagrama del proceso de implementación de software[[4]](#footnote-4).

* Productos o artefactos:

La ISO/IEC 29110 define unos productos o artefactos obligatorios para el perfil básico. En la siguiente tabla se listan los productos o artefactos en orden alfabético, sus descripciones y la fuente de donde provienen (ver tabla 4).

Tabla 2. Productos o artefactos según la ISO/IEC 29110[[5]](#footnote-5)

|  |  |  |  |
| --- | --- | --- | --- |
| **#** | **Nombre** | **Descripción** | **Proceso de donde proviene** |
| 1. | Acceptance Record (Registro de aceptación) | Documenta la aceptación del Cliente de los entregables del proyecto. | Gestión del proyecto. |
| 2. | Change Request (Solicitud de cambio) | Identifica un problema y solicita modificaciones. | Implementación de software.  Cliente.  Gestión del proyecto. |
| 3. | Correction Register | Identifica las actividades establecidas para corregir una desviación o problema relacionado con el cumplimiento de un plan. | Gestión del proyecto. |
| 4. | Maintenance Documentation (Manual de mantenimiento) | Describe la configuración del software y el entorno utilizado para el desarrollo y las pruebas (compiladores, herramientas de diseño, construcción y pruebas.  Está escrito en términos que el personal de mantenimiento puede entender. | Implementación de software. |
| 5. | Meeting Record (Acta de la reunión) | Registra los acuerdos establecidos con el Cliente y/o equipo de trabajo. | Gestión de proyectos. |
| 6. | Product Operation Guide (Manual de operación) | Contiene información necesaria para instalar y administrar el software. | Implementación de software. |
| 7. | Progress Status Record (Registro de estado de progreso) | Registra el estado del proyecto en relación con el Plan del Proyecto. | Gestión del proyecto. |
| 8. | Project Plan (Plan del Proyecto) | Presenta cómo se ejecutarán los procesos y actividades del proyecto para asegurar su finalización exitosa y la calidad de los productos entregados. | Gestión del proyecto. |
| **#** | **Nombre** | **Descripción** | **Proceso de donde proviene** |
| 9. | Project Repository (Repositorio del proyecto) | Contenedor electrónico para almacenar los productos y entregas del proyecto. | Gestión del proyecto. |
| 10. | Project Repository Backup (Copia de seguridad del repositorio del proyecto) | Repositorio usado para respaldar el Repositorio del proyecto y, si es necesario, para recupera la información. | Gestión del proyecto. |
| 11. | Requirements Specification (Especificación de requerimientos) | Identifica los requerimientos de software. | Implementación de software. |
| 12. | Software | Elemento de software (código fuente y código ejecutable) para el Cliente, constituido por un conjunto de Componentes de software. | Implementación de software. |
| 13. | Software Components (Componentes de software) | Un conjunto de unidades de código relacionadas. | Implementación de software. |
| 14. | Software Configuration (Configuración del software) | Un conjunto de productos de software identificados de forma única y consistentes, incluyendo:   * Especificación de requerimientos * Diseño de software * Registro de trazabilidad * Componentes de software * Software * Casos de prueba y procedimientos de prueba * Reporte de pruebas * Manual de operación * Manual de usuario * Manual de mantenimiento | Implementación de software. |
| **#** | **Nombre** | **Descripción** | **Proceso de donde proviene** |
| 15. | Software Design (Diseño de software) | Información textual y gráfica sobre la estructura del software.  Diseño arquitectónico de alto nivel del software - Describe la estructura general del software  Diseño detallado de bajo nivel del software – incluye detalles de los *Componentes de software* para facilitar su construcción y prueba dentro del entorno de programación. | Implementación de software. |
| 16. | Software User Documentation (Manual de usuario) | Describe la forma de utilizar el software a partir de la interfaz de usuario.  Está escrito en términos comprensibles para los usuarios. | Implementación de software. |
| 17. | Statement of Work (Declaración de trabajo) | Descripción del trabajo a realizar en relación con el desarrollo del software. | Cliente. |
| 18. | Test Cases and Test Procedures (Casos de prueba y procedimientos de prueba) | Elementos necesarios para probar el código. | Implementación de software. |
| 19. | Test Report (Reporte de pruebas) | Documenta la ejecución de las pruebas. | Implementación de software. |
| 20. | Traceability Record (Registro de trazabilidad) | Documenta la relación entre los requerimientos incluidos en la Especificación de requerimientos, elementos del Diseño de software, Componentes de software, Casos de prueba y Procedimientos de prueba. | Implementación de software. |
| **#** | **Nombre** | **Descripción** | **Proceso de donde proviene** |
| 21. | Verification Results (Resultados de la verificación) | Documento de la ejecución de la verificación. | Implementación de software. |
| 22. | Validation Results (Resultados de la validación) | Documento de la ejecución de la validación. | Implementación de software. |

Y finalmente, la ISO/IEC 29110 tiene por definición unos roles para el proyecto. Estos son: analista, cliente, diseñador, programador, gestor del proyecto, líder técnico y equipo de trabajo.

### DevOps

El término DevOps es una mezcla de development (desarrollo) (representando a los desarrolladores de software, incluyendo programadores, probadores de software y personal de aseguramiento de la calidad) y operations (operaciones) (representando a los expertos, quienes son los que ponen el software en producción y gestionan la infraestructura de producción, incluyendo los administradores de sistemas, bases de datos y los técnicos de red). DevOps describe prácticas que agilizan el proceso de entrega de software, enfatizando el aprendizaje al transmitir retroalimentación desde la producción hasta el desarrollo y mejorando el tiempo de ciclo. DevOps no sólo permitirá la entrega de software más rápido, sino que también ayudará a producir software de alta calidad que esté más alineado con los requerimientos individuales y las condiciones básicas.(Hüttermann, 2012, p. 4).

* Actividades y aspectos de DevOps:

DevOps abarca numerosas actividades y aspectos, entre estos se encuentran (Hüttermann, 2012, p. 4):

* **Cultura:** las personas sobre los procesos y las herramientas. El software está hecho por y para las personas.
* **Automatización:** la automatización es esencial para que DevOps obtenga una retroalimentación rápida.
* **Medición:** DevOps encuentra una ruta específica para la medición. La calidad y los incentivos compartidos son fundamentales.
* **Compartir:** crea una cultura en la que las personas comparten ideas, procesos y herramientas.

DevOps es un movimiento que trata de mejorar la agilidad en la prestación de servicios, fomentando una mayor colaboración y comunicación entre los equipos de desarrollo y operaciones, así mismo, evita que los problemas y las necesidades operacionales afecten la calidad del software (Farías & Ivonne, 2017, p. 38).



Figura 6. Definición de DevOps[[6]](#footnote-6).

* Influencias y orígenes (Hüttermann, 2012, p. 5):

Patrick Debois acuñó el término DevOps en el año 2009 mientras organizaba la conferencia DevOpsDays en Bélgica. Esta fue la primera de una serie de conferencias relevantes dedicadas al concepto que ayudó a difundir el término popularmente. En numerosos movimientos pasados, adoptadores tempranos e influencias ayudaron a acuñar el término DevOps y transformarlo en un término aceptado:

* Patrick Debois dirigió una sesión llamada “Agile Operations and Infrastructure: How Infra-gile Are You?” en la conferencia Agile 2008 en Toronto y publicó un artículo con un nombre similar.
* Marcel Wegermann publicó una lista de correo electrónico llamada “Agile System Administration”.
* John Allspaw hizo una presentación titulada “10+ Deploys per Day: Dev and Ops Cooperation” en la conferencia Velocity de 2009 en San José.
* Erick Ries publicó *The Lean Startup*.
* *The 451 group* publicó el primer informe de analistas sobre DevOps (titulado “The Rise of DevOps”) en septiembre de 2010.
* Lo que no es DevOps (Hüttermann, 2012, p. 9):

El término DevOps es un término ligeramente sobrecargado. Comprender el alcance del concepto DevOps ayuda a discutir lo que DevOps no es. DevOps no es un término de marketing. El enfoque de DevOps acepta los desafíos diarios en la entrega de software y proporciona pasos para abordarlos. DevOps no permite que los desarrolladores trabajen en el sistema de producción. No es un “proceso” gratuito que abre aspectos relevantes para la producción a los desarrolladores. En cambio, DevOps se trata de disciplina, convenciones y un proceso definido que sea transparente para todos.

* Roles y estructuras:

DevOps no es un departamento nuevo. Cada intento de establecer un departamento de tipo DevOps conduce a construcciones extrañas. Algunas personas creen que “NoOps” (No Operations) es el futuro, donde los desarrolladores se encargan de todos los aspectos relevantes de la producción de software. Por supuesto, tal escenario es imposible; los desarrolladores y los ingenieros de operaciones tienen diferentes prioridades y habilidades. Del mismo modo, es falso creer que en DevOps los expertos en operaciones se hacen cargo de todo.



Figura 7. Ejemplo de una estructura organizacional en la que se visualiza erróneamente la instauración de un departamento de tipo DevOps[[7]](#footnote-7).

Las responsabilidades pueden (y de hecho lo hacen) cambiar con el tiempo, y a medida que cambian, obviamente lo hacen las descripciones de los puestos de trabajo, pero no importa cómo se haga, se realizarán los mismos trabajos, uno de ellos son las operaciones y el otro es el desarrollo. DevOps describe patrones de colaboración, procesos y herramientas; no es un título de trabajo.

* DevOps y conjuntos de herramientas:

Algunas personas prefieren pensar en las herramientas en lugar de en las personas y los procesos. Sin respetar otras personas y comprender los procesos, estas personas simplemente introducen herramientas. Con DevOps surge el mismo problema. Sin entender la idea detrás del enfoque, intentar mejorar la colaboración y compartir los procesos de una manera concreta, todo intento de adoptar DevOps fracasará.

Etiquetar herramientas individuales como herramientas DevOps es aceptable, pero no hay que pensar en DevOps como una nueva herramienta para eliminar el personal de operaciones o como un conjunto de herramientas; más bien, es un enfoque para liberar el tiempo del personal actual para centrarse en problemas más difíciles que pueden aportar aún más valor al negocio.



Figura 8. DevOps no es una suite de herramientas[[8]](#footnote-8).

En la mayoría de los casos en los que interviene DevOps, las herramientas prexistentes están orquestadas e integradas para ser utilizadas por los equipos de desarrollo y operaciones.

* Prácticas ágiles de DevOps:

DevOps maneja unas prácticas ágiles que son utilizadas durante toda la etapa de desarrollo y despliegue de aplicaciones, las cuales son: integración continua, entrega continua y despliegue continuo.

Es pertinente mencionar, que en este documento se va a trabajar con las dos primeras prácticas ágiles de las tres mencionadas anteriormente (integración y entrega continua), motivo por el cual, el enfoque estará centrado únicamente en estas dos.

* Integración Continua (IC):

Es el proceso de integración de código nuevo, escrito por los desarrolladores con una línea principal o rama “maestra” con frecuencia a lo largo del día. Esto contrasta con el hecho de que los desarrolladores trabajen en ramas de funciones independientes durante semanas o meses a la vez, fusionando su código con la rama maestra sólo cuando esté completamente terminado (Davis & Daniels, 2016, p. 38).

Para asegurarse de que las integraciones fueron exitosas, los sistemas de IC generalmente ejecutarán una serie de pruebas automáticamente al fusionarse nuevos cambios. Cuando estos cambios se confirman y se fusionan, las pruebas comienzan a ejecutarse automáticamente. El resultado de estas pruebas a menudo se visualiza así: “verde” cuando se pasan las pruebas, concluyendo que la compilación recién integrada se considera limpia, y “rojo” cuando las pruebas fallan, concluyendo que la compilación posee fallas que deben repararse, cuando esto ocurre, según (Farías & Ivonne, 2017, p. 35) el equipo de desarrollo detiene lo que está haciendo y lo corrige inmediatamente. Con este tipo de flujo de trabajo, los problemas se pueden identificar y solucionar de forma más rápida, debido a que se realizan continuas pruebas automatizadas (Davis & Daniels, 2016, p. 38).

* Entrega Continua (EC):

Esta práctica ha sido diseñada para brindar mayor rendimiento y que los equipos se mantengan en constante retroalimentación sobre su aplicación. Es un conjunto de principios generales de ingeniería de software que permiten lanzamientos frecuentes de software nuevo mediante el uso de pruebas automatizadas e integración continua (Farías & Ivonne, 2017, p. 34).

La entrega continua requiere una aprobación manual. Los periodos de tiempo para lograr entregas continuas dependen de que la organización cuente con el equipo de desarrollo y operaciones adecuado. De esta manera se pueden presentar versiones candidatas en tiempos mínimos (Belalcázar, 2017, p. 24).



Figura 9. Arquitectura de referencia de DevOps[[9]](#footnote-9).

### Metodología ICONIX

Es una metodología de desarrollo de software de tamaño medio, cuyo análisis y capacidad de diseño se basa en UML (Unified Modeling Language (UML))(Mnkandla & Dwolatzky, 2004, p. 1). Fue elaborada por Doug Rosenberg y Kendall Scott. Se dice que ICONIX se encuentra entre la complejidad de RUP (Rational Unified Processes) y la simplicidad y pragmatismo de XP (eXtreme Programming), sin eliminar las tareas de análisis y de diseño que XP no contempla (Amavizca Valdez, García Ruíz, Jiménez López, Duarte Guerrero, & Vásquez Brindis, 2014, p. 3).

El fundamento de ICONIX es el hecho de que un 80% de los casos pueden ser resueltos tan solo con un uso del 20% del UML, con lo cual se simplifica en gran medida el proceso sin perder documentación al dejar solo aquello que es necesario. Esto implica un uso dinámico del UML de tal forma que siempre se pueden utilizar otros diagramas aparte de los ya estipulados si se cree conveniente (ICONIX Brand Group, 2016, p. 1).

El proceso de ICONIX describe cómo pasar de los casos de uso a la codificación de forma fiable, en el menor tiempo posible. Por tal motivo, la principal preocupación de ICONIX es el análisis y diseño de los aspectos de modelado de la producción de software. (Rosenberg, Stephens, & Collins-Cope, 2005, p. 41).

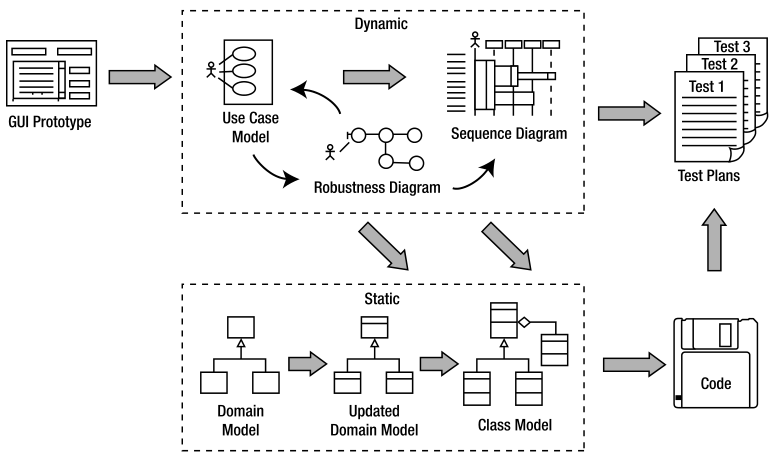


Figura 10. Proceso de ICONIX[[10]](#footnote-10).

* Orígenes (Rosenberg et al., 2005, p. 40):

El proceso ICONIX se originó varios años antes de UML y el proceso unificado como una síntesis y destilación de las mejores técnicas de las metodologías originales que formaron UML: la Técnica de modelado de objetos (Object Modeling Technique (OMT)) de Jim Rumbaugh, el método Objectory de Ivar Jacobson y el método Booch de Grady Booch.

Se realizó una síntesis de estas tres escuelas diferentes de pensamiento orientado a objetos (OO) porque las fortalezas y debilidades de estas metodologías se complementaban entre sí.

* Características de ICONIX:

Se destacan tres características fundamentales de ICONIX:

* Iterativo e incremental: múltiples iteraciones ocurren entre el desarrollo del modelado del dominio y la identificación de los casos de uso. El modelo estático se refina incrementalmente durante las iteraciones sucesivas a través del modelo dinámico (Fiestas Jacinto, 2015, p. 40).
* Trazabilidad: Cada paso está referenciado por algún requerimiento. Nunca hay un punto en que el proceso le permita desviarse lejos de las necesidades del usuario (Fiestas Jacinto, 2015, p. 40). Según (Amavizca Valdez et al., 2014) trazabilidad es la capacidad de seguir una relación entre los diferentes artefactos producidos.
* Aerodinámica del UML: la metodología ofrece un uso “aerodinámico” del UML, como lo son los diagramas de caso de uso, diagramas de secuencia y de colaboración (Amavizca Valdez et al., 2014, p. 3).

Como se dijo anteriormente, ICONIX basa su análisis y diseño en UML, por lo tanto, utiliza un conjunto de diagramas que auxilian a la metodología a proveer la mejor respuesta. Los principales diagramas que maneja ICONIX son:

* Modelado de dominio:

Es la tarea de crear un glosario de proyecto o un diccionario de términos utilizados en el proyecto. Su propósito es asegurarse de que todos en el proyecto entiendan el espacio problemático en términos inequívocos. El modelado de dominio para un proyecto define el alcance y forma la base sobre la cual construir los casos de uso (Rosenberg et al., 2005, p. 7).

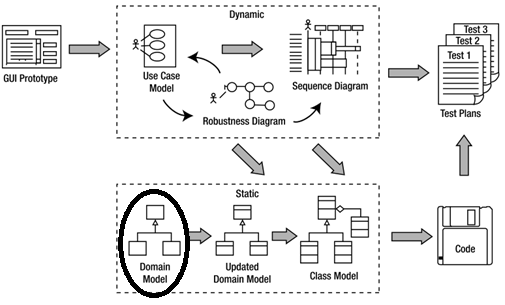


Figura 11. Ubicación del modelado de dominio en el proceso de ICONIX[[11]](#footnote-11).

* Diagramas de caso de uso:

Es el centro conceptual del desarrollo, porque guía todo el proceso de ICONIX (Bona, 2002, p. 68).

Un caso de uso es una secuencia de acciones que un actor (generalmente una persona, pero quizás una entidad externa, como otro sistema) realiza dentro de un sistema para lograr un objetivo particular. Un caso de uso completo e inequívoco, describe un aspecto del sistema sin suponer ningún diseño o implementación específica. El resultado del modelado de los casos de uso debería ser, que toda la funcionalidad requerida del sistema, se describa en los casos de uso. Si no se adhiere a este principio básico, se corre el riesgo de que los ingenieros construyan un sistema excelente que no es el que desea el cliente (Rosenberg et al., 2005, p. 42).

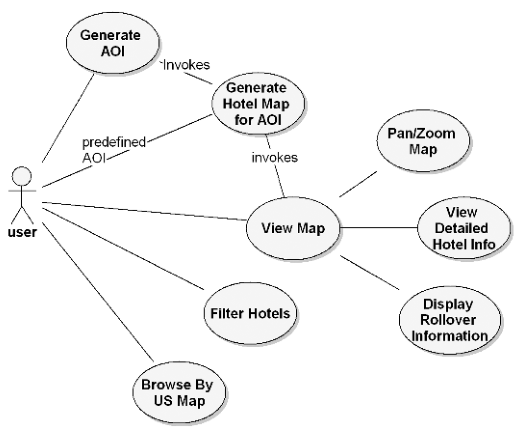


Figura 12. Ejemplo de diagrama de caso de uso[[12]](#footnote-12).

* Análisis de robustez:

Es una forma de analizar el texto del caso de uso e identificar un primer conjunto de posibles objetos de primera estimación para cada caso de uso. Estos objetos se clasifican en tres estereotipos (Rosenberg et al., 2005, p. 49):

* Objetos de límite (boundary objects): los actores los usan para comunicarse con el sistema (a menudo referidos como objetos de interfaz o frontera).
* Objetos de entidad (entity objects): son normalmente objetos del modelado de dominio, generalmente asignados en tablas de una base de datos.
* Objetos de control (control objects): funcionan como integradores entre los objetos de límite y los objetos de identidad. Generalmente, se convierten en métodos de objetos de entidad o de objetos de límite.

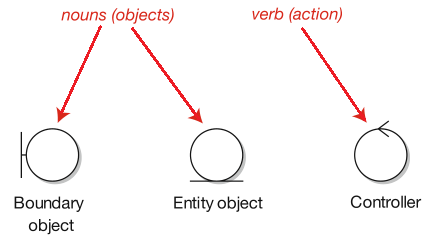


Figura 13. Objetos del diagrama de robustez[[13]](#footnote-13).

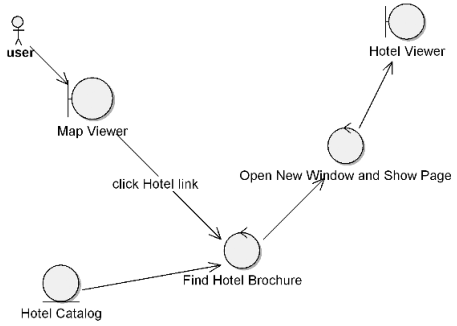


Figura 14. Ejemplo de un diagrama de robustez[[14]](#footnote-14).

Las reglas básicas que se deben aplicar al diagrama de robustez son (Rosenberg & Scott, 2001, pp. 61-62):

* Los actores sólo se pueden comunicar con objetos de límite.
* Los objetos de límite sólo se pueden comunicar con actores y objetos de control.
* Los objetos de entidad sólo se pueden comunicar con los objetos de control.
* Los objetos de control sólo se pueden comunicar con los objetos de límite y de control.
* Diagrama de secuencia:

El diagrama de secuencia muestra la colaboración dinámica entre los distintos objetos del sistema (Bona, 2002, p. 74). Es el núcleo del modelo dinámico y muestra todos los cursos alternos que pueden tomar los casos de uso (Carbajal & Martín, 2013, p. 25).

En (Rosenberg & Scott, 2001, pp. 83-84), se destacan cuatro tipos de elementos en un diagrama de secuencia, que son:

* Texto del caso de uso (flujo de acción). Copiar este elemento al margen izquierdo del diagrama de secuencia.
* Objetos de los diagramas de robustez. Están representado en cajas rectangulares con el nombre del objeto y opcionalmente con el nombre de la clase a la que pertenece (objeto: clase).
* Mensajes. Representados como flechas entre objetos.
* Los métodos (operaciones) se muestran como rectángulos que se encuentran encima de las líneas de puntos que pertenecen a los objetos a los que se asignan los métodos.

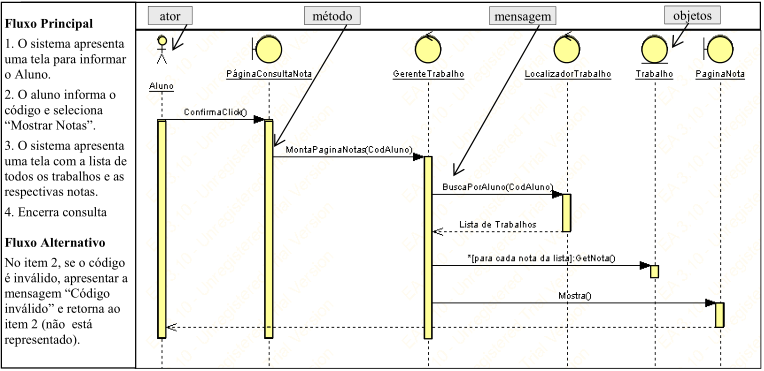


Figura 15. Elementos de un diagrama de secuencia[[15]](#footnote-15).

En (Rosenberg & Scott, 2001, p. 23) se presentan las tareas principales de la metodología ICONIX. Estas tareas incluyen el enfoque completo de la metodología con unos hitos específicos asociados, como se va a mostrar a continuación.

* Análisis de requerimientos:

Las actividades que se llevan a cabo en esta tarea son (Rosenberg & Scott, 2001, p. 23) (Fiestas Jacinto, 2015, p. 46):

* Identificar en el “mundo real” los objetos y todas las relaciones de agregación de generalización entre ellos. Utilizar el diagrama de clases de alto nivel definido como **modelado de dominio**.
* Presentar, si es posible, un **prototipado** rápido de la interfaz del sistema, o diagramas de navegación, de manera que el cliente pueda comprender mejor el sistema propuesto.
* Identificar los casos de uso del sistema, mostrando los actores involucrados. Utilizar **diagramas de caso de uso**.
* Organizar los casos de uso en grupos. Capturar esta organización en un **diagrama de paquete**.
* Asociar los requerimientos funcionales en los casos de uso y objetos de dominio en este escenario.

**Hito:** Revisión de los requisitos.

* Análisis y diseño preliminar:

Las actividades que se llevan a cabo en esta tarea son (Rosenberg & Scott, 2001, p. 24):

* Escribir los **casos de uso**, como flujo principal de acciones, pudiendo contener el flujo alternativo y el flujo de excepción.
* Presentar el **análisis de robustez**. Siendo que, para cada caso de uso se debe identificar un conjunto de objetos y actualizar el diagrama de clases del modelado de dominio.
* Finalizar la actualización del **diagrama de clases.**

**Hito:** Revisión del diseño preliminar.

* Diseño detallado:

Las actividades que se llevan a cabo en esta tarea son (Rosenberg & Scott, 2001, p. 24) (Bona, 2002, p. 64):

* Especificar comportamiento a través del **diagrama de secuencia**. Para cada caso de uso, identificar los mensajes entre los diferentes objetos. Si es necesario, utilizar diagrama de colaboración para representar las transacciones claves entre los objetos.
* Terminar el modelo estático, añadiendo detalles al proyecto en el **diagrama de clase**.
* Verificar con el equipo si el proyecto satisface todos los requerimientos identificados.

**Hito:** Revisión crítica del diseño.

* Implementación:

Las actividades que se llevan a cabo en esta tarea son (Rosenberg & Scott, 2001, p. 25):

* Si es necesario, generar diagramas de implementación y componentes para apoya la fase de implementación.
* Escribir/generar **código**.
* Realizar pruebas unitarias y de integración.
* Realizar pruebas de aceptación del usuario.

**Hito:** Entrega.

## Marco conceptual

### Proceso de software

Un proceso de software es un marco de trabajo de las tareas que se requieren para construir software de alta calidad. Un proceso de software define el enfoque que se toma cuando el software es tratado por la ingeniería (Pressman, 2002, p. 13). Es aquel conjunto de actividades relacionadas que se enfocan en la creación de un producto de software. Este software puede ser un producto totalmente nuevo o la extensión o actualización de un software ya existente (Ramos & Mendoza, 2014, p. 13).

Según (Sommerville, 2005, p. 7), existe cuatro actividades de procesos que son comunes para todos los procesos de software. Estas actividades son:

* Especificación del software. Se definen el software a producir y las restricciones sobre su operación.
* Desarrollo del software. Es donde el software se diseña y programa.
* Validación del software. Es donde el software se valida para asegurar que es lo que el cliente requiere.
* Evolución del software. Aquí es donde el software se modifica para adaptarlo a los cambios requeridos por el cliente y el mercado.

Aparte de las actividades, es importante que un proceso incluya los siguientes componentes en su descripción (Ramos & Mendoza, 2014, p. 14):

* Productos: son las salidas de las actividades de un proceso.
* Roles: reflejan las responsabilidades de las personas involucradas en el proceso. Ejemplo de roles son: diseñador, desarrollador, etc.
* Pre y post condiciones: son enunciados que tienen una validez antes y después de que una actividad de un proceso ha sido realizada.

### Metodología de desarrollo de software

Una metodología de desarrollo de software tiene como función estructurar, planificar y controlar el proceso de desarrollo en sistemas de información (Hugues, Fisher, & Mc Daniel, 2010).

Según (Cataldi, 2000, p. 40), una metodología de desarrollo de software es un conjunto de componentes que especifican:

* Cómo se debe dividir un proyecto en etapas.
* Qué tareas se llevan a cabo en cada etapa.
* Qué salidas se producen y cuando se debe producir.
* Qué restricciones se aplican.
* Qué herramientas se van a utilizar.
* Cómo se gestiona y controla un proyecto.

Las metodologías de desarrollo de software se clasifican en:

* Metodologías tradicionales (Figueroa, Solis, & Cabrera, s. f., p. 1) (Navarro Cadavid, Fernández Martínez, & Morales Vélez, 2013, p. 31):

Estas metodologías centran su atención en llevar una documentación exhaustiva de todo el proyecto y en cumplir con un plan de proyecto, definido todo esto, en la fase inicial del desarrollo del proyecto.

En las metodologías se concibe un solo proyecto, de grandes dimensiones y estructura definida; se sigue un proceso secuencial, en una sola dirección y sin marcha atrás; el proceso es rígido y no cambia; los requerimientos son acordados de una vez y para todo el proyecto, demandando grandes plazos de planeación previa y poca comunicación con el cliente una vez ha terminado ésta.

Entre las principales metodologías tradicionales, se encuentran: RUP (Rational Unified Process) y MSF (Microsoft Solution Framework).

* Metodologías ágiles (Cáceres & Marcos, s. f., p. 2) (Figueroa et al., s. f., pp. 1; 5) (Timoco Gómez, Rosales López, & Salas Bacalla, 2010, p. 72):

Se basa en dos aspectos puntuales: el retrasar las decisiones y la planificación adaptativa; permitiendo potenciar aún más el desarrollo de software a gran escala. Estas metodologías apuestan por una cantidad apropiada de procesos, es decir, ni se pierden en una excesiva cantidad de cuestiones burocráticas ni defienden tampoco la falta total de proceso.

Las metodologías ágiles proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos.

Entre las principales metodologías ágiles, se encuentran: XP (eXtreme Programming), Scrum, Cristal Methods y AUP (Agile Unified Process).

### UML (Unified Modeling Language)

Es una familia de notaciones gráficas, respaldada por un único metamodelo, que ayuda a describir y diseñar sistemas de software, particularmente sistemas de software construidos utilizando el estilo OO (orientado a objetos). Fue creado por Grady Booch y Jim Rumbaugh en 1995 (Fowler, s. f., p. 1).

El Lenguaje Unificado de Modelado (UML) es un lenguaje estándar de modelado para software que permite visualizar, especificar, construir y documentar los artefactos de sistemas en los que el software juega un papel importante. Básicamente, UML permite a los desarrolladores visualizar los resultados de su trabajo en esquemas o diagramas estandarizados (Jacobson, Booch, & Rumbaugh, 2000, p. 407).

UML se centra en la descripción de métodos y procesos con lo cual permite definir de igual manera las interacciones que se llevan a cabo de parte de los programadores y los usuarios con el software que se está desarrollando, esto se hace a través de diferentes tipos de diagramas, que se encuentran organizados por categorías de la siguiente manera (Matla Cruz, 2014, pp. 24-26):

Tabla 3. Diagramas de UML por categoría.

|  |  |  |
| --- | --- | --- |
| **De estructura** | **De comportamiento** | **De interacción** |
| Diagrama de clases | Diagrama de estado | Diagrama de secuencia |
| Diagrama de objetos | Casos de uso | Diagrama de colaboración |
| Diagrama de componentes |  |  |
| Diagrama de paquetes |  |  |

(Elaboración propia)

### Calidad de software

La calidad del software es una compleja combinación de factores que variaran entre diferentes aplicaciones, diversos autores como Pressman. McCall y estándares, como ISO 9126 han tratado de determinar y categorizar los factores que afectan a la calidad del software.

Una definición amplia de calidad, planteada en la norma UNE-EN ISO 8402, expresa que “la calidad es el conjunto de propiedades y características de un producto o servicio que le confieren su aptitud para satisfacer unas necesidades explicitas o implícitas”. Llevada esta definición al campo de la ingeniería de software, la IEEE Std 610, señala que “la calidad del software es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario”.

Por su parte Pressman, se refiere a la calidad del software como “la concordancia con los requisitos funcionales y de rendimientos explícitamente establecidos, estándares de desarrollo explícitamente documentados y características implícitas que se espera de todo software desarrollado profesionales”.

Sommerville sostiene que la calidad del software es un concepto complejo que no es directamente comparable con la calidad de la manufactura de productos. En la manufacturación, la noción de calidad viene dada por la similitud entre el producto, pero para sistemas de software existen cuestiones específicas que impiden aplicar este mecanismo.

### Modelo de calidad de software

A lo largo del tiempo se han desarrollado diferentes modelos para evaluar la calidad del software que intentan descomponer la calidad en una categoría de características más sencilla. En el caso de la calidad de software, el modelo debe ir enfocado a hacer seguimiento y evaluación a cada etapa de construcción del producto de software (Callejas Cuervo, Alarcón Aldana, & Álvarez Carreño, 2017, p. 238).

Según (Scalone, 2006), los modelos de calidad son aquellos documentos que integran la mayor parte de las mejores prácticas, proponen temas de administración en los que cada organización deben hacer énfasis, integran diferentes prácticas dirigidas a los procesos clave y permiten medir los avances en calidad.

En el ámbito de la construcción de software, el modelo de calidad debe permitir evaluar el sistema, bien sea cualitativa o cuantitativamente, y de acuerdo con esta evaluación la organización podrá proponer e implementar estrategias que permitan la mejora del proceso dentro de las etapas de análisis, diseño, desarrollo y pruebas del software (Callejas Cuervo et al., 2017, p. 238).

## Marco legal

Todas las actividades propuestas y desarrolladas dentro del proyecto quedan inmersas dentro del acuerdo de cooperación interinstitucional realizado entre la empresa Sanambiente y la Institución Universitaria Antonio José Camacho, firmado en el mes de abril de 2019.

Para la realización e implementación de este proyecto se tomará como base el estándar ISO/IEC 29110, bajo el título general *Ingeniería de Software – Perfiles del ciclo de vida para entidades muy pequeñas* (Very Small Enterprises (VSEs)).

# Desarrollo

A partir del estudio realizado en el marco teórico se determinaron las siguientes fases:

# Conclusiones

# Referencias

Amavizca Valdez, L. O., García Ruíz, A. C., Jiménez López, E., Duarte Guerrero, G. L., & Vásquez Brindis, J. C. (2014). *Aplicación de la metodología semi-ágil ICONIX para el desarrollo de software: implementación y publicación de un sitio WEB para una empresa SPIN -OFF en el Sur de Sonora, México.* [Científico].

Belalcázar, A. (2017). *Arquitectura de un Data Center con herramientas DevOps*. Universidad Nacional de la Plata.

Bona, C. (2002). *AVALIAÇÃO DE PROCESSOS DE SOFTWARE: UM ESTUDO DE CASO EM XP E ICONIX*. Universidade Federal de Santa Catarina, Florianópolis.

Cáceres, P., & Marcos, E. (s. f.). *Procesos ágiles para el desarrollo de aplicaciones web* (p. 4) [Científico]. Madrid: Universidad Rey Juan Carlos.

Callejas Cuervo, M., Alarcón Aldana, A. C., & Álvarez Carreño, A. M. (2017). *Modelos de calidad del software, un estado del arte*. *13, núm. 1*(enero-junio (2017)), 236-250.

Carbajal, W., & Martín, D. (2013). *Implementación de un sistema informático web para la gestión de compras de la empresa Certicom S.A.C usando la metodología ICONIX y frameworks Spring, Hibernate y Richfaces*. Universidad Privada Antenor Orrego, Trujillo.

Cataldi, Z. (2000). *Una metodología para el diseño, desarrollo y evaluación de software educativo*. Universidad Nacional de la Plata, La Plata.

Davis, J., & Daniels, K. (2016). *Effective DevOps*. O’Really Media.

Farías, A., & Ivonne, K. (2017). *Definición de un ambiente de construcción de aplicaciones empresariales a través de DevOps, microservicios y contenedores*. Universidad Técnica Particular de Loja, Loja.

Fiestas Jacinto, J. E. (2015). *La implementación de un sistema de inteligencia de negocios que permita mejorar la toma de decisiones respecto a las remuneraciones de la empresa pesquera Carlos Eduardo S.R.L.-2014*. Universidad Nacional de Trujillo, Trujillo.

Figueroa, R. G., Solis, C. J., & Cabrera, A. A. (s. f.). *Metodologías tradicionales vs. Metodologías ágiles* (p. 9) [Científico]. Loja: Universidad Técnica Particular de Loja.

Fowler, M. (s. f.). *UML Distilled: A brief guide to the standard object modeling language* (Third Edition). Addison Wesley.

Hugues, P., Fisher, P., & Mc Daniel, J. (2010). *System development life cycle models and methodologies*. Canadian Society for International Health Certificate Course in Health Information Systems.

Hüttermann, M. (2012). *DevOps for Developers*. Apress.

ICONIX Brand Group. (2016). *Manual introductorio de ICONIX*.

ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*.

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *El proceso unificado de desarrollo de software* (S. Sánchez, M. Á. Sicilia, C. Canal, & F. J. Durán, Trads.). Madrid: Addison Wesley.

Laporte, C., O´connor, R., & García, L. (2016). *THE IMPLEMENTATION OF ISO/IEC 29110 SOFTWARE ENGINEERING STANDARDS AND GUIDES IN VERY SMALL ENTITES*. École de technologie supérieure, Montréal, Canada, School of Computing, Dublin City University, Dublin, Ireland, Universidad Peruana de Ciencias Aplicadas, Lima, Peru.

LEGARIA. (2018). *IMPLEMENTACIÓN DE PROCESO ORGANIZACIONAL DE GESTIÓN DE PROYECTOS EN DEVELOPIT*. SANTIAGO DE CHILE.

Madruñero Padilla, E. R. (2018). *Implementación del estándar ISO/IEC 29110 en el proceso de desarrollo de software de la dirección de desarrollo tecnológico e informático de la Universidad Técnica del Norte*. Universidad Técnica del Norte, Ibarra.

Matla Cruz, E. O. (2014). *Desarrollo de software guiado por la norma ISO/IEC 29110 y Scrum: SIDEP v.2.0*. Universidad Nacional Autónoma de México, Ciudad de México.

Mnkandla, E., & Dwolatzky, B. (2004). *A Survey of Agile Methodologies* [Científico].

Navarro Cadavid, A., Fernández Martínez, J. D., & Morales Vélez, J. (2013). *Revisión de metodologías ágiles para el desarrollo de software*. *11, núm. 2*(julio-diciembre (2013)), 30-39.

PORRAS. (2019). *“METODOLOGÍA ÁGIL ICONIX EN LA CALIDAD DEL PRODUCTO SOFTWARE, LIMA, 2017”*. Universidad Nacional Federico villareal, LIMA – PERÚ.

Pressman, R. S. (2002). *Ingeniería del Software. Un enfoque práctico* (Quinta Edición; R. Martín Ojeda, V. Yagüe Galaup, I. Morales Jareño, & S. Sánchez Alonso, Trads.). Madrid: Concepción Fernández Madrid.

Ramos, C., & Mendoza, L. (2014). *Implementación del estándar ISO/IEC 29110-4-1 para pequeñas organizaciones de desarrollo de software*. Universidad Peruana de Ciencias Aplicadas, Lima.

Rosenberg, D., & Scott, K. (2001). *Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example* (First Edition). Addison Wesley.

Rosenberg, D., Stephens, M., & Collins-Cope, M. (2005). *Agile Development with ICONIX Process*. United States of America: Apress.

Scalone, F. (2006). *Estudio comparativo de los modelos y estándares de calidad del software*. Universidad Tecnológica Nacional Facultad Regional Buenos Aires, Buenos Aires.

Sommerville, I. (2005). *Ingeniería del software* (Séptima Edición). Madrid: Pearson Addison Wesley.

Timoco Gómez, O., Rosales López, P. P., & Salas Bacalla, J. (2010). *Criterios de selección de metodologías de desarrollo de software*. *13, núm. 2*(julio (2010)), 70-74.

1. ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*, p.vi. [↑](#footnote-ref-1)
2. ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*, p.4. [↑](#footnote-ref-2)
3. ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*, p.9. [↑](#footnote-ref-3)
4. ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*. [↑](#footnote-ref-4)
5. ISO/IEC. (2011). *Software engineering - Lifecycle profiles for Very Small Entities (VSEs) Part 5-1-2: Management and engineering guide: Generic profile group: Basic profile*, p.33-40. [↑](#footnote-ref-5)
6. Farías, A., & Ivonne, K. (2017). *Definición de un ambiente de construcción de aplicaciones empresariales a través de DevOps, microservicios y contenedores*. Universidad Técnica Particular de Loja, Loja, p.39. [↑](#footnote-ref-6)
7. Hüttermann, M. (2012). *DevOps for Developers*. Apress, p.9. [↑](#footnote-ref-7)
8. Hüttermann, M. (2012). *DevOps for Developers*. Apress, p.11. [↑](#footnote-ref-8)
9. Farías, A., & Ivonne, K. (2017). *Definición de un ambiente de construcción de aplicaciones empresariales a través de DevOps, microservicios y contenedores*. Universidad Técnica Particular de Loja, Loja, p.36. [↑](#footnote-ref-9)
10. Rosenberg, D., Stephens, M., & Collins-Cope, M. (2005). *Agile Development with ICONIX Process*. United States of America: Apress, p.45. [↑](#footnote-ref-10)
11. Rosenberg, D., & Scott, K. (2001). *Applying Use Case Driven Object Modeling with UML: An Annotated e-Commerce Example* (First Edition). Addison Wesley, p.27. [↑](#footnote-ref-11)
12. Bona, C. (2002). *AVALIAÇÃO DE PROCESSOS DE SOFTWARE: UM ESTUDO DE CASO EM XP E ICONIX*. Universidade Federal de Santa Catarina, Florianópolis, p.44. [↑](#footnote-ref-12)
13. Rosenberg, D., & Stephens, M. (2007). *Use Case Driven Object Modeling with UML*. United States of America: Apress, p.103. [↑](#footnote-ref-13)
14. Rosenberg, D., Stephens, M., & Collins-Cope, M. (2005). *Agile Development with ICONIX Process*. United States of America: Apress, p.48. [↑](#footnote-ref-14)
15. Bona, C. (2002). *AVALIAÇÃO DE PROCESSOS DE SOFTWARE: UM ESTUDO DE CASO EM XP E ICONIX*. Universidade Federal de Santa Catarina, Florianópolis, p.75. [↑](#footnote-ref-15)