ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

Diseño de un Modelo de Trabajo Basado en las Mejores Prácticas de las Metodologías Tradicionales y no Tradicionales para Ayudar al Proceso de Desarrollo de Software en Sistemas Embebidos del Laboratorio de Investigación del Departamento de Eléctrica y Electrónica.

Vicerrectorado de Investigación, Innovación y Transferencia de Tecnológica

Centro de Estudios de Posgrado

Maestría en Ingeniería en Software

Trabajo de titulación, previo la obtención de título de Magister en Ingeniería en Software

Autora: Ing. Chicaiza Angamarca, Doris Karina

Director: Ing. Espinosa Gallardo, Edison Gonzalo PhD

31 de marzo de 2021

Certificado de Tutor

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y

TRANSFERENCIA TECNOLÓGICA

CENTRO DE POSGRADOS

MAESTRÍA EN INGENIERÍA DE SOFTWARE

CERTIFICACIÓN

Certifico que el trabajo de titulación "DISEÑO DE UN MODELO DE TRABAJO BASADO EN LAS MEJORES PRÁCTICAS DE LAS METODOLOGÍAS TRADICIONALES Y NO TRADICIONALES PARA AYUDAR AL PROCESO DE DESARROLLO DE SOFTWARE EN SISTEMAS EMBEBIDOS DEL LABORATORIO DE INVESTIGACIÓN DEL DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA." fue realizado por la Sra. Ing. CHICAIZA ANGAMARCA DORIS KARINA, el mismo que ha sido revisado en su totalidad, analizado por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Latacunga, 18 enero de 2021

.....

Ing. Espinosa Gallardo Edison Gonzalo Ph.D

C.C:0501577910

Reporte de Urkund

URKUND

Document Information

Analyzed document	Tesis 3.6 Urkund Chicaiza Doris.docx (D93013076)
Submitted	1/20/2021 10:41:00 PM
Submitted by	Espinosa Gallardo Edison Gonzalo
Submitter email	egespinosa1@espe.edu.ec
Similarity	6%
Analysis address	egespinosa1.espe@analysis.urkund.com

Firmado electrónicamente por:
EDISON GONZALO
ESPINOSA
GALLARDO

Sources included in the report

W	URL: https://dialnet.unirioja.es/descarga/articulo/5351802.pdf Fetched: 1/20/2021 10:58:00 PM	3
W	URL: https://es.slideshare.net/yeltsintorres18/metodologias-para-el-desarrollo-del-software Fetched: 11/22/2019 6:19:37 PM	1
W	URL: http://www.culiacan.tecnm.mx/wp-content/uploads/2020/07/Tesis-Miguel-26-03-2019.pdf Fetched: 12/9/2020 11:01:19 PM	1
SA	**TESIS - RONALD GEORGE v5 -URKUND.docx** Document TESIS - RONALD GEORGE v5 -URKUND.docx (D61920416)	1
SA	**computaciòn.docx** Document computaciòn.docx (D71785149)	1
W	URL: https://docplayer.es/9901811-Desarrollo-de-una-aplicacion-de-gestion-de-quejas-y-s ... Fetched: 11/19/2019 7:06:26 AM	1
W	URL: https://www.researchgate.net/profile/Jimmy_Molina2/publication/311909956_Nociones_ ... Fetched: 7/27/2020 1:37:39 AM	1
W	URL: https://docplayer.es/74457871-Desarrollo-de-software.html Fetched: 10/18/2019 8:39:10 AM	3
W	URL: https://es.wikipedia.org/wiki/Metodolog%C3%ADa_de_desarrollo_de_software Fetched: 1/20/2021 10:58:00 PM	1
SA	**FT_SrCaicedo.pdf** Document FT_SrCaicedo.pdf (D87076507)	1
SA	**Tesis Maestria2015 ESPELV6_3.pdf** Document Tesis Maestria2015 ESPELV6_3.pdf (D16352801)	1
W	URL: https://docs.google.com/spreadsheets/d/1YKzUqDmQJcBi42pRmFeBrdcnkfGn-zt-8ioOIRK0uI ... Fetched: 1/20/2021 10:58:00 PM	3

Certificado de Implementación

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

CERTIFICADO

LI-ARSI-01-2021

Certifico que el trabajo de titulación "DISEÑO DE UN MODELO DE TRABAJO BASADO EN LAS MEJORES PRÁCTICAS DE LAS METODOLOGÍAS TRADICIONALES Y NO TRADICIONALES PARA AYUDAR AL PROCESO DE DESARROLLO DE SOFTWARE EN SISTEMAS EMBEBIDOS DEL LABORATORIO DE INVESTIGACIÓN DEL DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA." se aplicó en el análisis y diseño del Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos, que es parte de un proyecto que se ejecuta en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas ESPE - Sede Latacunga.

A pedido del interesado y para los fines que estime conveniente, se expide el presente certificado a los 18 días del mes de enero de 2021.

Firmado electrónicamente por:
VICTOR HUGO ANDALUZ

.....

Ing. Víctor H. Andaluz, Ph.D.
Jefe del Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes
Profesor Titular Principal 1 ESPE

Autoría de Responsabilidad

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA TECNOLÓGICA

CENTRO DE POSGRADOS

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORÍA DE RESPONSABILIDAD

Yo, CHICAIZA ANGAMARCA, DORIS KARINA con cédula de ciudadanía N° 0502986508 declaro que el contenido, ideas y criterios del trabajo de titulación: "Diseño De Un Modelo De Trabajo Basado En Las Mejores Prácticas De Las Metodologías Tradicionales Y No Tradicionales Para Ayudar Al Proceso De Desarrollo De Software En Sistemas Embebidos Del Laboratorio De Investigación Del Departamento De Eléctrica Y Electrónica", es de mi autoría y responsabilidad, cumpliendo con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas. Consecuentemente el contenido de la investigación mencionada es veraz.

Latacunga, enero del 2021

……

Chicaiza Angamarca, Doris Karina

C.C: 0502986508

Autorización de Publicación

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA TECNOLÓGICA

CENTRO DE POSGRADOS

MAESTRÍA EN INGENIERÍA DE SOFTWARE

AUTORIZACIÓN DE PUBLICACIÓN

Yo, CHICAIZA ANGAMARCA DORIS KARINA, con cédula de ciudadanía 0502986508, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación "Diseño De Un Modelo De Trabajo Basado En Las Mejores Prácticas De Las Metodologías Tradicionales Y No Tradicionales Para Ayudar Al Proceso De Desarrollo De Software En Sistemas Embebidos Del Laboratorio De Investigación Del Departamento De Eléctrica Y Electrónica." en el Repositorio Institucional, cuyo contenido, ideas y criterios son de mi responsabilidad.

Latacunga, enero del 2021

……

Chicaiza Angamarca, Doris Karina

C.C: 0502986508

Dedicatoria

A mi querido esposo Javier, por su amor, cariño, ayuda y compresión, por ser siempre mi fiel amigo y compañero en buenos y difíciles momentos.

A mis padres: Alfredo y Diocelina por ser mi inspiración y aliento, quienes siempre me han apoyado de manera incondicional en este largo camino.

A mis hermanas y en especial a mi hermano Cristian Alfredo por creer en mí y brindarme fortaleza.

Agradecimiento

A mi esposo Javier Montaluisa por ser mi guía en todas las etapas de este proceso de educación. Por su entrega y dedicación para la culminación del presente trabajo.

A mis padres Alfredo y Diocelina por ser un gran soporte dentro de mi educación y haberme otorgado la oportunidad de seguirme superando en mi vida profesional.

Al Centro de Investigación del Departamento de Eléctrica y Electrónica, por abrirme las puertas y depositar su confianza para contribuir en los procesos que realizan para la construcción de Sistemas Embebidos.

Mi agradecimiento al Centro de Posgrados de la Universidad de las Fuerzas Armadas sede Latacunga, por permitirme continuar con mis estudios y especializarme.

Al P.h.d Edison Gonzalo Espinosa Gallardo por ser una guía para realizar mis objetivos planteados en mi proyecto de titulación, gracias a su apoyo ha cultivado en la confianza profesional y desenvolverme en mi especialidad.

Tabla de Contenidos

Índice de Tablas

Índice de figuras

Resumen

La Ingeniería de Software se considera una disciplina que se encarga del desarrollo, gestión y mantenimiento de los sistemas software utilizando prácticas, métodos y técnicas. Existen algunos tipos de software como los de base, desarrollo y aplicación. Dentro de los de aplicación podemos citar los ofimáticos, médico, videojuegos y sistemas embebidos. Estos últimos son sistemas de computación para realizan tareas de control y se aplican comúnmente en los ámbitos salud, agricultura, militar, telecomunicaciones e industria. En el desarrollo de este tipo de sistemas comúnmente se identifican problemas como: definición incompleta del alcance del sistema, falta de estandarización de modelos para el análisis y diseño entre otros que ocasionan la necesidad de contar con metodologías que guíen su desarrollo. Lo detallado se utiliza como base para plantearnos como **objetivo el de** adoptar las mejores prácticas de desarrollo de software de las metodologías tradicionales y no tradicionales para crear un modelo de trabajo que permita desarrollar un software de calidad para sistemas embebidos. Para cumplir este objetivo aplicamos los **métodos** revisión bibliográfica que nos permitió identificar publicaciones científicas semilla que se obtuvo de bases de datos bibliográficas científicas. A partir de estas publicaciones generamos una revisión sistemática de procesos de desarrollo de software donde localizamos algunas investigaciones que permitió identificar la aplicación habitual en el desarrollo de sistemas embebidos de modelos, técnicas y herramientas, que permitió adoptar estas prácticas y construir un modelo de trabajo denominado BPSSEM (Mejores Prácticas de Software para Sistemas Embebidos). Cómo **resultado** se seleccionó un conjunto de artefactos como los casos de uso, diagrama de clases y se generaron algunas plantillas para soportar la especificación de requisitos, análisis de requisitos, documentación de código y Programación Orientada a Objetos, plan de pruebas funcionales y unitarias que se adoptaron para soportar a cada fase del proceso de desarrollo de software de sistemas embebidos. La BPSSEM se aplicó en el desarrollo del Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos. Finalmente, podemos **concluir** que BPSSEM es un modelo de trabajo para soportar al proceso de desarrollo de software para sistemas embebidos dentro del grupo de investigación.

Palabras claves:

- **EMBEBIDO**
- **METODOLOGÍA**
- **SOFTWARE**
- **SISTEMA**

Abstract

Software Engineering is considered a discipline that deals with the development, management and maintenance of software systems using practices, methods and techniques. There are some types of software, such as basic, development and application software. Among the application software we can mention office, medical, video games and embedded systems. The last ones are computer systems that perform control tasks and are commonly applied in the health, agriculture, military, telecommunications and industrial fields. In the development of this type of systems, problems are commonly identified such as: incomplete definition of the scope of the system, lack of standardization of models for analysis and design, among others that cause the need of methodologies to guide their development. The detailed information is used as a basis for our objective to adopt the best software development practices of traditional and non-traditional methodologies to create a working model to develop quality software for embedded systems. To meet this objective, we applied the methods of bibliographic review, which allowed us to identify seed scientific publications obtained from scientific bibliographic databases. From these publications we generated a systematic review of software development processes where we located some research that allowed us to identify the usual application in the development of embedded systems of models, techniques and tools, which allowed us to adopt these practices and build a working model called BPSSEM (Best Software Practices for Embedded Systems). As a result, a set of artifacts such as use cases, class diagrams were selected and some templates were generated to support the requirements specification, requirements analysis, code documentation and object-oriented programming, functional test plan and units that were adopted to support each phase of the embedded systems software development process. BPSSEM was applied in the development of the Collaborative Aerial Robot System for Cargo Handling with Optimal Resource Consumption. Finally, we can conclude that BPSSEM is a working model to support the software development process for embedded systems within the research group.

Key words:

- **EMBEDDED**
- **METHODOLOGY**
- **SOFTWARE**
- **SYSTEM**

Generalidades

Introducción

En este capítulo se describen las generalidades, definiendo antecedentes históricos con los cuales se determina la evolución que ha sufrido las metodologías y modelos del proceso de desarrollo de software en ingeniería de software. En base a lo expuesto de los antecedentes históricos se procede a definir el planteamiento del problema, así también corresponde al detalle de los objetivos para el lograr a buena consecución el proyecto de investigación.

Por lo tanto, es necesario describir la justificación e importancia del tema propuesto para la investigación y para finalizar se define las hipótesis que es un enunciado no verificado, una vez refutado o confirmado dejará de ser hipótesis y sería un enunciado verificado, en el caso del proyecto de investigación será una hipótesis de trabajo.

Antecedentes

Las metodologías y modelos de Desarrollo de Software han experimentado un proceso evolutivo desde la década de los años 40, que se inicia con las primeras computadoras, y estaba caracterizada por el desconocimiento de modelos y metodologías de desarrollo de software. El software que se desarrollaba, se hacía de manera empírica, lo cual comúnmente llevó a generar proyectos fallidos que no cubrían las necesidades del usuario, provocando la crisis del software (Rai et al., 2014) relacionados al software de sistemas de información como para el software de dispositivos electrónicos. Como ejemplo citamos el caso de Therac-25, máquina de radioterapia que causó la muerte de varios pacientes en diversos hospitales de Estados Unidos y Canadá, debido a las radiaciones de alto poder aplicadas sin

control, las cuales fueron atribuidas a la falta de control de calidad del software embebido de carácter médico.

En respuesta al desarrollo de software empírico, se desarrollaron y adoptaron metodologías y modelos clásicos que progresivamente se fueron incorporando en el proceso de desarrollo de software formalmente. Sin embargo, el crecimiento del desarrollo de software no se detuvo, más aún con la llegada del internet surgen proyectos que se requerían metodologías y modelos que permitan aligerar el proceso de desarrollo, además de contar con la comunicación entre el usuario y equipo de desarrollo, y surge las metodologías agiles. A igual que las metodologías y modelos clásicos, las ágiles o no tradicionales permiten el desarrollo de software para sistemas embebidos, como es el caso de Renault, en sus proyectos de automotriz aplican principios agiles para la estimación del proceso de desarrollo de software embebido (Oriou et al., 2014).

En la actualidad, existen empresas que demandan de software eficiente para mejorar su productividad, calidad, tiempo y costes. Por ello se ha decidido profundizar en la búsqueda metodologías y modelos para el desarrollo de software, con la finalidad de identificar métodos y técnicas para adoptar como mejores prácticas para crear un producto de calidad, de manera que se convierta un modelo de trabajo guía para mejorar el desarrollo de software embebido. Este tipo de software forman parte de un subsistema electrónico de procesamiento programado que realiza funciones específicas (Vega, 2010).

En el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas Sede Latacunga, se desarrollan proyectos de investigación de construcción Sistemas Embebidos centrándose únicamente en el diseño físico hardware y la funcionalidad, donde se puede evidenciar la carencia de procesos de desarrollo de software que garanticen la calidad requerida, programada y entregada a

los usuarios. Lo detallado anteriormente, provoca la necesidad de establecer una propuesta de modelo de trabajo basado en las mejores prácticas de los metodologías tradicionales y no tradicionales de desarrollo de software.

La propuesta de este modelo de trabajo está basada en las metodologías tradicionales y no tradicionales para el desarrollo de software en sistemas embebidos y tiene como punto partida el estudio de metodologías orientadas al desarrollo de software con el propósito de identificar las mejores prácticas que permitan definir un proceso de desarrollo de software para sistemas embebidos.

Planteamiento del problema

El objetivo de la ingeniería de software es la calidad de los productos a desarrollar (Mumphrey, s. f.). Actualmente, encontramos aplicaciones software que apoyan en la toma de decisiones en todas las áreas de conocimiento como la medicina, agricultura, industria, mecánica, electrónica entre otros (Mejía-Neira et al., 2019). En estas áreas en las últimas décadas se evidencia un incremento en la demanda de software embebido que es un tipo de aplicación software.

Los sistemas embebidos están compuestos por hardware y software, dedicados a una sola actividad. Una de las características más importantes de los sistemas embebidos es la capacidad de realizar tareas muy importantes con pocos recursos computacionales, lo cual representa un alto grado de limitaciones para su desarrollo y operación. Este tipo de sistemas se encuentran en flexibilidad con los procesos de desarrollo de software, lo cual hace aún más crítico para su desarrollo. Por ello requiere procedimientos apropiados y específicos para la construcción (Silvério Miyashiro & Ferreira, 2014). Uno de los problemas que comúnmente enfrenta la Ingeniería de software es contar con métodos y técnicas que se apliquen como mejores prácticas de desarrollo para satisfacer la demanda de este tipo de software, y permitirá conocer la especificidad de cada componente de software a desarrollar.

El proceso de desarrollo de software embebido es un contexto muy complejo por el número de componentes físicos y lógicos que lo integran. Y pequeños cambios dentro los procesos establecidos puede causar defectos al momento del desarrollo ocasionando situaciones que ponen en peligro la vida, los retrasos pueden generar enormes costos y una productividad insuficiente (Ebert, 2009), así también el descontento de los clientes de este tipo de sistemas, ya que por mucho tiempo buscan obtener características como reutilización, mantenibilidad y flexibilidad, pues históricamente no han sido prioridad en el desarrollo de software para sistemas embebidos (Vega, 2010). Además, desarrollar software embebido para sistemas embebidos o empotrados, presenta grandes retos debido a sus particularidades, se preocupa por el mundo físico y por supuesto por la parte del hardware haciendo a un lado al software, lo cual ha dificultado incluir y aplicar las mejores prácticas de las metodologías de desarrollo.

En el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas Sede Latacunga tiene por objetivo construir Sistemas Embebidos. Para construir este tipo de sistemas no disponen de una guía de buenas prácticas para el proceso de desarrollo de software que se ajuste a la realidad y necesidades de este tipo de sistemas, ya que el proceso de desarrollo de software para SE[1] en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes lo realizan de manera empírica ya que no existe procesos definidos y documentación lo cual dificulta el desarrollo de proyectos software. Los desarrollos que se han realizado han permitido adquirir experiencia, pero en muchos de los casos sacrificando la calidad y la escalabilidad de las aplicaciones software. Los expuesto es el resultado de no aplicar buenas prácticas de desarrollo de software basadas en las metodologías tradicionales y no tradicionales, que le permita al desarrollador conocer exactamente como aplicar o qué tipo de mejores prácticas utilizar, muchas veces la construcción de SE, se centra

en la funcionalidad y el hardware dejando de lado el software por lo que se ha generado conflictos al momento implementar estos sistemas, por lo cual se busca desarrollar un modelo de trabajo que soporte a los profesionales de laboratorio al generar sistemas embebidos.

En base a lo expuesto, se plantea el siguiente problema:

¿Cómo ayudar al proceso de desarrollo de software en sistemas embebidos en el laboratorio de investigación en automatización, robótica y sistemas inteligentes?

Objetivos generales y específicos

Objetivo General

Diseño de un modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales para ayudar al proceso de desarrollo de software en sistemas embebidos.

Objetivos Específicos

- Construir el Marco Teórico para fundamentar las metodologías y modelos tradicionales y no tradicionales de desarrollo de software.
- Diseñar un modelo de trabajo para ayudar al proceso de desarrollo de software de sistemas embebidos.
- Implementar en modelo de trabajo para el desarrollo de software en sistemas embebidos: caso de estudio proyectos de Laboratorio de Investigación del Departamento de Eléctrica y Electrónica.
- Validar el modelo trabajo para ayudar al proceso de desarrollo del producto software en sistemas embebidos.

Justificación e importancia

En la actualidad la tendencia del desarrollo de software se relaciona de forma directa con los sistemas embebidos, el cual consiste en un sistema de computación cuyo hardware y software están diseñados y optimizados para cubrir necesidades específicas en tiempo real, y se encuentran disponibles en cualquier aspecto de nuestra vida. Un horno de microondas, teléfonos celulares, un automóvil y otros dispositivos electrónicos están controlados mediante este tipo de sistemas, por ello es difícil encontrar un dispositivo cuyo funcionamiento no esté basado en un SE.

El desarrollo de Sistemas Embebidos está adquiriendo una importancia significativa y estratégica. En la actualidad, es indiscutible que estos sistemas aportan un valor añadido a los productos en términos de innovación y competitividad. Y es precisamente por la importancia que tiene el tema tratado para la industria de la electrónica y el software, por lo cual se ha decidido realizar la investigación, en donde se propone aplicar las mejores prácticas de las metodologías tradicionales y no tradicionales para desarrollar software de SE. Lo descrito, pretende servir como un punto de partida para aquellas personas que se desenvuelven en el campo objeto de estudio. Asimismo, este estudio servirá como un documento base para todo aquel que pueda estar implicado en el campo del software y la electrónica desde, los docentes, investigadores, estudiantes y hasta la propia industria.

Adicional el modelo de trabajo, obtenido como resultado de la investigación, servirá de instrumento guía para el proceso de desarrollo de software para sistemas embebidos que trabajan en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica; y ellos se benefician desde un primer momento ya que conocerán y aplicaran el modelo que contiene **un conjunto de buenas prácticas**; y a la vez adquirirán el conocimiento para llevar una parte del proceso de ingeniería de software.

Realizar esta propuesta generaría una serie de beneficios relevantes para los profesionales que laboran en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica, entre los que se describe a continuación.

En primer lugar, se podrá contar con las mejores prácticas entre métodos y técnicas para el desarrollo de software que permitan llevar a cabo de manera óptima un conjunto de actividades que comprenden el desarrollo de software para sistemas embebidos. Así también, se tendrá una guía en donde se detalle las mejores prácticas y etapas que se deben llevar a cabo para el desarrollo de software. También permitirá minimizar los principales errores y problemáticas al construir y desarrollar software embebido para sistemas electrónicos. Así también, se puede contar con un modelo que permita estandarizar procesos desarrollo de software aplicando las mejores prácticas para la construcción de sistemas embebidos.

Hipótesis

Si se diseña un modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales **entonces** ayudara al proceso de desarrollo de software en sistemas embebidos del Laboratorio de Investigación del Departamento de Eléctrica y Electrónica.

Variables de la investigación

Variable Independiente. - Diseñar un modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales.

Conceptualización de la variable independiente. - Un modelo de trabajo basado en metodologías tradicionales y no tradicionales de desarrollo es un marco de trabajo que contiene **un conjunto de buenas prácticas, modelos, herramientas, métodos y técnicas** que ayuden al proceso de desarrollo de software de sistemas embebidos.

Variable Dependiente. -Mejorar el proceso de desarrollo de software en sistemas embebidos en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas Sede Latacunga.

Operacionalización de la variable dependientes (Indicadores)

1. Tiempo para desarrollar software embebido.
2. Ayuda al proceso de desarrollo software.
3. Reutilización de recursos.
4. Evaluar a los desarrolladores.
5. Modelo trabajo para desarrollar software de sistemas embebidos.
6. Generar artefactos y documentación.

Marco Teórico

Introducción

En este capítulo se describe la fundamentación teórica de la investigación, que abarca las siguientes secciones: Antecedentes históricos en donde se detalla la evolución cronológica de las metodologías y modelos en el proceso de ingeniería de Software.

Antecedentes conceptuales y referenciales, en donde se habla de la caracterización gnoseológica del proceso de desarrollo de software, ingeniería de software y modelos de desarrollo; además, se incorpora la propuesta de la caracterización tecnológica de las metodologías tradicionales y no tradicionales, sistemas embebidos y componentes. Se finaliza con la sección de los antecedentes contextuales, en donde se ha recopilado una serie de estudios permiten justificar la existencia del problema planteado en la presente investigación.

Evolución de las metodologías y modelos en la ingeniería del software

En este contexto se describe la evolución del tiempo de las metodologías junto a los métodos, técnicas y mejores prácticas aplicadas para el proceso de desarrollo de software.

Primera Etapa Cronológica 1940 – 1950

En esta etapa cronológica, el coste del hardware era superior al software, por lo tanto, su importancia era relativamente menor. Se consideraba además que el software se podía desarrollar de la misma forma que se desarrolla el hardware; y, de hecho, los primeros ingenieros que se ocupaban del software eran los mismos que desarrollaban el hardware (Piattini, s. f.).

Segunda Etapa Cronológica 1960 - 1970

En esta etapa cronológica, se presenta uno de los primeros sistemas auto-contenidos, usados en el continente americano fue la computadora de guía Apollo, desarrollada en el laboratorio de instrumentación de la Universidad de tecnología de Massachusetts, el sistema está basado en microprocesadores y microcontroladores comenzaron a emplearse en el control de tareas aeronáuticas y espaciales.

En la década de 1960, se trabaja con la idea de Codificar y Corregir (Code and Fix), este modelo se inicia con la idea general de lo que se necesita construir. El modelo implementaba código y después se pensaba en los requisitos, diseño, validación y mantenimiento. En esta década, se inicia la Crisis del Software, llamada así por los problemas que surgieron a medida que se desarrollaba software.

En la década de 1970, las empresas iniciaron a comprobar que el valor del software, era superior a los del hardware. Para esta época se propone utilizar un modelo de ciclo de vida en cascada y la formación de los ingenieros de Software se centra en aplicar Metodologías estructuradas para el desarrollo de software de Sistemas de Información y Sistemas Embebidos. En esta década aparece una nueva disciplina llamada “Ingeniería Software”, en donde se promueve aspectos técnicos del software y gestión de datos.

Tercera Etapa Cronológica 1970 - 1985

Esta etapa está marcada, establecer soluciones para resolver la “Crisis del Software”. En 1968 en la Conferencia de Ingeniería de Software OTAN, establece el termino Ingeniería de Software para minimizar la crisis. La historia de la ingeniería de software entrelazada con el hardware y software.

En el año de 1970, se inicia a tomar importancia para resolver problemas complejos, para ello se inicia por el análisis por partes o etapas, esto lo realiza en la planeación y administración.

En etapa para el proceso de desarrollo de software, se emplea los Ciclos de vida, con la finalidad de establecer estados por donde pasan un producto software de Sistemas de información y embebidos. Esto permitió definir una estructura para el proceso de desarrollo software como un aporte a la resolución de problemas. A este proceso se denomina Modelos de ciclo de vida, estos modelos pretendían abarcar el proceso de desarrollo desde la fase inicial hasta final ajustando las necesidades de la empresa. El aporte de dichos modelos al desarrollo del software se focalizo para software de sistemas de información y de sistemas embebidos.

Los modelos aplicados al proceso de desarrollo de software para sistemas de información y embebidos fueron: Modelo en Cascada que es un método convencional permite reconocer requerimientos al iniciar el proyecto, brinda la flexibilidad mayor a la hora de corregir errores y el proyecto se divide en etapas de planeación, análisis, diseño del sistema e implementación. Así, también se llegó a utilizar el Modelo de ciclo de vida en V, un modelo diseñado por Alan Davis, contiene las mismas etapas del modelo en cascada más dos sub etapas de retroalimentación. Como respuesta a las debilidades del modelo en cascada, se utiliza el Modelo evolutivo, este acepta que los requerimientos pueden ser cambiados en cualquier momento, las etapas son planeación y análisis, diseño e implementación y evaluación.

Para el año de 1980, Harlan Mills propone el Modelo de Desarrollo Incremental lo cual es una combinación de elementos del modelo en cascada y la construcción de prototipos. Surge como una forma de reducir el trabajo en el proceso de desarrollo.

Por otra parte, en la década de los 80 el desarrollo para sistemas embebidos no es un tema aislado para el proceso de desarrollo de software de sistemas embebidos. Se inicia el uso y aplicación de metodologías de diseño a continuación, se detalla:

Metodologías de Descripción y Síntesis, el diseñador especifica en primer lugar lo que quería en ecuaciones booleanas o descripciones FSM (Finite-State

Machine), método matemático basado en estados según las entradas al sistema), a continuación, la síntesis herramientas generadas a la aplicación en términos de una netlists (netlists es un método de expresar circuitos electrónicos en texto plano) de nivel lógico (Gallo et al., s. f.).

La metodología KAOS (Knowledge Acquisition Automated Specificaction) se basan en técnicas de Inteligencia Artificial para definir metas y acciones. Una de las desventajas de usar Kaos es realizar el tratamiento de requisitos de manera aislada (Lezcano et al., 2015).

Cuarta Etapa Cronológica 1985-1999

Esta época está marcada por la conceptualización de la Ingeniería de Software, cuya importancia toma fuerza dentro de las empresas. Se inicia el estudio de los objetos como unidades de información, iniciando el proceso evolutivo de desarrollo software.

En 1988 se propone el Modelo Espiral, básicamente consiste en una serie de ciclos que se repiten de forma espiral que se inicia en el centro.

En 1990, la evolución del internet y un entorno cambiante, dio lugar a requisitos rápidos e imprecisos, lo cual dio lugar a trabajar en ciclos cortos para desarrollar un proyecto. Con esta necesidad surge las Metodologías Agiles para el desarrollo de software, estos métodos reducirían el tiempo de ciclo de vida del software lo cual permitió acelerar el proceso de desarrollo de software.

Quinta Etapa Cronológica 2000 al presente

La creciente demanda de software, llevo al crecimiento de métodos más simples y rápidos. Esto llevo al uso de prototipos rápidos, destacando la utilización de las metodologías ligeras completas como Programación Extrema (XP). Es así que las metodologías ligeras se aplican a sistemas más pequeños de tareas específicas, este tipo de sistemas tienen un enfoque alternativo más simple y rápido. Para el año 2001,

adoptan el nombre de métodos agiles, y estas contempla un conjunto de métodos de Ingeniería de Software basado en el desarrollo iterativo e incremental, teniendo presente los cambios y respondiendo a estos mediante la colaboración de un grupo de desarrolladores.

En esta década las tendencias de desarrollo de software para sistemas empotrados, se consideran una necesidad urgente por lo tanto es urgente emplear técnicas y métodos para el proceso de desarrollo. Crear desde cero un software embebido, es una práctica común sin embargo las necesidades y el avance tecnológico resulta complejo tener un proceso de desarrollo que permita obtener un producto de calidad, debido a que las aplicaciones empotradas utilizan en tiempo real con entornos con alto grado de seguridad y por el tamaño.

A continuación, se exponen varias metodologías propuestas en la literatura de sistemas embebidos, cada una de estas metodologías sigue de proceso o flujo de ejecución que se ajusta a uno de modelos de ciclo de vida, ya se lineal, en cascada, cíclico o evolutivo, etc. lo cual los hace a cada uno más propenso a algún tipo de aplicación, posterior al desarrollo de estas metodologías, se incluye una Tabla resumen con las ventajas, desventajas y aplicaciones más comunes de cada metodología.

La Metodología de Bottom-Up. -Se abordan inicialmente los problemas físicos, lo cual produce buena robustez a nivel de hardware, se puede aplicar en sistemas pequeños y pocos complejos. En cambio, la metodología Top-Down permite tener una meta fija de desarrollo se pueden aplicar para sistemas de complejidad media. La metodología Basada en la plataforma, esta consta de cinco fases Exploración, Planificación, Desarrollo, Lanzamiento y Mantenimiento. La metodología de la integración es utilizada en las empresas en donde trabajan con diferentes equipos y las tareas son repartidas, esta metodología desarrolla el hardware y software por separado, y una vez se posea todos los componentes se realiza una

tarea de integración. Las fases de la metodología de integración son el diseño inicial, diseño hardware, prototipo de hardware, diseño de software, prototipo de software, integración hardware – software prototipo, pruebas - depuración y producto final. En los sistemas embebidos hay que tener en cuenta durante su diseño tanto el hardware como el software. Por lo tanto, utilizar la metodología Codiseño de Hardware/Software, el objetivo es encontrar la correcta combinación de hardware y software que resulte en el producto más eficiente de acuerdo a los requisitos. Los beneficios del desarrollo de hardware y software al mismo tiempo incluyen evitar el tiempo extra que participan en el desarrollo, una tras otra, y la detección temprana de errores en el juego de software y hardware (Gallo et al., s. f.), las fases son requerimientos y compromisos, particionamiento, requisitos y compromisos de hardware, diseño de hardware, verificación de hardware, requisitos y compromisos de software, diseño de software, verificación de software, manufactura y pruebas. Hoy en día, las empresas de productos electrónico de consumo tienen sus propias metodologías de sistemas desarrolladas a partir de las funciones de sus productos. A continuación, se propone utilizar las metodologías mixtas o propias, es decir se toma una parte de la metodología de integración para diseñar el concepto del producto, desarrollo y se toma partes de la metodología de codiseño.

Al revisar bibliografía también se encontró que en la actualidad se han desarrollado diferentes investigaciones respecto a la construcción de sistemas embebidos, lo cual se busca un desarrollo de software embebido de calidad, para ello también se emplean marcos de desarrollo que tienen por objetivo establecer fases para guiar a los desarrolladores en proceso de desarrollo de sistemas embebidos. A continuación, se presentan propuestas actuales para el desarrollo de sistemas empotrados, que parten a partir del 2000 hasta la fecha.

Embedded Software Component Model (ESCM). – El modelo se orienta a la especificación, verificación y la composición de software embebido. ESCM cuenta con

tres elementos interfaz, componentes y conector en combinación con contratos. ESCM hace énfasis en la especificación de requisitos funcionales de los sistemas embebidos, mediante un contrato de NFPS. El modelo se orienta se basa en tres elementos interfaz, componentes y conector.

Save Integrated Development Enviroment (Save-IDE). - El modelo reúne las herramientas y técnicas necesarias para el desarrollo de sistemas embebidos. Este modelo está diseñado con un enfoque top-down que permite el re utilización de componentes. El modelo consta de tres fases como es diseño, análisis y reutilización.

Resource Model For Embedded Systems (Remes). Es un marco de modelado y análisis de recursos embebidos. El modelo da soporte a recursos genéricos como memoria, puertos, almacenamiento, energía, comunicación, CPU y buses.

Rapid Object-Oriented Process For Embedded Systems (ROPES). Se plantea como un proceso completo para el desarrollo de Sistemas Empotrados o Embebidos. Este modelo se fundamenta en UML y trabaja con proceso de desarrollo iterativo, se basa en las tendencias de Ingeniería de Software, análisis de riesgos y calidad del software. El proceso de desarrollo de software se organiza en cuatro fases análisis, diseño, traducción y pruebas, estos son los artefactos resultantes por cada fase en base a UML.

Model Driven Design of Embedded Systems (ModES). Este modelo se basa en un meta modelo, el cual puede ser utilizado por varias tareas durante el proceso de diseño de sistemas embebidos. Su aplicación se inicia desde la especificación hasta la generación y la síntesis del software/hardware de esta manera se generan artefactos específicos del diseño, que servirán cumplir las tareas del diseño.

Simplified Parallel Processes (SPP). Este modelo proporciona una solución enfocada a mejorar los procesos de software y se relaciona con el modelo CMMI

(Modelo de Madurez de Capacidades de Integración) en el nivel 2 y3, con la particularidad que se aplican a pequeñas y medianas empresas de desarrollo de software embebido o empotrado. El modelo expuesto consta de tres capas, en el nivel superior se establecen los procesos para la gestión del proyecto, nivel intermedio se encuentra los procesos para el desarrollo del proyecto y en el nivel inferior se encuentra el apoyo a los procesos del proyecto. Es así que el modelo SPP, se basa en un ciclo de vida que se divide en seis fases Desarrollo de requisitos, Investigación técnica, Diseño del sistema, Implementación y pruebas, Pruebas del sistema, Prueba Beta – Aceptación y Servicio y mantenimiento.

Product Focused Software Process Improvement. Este modelo es el resultado de un trabajo doctoral, pues se enfoca en mejorar los procesos de software y calidad de sistemas embebidos. Se compone de la combinación de tres áreas de trabajo y son: Ingeniería de Requisitos es decir lo que hay que hacer; Ingeniería de Procesos que permite diseñar, construir, y adaptar métodos, técnicas y herramientas para el desarrollo de un producto de software específico y; el Programa de medición que se enfoca en el diseño e implementación de procesos.

Conceptos generales en el proceso de desarrollo de software

Los conceptos para conocer y definir de una mejor manera el proceso de desarrollo de software, se cita los siguientes:

Ingeniería de Software

Según Ian Sommerville en su libro “Ingeniería de Software” lo define: “Como una disciplina de ingeniería que comprende todos los aspectos de la producción de software (Ignacio & Paola, 2015)”.

En su libro “Metodologías del análisis estructurado del sistema” de Jesús Barranco de Areba para quien la Ingeniería de Software: “Es el establecimiento y uso de principios de ingeniería orientados a obtener software económico, que sea fiable y funcione de manera eficiente sobre maquinas reales (Areba, 2001)”.

De los conceptos investigados puedo concluir que la Ingeniería de Software, es un conjunto de métodos, herramientas y técnicas que se aplica en el proceso de desarrollo de software.

Software

De acuerdo al estándar 729 de la IEEE, es un conjunto de programas de cómputo, procedimientos, reglas, documentación y datos que forman parte de las operaciones de un sistema de computación.

En su libro “Introducción al análisis de sistemas y la Ingeniería de Software” de Roberto Cortes Morales lo define como: “Los programas de computadora, estructuras de datos y la documentación asociada, que sirven para realizar el método lógico, procedimiento o control requerido (*Introducción Al Análisis de Sistemas Y la Ingeniería de Software*, s. f.) “.

Con la revisión bibliográfica se puede comprender que el software son instrucciones para comunicarse con el ordenador, en si el software son programas de computador y dispositivos.

Tipos de Software

A continuación, se describe la clasificación y tipos de tipos de software, que se muestran en la ***Tabla 1.***

Tabla 1

Clasificación y Tipos de Software

Clasificación	Tipo	Descripción
Por su función	Software de sistemas	Son programas escritos para servir a otros programas.
	Software de gestión de producto	Son programas que permiten gestionar y administrar procesos de negocios.
Por su estructura	Funcional	Son funciones de tipo matemático, y definen objetos específicos de la aplicación.
	Orientada a objetos	El software es una colección de objetos discretos.
Por su plataforma de ejecución	Sistemas embebidos	Software diseñado para realizar una o pocas funciones, utilizadas en tiempo real.
	Sistemas de computación distribuida	Colección de computadores separadas de manera física y conectadas mediante una red.
	Sistemas de cómputo ubicuos	Es un dispositivo electrónico que puede portar una persona en su vestimenta.

Nota: En esta Tabla muestra la clasificación y tipo de software.

Proceso de desarrollo de software

Los procesos de desarrollo de software poseen reglas preestablecidas, son aplicados en la creación del software. El objetivo de los procesos de desarrollo es transformar las necesidades del usuario en un producto software.

El proceso de desarrollo de software no es único. No existe una manera universal de manejar este proceso para abastecer los diferentes contextos en los que se aplica, pero a pesar de esto existe un conjunto de actividades comunes y fundamentales que son:

- **Especificación de software:** Define la funcionalidad y restricciones de operación que debe cumplir el software.
- **Diseño e Implementación:** Permite el diseño y construcción del software de acuerdo a la especificación.
- **Validación:** El software creado, es necesario validar para cerciorar que cumpla con los requisitos del cliente.
- **Evolución:** El software desarrollado, debe evolucionar para adaptarse a las necesidades del cliente.

Modelos de desarrollo de software

El modelo de desarrollo de software es una representación resumida del proceso para el desarrollo de software. Se puede considerar los modelos como marcos de trabajo del proceso y se pueden adaptar para crear procesos más específicos. A continuación, se describe las más importantes:

Modelo en Cascada

Este modelo se derivó de procesos de sistemas más generales y sus principales fases son:

1. Análisis y definición de requerimientos
2. Diseño del sistema y software
3. Implementaciones pruebas de unidades
4. Integración y prueba del sistema

5. Funcionamiento y mantenimiento

6. Modelo de desarrollo evolutivo Espiral

Boehm propone un modelo de proceso de software evolutivo basado en la naturaleza iterativa de la construcción de prototipos con aspectos controlados del modelo en cascada. Existen dos tipos de desarrollo evolutivo, a continuación, se detalla:

- Desarrollo exploratorio permite trabajar con el cliente y explorar requerimientos, se agregar atributos para la entrega de un sistema final.
- Prototipos desechables permite comprender los requerimientos del cliente, para obtener una versión mejorada de los requerimientos para el sistema.

Modelo de desarrollo basado en componentes

En 1988, Boehm propone el modelo de desarrollo de software basado en componentes, este permite reducir la cantidad de software a desarrollar, permite reducir costos y riesgos con una entrega rápida de software.

En gran parte de proyectos de software existe la reutilización. Tiene un enfoque evolutivo, la reutilización es indispensable para el desarrollo ágil del sistema. Entre sus fases puedo mencionar:

1. Análisis de componentes

2. Modificación de requerimientos

3. Diseño del sistema con reutilización

4. Desarrollo e integración

Notaciones gráficas en el desarrollo de software

Lenguaje Unificado de Modelado UML

Se considera como un lenguaje grafico para el modelado de sistemas software que permite describir y especificar métodos o procesos. UML al ser un lenguaje de modelado permite visualizar, especificar, construir y documentar un sistema informático.

Metodología

Para Johana Gamboa y Cecibel Arreaga en su artículo “Evolución de las metodologías y modelos en el desarrollo de software” lo define como: “- Una referencia al conjunto de procedimientos racionales utilizados para alcanzar el objetivo o la gama de objetivos que rige una investigación científica, una exposición doctrinal o tareas que requieran habilidades, conocimientos o cuidados específicos (Gamboa & Arreaga, 2018)”.

Metodologías en el desarrollo de software

Para Johana Gamboa y Cecibel Arreaga en su artículo “Evolución de las metodologías y modelos en el desarrollo de software” lo define como: “Un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. En un proyecto de desarrollo de software la metodología ayuda a definir: Quién debe hacer Qué Cuándo y Cómo debe hacerlo (Gamboa & Arreaga, 2018)”.

El desarrollo de software es parte fundamental dentro de la Ingeniería de Software. Los equipos de desarrollo trabajan construyendo e integrando software a través de técnicas de análisis, codificación, validación y pruebas. En el caso del software de sistemas embebidos de la misma manera que el software tradicional se debería considerar al menos las etapas de un ciclo de vida. Estas permiten tener un

mayor control en un proceso de desarrollo por parte del equipo involucrado, generando un producto de software eficiente y de calidad (Caiza et al., 2020).

En proyectos de desarrollo de software es primordial la elección de una metodología. Es así, que la elección de la metodología se realiza de acuerdo a las necesidades y actividades del software a desarrollar. Las metodologías de desarrollo de software aplican distintos modelos como es la cascada, incremental, evolutivo y espiral (Castrillón, 2011).

Con respecto a las metodologías, para el desarrollo de software existen dos tipos de corrientes metodológicas llamadas tradicionales y no tradicionales. La diferencia entre estas, es que las llamadas tradicionales buscan desarrollar un software mediante el orden y la documentación, en cambio las llamadas no tradicionales buscan desarrollar un software utilizando la comunicación directa entre las personas involucradas en el proceso de desarrollo. A continuación, se describe cada una de las corrientes metodológicas:

Características de las metodologías de desarrollo del software

A continuación, se describe las características más importantes de las metodologías:

1. Reglas definidas
2. Trabaja con un ciclo de desarrollo
3. Planificación y control
4. Comunicación efectiva cliente y equipo
5. Útiles para proyectos de desarrollo
6. Aplicación de herramientas CASE
7. Adopción de estándares

Metodologías Tradicionales

En la obra “Metodologías de desarrollo de software”, de Carlos Ignacio Rivas y otros, expone que una “metodología tradicional son modelos de proceso prescriptivo, fueron propuestas para poner en orden el caos del desarrollo de software, estos modelos tradicionales fueron propuestos en la década de los 60, con una estructura útil basada en procesos de Ingeniería de Software (Ignacio & Paola, 2015)”.

De acuerdo Andrés Navarro Cadavid y otros, mencionan que las metodologías tradicionales de desarrollo de software “se conciben un solo proyecto de grandes dimensiones y estructura definida, la aplicación de este tipo de metodologías tienen procesos rígidos y no cambian, los procesos son secuenciales con una sola dirección, los requisitos de software se definen al inicio y no se cambian hasta finalizar el proyecto; el tiempo de desarrollo es extenso con poca comunicación con el cliente” (Cadavid, 2013).

Las metodologías tradicionales poseen una guía de trabajo para el proceso de desarrollo del software, esto con la finalidad de obtener un producto software de calidad. Esta metodología inicia con una perfecta planificación y posterior, se inicia el ciclo de desarrollo de software. La metodología es un rigurosa al generar artefactos, definir actividades, roles y herramientas con la finalidad de obtener una documentación detallada del proceso de desarrollo de software.

Metodologías no Tradicionales

En su artículo “Revisión de las metodologías agiles para el desarrollo de software”, de Andrés Navarro Cadavid y otros, mencionan que:” En los años 90 surgieron, nacen las metodologías desarrollo de software ligeras, y más adelante las llamaron agiles o también conocidas como no tradicionales. Estas metodologías nacen como reacción a los modelos y metodologías existentes con la finalidad

aminorar la documentación que implica el uso y aplicación de las metodologías tradicionales (Cadavid, 2013) “.

Según Carmen de Pablos Heredero y otros, las metodologías agiles o no tradicionales “surgen como respuesta a los problemas fundamentales para corregir cierta mala praxis en los proyectos de desarrollo de software (Heredero et al., 2019)”.

Las metodologías agiles o no tradicionales son flexibles, se adaptan a la realidad de lo equipos de desarrollo o proyectos a construir. Estos proyectos son divididos en proyectos más pequeños.

Mejores Prácticas de desarrollo de software

Las mejores prácticas son una práctica técnica o de gestión que ha permitido una mejora sobre procesos de desarrollo que involucra la productividad, costo, tiempo, calidad y satisfacción del usuario (Withers, 2000).

La importancia de utilizar las mejores prácticas ha permitido que instituciones de prestigio como el Instituto de Ingeniería del Software (Software Engineering Institute, SEI), el Instituto de Gestión de Proyectos (Project Management Institute, PMI) y otros, se han enfocado en el estudio y aplicación de las mejores prácticas para el desarrollo de modelos y estándares, mismos que son ampliamente aceptados en el sector gubernamental y privados.

Sistemas embebidos

Definición de Sistemas Embebidos

En el libro “Todo sobre sistemas embebidos: Arquitectura, programación y diseño de aplicaciones prácticas con el PIC18F”, de Sergio Salas Arriarán define como: “Un circuito electrónico digital capaz de realizar operaciones de computación, generalmente en tiempo real, que sirve para cumplir una tarea específica en un producto” (Arriarán, 2017).

Según José Antonio Mercado Fernández en su libro “Sistemas programables avanzados” lo define: “Como un conjunto especifico de operaciones, con un hardware y software dedicados a la maquina donde se encuentran (Fernández, 2019)”.

Según Daniel Benchimol, define “Los Sistemas Embebidos es un dispositivo controlado por un procesador, dedicado a realizar una única tarea o una serie de ella (Benchimol, 2011)”.

De los conceptos investigados puedo concluir que los Sistemas Embebidos o también conocido como Sistemas Empotrado o Integrado, se puede definir que los sistemas embebidos son sistemas de cómputo cuyo hardware y software están diseñados para resolver problemas específicos en tiempo real. Por ello, es necesario trabajar tanto con el hardware y software al crear un sistema de este tipo.

Hardware de Sistemas Embebidos

Hardware es un término que hace referencia a los componentes físicos que lo constituyen, cabe destacar que estos componentes permiten realizar ciertas tareas al ejecutar un software(Deschamps et al., 2009). Por lo antes mencionado es necesario tener conocimiento del funcionamiento de hardware para poder construir sistemas embebidos y el software que se ejecutara en él. A continuación, se procede a describir los componentes más importantes del hardware generalmente es usado en la construcción de sistemas embebidos.

Los microprocesadores son una tecnología de microprocesadores y trabaja en una arquitectura de núcleo computacional que se utiliza para implementar funcionalidades que se necesita en un sistema. Un microcontrolador también forma parte de un sistema embebido, es un circuito integrado que posee las mismas cualidades de un computador de escritorio como un CPU (Unidad Central de Proceso), este componente no incluye algún tipo de dispositivo de comunicación humano máquina. Así también, los sensores son dispositivos eléctricos, electrónicos y

electromecánicos(Pedre, 2017). Para finalizar, uno de los componentes importantes del hardware embebido es la memoria, cuya finalidad es almacenar el procesamiento y transformación de datos. Es necesario mencionar que cualquier funcionalidad de los sistemas embebidos se compone de tres aspectos el procesamiento, almacenamiento y comunicación.

Software de Sistemas Embebidos

El software embebido o empotrado se encuentra ubicado en la memoria de lectura y se utiliza para ejecuta funciones o tareas específicas de un dispositivo electrónico (Vega, 2010). Es decir, se puede tener control de un microondas, una lavadora, un vehículo e incluso se puede aplicar en naves aeroespaciales.

Unas de las características principales del software embebido es interactuar con el mundo físico, a diferencias del desarrollo de aplicaciones empresariales se aleja del mundo físico enfocándose en abstracciones como entidades de información y proceso de negocios. Sin embargo, en el desarrollo del software embebido sucede lo contrario. El objetivo principal no es la transformación de datos, más bien la interacción con el mundo físico. El software embebido se ejecuta en máquinas que no suelen ser computadores. Las maquinas suelen ser automóviles, aviones, teléfonos, equipo del audio, robots, juguetes, los sistemas de seguridad, armas, menús de televisiones, copiadoras, escáneres, clima, control de sistemas, sistemas industriales, entre otros (Vega, 2010).

Categorías del Software Embebido

El software embebido se usa para controlar productos electrónicos que se ejecutan dentro de un microprocesador, microcontrolador, un procesador digital, una compuerta programable y en ciertas ocasiones en un computador adaptada tareas específicas(Pérez Abreu, 2009).

El software embebido presenta dos características el original o básico y el requerido por los equipos. El original o básico constituye un sistema operativo que es indispensable para el funcionamiento del dispositivo. Se crea utilizando lenguajes de programación assembler, C/C++ o VHDL, este tipo de software viene incorporado en el dispositivo desde su salida al mercado. El desarrollo de este tipo de software requiere de personal altamente especializado debido a su complejidad y no se modifica a lo largo de su vida activa.

El segundo tipo que es el requerido por algún dispositivo, trabaja en condiciones específicas cuando se necesite cumplir requerimientos de una determinada comunidad, modalidades o costumbres (Cetinkunt, 2007).

Ámbitos de aplicación del Software Embebido

El ámbito de la aplicación de software embebido, se encuentran en los dispositivos electrónicos, sistemas de comunicación, sistemas de automotores, sistemas domóticas y aplicaciones empresariales. Por lo antes mencionado, es necesario ejemplificar la incorporación del software embebido y se encuentran en una variedad de dispositivos electrónicos comunes, tales como consumibles electrónicos (teléfonos celulares, cámaras digitales, video juegos portátiles, calculadoras, PDAs, etc.), electrodomésticos (hornos microondas, máquinas contestadoras, termostatos, lavadoras, etc.), equipos de oficina (fax, copiadoras, impresoras, scanner), equipos de negocios (caja registradora, sistemas de alarma, lectores de tarjeta y cajeros automáticos), y automóviles (control de transmisión, control de viaje, inyección de combustible, ABS, etc.). Podría decirse que prácticamente cualquier dispositivo que se ejecute con electricidad o ya tiene un sistema computacional embebido o próximamente lo tendrá (Perez Abreu, 2009).

Características del Software Embebido.

Las características que se consideran para el desarrollo de software embebido son la confiabilidad, limitaciones de recursos de hardware y la respuesta en tiempo real (Vega, 2010).

Mejores prácticas de desarrollo de software

Son una práctica técnica o de gestión, que ha demostrado una mejora procesos cómo: desarrollo, costo, calendario, calidad del producto o satisfacción del cliente.

Por lo tanto la importancia de las mejores prácticas ha permitido que instituciones como el Instituto de Ingeniería del Software (Software Engineering Institute, SEI), el Instituto de Gestión de Proyectos (Project Management Institute, PMI), el Instituto de Ingenieros Eléctricos y Electrónicos (Institute of Electrical and Electronics Engineers, IEEE), el Instituto de Gobierno de Tecnologías de la Información (IT Governance Institute, ITGI), la Oficina de Gobierno de Comercio (Office of Government Commerce, OGC) y la Organización Internacional para la Estandarización (International Organization for Standardization, ISO) se enfoquen en el estudio de mejores prácticas para el desarrollo de modelos y estándares de referencia (Withers, 2000), lo anterior mencionado son ampliamente aceptados en los sectores públicos y privados.

Diseño del Modelo de Trabajo Basado en las Mejores Prácticas

Introducción

El diseño del modelo de trabajo para la construcción de sistemas embebidos, es parte fundamental para ayudar al proceso de desarrollo de software embebido. Al aplicar las mejores prácticas de las metodologías tradicionales y no tradicionales o proponer un nuevo modelo basado en las mejores prácticas da lugar a mejorar al proceso de desarrollo de software embebido como definir de requisitos, definir entregables o fases, identificar IDE que se adapten a las necesidades del software embebido, documentación, pruebas y otros. Por lo antes mencionado hace referencia al artículo realizado por (Sandoval et al., s. f.). El presente estudio, tomara como base este modelo de trabajo para ser aplicado en el Laboratorio de Investigación del Departamento de Eléctrica y Electrónica caso de estudio proyecto de investigación de Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos sobre el módulo de *"Desarrollo de plataforma de comunicación a través de Red Cedia".*

En este capítulo se desarrollará un modelo de trabajo aplicando las mejores prácticas de desarrollo de Ingeniería de Software, se iniciará con una revisión literaria de publicaciones que estén alineadas al proyecto de estudio las metodologías de tradicionales y no tradicionales, se realizará un segundo estudio de referencia de las mejores prácticas que se ha elegido con la finalidad de adoptarlas y estas servirá de base en el diseño del modelo de trabajo propuesto. Una vez establecido el modelo de trabajo de desarrollo de software embebido se continuará al desarrollo del proyecto propuesto.

Fases para extracción de publicaciones científicas

Para el desarrollo del proyecto propuesto, de acuerdo a las diferentes fuentes consultadas sobre el proceso metodológico para realizar la revisión de la literatura se

obtuvo que las fases principales como: Planeación, Selección, Extracción y Ejecución de publicaciones científicas enfocadas a la aplicar las mejores prácticas en el proceso de desarrollo de software embebido.

Planeación

En esta etapa se procede a identificar el campo de estudio y selección de las fuentes de información.

Descripción del contexto de la investigación. - Este trabajo pretende llevar a cabo una revisión de la literatura de las mejores prácticas de las metodologías tradicionales y no tradicionales aplicadas en el dominio del software embebido o sistemas embebidos en diferentes áreas como la industria, investigación, agricultura, educación y otros.

Estrategia de búsqueda bibliográfica. -En esta etapa se realiza la búsqueda y lectura artículos semillas para extracción de palabras claves.

La búsqueda se desarrolló entre las publicaciones indexadas en las bases de datos científicas como IEEExplorer, Elseiver, ScienceDirect, ResearchGate y Google Schoolar arrojando estudios de los últimos 20 años. Los resultados obtenidos serán una guía para identificar las mejores prácticas aplicadas para la construcción de un software o sistema embebido.

Una vez seleccionadas las bases de datos y artículos semillas, se eligieron los descriptores o palabras claves. Las palabras claves servirá para comenzar la búsqueda literaria, para el desarrollo de la investigación se utilizarán palabras como MEJORES PRACTICAS, SOFTWARE EMBEBIDO o SISTEMAS EMBEBIDOS y otros.

Criterios de inclusión y exclusión. Para filtrar los estudios obtenidos en diferentes bases de datos científicas se aplicaron los siguientes criterios de inclusión y exclusión:

Se pueden elegir todos aquellos artículos científicos que estén relacionadas con mejores prácticas, software embebido o sistemas embebidos, proceso de desarrollo, metodologías agiles o metodologías tradicionales.

1. Se deben incluir estudios en dos idiomas inglés y español.
2. Se deben incluir publicaciones científicas indexadas entre las más comunes son IEEExplorer, Google Scholar, Science Direct y otros de carácter latino.
3. Se incluyeron artículos científicos, revisiones sistemáticas, tesis, casos de estudios.
4. Se incluyeron publicaciones científicas publicadas en conferencias, workshops, actas, encuestas y otros.
5. Se excluyeron artículos que se no estén en los idiomas inglés y español.
6. Se excluyó publicaciones científicas que no esté bien definido su contenido y estructura.
7. Se excluyeron publicaciones hibridas que no estén a fin a las palabras claves y resumen.

Generar las cadenas de búsqueda. - En este estudio se revisó literatura a través de bases de datos bibliográficas de contenido científico-técnico

Para este estudio se revisó la literatura a través de bases de datos bibliográficas de contenido científico-técnico de artículos revistas, libros, tesis, congresos, etc, de contenido referente al tema de investigación. Entre las bases de datos consultadas son Elseiver, IEEE (Instituto de Ingenieros Eléctricos y Electrónicos), Springer y otros. Para realizar la búsqueda de publicaciones científicas se recurrió a uso de motores de búsqueda como la IEEExplorer, Science Direct,

Google Scholar, Research Gate, SpringerLink y otros, desde el año 2000 al 2020. Para esto se generó las siguientes cadenas de búsqueda:

- ((“BEST PRACTICE”)) AND ((SOFTWARE EMBEDDED)) OR ((SYSTEM EMBEDDED)).
- ((EMBEDDED)) AND ((SYSTEM)) AND ((TRADITIONAL)) AND ((METHODOLOGY)) AND ((SOFTWARE)) AND ((DEVELOPMENTE)) AND ((PROCCESS))

Mediante las cadenas de búsqueda arrojaron 5.654 publicaciones filtradas desde el año 2000 hasta el 2020. Las cadenas de búsqueda se ajustaron de acuerdo a las necesidades de la investigación hasta obtener las publicaciones necesarias y adecuadas para el estudio propuesto se filtró 53 artículos de las diferentes bases de datos científicas.

Generar el instrumento para la extracción de datos. - Una vez establecida las estrategias de búsqueda bibliográfica, definidas las bases de datos científicas, identificadas las palabras claves y generado las cadenas de búsqueda se procedió a crear un instrumento para extraer datos relevantes de las publicaciones científicas, véase **Figura 1**.

Figura 1

Instrumento de extracción de datos de publicaciones científicas

SID	Autores	Año	Fuente (Source)	Editorial (Publisher)	Tipo de publicación	Titulo
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	15a Conferencia y Taller Internacional Anual de IEEE sobre Ingeniería de Sistemas Basados en Computadoras (ecbs 2008)	IEEExplorer	Artículo	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil
S02	P. Manhart ; K. Schneider	2004	Actas. 26 ° Congreso Internacional de Ingeniería de Software	IEEExplorer	Artículo	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018		ELSEIVER	Artículo	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado

Nota: En esta figura, se observa el diseño de un instrumento de recolección de datos para el análisis de artículos científicos, en donde se puede observar el SID (código), autores, año, fuente, editorial y tipo de publicación y el título de la publicación. (ver instrumento completo: https://docs.google.com/spreadsheets/d/1wCIDFugcVi-6ODSLSeTgJt_q2VVr9GS_YgqXGLHPoyQ/edit?usp=sharing)

A continuación, se procede a describir cada uno de los campos que forman el instrumento de extracción de datos:

1. En el campo *SID* se asigna un identificador a las publicaciones extraídas de las bases de datos.
2. En el campo *Autores* se encuentra el nombre de los autores de las publicaciones científicas.
3. En el campo *Año* se encuentra el año de publicación de los artículos científicos.
4. En el campo *Fuente* se encuentra la fuente en donde fue realizada la publicación.
5. En el campo *Editorial* se refiere a la editorial, es decir se dedica a exponer los artículos científicos.
6. En el campo *Tipo de publicación* se describe que tipo de publicación es, entre los más comunes tenemos artículos, caso de estudios y otros.
7. En el campo *Titulo del artículo* se muestra el nombre de la publicación seleccionada.

Selección de estudios

Para la selección de las publicaciones o estudios primarios se realizó mediante los siguientes filtros de revisión.

Primer Filtro:

- **Título:** se realizó una revisión de los títulos de las publicaciones obtenidas en las bases de datos científicas.

- **Resumen o Abstract:** a partir de la revisión del título se procede a revisar la lectura del abstract.
- **Palabras claves:** permitirá relacionar las publicaciones con el tema de estudio.

Segundo filtro

- **Texto completo:** si las publicaciones pasaron el primer filtro se realiza una lectura y análisis completo de los artículos, enfatizando en el área de investigación.

Extracción

En la fase extracción, se procede a la gestión y depuración de los resultados es decir permitirá filtrar, seleccionar y clasificar los estudios ya identificados en las fases anteriores. Se procede a la evaluación de las publicaciones si está disponible o no para su extracción desde las bases científicas.

Ejecución

En esta etapa se procede a ingresar las cadenas de búsqueda y se procede a chequear en las bases de datos científicas y otras bases electrónicas para extraer publicaciones relacionadas con las mejores prácticas del proceso de desarrollo de desarrollo de software embebido o sistemas embebidos, para lo cual se estableció los siguientes parámetros.

Ingreso de las cadenas de búsqueda en motores. - En esta sección se procede a ingresar las cadenas de búsqueda en IEEExplore, Science Direct, Google Scholar, Research Gate y otros. A continuación, se muestra las cadenas de búsqueda establecidas por cada motor de búsqueda.

- **Cadena de IEEExplore**

Los filtros establecidos para este motor de búsqueda son desde el año 2000 al 2020, y los resultados obtenidos fueron 47 publicaciones tanto en conferencias, cursos, revistas, artículos de acceso y libros.

(((((("All Metadata":EMBEDDED) AND "All Metadata":SYSTEMS) AND "All Metadata":SOFTWARE) AND "All Metadata":DEVELOPMENT) AND "All Metadata":PROCESS) AND "All Metadata":TRADITIONAL) AND "All Metadata":METHODOLOGIES)

Los filtros establecidos para este motor de búsqueda son desde el año 2000 al 2020, y los resultados obtenidos fueron 30 publicaciones tanto en conferencias, cursos y revistas.

(((((("All Metadata":EMBEDDED) AND "All Metadata":SYSTEMS) AND "All Metadata":SOFTWARE) AND "All Metadata":DEVELOPMENT) AND "All Metadata":PROCESS) AND "All Metadata":agile) AND "All Metadata":METHODOLOGIES)

Los filtros establecidos para este motor de búsqueda son desde el año 2000 al 2020, y los resultados obtenidos fueron 131 publicaciones tanto en conferencias, cursos, revistas, libros y otros.

(((((("All Metadata":good) AND "All Metadata":practice) AND "All Metadata":embedded) AND "Al Metadata":software) AND "All Metadata": development)

- **Cadena de Science Direct**

  ((“BEST PRACTICE”)) AND ((SOFTWARE EMBEDDED)) OR ((SYSTEM EMBEDDED)).

- **Cadena Google Scholar**

((“BEST PRACTICE”)) AND ((SOFTWARE EMBEDDED)) OR ((SYSTEM EMBEDDED)) OR ((SOFTWARE INTEGRATED))

- **Research Gate**

  ((“BEST PRACTICE”)) AND ((SOFTWARE EMBEDDED)) OR ((SYSTEM EMBEDDED)).

  (METHODOLOGY) OR (PRACTICE) OR (TECHNIQUE) AND ((SYSTEM EMBEDDED)) OR ((SOFTWARE INTEGRATED))

Generar lista de artículos que cumplen criterios. - Para generar la lista de artículos y que cumplan con los criterios antes mencionados se consideró los siguientes parámetros:

- **Selección de estudios primarios. -**Después de aplicar los filtros ya mencionados con anterioridad, se procede a seleccionar 52 publicaciones de tipo primario, y los resultados se pueden evidenciar en la **Figura 1,** descrito en secciones anteriores**.**
- **Definición de criterios para el análisis de artículos. -**Para el respectivo análisis de los trabajos primarios seleccionados se definieron una lista de criterios los cuales servirán de guía para evaluar y comparar las publicaciones entre sí, y obtener las mejores prácticas de desarrollo aplicadas en la construcción de sistemas embebidos.

- **Registrar los datos en el instrumento de selección. -**Para el registro de las publicaciones que cumplen con los criterios de selección, se generó un instrumento el cual se fue adaptando de acuerdo a las necesidades del registro de publicaciones.

A continuación, se presenta un análisis de los artículos que estarán en discusión por los criterios definidos en el esquema de caracterización. Se puede observar que la cantidad de estudios primarios utilizados en el desarrollo del proyecto de investigación. Para ello se contempló los siguientes parámetros.

En una revisión de la literatura, las publicaciones ya seleccionadas son cuidadosamente leídas y los resultados de los estudios primarios identificados serán utilizados para un análisis exhaustivo del título, resumen y palabras claves.

Cada uno de los campos antes mencionados permitió obtener información coherente y relacionada al tema de investigación que se enfoca en la identificación de las mejores prácticas para el proceso de desarrollo de software de metodologías tradicionales y no tradicionales para sistemas embebidos.

Con las cadenas de búsqueda generadas y la selección de publicaciones se identificó alrededor de 300 estudios entre las diferentes bases bibliográficas.

Los nuevos campos establecidos en el instrumento de extracción de datos y de acuerdo a los criterios establecidos en secciones anteriores se obtuvo 52 publicaciones primarias que servirán para el estudio.

Para el análisis de las investigaciones se agregó un nuevo campo denominado **"*Enfoque de investigación*"** que describe la aplicación de un modelo, metodología o herramienta para la construcción de Sistemas Embebidos, como se muestra en la **Figura 2**.

Figura 2

Instrumento de análisis de artículos científicos

SID	Autores	Año	Tipo de publicación	Titulo
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	Artículo	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil
S02	P. Manhart ; K. Schneider	2004	Artículo	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Artículo	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado

Nota: En esta figura, se observa el diseño de un instrumento de recolección de datos para el análisis de artículos científicos, en donde se puede observar el SID (código), autores, año, tipo de publicación y el título de la publicación.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1YKzUqDmQJcBi42pRmFeBrdcnkfGn-zt-8ioOIRK0uIs/edit?usp=sharing)

Identificación de las mejores prácticas

De las publicaciones científicas presentadas en los apartados anteriores se ha identificado las mejores prácticas del proceso de desarrollo de software de las metodologías tradicionales y no tradicionales aplicadas en la construcción de Sistemas Embebidos. La **Figura 3,** tiene como objetivo conocer el alcance de la aplicación de las mejores prácticas al aplicarlos en el desarrollo de SE, así como conocer el campo o dominio en donde se presenta su uso. Los criterios están basados en la revisión de la literatura.

Figura 3

Identificación de las mejores practicas

SID	Autores	Año	Titulo	Tipo de contribución	Campo/Dominio
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño	Pruebas unitarias y funcionales. Requisitos funcionales y no funcionales. Ciclos de vida. Funciones y responsabilidades.	Experiencia Industrial
S02	P. Manhart ; K. Schneider	2004	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo	Prueba unitaria. Especificación de requerimientos. Ciclo de vida.	Automotriz
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado	UML. Casos de uso y Actividad. Requerimientos funcionales y no funcionales.	Industria de Software

Nota: En esta figura, se observa el diseño de un instrumento de recolección de datos para identificar las mejores prácticas del proceso de desarrollo de software, en donde ya se define el campo o dominio de los artículos científicos seleccionados que tratan de la construcción de sistemas embebidos.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1qA5E-Y1vR2RPsY8iY2zJLXmGyJD25Ar-QO5HOEP8GDY/edit?usp=sharing)

A continuación, se describen los criterios más relevantes:

SID: con este criterio se podrá tener un identificador único para las publicaciones científicas, que servirán para el proyecto de investigación.

Autor: en este campo se anota el nombre de la organización o investigador que realizo la publicación.

Año: en esta casilla se anota el año de publicación del artículo científico.

Título del artículo: este criterio permite conocer el título de articulo puede estar los idiomas inglés y español.

Tipo de contribución: describe el fin, es decir la contribución de una metodología en el proceso de desarrollo de software embebido. Las mejores prácticas que se puede considerar como una contribución son: requisitos funcionales y no funcionales, diagramas de uml, lenguajes de Programación Orientada a Objetos, pruebas unitarias, de aceptación, documentación, sprints y otros.

Campo/Dominio: se describe el campo o dominio donde se aplica o se usa las mejores prácticas de desarrollo de software. Con este criterio se puede determinar el impacto de la aplicación de las mejores prácticas en el proceso de desarrollo de SE en diferentes campos como la industria automotriz, aeroespacial, educativo, agricultura y otros. Por lo tanto, se procede a mostrar el instrumento de selección de las mejores prácticas como se pudo observar en la **Figura 3.**

El contexto y análisis de las mejores practicas

El uso y aplicación de las mejores prácticas para el desarrollo de software, son un conjunto de métodos, técnicas y herramientas que permiten llevar a cabo el adecuado proceso de desarrollo de software.

Las mejores prácticas de desarrollo de software se han creado a partir de la experiencia en miles de proyectos de software con los que es posible eliminar los problemas que se presentan al desarrollar software sea para sistemas de información y sistemas embebidos.

Para establecer un buen desarrollo de software es necesario contemplar algunos las fases más comunes aplicados en el desarrollo de software como Análisis, Diseño, Codificación, Pruebas e Implementación (Sandoval et al., s. f.), estas fases permiten identificar las mejores prácticas y como soporte a estas prácticas se elaboran artefactos. Estos artefactos pueden estar presentes de acuerdo a cada fase de desarrollo.

Identificación formal de las mejores practicas

En esta sección se diseña un instrumento para la identificación y clasificación de las mejores prácticas utilizadas en el proceso de desarrollo software de Sistemas Embebidos.

Una vez analizada las publicaciones científicas y como resultado de esta investigación, los métodos, herramientas y técnicas consideradas como mejores prácticas y usadas para el desarrollo de Sistemas Embebidos son variantes dependiendo del campo o dominio, estas variaciones se presentan entre compañías e institutos de investigación, así también dentro de las mismas empresas y universidades, lo cual es un elemento crucial para mejorar proceso de desarrollo del producto final.

A partir de la revisión de la literatura se identificaron las mejores prácticas de desarrollo. Dentro de las publicaciones científicas se observó la aplicación de las mejores prácticas para el desarrollo de SE y se encuentran en las fases como: análisis, diseño, codificación y pruebas (Gabriel, s. f.).

Para la construcción de SE es necesario diseñar las funcionalidades del software de Sistema Embebido para ello, de las 52 publicaciones científicas motivo de investigación se extrajo las mejores prácticas que ayuden a realizar esta actividad. Para la identificación se estableció valores **“0” si no existe una mejor práctica** y **“1” si existe una mejor práctica**.

A continuación, se procede a describir la estructura del instrumento de análisis e identificación de mejores prácticas identificadas en las distintas fases.

Fase de Análisis

Es la primera etapa que se debe realizar para desarrollar un producto de software de cualquier tipo. Esta fase corresponde a tener una comunicación directa

con el cliente para obtener los requisitos necesarios para la construcción del software.

Como resultado del análisis de requisitos con cliente se genera artefactos dependiendo del tipo de metodología que se vaya a utilizar para el proceso de desarrollo.

El resultado de un adecuado análisis de requisitos permite generar documentación como *Especificación de requerimientos, Diagramas UML, Producto backlog, Sprints, Historias de Usuario*, estos son considerados como mejores prácticas de desarrollo. Los artefactos antes mencionados se generan a partir de una metodología tradicional o no tradicional.

Descripción del instrumento

En resumen, se ha identificado las mejores prácticas de la Ingeniería de Software aplicadas en el proceso de desarrollo de software para Sistemas Embebidos. La Figura 4, resume las mejores prácticas identificadas en la fase de análisis.

A continuación, se procede a describir los nuevos campos agregados al instrumento.

- **Metodología:** este criterio determina si la publicación científica seleccionada es de tipo tradicional (tradicional) y no tradicional (ágil).
- **Especificación de requerimientos**: este campo se define el comportamiento del sistema que se va a desarrollar, dentro de la especificación de requerimientos se ha identificado: *Roles, Requisitos Funcionales, Requisitos no Funcionales, Diagrama de casos de uso, Diagrama de actividades, Diagrama de flujo*
- **Product Backlog:** describe un listado de todas las tareas que pretenden realizar dentro de un proyecto software.

- **Sprints:** describe una pequeña lista de tareas que se deben realizar un en corto tiempo.
- **Historias de usuario:** este campo se permite conocer historias de usuarios y son descripciones cortas y simples.

Figura 4

Identificación de las Mejores Prácticas: Fase Análisis

SID	Autores	Año	Titulo del Artículo	Metodología	Análisis								
					Especificación de requeremientos			Diagramas UML			Product Backlog	Sprints	Historias de usuario
					Roles	Requisitos funcionales	Requisitos no funcionales	Diagramas de Caso de Uso	Diagrama de Actividades	Diagrama de flujo			
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil	Tradicional	1	1	1	0	0	0	0	0	0
S02	P. Manhart ; K. Schneider	2004	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria	Tradicional	0	1	1	0	0	0	0	0	0
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado	Tradicional	0	0	0	1	1	0	0	0	0

Nota: Esta figura, muestra el instrumento de identificación de las mejores prácticas en la fase de Análisis, se puede observar criterios como metodología de desarrollo de software, en el criterio de Análisis tenemos sub criterios como especificación de requerimientos, diagramas UML, product backlog, sprints e historias de usuario. (ver instrumento completo: https://docs.google.com/spreadsheets/d/1XoBnXkQ16OI5at3LC2NadtQ0bg1M6tFo3URnhRLBWU/edit?usp=sharing)

Fase de Diseño

Esta fase determina la infraestructura que va sostener el producto software. Aquí se preparan todas las funcionalidades necesarias en función a los requerimientos detectados en el análisis. En esta etapa definimos *Modelo de datos, Diagramas UML y Prototipos*. Los artefactos antes mencionados se generan a partir de una metodología tradicional o no tradicional.

Descripción del instrumento

En la **Figura 5**, se resume las mejores prácticas identificadas en la fase de diseño. A continuación, se procede a describir los nuevos campos agregados al instrumento.

- **Modelo de datos:** este campo determina los modelos de datos a utilizar dentro del diseño del sistema para establecer de forma visual el proceso de desarrollo de software de Sistemas Embebidos, y se basa principalmente en: *Módulo físico*, *Modelo entidad relación*, *Modelo conceptual, Diagrama de secuencia*, *Diagrama de estados*, *Diagrama componentes*, *Diagrama de paquetes*, *Diagrama de clases.*
- **Prototipo:** en este campo se define el diseño o prototipo que serán una guía para el proceso de desarrollo de los Sistemas Embebidos. Y este campo se basa en: *Diseño software* y *Diseño hardware.*

Figura 5

Identificación de las Mejores Prácticas: Fase Diseño

SID	Autores	Año	Titulo del Artículo	Metodología	Diseño							Prototipo		
					Modelo de datos			Diagramas UML						
					Modelo Físico	Modelo Entidad Relación	Modelo Conceptual	Diagramas de secuencia	Diagramas de estados	Diagramas de componentes	Diagramas de paquetes	Diagrama de clases	Diseño Software	Diseño de Hardware
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil	Tradicional	0	0	0	0	0	0	0	0	0	0
S02	P. Manhart ; K. Schneider	2004	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria	Tradicional	0	0	0	0	0	0	0	0	0	0
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado	Tradicional	0	0	0	0	0	0	0	0	0	0

Nota: Esta figura, muestra el instrumento de identificación de las mejores prácticas en la fase de Diseño, se puede observar criterios como metodología de desarrollo de software, en el diseño tenemos el modelado de datos y diagramas UML, y prototipo.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1XzYTz-7kglekL4xY5E0j1G1g8ts7L0VX7kZ8ivoWRkw/edit?usp=sharing)

Fase de Codificación

En esta etapa se inicia a codificar algoritmos y de ser necesario la estructura de datos, mismos que ya están diseñados en etapas anteriores. Reducir un diseño a código puede es la parte más importante de los procesos de Ingeniería de Software, esto es determinante para obtener un producto de calidad.

Descripción del instrumento

En ciertos proyectos de software para Sistemas Embebidos, se omite ciertas etapas pasando directamente a la etapa de codificación. A continuación, se detalla las mejores prácticas véase la Figura 6, estas son aplicadas en la fase de codificación como:

- **Documentación de Código:** este campo muestra a la documentación de código como una de las mejores prácticas que servirá de respaldo para los desarrolladores de Sistemas Embebidas.
- **Lenguaje de programación**: permite determinar qué tipo de paradigma de programación es lo más utilizado para la construcción de software para Sistemas Embebidos.

Figura 6

Identificación de las Mejores Prácticas: Fase Codificación

SID	Autores	Año	Titulo del Artículo	Metodología	Codificación	
					Documentación de Código	P.O.O
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil	Tradicional	0	0
S02	P. Manhart ; K. Schneider	2004	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria	Tradicional	0	0
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado	Tradicional	0	1
S04	Kaisa Könnölä, Samuli Suomi, Tuomas Mäkilä, Tero Jokela, Ville Rantala , Teijo Lehtonen	2016	Métodos ágiles en el desarrollo de sistemas integrados: estudio de casos múltiples de tres casos industriales	Tradicional	0	0

Nota: Esta figura, muestra el instrumento de identificación de las mejores prácticas en la fase de Codificación, se puede observar criterios como metodología de desarrollo de software, para la codificación se consideró los criterios de documentación de código y Programación Orientada a Objetos donde se revisó los lenguajes de programación utilizados para la construcción de Sistemas Embebidos.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1HPVrkuOX9pr09agZKtoTNBTbAxsg3AyRFKN9qvUzl6U/edit?usp=sharing)

Fase de Pruebas

Permite comprobar que el software realice de forma correcta las actividades mencionadas en las etapas anteriores, como es en el caso de la etapa de pruebas. Es recomendable probar el software por módulos y luego probarlo de forma integral. En la **Figura 7**, se describe cada uno de los componentes del instrumento de las mejores prácticas.

Descripción del instrumento

Los artículos científicos analizados son producto de una revisión de literatura de las bases de datos científicas con proyectos en la industria e instituciones académicas, cuyo objetivo es incorporar pruebas al proceso de desarrollo de los Sistemas Embebidos.

- **Pruebas Funcionales:** este criterio se estableció para determinar las pruebas funcionales usadas de las metodologías tradicionales y no tradicionales con características propias.
- **Pruebas no Funcionales:** identifica las pruebas no funcionales que ofrecen las publicaciones científicas establecidas dentro de la matriz de hallazgos establecida para esta investigación.
- **Pruebas Unitarias:** de acuerdo a la revisión literaria realizada, se han podido identificar pruebas unitarias utilizadas en las metodologías tradicionales y no tradicionales.
- **Pruebas de Componentes:** especifica la integración de componentes considerada de como una de las mejores prácticas que establecen las metodologías de desarrollo para SE.
- **Pruebas de Aceptación:** este criterio son las últimas pruebas que se realiza al software esto con la presencia del cliente.

- **Pruebas de Integración:** este criterio es identifica pruebas de integración con la finalidad de comprobar la interacción entre los componentes del SE.
- **Pruebas de Sistema:** este criterio se estableció con la finalidad de identificar a las pruebas de sistema como una de las mejores prácticas aplicadas al momento de probar un SE, hay que considerar que este criterio seria la finalización de las pruebas al construir un Sistema Embebido.

Figura 7

Identificación de las Mejores Prácticas: Fase Pruebas

SID	Autores	Año	Titulo del Artículo	Pruebas						
				Pruebas Funcionales	Pruebas no Funcionales	Pruebas Unitarias	Pruebas de Componentes	Pruebas de Aceptación	Pruebas de Integración	Pruebas de Sistema
S01	Lucas Cordeiro ; Carlos Mar ; Eduardo Valentin ; Fabiano Cruz ; Daniel Patrick ; Raimundo Barreto ; Vicente Lucena	2008	A Platform-Based Software Design Methodology for Embedded Control Systems: An Agile Toolkit - Una metodología de diseño de software basada en plataforma para control integrado Sistemas: un conjunto de herramientas ágil	1	1	1	0	0	0	0
S02	P. Manhart ; K. Schneider	2004	Breaking the Ice for Agile Development of Embedded Software: An Industry Experience Report-Rompiendo el hielo para el desarrollo ágil de software integrado: Un informe de experiencia de la industria	0	0	1	0	0	0	0
S03	Deniz Akdur -Vahid Garousi -Onur Demirörs	2018	Una encuesta sobre el modelado y las prácticas de ingeniería impulsadas por modelos en la industria del software integrado	0	0	0	0	0	0	0

Nota: Esta figura, muestra el instrumento de identificación de las mejores prácticas en la fase de Pruebas, se puede observar criterios como metodología de desarrollo de software, en el criterio de pruebas tenemos las pruebas funcionales, no funcionales, unitarias, componentes, aceptación, integración y sistema. (ver instrumento completo: https://docs.google.com/spreadsheets/d/1gIGTC7a8bcVlNlnqYZmXPHxFlVC9e5RUSvpmH8g8tSg/edit?usp=sharing

Evolución de las mejores prácticas en el desarrollo de Sistemas Embebidos

Hoy en día, los Sistemas embebidos se encuentran presentes en cualquier parte del mundo y al menos el 50% de hogares tienen algún dispositivo que contenga un SE (Srinivasan et al., 2009). Es así, que el mercado de los Sistemas empotrados ha mostrado un gran crecimiento, lo cual ha contribuido a desarrollar software para este tipo de sistemas incorporando procesos de Ingeniería de Software. Para el análisis y distribución se consideró las fases de un ciclo de vida como el análisis de requerimientos, diseño, codificación y pruebas (Gómez, 2012). Para tener un mejor análisis y distribución de las mejores prácticas identificadas en la revisión literaria se estableció rangos por años desde el 2000 al 2010 y 2011 al 2020, así también se clasifica los artículos científicos de acuerdo a las metodologías de desarrollo tradicionales y no tradicionales.

Descripción de criterios generales del instrumento de las mejores practicas

Luego de haber realizado el análisis de la revisión de la literatura del desarrollo de software se procedió a seleccionar y distribuir las mejores prácticas, mismas que servirán para construir el modelo de trabajo de SE.Los criterios comunes que se aplican dentro del instrumento **(ver figura 7)**, son:

SID: Este criterio permite identificar y contabilizar a las publicaciones científicas seleccionadas que contengan una o varias mejores prácticas del proceso de desarrollo de software de SE.

Año: Es necesario utilizar este criterio lo cual permite clasificar.

Metodología: Se ha elegido este criterio ya que permite conocer en clasificación de las metodologías se encuentra cada publicación motivo de investigación.

Selección de mejores prácticas de las metodologías tradicionales 2000-2010

Para esta sección se ha diseñado un instrumento con ciertos criterios, en donde se puede seleccionar y distribuir las mejores prácticas identificadas en las 52 publicaciones científicas.

En esta sección se puede observar el selección y distribución de las mejores prácticas aplicadas a la construcción de SE, respecto a tratamiento de los rangos por año, en este caso del 2000 al 2010 se identificaron 10 publicaciones científicas de un total de 52 publicaciones identificadas en secciones anteriores, en donde se puede observar la tendencia de las mejores prácticas hacia las metodologías tradicionales.

Instrumento de selección de mejores prácticas para SE: Fase Análisis

Descripción del instrumento

Los criterios que se describen a continuación son el resultado de un análisis en realizadas en etapas anteriores. El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Requisitos funcionales y Requisitos no Funcionales:** El criterio forma parte del documento de especificación de requisitos y su participación dentro del proceso de desarrollo es considerable.
- **Diagramas de Caso de Uso:** Permite una representación más específica y claro sobre los procesos de un SE.

En la **Figura 8,** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software. Este instrumento permite visualizar la selección y distribución de buenas prácticas desarrollo por cada ciclo de vida del software en la construcción de

sistemas embebidos. Para la estructura del instrumento se considera los criterios como *SID, Año, Metodología, Requisitos funcionales, Requisitos no funcionales y Diagramas de Caso de Uso.*

Luego de haber seleccionado las publicaciones se puede observar que entre las mejores prácticas seleccionadas están los *Requisitos funcionales, Requisitos no funcionales y Diagramas de Caso de Uso*, estas prácticas forman parte de las metodologías tradicionales, lo cual hace entender a principios de los años 2000 hacia el 2010, la construcción de SE basan en la aplicación de mejores prácticas. Cabe señalar, que las metodologías tradicionales o clásicas de desarrollo de software se basa en el ciclo de vida de desarrollo de software y esto serviría para la construcción centralizada de software (Montero et al., s. f.). Por tal razón los primeros modelos publicados se acercan a la idea de establecer estados con procesos de Ingeniería de Software para sistemas de información y estos son extraídos para el desarrollo de software de sistemas embebidos.

Figura 8

Instrumento de selección de mejores prácticas: Fase Análisis

SID	Año	Metodología	Requisitos funcionales	Requisitos no funcionales	Diagramas de Caso de Uso
S01	2008	Tradicional	1	2	0
S02	2004	Tradicional	1	2	0
S09	2006	Tradicional	1	2	0
S23	2010	Tradicional	0	0	3
S36	2008	Tradicional	0	0	3

Nota: Esta figura, muestra la selección de los criterios de requisitos funcionales se identifica con el número 1 y no funcionales se identifica con el número 2, y diagramas de caso de uso con el número 3.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1BddHw2wCOrnHR2gc4JbrLavaTBkVd7hSp6A0v7JV24A/edit?usp=sharing)

Como se puede observar en la ***Tabla 2***, para contabilizar y clasificar se puede asignar a cada mejor practica identificada una valoración numérica para cada uno de estos criterios, para conseguir el número total de frecuencias aplicando pesos a cada criterio. Adicional se establece una semaforización para una mejor interpretación visual.

Tabla 2

Mejores prácticas de la Fase de Análisis 2000-2010

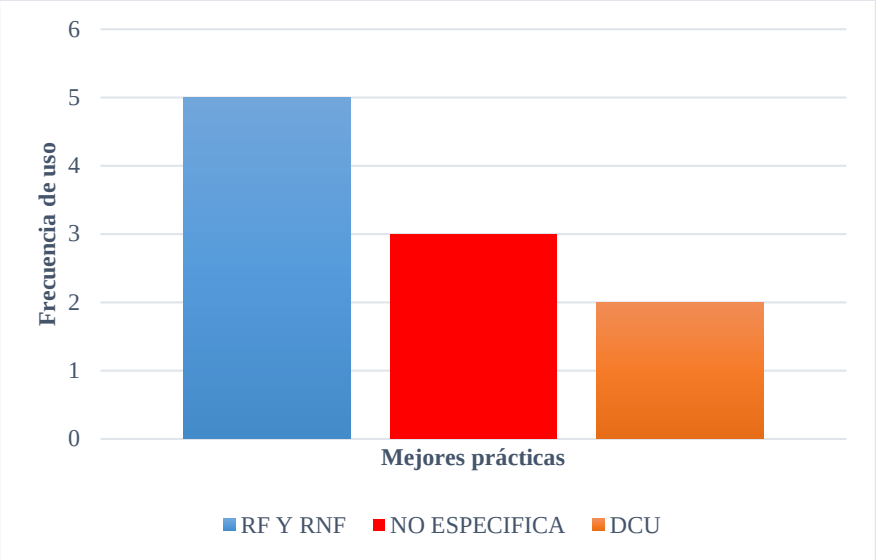
Criterios	Peso	Abreviaturas	Colores
No específica	0		
Requisitos funcionales	1	RF	
Requisitos no funcionales	2	RNF	
Diagrama de Caso de uso	3	DC	

Nota: En esta Tabla muestra las mejores prácticas de las Metodologías tradicionales identificadas del 2000 al 2010 con frecuencia de uso y semaforización.

Demostración en gráficos de barras

En la Figura 9, se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso.

Figura 9 *Fase Análisis: Frecuencia de uso de las mejores practicas*



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 10 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *5 publicaciones* con *Requisitos funcionales, Requisitos no funcionales*. Sin embargo, se ha identificado que de las 10 publicaciones son 3 *publicaciones* que no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos dentro de la fase de análisis. Entonces los *Requisitos funcionales y Requisitos no Funcionales* son las mejores prácticas que mayor frecuencia de uso y aplicación se presentan en las publicaciones científicas revisadas.

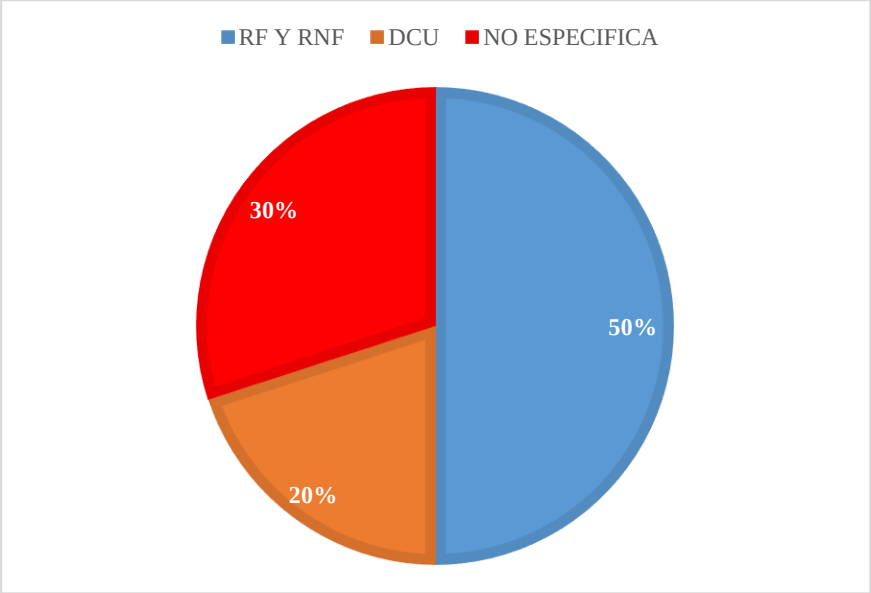
Demostración en grafico de pastel

En el **Figura 10**, se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se

procede a trabajar con la semaforización establecida en esta ocasión se interpretará la cantidad de artículos científicos en porcentajes.

Figura 10

Fase Análisis: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, en la demostración de pastel muestra las mejores prácticas aplicadas en la construcción de SE.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 10 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 10 publicaciones 5 publicaciones equivalen al 50% y hacen referencia a las mejores prácticas de *Requisitos funcionales* y *Requisitos no funcionales*, de igual manera se puede observar que 2 artículos científicos equivalen a 20% y corresponden a la aplicación de *Diagramas de casos de usos*. Para terminar, se ha identificado 3 publicaciones que es equivalente al 30% no especifican el uso o

aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Diseño

En esta etapa se estudió el diseño para el proceso de desarrollo usado para sistemas embebidos. En detalle, desde la misma se procedió a identificar publicaciones que contribuyan con mejores prácticas enfocadas al diseño de software de SE. Se realizó un instrumento en donde se puede apreciar 10 publicaciones de 52 que son el total de artículos propuestos para la investigación.

Descripción del instrumento

El instrumento de selección y clasificación de las mejores prácticas se estructura de la siguiente forma para el proceso de investigación aplicado al proceso de desarrollo de SE, los criterios que se describen a continuación son el resultado de la selección en realizada en etapas anteriores.

- **Diagrama de secuencia:** Permite conocer la interacción y describir el comportamiento dinámico de un Sistema Embebido.
- **Diagrama de clases:** Se estableció este criterio ya que permite utilizar el paradigma Orientado a Objetos, y es una guía para la codificación del software de SE.

En la **Figura 11**, *se* puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 11

Instrumento de selección de mejores prácticas: Fase Diseño

Diagramas de secuencia	Diagrama de clases
0	0
0	0
0	0
0	0
0	0
0	0
1	2

Nota: Esta figura, muestra la selección de los criterios de diagrama de secuencia que se identifica con el número 1 y diagrama de clases se identifica con el número 2, esto en la fase de diseño.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1BddHw2wCOrnHR2gc4JbrLavaTBkVd7hSp6A0v7JV24A/edit?usp=sharing)

Como se puede observar en la **Tabla 2**, este instrumento permite visualizar la selección y distribución de buenas prácticas desarrollo por cada ciclo de vida del software en la construcción de sistemas embebidos. Para la estructura del instrumento se considera los criterios como *Diagramas de secuencia y Diagrama de clases*.

Una vez seleccionadas las mejores prácticas como Diagramas *de Caso de secuencia y Diagramas de clases*, mismos que están presentes en las metodologías

tradicionales se puede considerar como las más aplicadas dentro del desarrollo de software de sistemas embebidos. Pero de acuerdo a la literatura revisada su aplicación está presente en los años 2000 hacia el 2010 para la construcción de SE basan en la aplicación de mejores prácticas. Cabe señalar, que las metodologías tradicionales o clásicas están presentes desde 1966 (Gabriel, s. f.), lo cual ha permitido establecer una base para utilizar en el desarrollo de diferentes tipos de software hasta las presentes fechas.

Como se puede observar en la ***Tabla 3***, para contabilizar y clasificar se puede asignar a cada mejor practica identificada una valoración numérica para cada uno de estos criterios, para conseguir el número total de frecuencias aplicando pesos a cada criterio. Adicional se establece una semaforización para una mejor interpretación visual.

Tabla 3

Mejores prácticas de la Fase de Diseño 2000-2010

Criterios	Peso	Abreviaturas	Colores
No específica	0		
Diagrama de secuencia	1	D SEC	
Diagrama de clase	2	D CLASES	

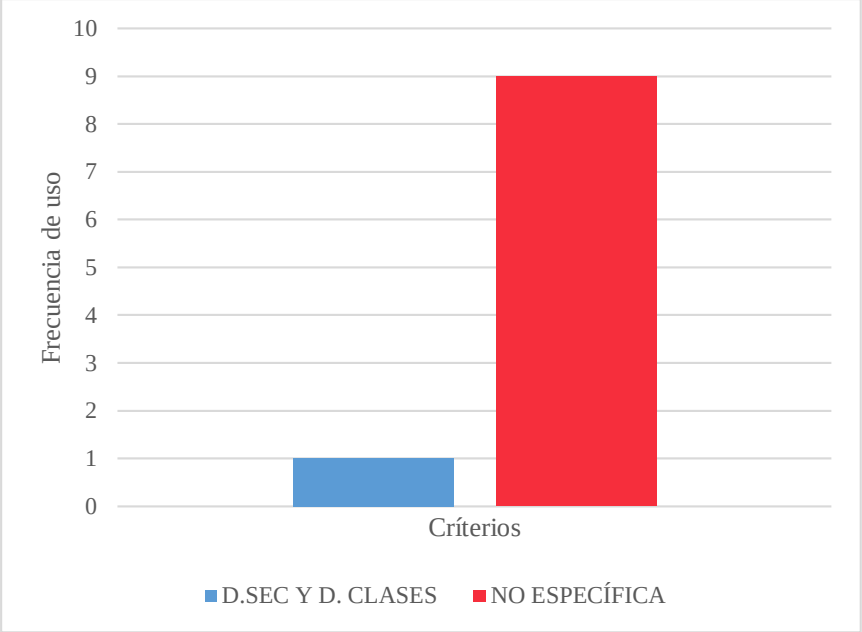
Nota: En esta Tabla muestra las mejores prácticas de las Metodologías tradicionales identificadas del 2000 al 2010 con frecuencia de uso y semaforización en la fase de diseño.

Demostración *en grafico de barras*

En la Figura 12**,** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso.

Figura 12

Fase Diseño: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de diseño y los criterios seleccionados son diagrama de clases y secuencia.

Interpretación:

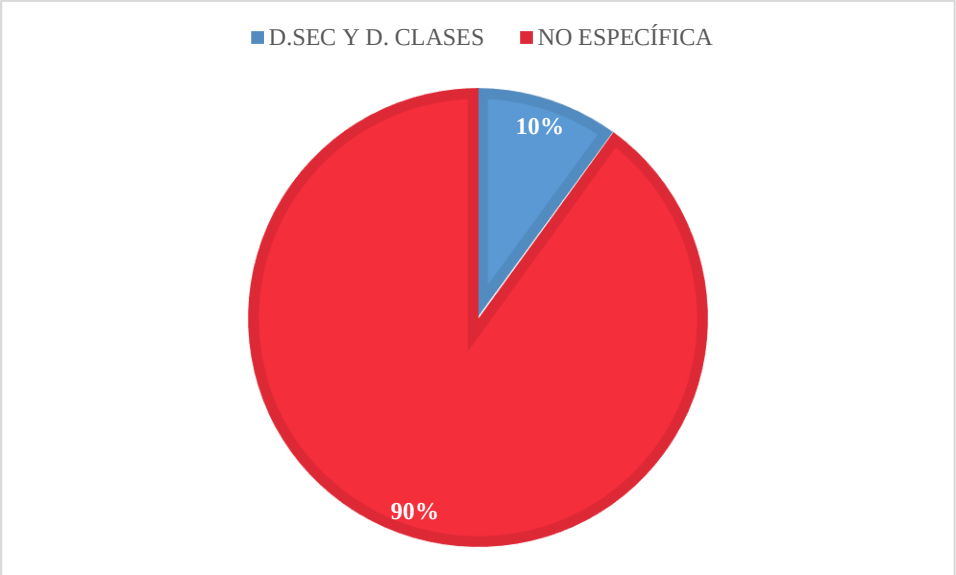
Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 10 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *1 publicación* con la aplicación de *Diagrama de secuencia y Diagrama de clases*. Sin embargo, se puede apreciar que 10 *publicaciones* son *9 publicaciones* no

especifican la aplicación de las mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar desde el año 2000 al 2010 no existe una aplicación relevante de las buenas prácticas de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 13

Fase Diseño: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, en la demostración de pastel muestra las mejores prácticas aplicadas en la construcción de SE.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 10 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 10 publicaciones 1 publicación equivale al 10% y hacen referencia a las mejores prácticas de *Diagrama de secuencia y Diagrama de clases*. Sin embargo, concluir se ha identificado 9 publicaciones que es equivalente al 90% no especifican

el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Codificación

En esta etapa se estudió la codificación para el proceso de desarrollo usado para sistemas embebidos. En detalle, desde la misma se procedió a identificar publicaciones que contribuyan con mejores prácticas enfocadas al diseño de software de SE. Se realizó un instrumento en donde se puede apreciar 10 publicaciones de 52 que son el total de artículos propuestos para la investigación.

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Documentación de código:** Permite añadir información para explicar lo que hace el código, de esta manera no solo el computador sabrá que hacer, sino que los humanos entenderán que es lo que se está haciendo y por qué.
- **Paradigma de Programación Orientada a Objetos*:*** Este criterio permite la capacidad para administrar la complejidad del software, este tipo de paradigma es aplicable cuando se desarrollan y mantienen aplicaciones y una estructura de datos de gran tamaño.

En la **Figura 14**, se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 14

Instrumento de selección de mejores prácticas: Fase Codificación

Documentación de Código	P.O.O
0	0
0	0
0	0
0	2
0	2

Nota: Esta figura, muestra la selección de los criterios de documentación de código que se identifica con el número 1 y Programación Orientada a Objetos con el número 2, esto en la fase de diseño.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1BddHw2wCOrnHR2gc4JbrLavaTBkVd7hSp6A0v7JV24A/edit?usp=sharing)

Como se puede observar en la **Figura 14**, este instrumento permite visualizar la selección y clasificación de buenas prácticas desarrollo por cada ciclo de vida del software en la construcción de sistemas embebidos. Para la estructura del instrumento dentro de la fase diseño se considera los criterios como *Documentación de código* y la aplicación de *Lenguajes de Programación Orientada a Objetos*, este último se puede observar que no tiene una relevancia dentro de la frecuencia de uso pues se puede evidenciar que dentro de las *10 publicaciones* extraídas *3 publicaciones* son las que aplican algún tipo de lenguaje orientado a objetos, así también al considerarse como POO es claro su apego a trabajar con la técnica de

UML *(Lenguaje Modelado Universal)* (Lagos, s. f.) pues UML se caracteriza por tener una semántica definida lo cual permite su integración para describir los sistemas, en los ciclos desarrollo software orientado a objetos.

Como se puede observar en la ***Tabla 4***, para contabilizar y clasificar se puede asignar a cada mejor practica identificada una valoración numérica para cada uno de estos criterios, para conseguir el número total de frecuencias aplicando pesos a cada criterio. Adicional se establece una semaforización para una mejor interpretación visual.

Tabla 4

Mejores prácticas de la Fase de Codificación 2000-2010.

Criterios	Peso	Abreviaturas	Colores
No específica	0		
Documentación de código	1	DOC	
Programación Orientada a Objetos	2	POO	

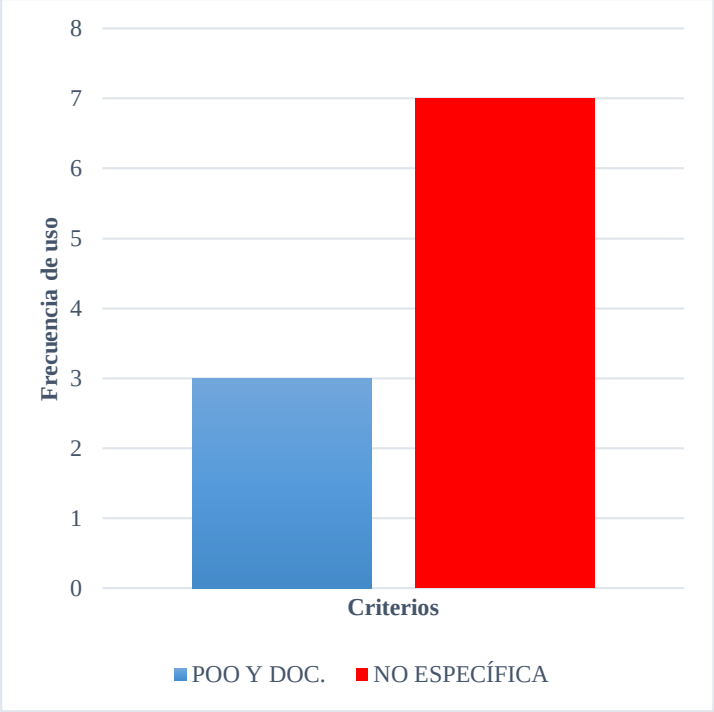
Nota: En esta Tabla muestra las mejores prácticas de las metodologías tradicionales identificadas del 2000 al 2010 con frecuencia de uso y semaforización en la fase de codificación.

Demostración en grafico de barras

En la **Figura 15** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización.

Figura 15

Fase Codificación: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE fase de codificación con los criterios de Programación Orientada a Objetos y documentación.

Interpretación:

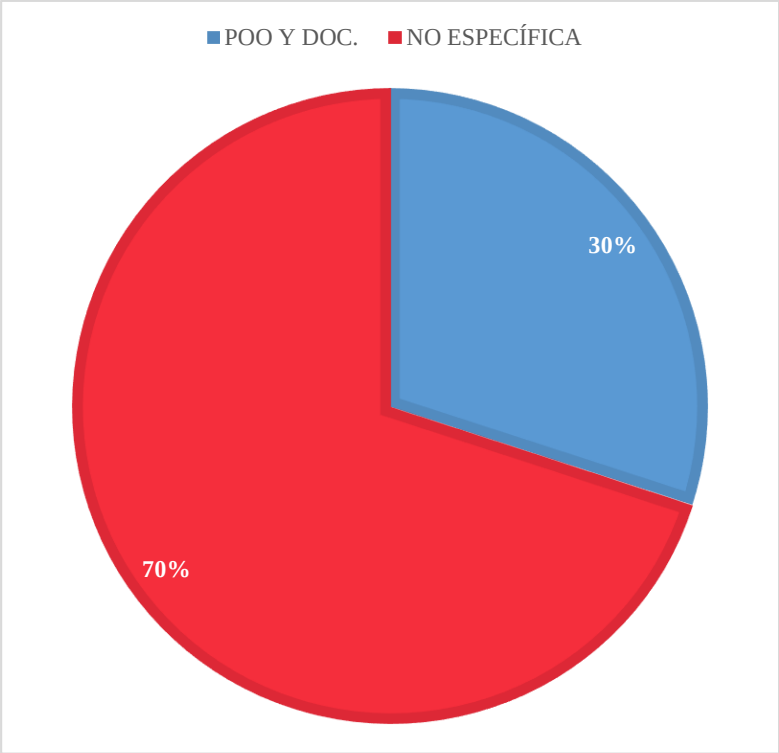
Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 10 publicaciones científicas de las enfocadas a metodologías tradicionales, se ha identificado que 3 *artículos de investigación* en donde utiliza el *Paradigma de Programación Orientada a Objetos.* Sin embargo, se ha identificado que de las 10 publicaciones son *7 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos en la fase de codificación. Como se puede ver se ha identificado las mejores prácticas como el uso de *Paradigma de Programación Orientada a Objetos* se presenta 3 publicaciones lo cual permite

observar que esta práctica sería la más óptima de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 16

Fase Codificación: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE fase de codificación con los criterios de Programación Orientada a Objetos y documentación.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 10 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que *3 artículos científicos* equivalen a 30% y corresponden a la aplicación de *Paradigma de Programación Orientada a Objetos*. Sin embargo, se ha identificado *7*

publicaciones que es equivalente al 70% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos en la fase de codificación.

Instrumento de selección de mejores prácticas para SE: Fase Pruebas

Las pruebas han adquirido importancia dentro del desarrollo de software. Sin embargo, las pruebas pueden ser aplicadas dependiendo del ámbito y área en donde se encuentre desarrollando un software o sistema. Por lo tanto, se procedió a identificar publicaciones que contribuyan con mejores prácticas enfocadas a la aplicación de pruebas dentro de la construcción de SE. Se realizó un instrumento en donde se puede apreciar 10 publicaciones de 52 que son el total de artículos propuestos para la investigación.

Descripción del instrumento

El instrumento estructura de la siguiente forma para el proceso de investigación aplicado al proceso de desarrollo de SE, los criterios que se describen a continuación son el resultado de un análisis en realizadas en etapas anteriores.

- **Pruebas funcionales y no funcionales:** Permite comprobar que el SE funcionen a de acuerdo a las especificaciones de los requisitos funcionales.
- **Pruebas unitarias o componentes***:* Este criterio permite comprobar el correcto funcionamiento de una unidad código.
- **Pruebas de aceptación:** Este tipo de prueba son realizadas con el cliente y este quien verifica las funcionalidades del producto software embebido.
- **Pruebas de integración:** Permite probar la comunicación e interacción entre componentes de hardware y software.

En la **Figura 17** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos,

gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 17

Instrumento de selección de mejores prácticas: Fase Pruebas

Pruebas Funcionales	Pruebas no Funcionales	Pruebas Unitarias	Pruebas de Aceptación
1	2	3	0
0	0	3	0

Nota: Esta figura, muestra la selección de los criterios como pruebas funcionales se identifica con el número 1, pruebas no funcionales se identifica con el número 2, pruebas unitarias se identifica con el número 3 y las pruebas de aceptación se identifica con el numero 0 debido a que no existen publicaciones que contengan este criterio.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1BddHw2wCOrnHR2gc4JbrLavaTBkVd7hSp6A0v7JV24A/edit?usp=sharing)

Como se puede observar en la **Figura 17**, este instrumento permite visualizar la selección y distribución de buenas prácticas. Luego de haber seleccionado las publicaciones se puede observar que entre las mejores prácticas seleccionadas esta la aplicación de *Pruebas funcionales, Pruebas no funcionales, Pruebas unitarias y Pruebas de aceptación*. Las pruebas son parte de la verificación y validación de software (Leopoldo Pauta Ayabaca & Moscoso Bernal, 2018), las pruebas de software es proceso para asegurar que el software sea entregado al cliente sin ningún defecto.

Como se puede observar en la ***Tabla 5***, para contabilizar y clasificar se puede asignar a cada mejor practica identificada una valoración numérica para cada uno de estos criterios, para conseguir el número total de frecuencias aplicando pesos a cada criterio. Adicional se establece una semaforización para una mejor interpretación visual.

Tabla 5

Mejores prácticas de la Fase de Pruebas 2000-2010

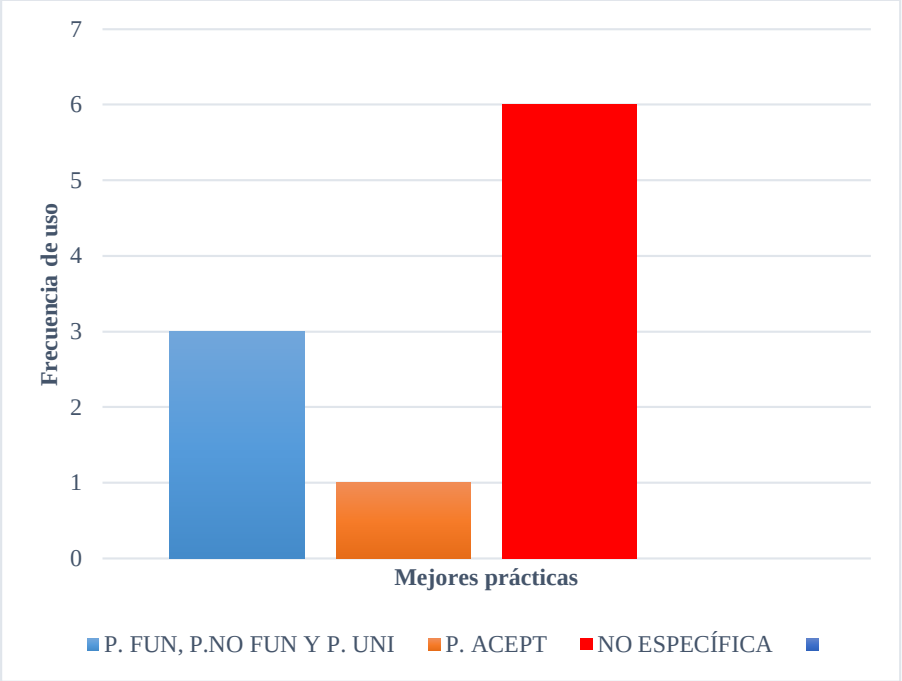
Criterios	Peso	Abreviaturas	Colores
No específica	0		
Pruebas funcionales	1	P.FUN	
Pruebas no funcionales	2	P.NO FUN	
Pruebas unitarias o componentes	3	P.UNI	
Pruebas de aceptación	4	P. ACEPT	

Nota: En esta Tabla muestra las mejores prácticas de las Metodologías tradicionales identificadas del 2000 al 2010 con frecuencia de uso y semaforización en la fase de diseño.

Demostración en grafico de barras

En la **Figura 18** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización.

Figura 18 *Fase Pruebas: Frecuencia de uso de las mejores practicas*



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas y los criterios seleccionados son pruebas funcionales y no funcionales, pruebas de unitarias, pruebas de aceptación.

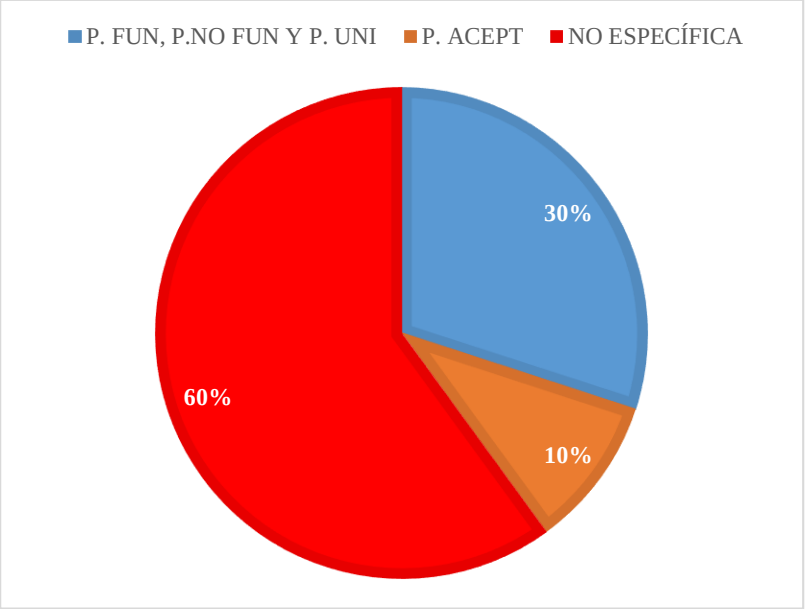
Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 10 publicaciones científicas de las enfocadas a metodologías tradicionales, se ha identificado que 3 *artículos de investigación* en donde utilizan las *Pruebas funcionales, Pruebas no funcionales y Pruebas unitarias*. También se puede observar *1 publicación* donde se hace uso de las *Pruebas de aceptación*. Sin embargo, se ha identificado que de las 10 publicaciones son *7 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos en la fase de codificación.

Demostración en grafico de pastel

Figura 19

Fase Pruebas: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas y los criterios seleccionados son pruebas funcionales y no funcionales, pruebas de unitarias, pruebas de aceptación.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 10 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, se puede observar que 3 artículos científicos equivalen a 30% y corresponden a la aplicación de *Pruebas Funcionales, Pruebas no funcionales y Pruebas unitarias*. En la gráfica también se demuestra que la aplicación mínima de *Pruebas de aceptación* se ha identificado en *1 publicación* que equivalen al 10%. Sin embargo, se ha demostrado que *6 publicaciones* corresponden al 60% no especifican el uso o aplicación de las

mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos en la fase de *Pruebas.*

Selección de mejores prácticas de metodologías tradicionales 2011-2020

Para la elaboración del instrumento de selección y clasificación de las mejores prácticas de las metodologías tradicionales se consideró ciertos criterios y rangos de años entre el 2011 al 2020.

En esta sección se procede a la selección y clasificación de las mejores prácticas aplicadas a la construcción de SE, respecto a tratamiento de los rangos por año, en este caso del 2011 al 2020 se identificaron 27 de las 52 publicaciones científicas en donde se puede observar la tendencia de las mejores prácticas hacia las metodologías tradicionales.

Lo anterior explicado conlleva a describir cada una de las fases y las mejores prácticas identificadas en el proceso de desarrollo de software para sistemas embebidos lo expresamos a continuación.

Instrumento de selección de mejores prácticas para SE: Fase Análisis

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Roles:** El criterio de los Roles forma parte del documento de especificación de requisitos y su participación dentro del proceso de desarrollo es considerable.

- **Requisitos Funcionales y Requisitos no Funcionales:** Este criterio forma parte del documento de especificación de requisitos y permite conocer la funcionalidad del software embebido a desarrollar.
- **Diagramas de Caso de Uso***:* Se usan para especificar la comunicación y el comportamiento de un sistema con la interacción de los usuarios y el sistema.
- **Product Backlog:** Este criterio dentro de las metodologías agiles permite conocer una lista de requisitos necesarios.
- **Historias de usuario:** Permite la representación de un requisito funcional en lenguaje en común del usuario.

En la **Figura 20** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 20

Instrumento de análisis de mejores prácticas: Fase Análisis

SID	Año	Metodología	Roles	Requisitos funcionales	Requisitos no funcionales	Diagramas de Caso de Uso	Product Backlog	Historias de usuario
S03	2018	Tradicional	0	2	3	4	0	0
S04	2016	Tradicional	1	2	3	0	0	0
S05	2013	Tradicional	1	2	3	0	0	0

Nota: Esta figura, muestra la selección de los criterios como roles que se identifica con el número 1, requisitos funcionales se identifica con el número 2, requisitos no funcionales se identifica con el número 3, diagrama de casos de uso se identifica con el número 4, así también el product backlog y las historias de usuario se

identifican con el numero 0 ya que dentro de las publicaciones estudiadas no se encontraron estos criterios esto en la fase de análisis.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1TCv6GAQ8ZzISAZS5F6DaB7GRDPexCbsEUkkBKmtRSss/edit?usp=sharing)

Este instrumento permite visualizar el análisis y clasificación de buenas prácticas. Para la estructuración del instrumento se consideró la frecuencia de uso. Se identificó las buenas prácticas de acuerdo a las metodologías, en este caso se trabaja con 27 artículos científicos que forman parte de las metodologías tradicionales.

Criterios de valoración y semaforización

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación, a continuación, se describe la ***Tabla 6***.

Tabla 6

Semaforización de las mejores prácticas de la Fase de Análisis

Color	Criterio	Abreviatura
	Requisitos Funcionales, Requisitos no Funcionales, Diagramas de Caso de Uso	RF, RNF, DCU
	Roles, Requisitos funcionales, Requisitos no Funcionales	ROLES, RNF, DCU
	Requisitos funcionales	RF
	Diagramas de Caso de Uso	DCU
	Requisito funcionales Requisitos no funcionales	RF, RNF
	Requisitos funcionales Diagramas de Caso de Uso	RF, DCU
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías tradicionales del 2011 al 2020.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno.

Como se puede observar en la ***Tabla 7*****,** se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 7

Valorización de las mejores prácticas Fase de Análisis

Criterios	Peso
Roles	1
Requisitos funcionales	2
Requisitos no funcionales	3
Diagramas de Casos de Uso	4
Product Backlog	5
Historias de usuarios	6

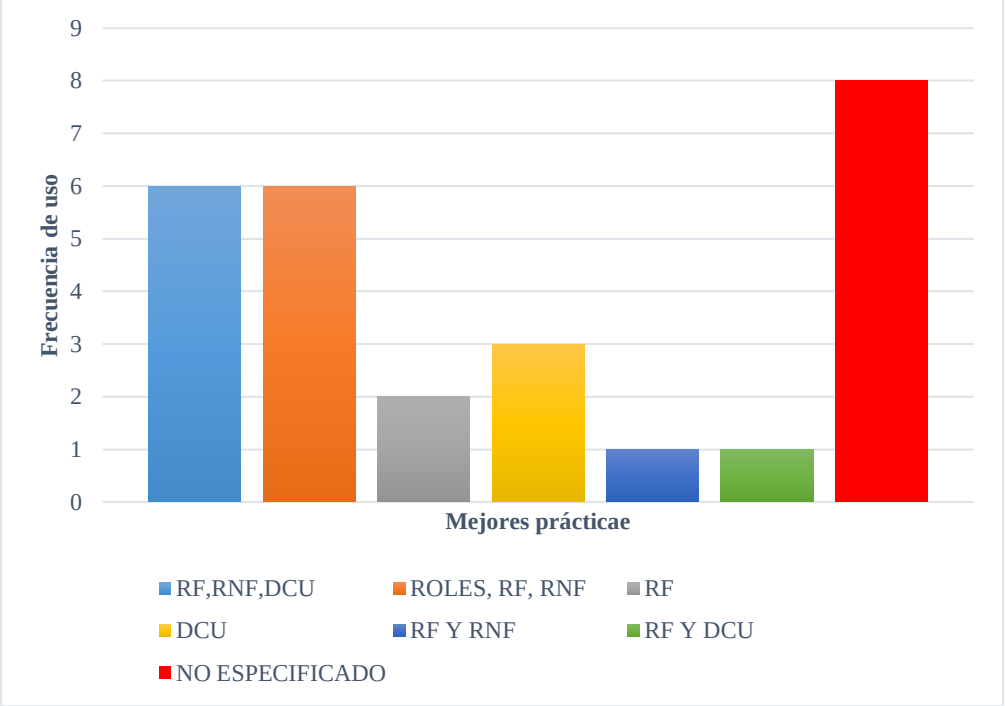
Nota: En esta Tabla muestra los criterios de valoración de las mejores prácticas identificadas en las metodologías tradicionales identificadas del 2011 al 2020.

Demostración en gráficos de barras

En la **Figura 21,** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 21

Fase Análisis: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de análisis tenemos requisitos funcionales, requisitos no funcionales, diagramas de caso de uso.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 27 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *6*

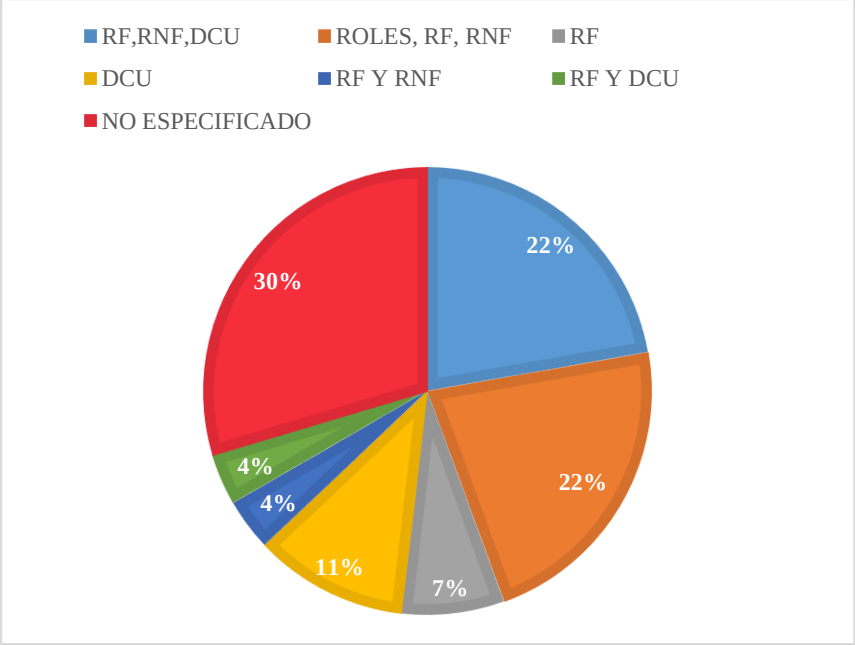
publicaciones con *Requisitos funcionales, Requisitos no funcionales y Diagramas de caso de uso*. Así también, se ha identificado *6 artículos de investigación con* los *Roles, Requisitos Funcionales y no Funcionales.* De la misma manera se observa *1 publicación científica* que hace referencia a los *Requisitos funcionales.* Hay que considerar la presencia de *3 publicaciones* para los *Diagramas de caso de uso*. Además, se ha identificado *1 publicación científica* tanto para *Requisitos funcionales y no funcionales, y 1 publicación* se ha identificado para *Requisitos funcionales y Diagramas de caso de uso.* Para finalizar se identificó que de las 27 publicaciones son *8 publicaciones* que no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar los *Requisitos funcionales, Requisitos no Funcionales, Diagramas de Caso de uso y Roles* con una valoración de 6 son las mejores prácticas que mayor frecuencia de aplicación tienen.

Demostración en grafico de pastel

En la **Figura 22** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida, en esta ocasión se interpretará la cantidad de artículos científicos en porcentajes.

Figura 22

Fase Análisis: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de análisis tenemos requisitos funcionales, requisitos no funcionales, diagramas de caso de uso y otros.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 27 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 27 publicaciones 6 publicaciones equivalen al 22% y hacen referencia a las mejores prácticas de *Requisitos funcionales, Requisitos no funcionales y Diagramas de Caso de uso*, de igual manera se puede observar que 6 artículos científicos equivalen a 22% y corresponden a la aplicación de *Roles, Requisitos funcionales y Requisitos no Funcionales*. En la gráfica de pastel también se demuestra que la aplicación de *Requisitos funcionales* se ha identificado en 2 publicaciones que equivalen al 7%, por otra parte, se ha demostrado 1 publicación

corresponde al 4% y se establece para *Requisitos funcionales y no funcionales y Diagramas de caso de uso.* Así también se observar 3 publicaciones sobre los *Diagramas de Caso de uso* mismo que equivalen al 11%. Para concluir se ha identificado 8 publicaciones que es equivalente al 30% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Diseño

Descripción del instrumento

El instrumento de selección y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Diagrama de secuencia***:* Permite conocer la interacción y describir el comportamiento dinámico de un Sistema Embebido.
- **Diagrama de estados:** Muestra el conjunto de estados por los cuales pasa un objeto durante su vida en una aplicación, lo cual permite describir el comportamiento de un sistema. Este tipo de diagrama se relaciona con técnicas Orientadas a Objetos y se relacionan con Diagramas de clases.
- **Diagrama de paquetes:** Permite obtener una visión clara de un sistema con orientación a objetos. El sistema es organizado en subsistemas y es más fácil de interpretar al construir un SE.
- **Diagrama de clases***:* El diagrama de clases permite comunicar el diseño de un programa (código) basado en el paradigma Orientado a Objetos, y es una guía para la codificación del software de SE.
- **Prototipos:** Se ha seleccionado el prototipo debido a que muestra la funcionalidad del producto software.

En la **Figura 23,** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 23

Instrumento de análisis de las mejores prácticas: Fase Diseño

Diagramas de secuencia	Diagramas de estados	Diagramas de paquetes	Diagrama de clases	Prototipos
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	2	0	0	5

Nota: Esta figura, muestra la selección de los criterios como diagrama de secuencia, diagrama de estados, diagrama de paquetes, diagrama de cases y prototipos esto en la fase de diseño.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1TCv6GAQ8ZzISAZS5F6DaB7GRDPexCbsEUkkBKmtRSss/edit?usp=sharing)

Este instrumento permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 27 publicaciones científicas.

Criterios de valoración y semaforización

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener

control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación, a continuación, se describe en la ***Tabla 8.***

Tabla 8

Semaforización de las mejores prácticas Fase de Diseño

Color	Criterio	Abreviatura
	Diagrama de Estados, Prototipos	D.EST PROTOTIPOS
	Diagrama de Paquetes	D. PAQ
	Diagrama de Secuencia Diagrama de Clases Diagrama de Paquetes	D.SEC D. CLASES D. PAQ
	Diagrama de Secuencia Diagrama de Estados	D. SEC D.EST
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías tradicionales identificadas del 2011 al 2020 en la fase de diseño.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 9*****,** se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 9

Valoración de las mejores prácticas Fase de Diseño

Criterios	Valoración
Diagrama de secuencia	1
Diagrama de estados	2
Diagrama de paquetes	3
Diagrama de clases	4
Prototipos	5

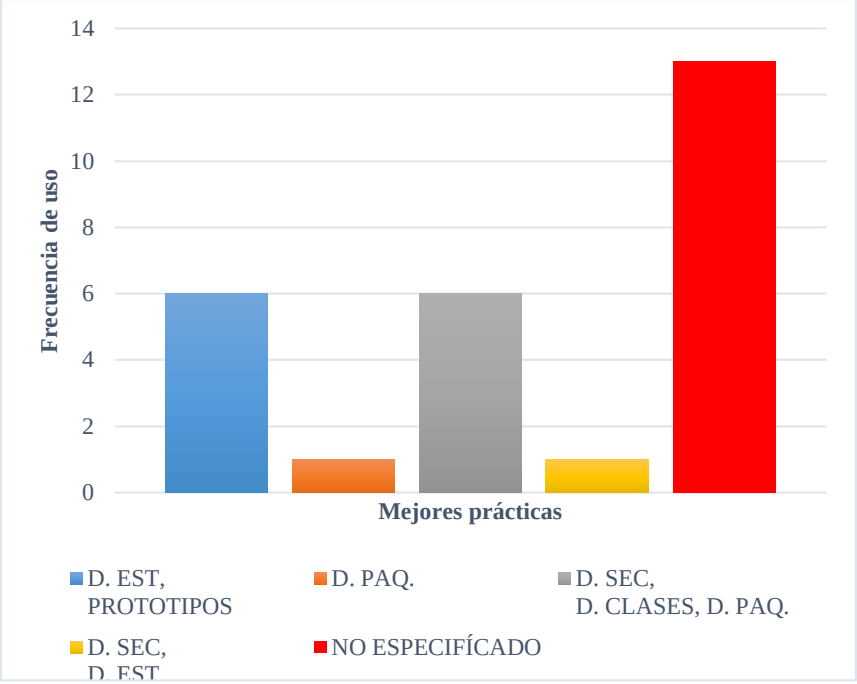
Nota: En esta Tabla muestra los criterios de valoración con las mejores prácticas identificadas en las metodologías tradicionales identificadas del 2011 al 2020.

Demostración en grafico de barras

En la **Figura 24,** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 24

Fase Diseño: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de diseño y los criterios seleccionados son diagrama de estados, prototipos, diagrama de secuencia, diagrama de paquetes y diagrama de clases.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 27 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *6 publicaciones* con la aplicación de *Diagrama de Estados y Prototipos*. Así también, se ha identificado 1 *artículo de investigación* en donde utiliza *Diagramas de Paquetes.* De la misma manera se observa *6 publicaciones científicas* que hace referencia al uso de *Diagramas de secuencia, Diagrama de clases y Diagramas de paquetes.* Además, se ha identificado *1 publicación científica* en donde se aplica *Diagrama de secuencia y Diagramas de estado.* Para finalizar se identificó que de

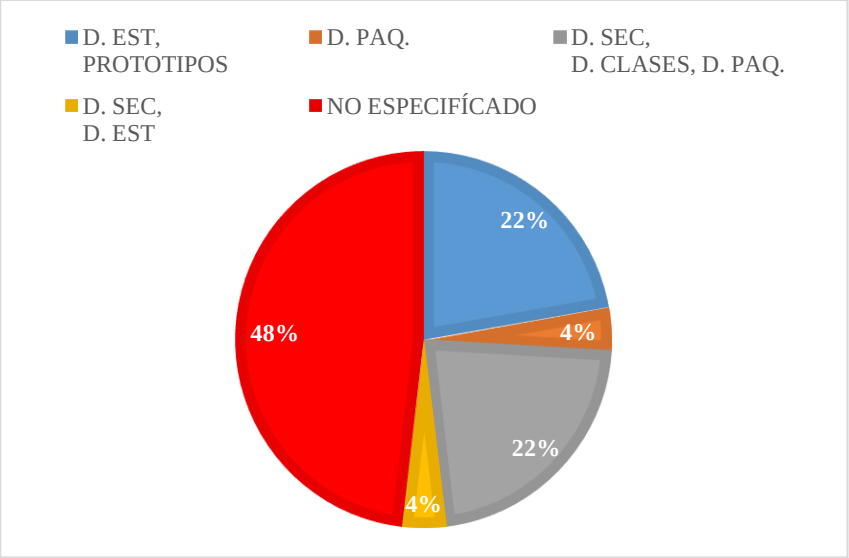
las 27 publicaciones son *13 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos.

Como se puede observar en la Figura 2.3 se ha identificado las mejores prácticas como *Diagrama de estados, Prototipos, Diagramas de secuencia, Diagrama de secuencia y Diagrama de Paquetes* tienen una valoración de 6 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 25

Fase Diseño: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de diseño y los criterios seleccionados son diagrama de estados, prototipos, diagrama de secuencia, diagrama de paquetes y diagrama de clases.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 27 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 27 publicaciones 6 publicaciones equivalen al 22% y hacen referencia a las mejores prácticas de *Diagrama de estados y Prototipos*, de igual manera se puede observar que 1 artículo científico equivalen a 4% y corresponden a la aplicación de *Diagramas de paquetes*. En la gráfica de pastel también se demuestra que la aplicación de *Diagramas de secuencia, Diagrama de clases y Diagrama de paquetes* se ha identificado en 6 publicaciones que equivalen al 22%, por otra parte, se ha demostrado 1 publicación corresponde al 4% y se establece el uso de *Diagrama secuencia y Diagrama de estados.* Para concluir se ha identificado 11 publicaciones que es equivalente al 48% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Codificación

Se realizó un instrumento en donde se puede apreciar 27 publicaciones de 52 que son el total de artículos propuestos para la investigación.

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Documentación de código:** Permite añadir información para explicar lo que hace el código, de esta manera no solo el computador sabrá que hacer, sino que los humanos entenderán que es lo que se está haciendo y por qué.

- **Paradigma de Programación Orientada a Objetos:** Este criterio permite la capacidad para administrar la complejidad del software, este tipo de paradigma es aplicable cuando se desarrollan y mantienen aplicaciones y una estructura de datos de gran tamaño.

Figura 26

Instrumento de análisis de las mejores prácticas: Fase Codificación

Documentación de Código	P.O.O
0	2
0	0
0	0
1	2

Nota: Esta figura, muestra la selección de los criterios como la documentación de código se identifica con el número 1 y Programación Orientada a Objetos se identifica con el número 2 esto en la fase de codificación.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1TCv6GAQ8ZzISAZS5F6DaB7GRDPexCbsEUkkBKmtRSss/edit?usp=sharing)

Este instrumento permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 27 publicaciones científicas.

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener

control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación.

Tabla 10

Semaforización de las mejores prácticas Fase Codificación

Color	Criterio	Abreviatura
	Documentación de código	DOC. CÓDIGO
	Paradigma de Programación Orientada a Objetos	P.O.O
	Diagrama de Secuencia Diagrama de Clases Diagrama de Paquetes	D.SEC D. CLASES D. PAQ
	Diagrama de Secuencia Diagrama de Estados	D. SEC D.EST
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores de las metodologías tradicionales identificadas del 2011 al 2020 en la fase de codificación.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 11***, se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 11 Valorización de las mejores prácticas Fase Codificación

Criterios	**Valoración**
Documentación de código	1
Programación Orientada a Objetos	2

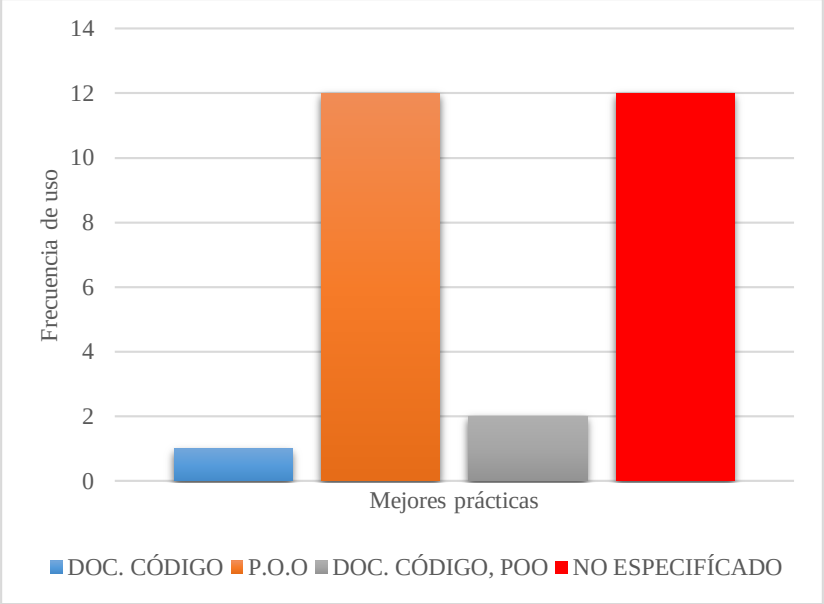
Nota: En esta Tabla muestra los criterios de valoración de las mejores prácticas identificadas en las metodologías tradicionales identificadas del 2011 al 2020.

Demostración en grafico de barras

En la **Figura 27** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 27

Fase Codificación: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de codificación con los criterios de documentación de código y la Programación Orientada a Objetos.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 27 publicaciones científicas de las enfocadas a metodologías tradicionales, se ha identificado 1 *artículo de investigación* en donde utiliza *Documentación de código.* De la misma manera se observa *12 publicaciones científicas* que hace referencia al uso del Paradigma *de Programación Orientada a Objetos*. Además, se ha identificado *2 publicaciones científicas* en donde se aplica *Documentación de código y* la *Programación Orientada a Objetos.* Para finalizar se identificó que de las 27

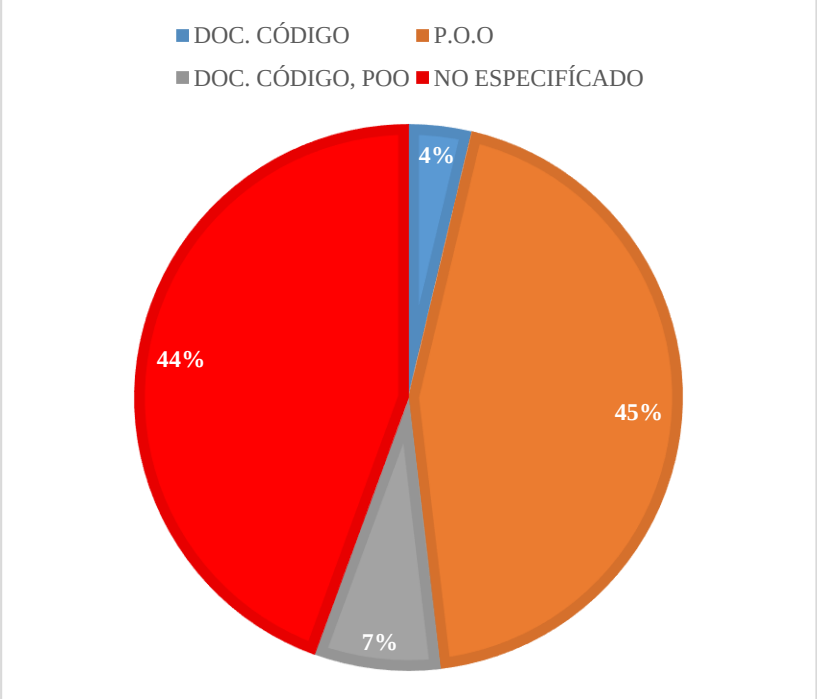
publicaciones son *12 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos.

Como se puede observar en la **Figura 28** se ha identificado las mejores prácticas como el uso de *Paradigma de Programación Orientada a Objetos* que tiene una valoración de 12 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 28

Fase Codificación: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de codificación con los criterios de documentación de código y la Programación Orientada a Objetos.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 27 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 27 publicaciones se puede observar que 1 artículo científico equivale a 4% y corresponden a la aplicación de Documentación de código. En la gráfica también se demuestra que la aplicación de Programación Orientada a Objetos se ha identificado en 12 publicaciones que equivalen al 45%, por otra parte, se ha demostrado que 1 publicación corresponde al 7% y se establece el uso de Documentación de código y la Programación Orientada a Objetos. Por último, se ha identificado 12 publicaciones que es equivalente al 45% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Pruebas

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Pruebas funcionales:** Permite comprobar que el SE funcionen a de acuerdo a las especificaciones de los requisitos funcionales.
- **Pruebas unitarias***:* Este criterio permite comprobar el correcto funcionamiento de una unidad código.
- **Pruebas de aceptación:** Este tipo de prueba son realizadas con el cliente y este quien verifica las funcionalidades del producto software embebido.
- **Pruebas de integración:** Permite probar la comunicación e interacción entre componentes de hardware y software.

En la **Figura 29** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 29

Instrumento de análisis de las mejores prácticas: Fase Pruebas

Pruebas Funcionales	Pruebas Unitarias	Pruebas de Aceptación	Pruebas de Integración
0	0	0	0
1	0	0	0
0	0	0	0
1	0	0	0

Nota: Esta figura, muestra la selección de los criterios como pruebas funcionales, pruebas unitarias, pruebas de aceptación y pruebas integración.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1TCv6GAQ8ZzISAZS5F6DaB7GRDPexCbsEUkkBKmtRSss/edit?usp=sharing)

Este instrumento permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 27 publicaciones científicas.

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener

control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación (*ver **Tabla 12**).*

Tabla 12

Semaforización de las mejores prácticas Fase Pruebas

Color	Criterio	Abreviatura
	Pruebas Funcionales	P. FUN
	Pruebas Funcionales	P.FUN
	Pruebas Unitarias	P.UNI
	Pruebas de Integración	P. INT
	Pruebas unitarias	P.UNI
	Pruebas de aceptación	P. ACEP
	Pruebas de integración	P.INT
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías tradicionales del 2011 al 2020.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 13***, se estableció una

valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 13 *Valoración de las mejores prácticas Fase Pruebas*

Criterios	Valoración
Pruebas funcionales	1
Pruebas unitarias	2
Pruebas de aceptación	3
Pruebas de integración	4

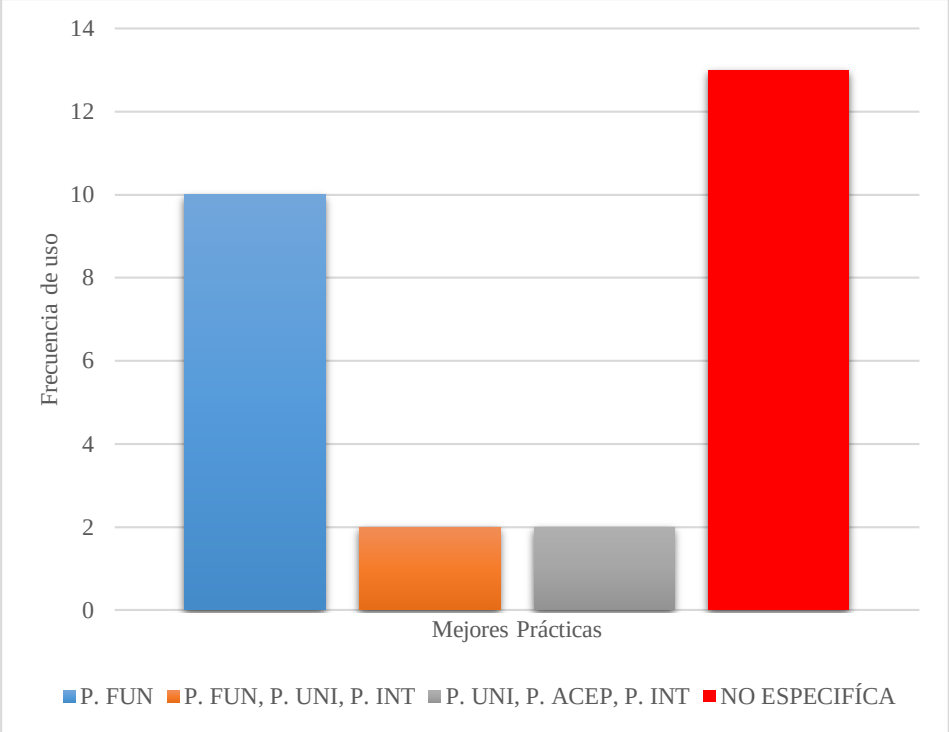
Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías tradicionales del 2011 al 2020.

Demostración en grafico de barras

En la **Figura 30** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual.

Figura 30

Fase Pruebas: Frecuencia de uso de las mejores prácticas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas las mejores prácticas identificadas son pruebas funcionales y no funcionales, unitarias, aceptación y de integración.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 27 publicaciones científicas de las enfocadas a metodologías tradicionales, se ha identificado 10 *artículos de investigaciones científicas en* donde utiliza *Pruebas funcionales.* De la misma manera se observa *2 publicaciones científicas* que hace referencia al uso *Pruebas funcionales, Pruebas unitarias y Pruebas de integración*. Además, se ha identificado *2 publicaciones científicas* en donde se aplica *Pruebas unitarias, Pruebas de aceptación y pruebas de integración.* Para finalizar se identificó que de las 27 publicaciones son *13 publicaciones* no reportan algún tipo de buenas o

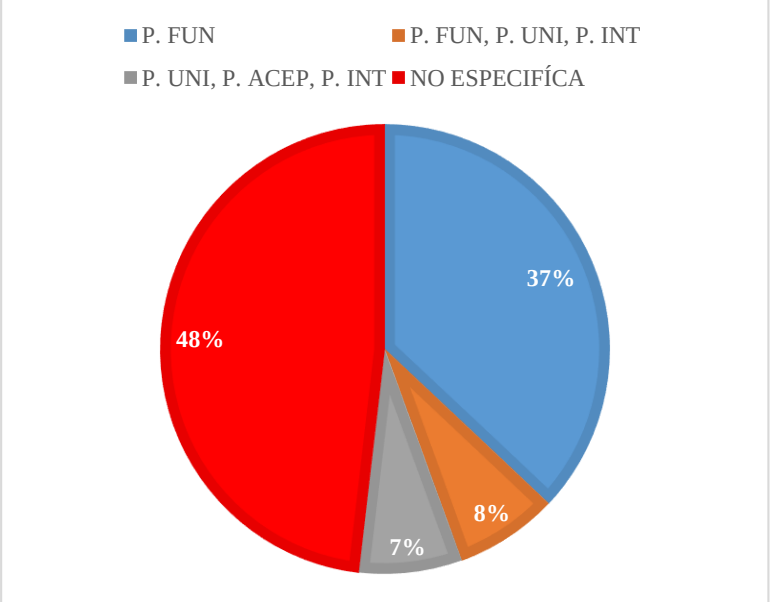
mejores prácticas para el proceso de desarrollo se software para sistemas embebidos.

Como se puede observar se ha identificado las mejores prácticas como el uso de *Pruebas funcionales* que tiene una frecuencia de aplicación 12 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 31

Fase Pruebas: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas las mejores prácticas identificadas son pruebas funcionales y no funcionales, unitarias, aceptación y de integración.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 27 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 27 publicaciones se puede observar que 10 artículos científicos equivalen a 37% y corresponden a la aplicación de *Pruebas Funcionales*. En la gráfica también se demuestra que la aplicación de *Pruebas funcionales, Pruebas unitarias, Pruebas integrales* se ha identificado en 2 publicaciones que equivalen al 8%, por otra parte, se ha demostrado que 2 publicaciones corresponden al 7% y se establece el uso de *Pruebas unitarias, Pruebas de aceptación y Pruebas de integración*. Por último, se ha identificado 13 publicaciones que es equivalente al 48% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos en la fase de *Pruebas.*

Selección de mejores prácticas de metodologías no tradicionales 2011-2020

En esta sección se realiza el análisis y distribución de las mejores prácticas aplicadas a la construcción de SE, respecto a tratamiento de los rangos por año, en este caso del 2011 al 2020 se identificaron 15 publicaciones científicas de 52 en donde se puede observar la tendencia de las mejores prácticas hacia las metodologías no tradicionales.

Instrumento de selección de mejores prácticas para SE: Fase Análisis

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Roles:** El criterio de los Roles forma parte del documento de especificación de requisitos y su participación dentro del proceso de desarrollo es considerable.
- **Requisitos Funcionales y Requisitos no Funcionales:** Este criterio forma parte del documento de especificación de requisitos y permite conocer la funcionalidad del software embebido a desarrollar.
- **Diagramas de Caso de Uso:** Permite una representación más específica y claro sobre los procesos de un SE.
- **Product Backlog:** Este criterio dentro de las metodologías agiles permite conocer una lista de requisitos necesarios.
- **Historias de usuario:** Permite la representación de un requisito funcional en lenguaje en común del usuario.

En la **Figura 32** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software.

Figura 32

Instrumento de análisis de las mejores prácticas: Fase Análisis

SID	Año	Metodología	Roles	Requisitos funcionales	Requisitos no funcionales	Diagramas de Caso de Uso	Product Backlog	Historias de usuario
S06	2015	No Tradicional	1	0	0	0	5	0
S13	2016	No Tradicional		2	0	4	5	6
S15	2015	No Tradicional	1	0	0	0	0	6
S18	2019	No Tradicional	1	0	0	0	5	0
S19	2016	No Tradicional	1	0	0	0	0	6

Nota: Esta figura, muestra la selección de los criterios como roles, requisitos funcionales y no funcionales, diagramas de caso de uso, product backlog e historias de usuario.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1QkpCaj7ISKHAlKxwWruW6F_JYcltMLOCEJ5P1tFaZiw/edit?usp=sharing)

Este instrumento permite visualizar el análisis y clasificación de buenas prácticas, para la estructuración del instrumento se consideró la frecuencia de uso. Se identificó las buenas prácticas de acuerdo a las metodologías, en este caso se trabaja con 15 artículos científicos que forman parte de las metodologías tradicionales.

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación.

Tabla 14

Semaforización de las mejores prácticas Fase Análisis

Color	Criterio	Abreviatura
	Roles	ROLES
	Product Backlog	P. BACK
	Roles,	RF
	Historias de usuario	DCU
	Requisitos funcionales	RF
	Diagramas de Caso de Uso	DCU
	Product Backlog	P. BACK
	Historias de Usuarios	H. USUARIOS
	Roles	DCU
	Requisitos funcionales	RF
	Requisitos no funcionales	RNF
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías no tradicionales del 2011 al 2020 para la construcción de Sistemas Embebidos.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 15***, se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 15

Valoración de las mejores prácticas Fase Análisis

Criterios	Valoración
Roles	1
Requisitos funcionales	2
Requisitos no funcionales	3
Diagramas de Casos de Uso	4
Product Backlog	5
Historias de usuarios	6

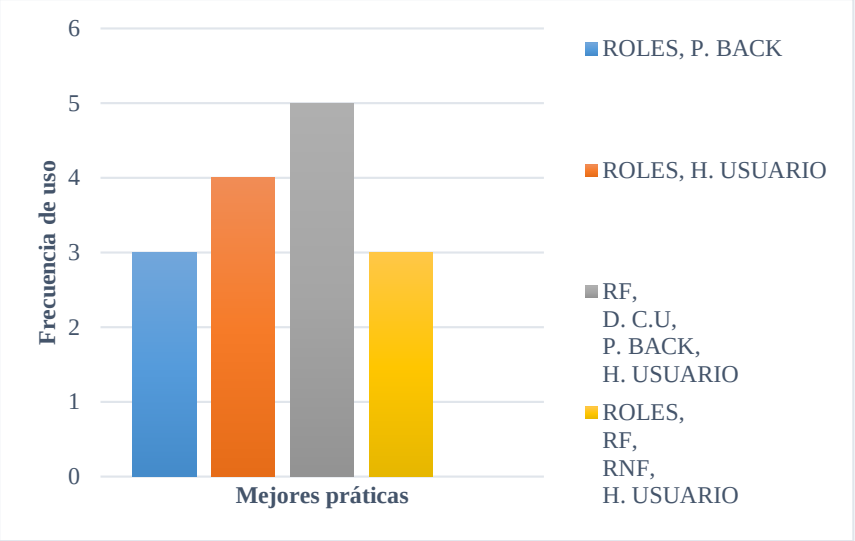
Nota: En esta Tabla muestra la valoración que se da a cada una de las mejores prácticas de las metodologías no tradicionales identificadas en las publicaciones científicas del 2011 al 2020 para la construcción de Sistemas Embebidos.

Demostración en gráficos de barras

En la **Figura 33** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 33

Fase Análisis: Frecuencia de uso de las mejores prácticas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de análisis y los criterios seleccionados como roles, product backlog, historias de usuario, requisitos funcionales y no funcionales.

Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 15 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *3 publicaciones* en donde aplican *Roles y Product Backlog*. De la misma manera, se ha identificado *4 artículos de investigación* con las mejores prácticas como *Roles e Historias de usuarios.* De la misma manera se observa *5 publicaciones científicas* que hace referencia a los *Requisitos funcionales, Diagramas de caso de uso, Product backlog e Historias de usuarios.* Para finalizar se identificó que de las 15 publicaciones todas tienen un aporte de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar los *Requisitos funcionales, Requisitos no Funcionales, Diagramas de Caso de uso,*

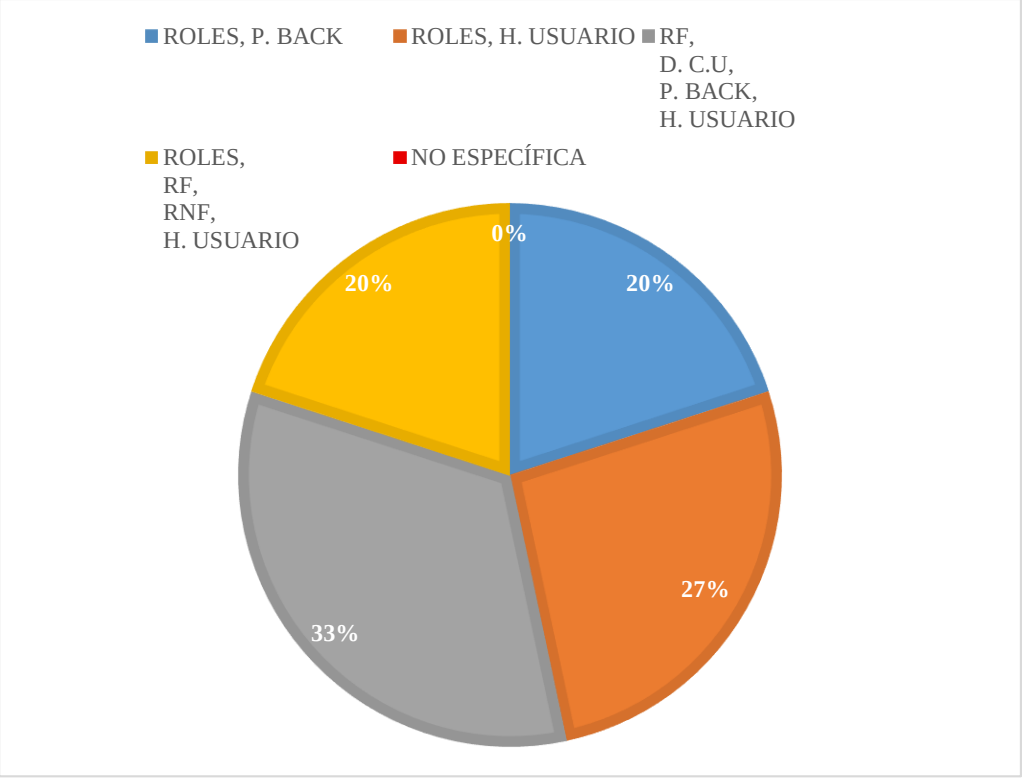
Roles e Historias de Usuarios se han presentado con más frecuencia de uso con una valoración y por lo cual sería lo más óptimo para su aplicación en el proceso de desarrollo de software para sistemas embebidos.

Demostración en grafico de pastel

En la **Figura 34** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida, en esta ocasión se interpretará la cantidad de artículos científicos en porcentajes.

Figura 34

Fase Análisis: Porcentaje de aplicación de las mejores prácticas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de análisis tenemos requisitos funcionales, requisitos no funcionales, diagramas de caso de uso y otros.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 15 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 15 publicaciones *3 publicaciones* equivalen al 20% y hacen referencia a las mejores prácticas de *Roles y Product Backlog*. De de igual manera se puede observar el uso de las mejores prácticas en 4 artículos científicos y estos equivalen a 27% y corresponden a la aplicación de *Roles* e *Historias de usuarios*. En la gráfica de pastel también se demuestra que la aplicación de *Requisitos funcionales, Diagramas de caso de uso, Product backlog e Historias de usuarios* están presente en *5 publicaciones* que equivalen al 33%. Por otra parte, se ha demostrado que 3 publicación corresponde al 20% y se establece para los *Roles, Requisitos funcionales y no funcionales e Historias de usuarios.* Para concluir, se puede apreciar que existe un 0% de publicaciones que no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Diseño

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente:

- **Modelo Entidad Relación:** Este criterio permitirá representar entidades como cosas u objetos para modelar una base de datos en el caso que en la fase de diseño se requiera el consumo de las bases de datos.

- **Modelo Conceptual:** Este criterio permite tener una representación del sistema embebido, el modelo ayuda a las personas a conocer, comprender y simular un sistema.
- **Diagrama de clases:** Se estableció este criterio ya que permite utilizar el paradigma Orientado a Objetos, y es una guía para la codificación del software de SE.

En la **Figura 35** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software. Como se puede observar permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 15 publicaciones científicas.

Figura 35

Instrumento de análisis de las mejores prácticas: Fase Diseño

Modelo Entidad Relación	Modelo Conceptual	Diagrama de clases
0	0	0
0	2	3
0	0	0
0	0	0
1	0	3

Nota: Esta figura, muestra la selección de las mejores prácticas como modelo entidad relación, modelo conceptual y diagrama de clases, esto en la etapa de diseño.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1QkpCaj7ISKHAlKxwWruW6F_JYcltMLOCEJ5P1tFaZiw/edit?usp=sharing)

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación.

Tabla 16

Valoración de las mejores prácticas Fase Análisis

Color	Criterio	Abreviatura
	Modelo. Conceptual, Diagrama de Clases	M. CONCEPTUAL D. CLASES
	Modelo Entidad Relación Diagrama de Clases	M. E. RELACIÓN D. CLASES
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías no tradicionales del 2011 al 2020 para la construcción de Sistemas Embebidos.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 17***, se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 17

Valoración de las mejores prácticas Fase Diseño

Criterios	Valoración
Modelo Entidad Relación	1
Modelo Conceptual	2
Diagrama de clases	3

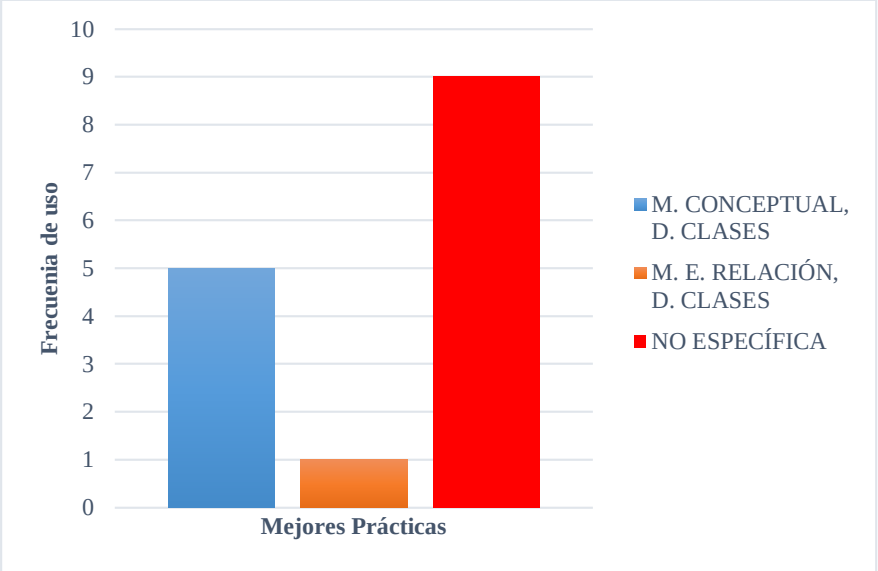
Nota: En esta Tabla muestra los criterios de valoración de las mejores prácticas identificadas en las metodologías no tradicionales del 2011 al 2020 para la construcción de Sistemas Embebidos.

Demostración en grafico de barras

En la **Figura 36** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 36

Fase Diseño: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de diseño tenemos el modelo conceptual, diagrama de clases, modelo entidad relación.

Interpretación:

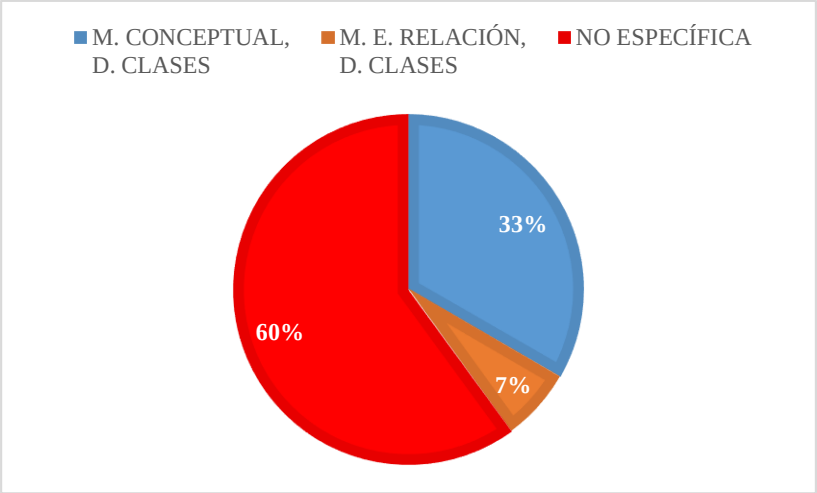
Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 15 publicaciones científicas de las enfocadas a metodologías tradicionales, se han identificado *5 publicaciones* con la aplicación de las mejores prácticas como Modelo *Conceptual y Diagramas de clases*. De la misma manera se observa *1 publicación científica* que hace referencia al uso de *Modelo entidad relación y Diagrama de clases.* Para finalizar se identificó que de las 15 publicaciones se observa la existencia *9 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar se ha identificado las mejores prácticas como *Modelo conceptual y Diagramas de clases*

poseen un nivel de frecuencia de 5 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 37

Fase Diseño: Porcentaje de aplicación de mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de diseño tenemos el modelo conceptual, diagrama de clases, modelo entidad relación.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 15 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 15 publicaciones 5 publicaciones equivalen al 33% y hacen referencia a las mejores prácticas de *Modelo Conceptual y Diagrama de Clases*, de igual manera se puede observar que 1 artículo científico equivalen a 7% y corresponden a la aplicación de *Modelo entidad relación y Diagrama de Clases.* Sin embargo, se ha identificado un número superior de 9 publicaciones que es equivalente al 60% y no

especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Codificación

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma para el proceso de investigación aplicado al proceso de desarrollo de SE, los criterios que se describen a continuación son el resultado de un análisis en realizadas en etapas anteriores.

- **Documentación de código:** Permite añadir información para explicar lo que hace el código, de esta manera no solo el computador sabrá que hacer, sino que los humanos entenderán que es lo que se está haciendo y por qué.
- **Paradigma de Programación Orientada a Objetos:** Este criterio permite la capacidad para administrar la complejidad del software, este tipo de paradigma es aplicable cuando se desarrollan y mantienen aplicaciones y una estructura de datos de gran tamaño.

En la **Figura 38** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software. Como se puede observar que permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 15 publicaciones científicas.

Figura 38

Instrumento de análisis de las mejores prácticas: Fase Codificación

Documentación de Código	P.O.O
	1
	1
1	1

Nota: Esta figura, muestra la selección de los criterios de las mejores prácticas como documentación de código y Programación Orientada a Objetos, esto en la fase de codificación.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1QkpCaj7ISKHAlKxwWruW6F_JYcltMLOCEJ5P1tFaZiw/edit?usp=sharing)

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación.

Tabla 18

Semaforización de mejores prácticas Fase Codificación

Color	Criterio	Abreviatura
	Documentación de código Programación Orientada a Objetos	DOC. CÓDIGO P.O.O
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías no tradicionales del 2011 al 2020 para la construcción de Sistemas Embebidos.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 19***, se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 19

Valoración de las mejores prácticas Fase Codificación

Criterios	Valoración
Documentación de código	1
Programación Orientada a Objetos	2

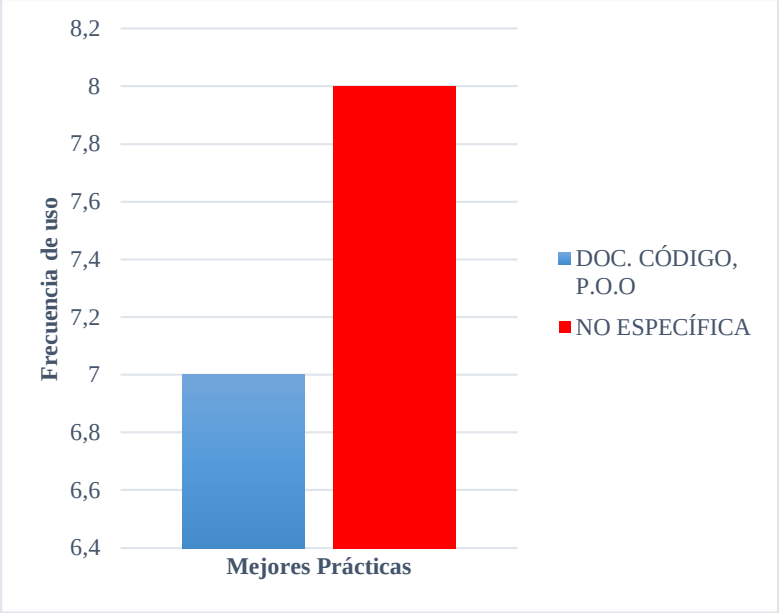
Nota: En esta Tabla muestra la valoración que se da a cada una de las mejores prácticas de las metodologías no tradicionales identificadas en las publicaciones científicas del 2011 al 2020 para la construcción de Sistemas Embebidos.

Demostración en grafico de barras

En la **Figura 39** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 39

Fase Codificación: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra la selección de los criterios de las mejores prácticas como documentación de código y Programación Orientada a Objetos, esto en la fase de codificación.

Interpretación:

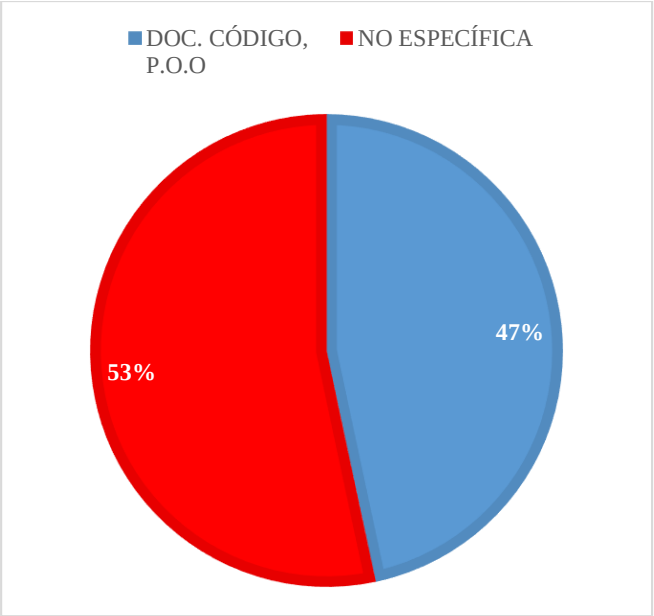
Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 15 publicaciones científicas de las enfocadas a metodologías no tradicionales, se ha identificado *7 artículos de investigaciones* en donde utiliza *Documentación de código y Programación Orientada a Objetos.* Sin embargo, se ha identificado que de las 15 publicaciones establecidas para la investigación y *8 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar se ha identificado las mejores prácticas como el uso de *Paradigma de Programación Orientada a Objetos* que tiene

una frecuencia de uso 7 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 40

Fase Codificación: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra la selección de los criterios de las mejores prácticas como documentación de código y Programación Orientada a Objetos, esto en la fase de codificación.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 15 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 15 publicaciones se puede observar que *7 artículos científicos* equivalen a 47% y corresponden a la aplicación de *Documentación de código y Programación Orientada a Objetos*. Sin embargo, se ha identificado 8 publicaciones que es

equivalente al 47% no especifican el uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos.

Instrumento de selección de mejores prácticas para SE: Fase Pruebas

Descripción del instrumento

El instrumento de análisis y clasificación de las mejores prácticas se estructura de la siguiente forma:

- **Pruebas funcionales y no funcionales***:* Permite comprobar que el SE funcionen a de acuerdo a las especificaciones de los requisitos funcionales y no funcionales.
- **Pruebas unitarias:** Este criterio permite comprobar el correcto funcionamiento de una unidad código.
- **Pruebas de aceptación:** Este tipo de prueba son realizadas con el cliente y este quien verifica las funcionalidades del producto software embebido.

En la **figura 41** se puede observar el análisis, selección y clasificación de las mejores prácticas del proceso de desarrollo de software para Sistemas Embebidos, gran número de las buenas practicas se derivan del proceso de Ingeniería de Software. Como se puede observar permite visualizar el análisis y clasificación de las mejores prácticas de la fase de diseño. Para esta etapa se utilizó 15 publicaciones científicas.

Figura 41

Instrumento de análisis de mejores prácticas: Fase Pruebas

Pruebas Funcionales	Pruebas no Funcionales	Pruebas Unitarias	Pruebas de Aceptación
0	0	0	0
0	0	0	0
0	0	3	4
0	0	0	0
0	0	3	0

Nota: Esta figura, muestra la selección de las mejores prácticas como pruebas funcionales y no funcionales, pruebas unitarias y pruebas de aceptación esto en la fase de pruebas.

(ver instrumento completo: https://docs.google.com/spreadsheets/d/1QkpCaj7ISKHAlKxwWruW6F_JYcltMLOCEJ5P1tFaZiw/edit?usp=sharing)

Semaforización de la frecuencia de uso de las mejores practicas

Para identificar la frecuencia de uso de las mejores prácticas en la construcción de SE, se agrupo las mejores prácticas para ello se estableció colores como una forma de identificarlos y contabilizarlos, de esta manera se puede tener control sobre el número de mejores prácticas y las publicaciones científicas que aportan para esta investigación.

Tabla 20

Semaforización de las mejores prácticas Fase Pruebas

Color	Criterio	Abreviatura
	Pruebas Funcionales	P. FUN
	Pruebas Funcionales	P.FUN
	Pruebas Unitarias	P.UNI
	Pruebas de Integración	P. INT
	Pruebas unitarias	P.UNI
	Pruebas de aceptación	P. ACEP
	Pruebas de integración	P.INT
	No Especificado	NO ESPECÍFICADO

Nota: En esta Tabla muestra la semaforización de las mejores prácticas identificadas en las metodologías no tradicionales del 2011 al 2020 para la construcción de Sistemas Embebidos.

Para contabilizar las mejores prácticas también se estableció criterio de valores por cada uno. Como se puede observar en la ***Tabla 21,*** se estableció una valoración para cada criterio con la finalidad de contabilizar e identificar el número de veces que se repite la aplicación de las mejores prácticas.

Tabla 21

Valoración de las mejores prácticas Fase Pruebas

Criterios	Valoración
Pruebas funcionales	1
Pruebas no funcionales	2
Pruebas unitarias	3
Pruebas de aceptación	4

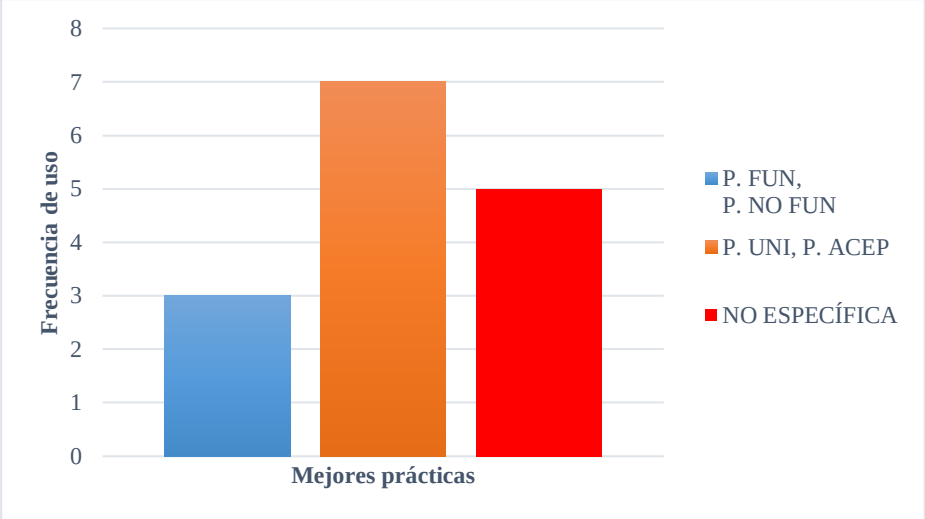
Nota: En esta Tabla muestra la valoración que se da a cada una de las mejores prácticas de las metodologías no tradicionales identificadas en las publicaciones científicas del 2011 al 2020 para la construcción de Sistemas Embebidos.

Demostración en grafico de barras

En la **Figura 42** se puede observar el resumen de las buenas prácticas de acuerdo al número de frecuencia de uso, para una mejor interpretación visual se procede a trabajar con la semaforización establecida.

Figura 42

Fase Pruebas: Frecuencia de uso de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas y los criterios seleccionados son pruebas funcionales y no funcionales, pruebas unitarias, pruebas de aceptación.

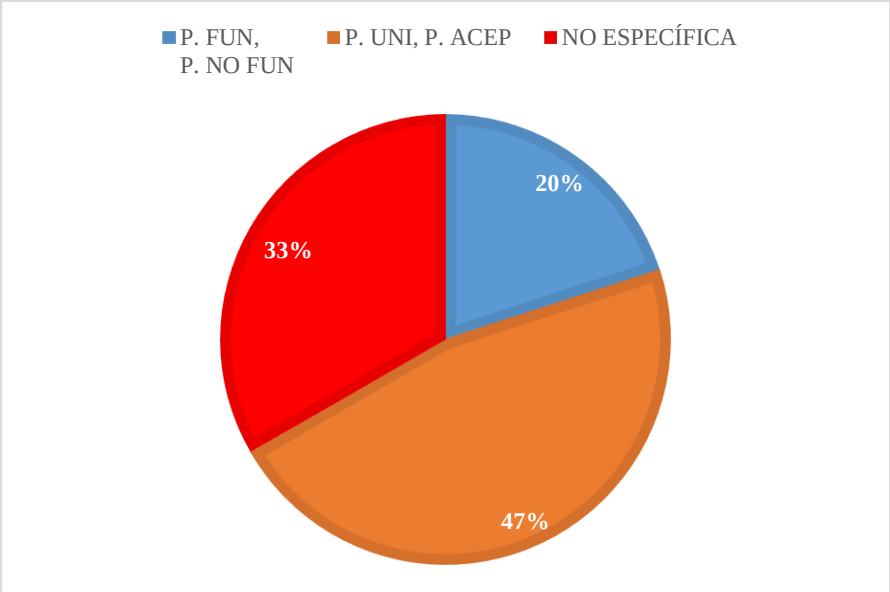
Interpretación:

Una vez analizados y clasificadas las mejores prácticas en el proceso de desarrollo de software embebido, demuestran que, de las 15 publicaciones científicas de las enfocadas a metodologías no tradicionales, se ha identificado *3 artículos de investigaciones científicas en* donde utiliza *Pruebas funcionales y no funcionales.* De la misma manera se observa *7 publicaciones científicas* que hace referencia al uso *Pruebas unitarias y Pruebas de aceptación*. Para finalizar se identificó que de las 15 publicaciones son *5 publicaciones* no reportan algún tipo de buenas o mejores prácticas para el proceso de desarrollo se software para sistemas embebidos. Como se puede observar se ha identificado las mejores prácticas como el uso de *Pruebas unitarias y pruebas de aceptación* tienen una frecuencia de aplicación 7 lo cual permite observar que estas prácticas serían las más optimas de aplicar al proceso de desarrollo de software para SE.

Demostración en grafico de pastel

Figura 42

Fase Pruebas: Porcentaje de aplicación de las mejores practicas



Nota: Esta figura, muestra las mejores prácticas aplicadas en la construcción de SE en la fase de pruebas y los criterios seleccionados son pruebas funcionales y no funcionales, pruebas unitarias, pruebas de aceptación.

Interpretación:

Los datos adquiridos con la revisión de la literatura, y mediante la semaforización establecida, demuestran que las 15 publicaciones científicas enfocadas a las metodologías tradicionales de desarrollo de software, demuestran que, de las 15 publicaciones se puede observar que 3 artículos científicos equivalen a 20% y corresponden a la aplicación de *Pruebas Funcionales y No Funcionales*. En la gráfica también se demuestra que la aplicación de *Pruebas unitarias y Pruebas de aceptación* se ha identificado en 7 publicaciones que equivalen al 47%. Por último, se ha identificado 5 publicaciones que es equivalente al 33% que no especifican el

uso o aplicación de las mejores prácticas para el proceso de desarrollo de software para Sistemas Embebidos en la fase de *Pruebas.*

Principales hallazgos de mejores prácticas para la construcción de SE

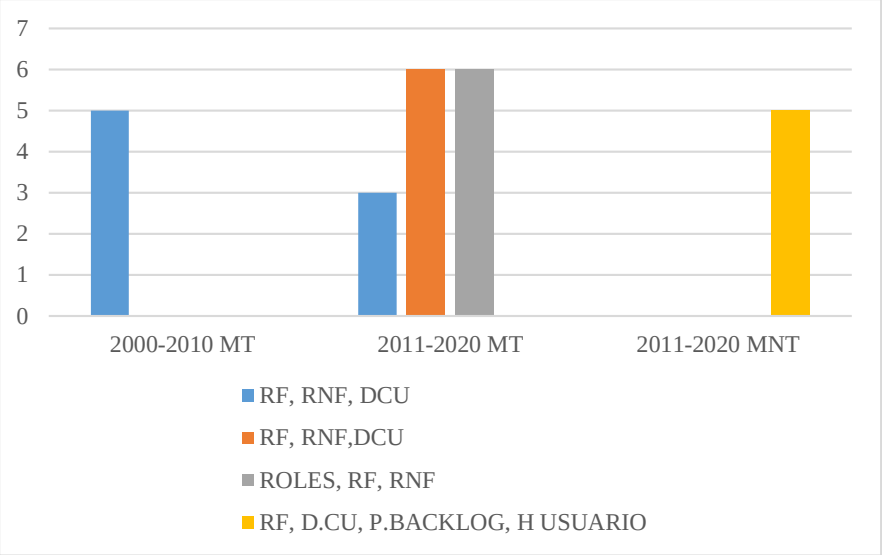
Las publicaciones científicas analizadas son el producto de una investigación que corresponden a la aplicación de las mejores prácticas del proceso de desarrollo de software para la construcción de sistemas embebidos. Con el objetivo de identificar las buenas practicas más utilizadas dentro de las metodologías tradicionales y no tradicionales se crean gráficos resumen que a continuación serán descritos de manera breve:

Mejores prácticas en la fase de análisis

Las mejores prácticas de desarrollo contribuyen a mejorar la construcción de software para sistemas embebidos. Como se puede observar la **figura 43***,* entre los años 2000 al 2010 se observa la aplicación de *Requisitos funcionales, Requisitos no funcionales y Diagramas de caso* en el desarrollo de SE, en donde se puede contabilizar que se aplican 5 veces este tipo de mejores prácticas. Entre los años del 2011 hasta el 2020 se continua aun con la aplicación de las metodologías tradicionales, y se identifican las mejores prácticas con *Roles* con una frecuencia 3 de aplicación, en este mismo rango se puede apreciar la aplicación de *Requisitos funcionales y no funcionales y Diagramas de caso de uso* con una frecuencia 6 de aplicación y de la misma manera se identifica *Roles, Requisitos funcionales y Requisitos no funcionales* con una frecuencia 6 de aplicación. Sin embargo, en el rango de años del 2011 al 2010 en lo que respecta a las metodologías no tradicionales se ha identificado una participación *Requisitos funcionales*, *Diagramas de Caso de Uso, Product Backlog e Historias de Usuario* con una frecuencia de aplicación de 5.

Figura 43

Principales hallazgos de las mejores prácticas: Fase Análisis



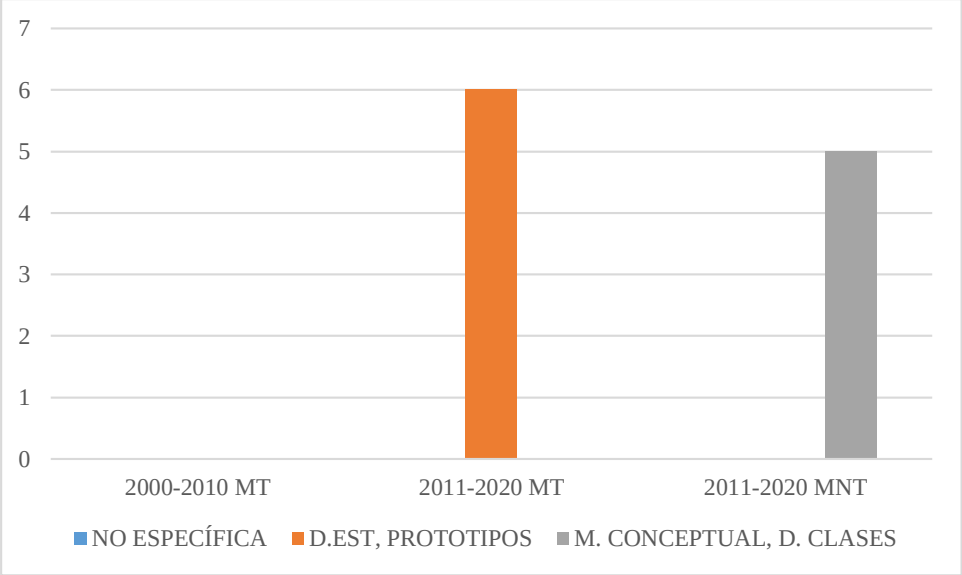
Nota: Esta figura muestra los principales hallazgos de las mejores prácticas identificadas en la fase de análisis propuesto desde el año 2000 al 2020.

Mejores prácticas en la fase de diseño

Como se puede observar en la **Figura 44**, correspondiente en los años 2000 al 2010 no se presenta ningún tipo de mejores prácticas. Respecto a los años 2011 al 2020 se encuentran distribuidas en metodologías tradicionales en donde se ha identificado que los *Diagramas de estado y Prototipos* poseen una frecuencia 6 de aplicación. En cambio, las metodologías no tradicionales y de la misma manera se encuentran identificadas por el *Modelo Conceptual y Diagramas de Clases* con una frecuencia de 5 de aplicación y encuentra en el rango del 2011 al 2020.

Figura 44

Principales hallazgos sobre las mejores prácticas: Fase diseño



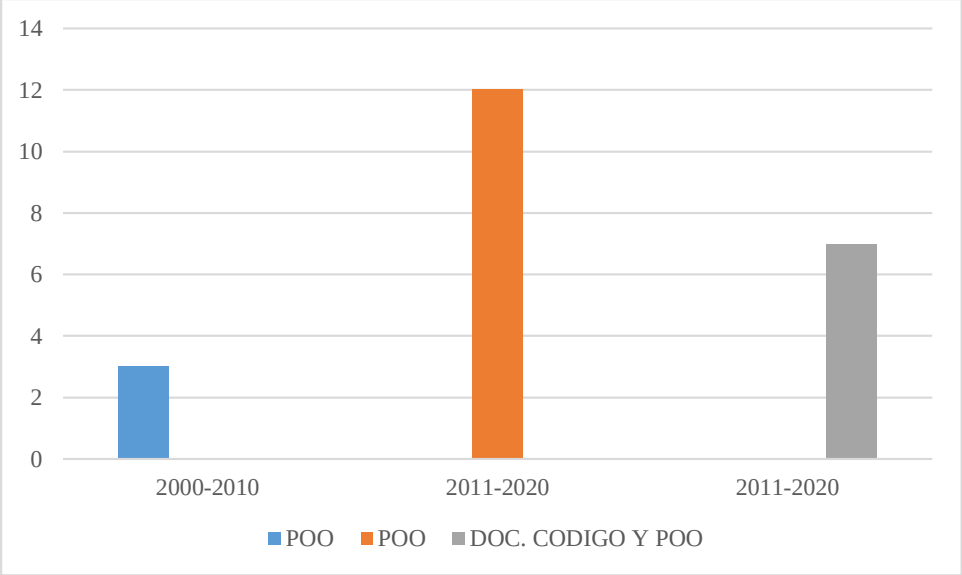
Nota: Esta figura muestra los principales hallazgos de las mejores prácticas identificadas en la fase de diseño propuesto desde el año 2000 al 2020.

Mejores prácticas en la fase de codificación

Como se puede observar **Figura 45**, entre los años 2000 al 2010 se observa el uso de *Lenguajes de Programación Orientada a Objetos* el desarrollo de SE, en donde se puede contabilizar que se aplican 3 veces este tipo de mejores prácticas. Entre los años del 2011 hasta el 2020 se continua aun con la aplicación de las metodologías tradicionales, y se identifican las mejores prácticas como la técnica de la *Programación Orientada a Objetos* con una frecuencia 12 de aplicación, en este mismo rango en referencia a las metodologías no tradicionales se puede apreciar la aplicación de *Documentación de código y Programación Orientada a Objetos con* una frecuencia 7 de aplicación.

Figura 45

Principales hallazgos de las mejores prácticas: Fase de Codificación



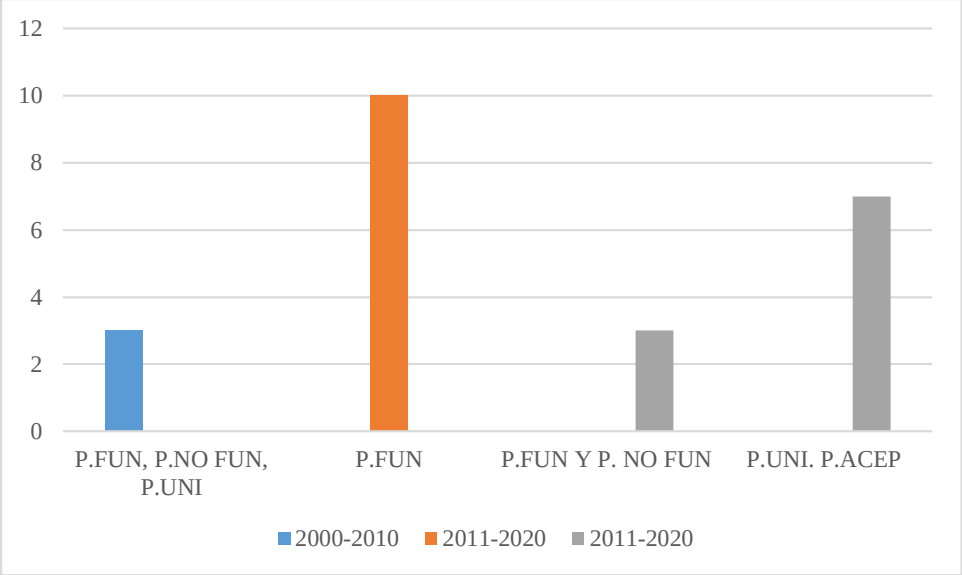
Nota: Esta figura muestra los principales hallazgos de las mejores prácticas identificadas en la fase de codificación propuesto desde el año 2000 al 2020.

Mejores prácticas en la fase de pruebas

Como se puede observar en la **Figura 46**, entre los años 2000 al 2010 se observa el uso de *Pruebas funcionales, pruebas no funcionales y pruebas unitarias* en el desarrollo de SE, en donde se puede contabilizar que se aplican 3 veces este tipo de mejores prácticas. Entre los años del 2011 hasta el 2020 se continua aun con la aplicación de las metodologías tradicionales, y se identifican la aplicación de una sola practica como *Pruebas funcionales* con una frecuencia 10 de aplicación, en este mismo rango en referencia a las metodologías no tradicionales se puede apreciar la aplicación de *Pruebas funcionales y no funcionales, Pruebas unitarias y Pruebas de aceptación con* una frecuencia 7 de aplicación.

Figura 46

Principales hallazgos de las mejores prácticas: Fase de Pruebas



Nota: Esta figura muestra los principales hallazgos de las mejores prácticas identificadas en la fase de pruebas propuesto desde el año 2000 al 2020.

Propuestas actuales de mejores prácticas para el desarrollo de SE

En la actualidad se han desarrollado diferentes investigaciones en el dominio y aplicación de las mejores prácticas que permitan desarrollar adecuadamente un producto software de calidad, buscando un equilibrio al construir un software de SE. Las metodologías de desarrollo tienen por objetivo establecer fases para guiar a los desarrolladores en el proceso de desarrollo de software embebido, así también entregar respuesta a los cambios que sufre el sistema o software durante las diferentes fases. A continuación, se analizarán las principales prácticas como propuesta para el desarrollo de SE con el objetivo de identificar buenas practicas tanto de metodologías tradicionales y no tradicionales y establecer así un proceso eficiente de un modelo de diseño para sistemas embebidos. Para las propuestas de las mejores prácticas, se consideran aquellas prácticas que tengan una frecuencia igual o mayor a 5.

Figura 47

Propuestas de las mejores prácticas para el desarrollo de SE

Mejores Prácticas	Fases					Puntos a favor	Puntos en contra
	Especificación Requerimientos	Análisis de Requisitos	Diseño	Desarrollo de Software	Pruebas		
Requisitos funcionales	✓					Especifican los aspectos funcionales que el software deba realizar.	Se enfoca únicamente en el levantamiento de requisitos funcionales y no contempla requisitos no funcionales.
Requisitos no funcionales	✓					Especifican los comportamientos específicos del software.	Se enfoca únicamente en el levantamiento de requisitos no funcionales.
Diagrama de Caso de Uso		✓				Describe las actividades que deberá realizar alguien o algo para llevar a cabo algún proceso.	Los caso de uso nos permiten tener la capacidad de saber explicitamente lo que el cliente necesita.
Historia de Usuario		✓				Documento informal de los requisitos, en caso de ausencia de comunicación, con relación al proceso de Prueba y Aceptación.	Difícil de medir la eficacia de un proyecto con los resultados de una historia
Diagrama de secuencia			✓			Describen el comportamiento de un sistema.	Representa estados que puede adquirir una clase.
Diagrama de clases			✓			Representa un conjunto de clases que forman parte de un sistema.	Los diagramas de clases sson estaticos
Modelo conceptual			✓			Identifica relaciones entre diferentes entidades.	No especifica ningún atributo y claves.
Documentación de código				✓		Permite evitar errores y herramientas de aprendizaje que están a mano para nuevos empleados.	La documentación puede retrasar un proyecto.
P. Orientada a objetos				✓		Código fácil de mantener.	Curva de aprendizaje. Comprensión de conceptos es un desafío.
Pruebas funcionales					✓	Centra en la experiencia del usuario.	Se limitan a probar qué tan bien una aplicación hace lo que se supone que hace posible perder errores fuera de este alcance
Pruebas de aceptación					✓	Se realizan por el cliente para determinar la confianza del sistema a nivel funcional y técnico.	El cliente no comprenda las limitaciones técnicas que puede tener el software.
Pruebas unitarias					✓	Permite comprobar un fragmento de código funciona de manera correcta.	No identifica todos los defectos del código.

Nota: Esta figura se observa las propuestas de las mejores prácticas para el desarrollo de SE que se han extraído de publicaciones científicas.

Conclusiones de las propuestas actuales de mejores practicas

El software se ha constituido en una parte importante en el desarrollo de Sistemas Embebidos. Sin embargo, es difícil de construir debido a su complejidad por estar empotrados en un entorno físico. Esto puede ser solventado con las metodologías tradicionales y no tradicionales con la ayuda de las mejores prácticas que están presentes en el ciclo de vida y se puede desarrollar software para SE.

De acuerdo al estudio realizado, es conveniente describir cada de una de las fases y mejores prácticas identificadas para establecer un modelo alternativo:

El análisis de requerimientos para el desarrollo de software para los SE realiza mediante el levantamiento de *Requisitos funcionales y Requisitos no funcionales*, esto permitirá especificar los aspectos funcionales del software.

Para la fase de diseño de un SE, está basado en el uso de *Diagramas de estado, Diagramas de clases y Modelo conceptual,* mismo que se centra en el uso de UML.

Las mejores prácticas seleccionadas para aplicar en la fase de Codificación son *Documentación de código y Programación de Orientada a Objetos*, estas prácticas proporcionaran una solución de código orientada a objetos al construir un SE.

Los trabajos revisados proveen mejores prácticas de *Pruebas funcionales, Pruebas de aceptación y Pruebas unitarias* para que se apliquen en la fase de pruebas de software lo cual permitirá obtener un producto de calidad.

Así, las mejores prácticas propuestas, integrará el nuevo modelo de diseño que ayude al proceso de desarrollo de software de SE y obtener un producto software de calidad. Además, el modelo que se construirá añade los principios de buenas prácticas de las metodologías clásicas y agiles que incorpora la capacidad

de administración y mejorar así la especificación del proceso; así también permite establecer un conjunto de documentos o artefactos para guiar a los desarrolladores en el proceso de desarrollo de software. Este modelo de diseño será denominado como BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems).*

Implementación Del Modelo De Trabajo Caso Estudio

Introducción a la solución

De acuerdo a la búsqueda de la literatura realizada en el Capítulo III, se puede observar los esfuerzos realizados para aplicar las mejores prácticas de desarrollo de software de sistemas embebidos. Por lo que, el problema actual del desarrollo de los SE, no es la falta de estándares, normas, modelos o metodología, más bien es la falta de procesos específicos y eficaces para aplicarlos con éxito. Con esto se ve la necesidad de optimizar el proceso de desarrollo de los SE para hacer frente a la calidad que exige este tipo de software.

A continuación, se tomará los resultados del análisis de las mejores prácticas de las metodologías tradicionales y no tradicionales presentadas en el capítulo anterior, con el objetivo de introducir el modelo de diseño alternativo para el proceso de desarrollo de software de sistemas embebidos, propuesta en este trabajo de investigación.

Propuesta formal de solución

Mejorar los procesos convencionales utilizados para el desarrollo de SE en áreas industriales y académicos, se ha constituido una preocupación latente lo cual ha propiciado el desarrollo de un sin número de propuestas para incorporar formalidades al proceso de desarrollo de software, pues no especifican las necesidades reales del proceso de desarrollo para este tipo de sistemas.

Como resultado de esta revisión literaria, los métodos, herramientas y técnicas utilizadas en el proceso de desarrollo de software para SE, no solo varían dentro de las empresas o universidades sino también entre compañías e institutos de investigación, por ende, los desarrolladores de software para sistemas embebidos tienden a desarrollar un producto software con estandarizaciones, esto

ha evitado tener un producto final de calidad. A partir de la revisión bibliográfica propuesta en el Capítulo II y Capítulo III, se compararon metodologías de acuerdos a las fases de desarrollo incorporando las mejores prácticas de desarrollo de software a SE. Observadas las propuestas de cada artículo científico para el proceso de desarrollo de SE, se enfoca en *Ayudar a Mejorar el Proceso de Desarrollo de Software*.

Estructura del modelo de trabajo del proceso de desarrollo de SE

Un modelo de trabajo consiste en una guía para especificar los elementos y las relaciones dentro de un software de SE, y un proceso completo de desarrollo como *planificación, roles, ciclos iterativos, diagramas ulm, pruebas* y otros. Bajo este concepto se diseñó BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems).* BPSSEM tiene por objetivo ayudar al proceso de desarrollo de software de los SE mediante un modelo de diseño empleando técnicas de modelado.

A continuación, se describirán las mejores prácticas y artefactos de la solución propuesta (BPSSEM), y estos elementos se utilizarán para responder a la hipótesis planteada para el proyecto de investigación.

Elementos del modelo de trabajo del proceso de desarrollo de SE

BPSSEM es un modelo de trabajo propuesto para el desarrollo de sistemas embebidos, pues es un conjunto integrado de actividades para ser una guía a los desarrolladores durante la creación de software de SE. BPSSEM consta de un ciclo de vida que está constituida por cinco fases.

Este modelo de trabajo incorpora principios de *Procesos de Desarrollo de Software* y consta de actividades y herramientas uml. El modelo está estructurado

por cinco fases con 11 buenas practicas aplicadas al proceso de desarrollo. BPSSEM ha sido diseñado con un enfoque iterativo para asegurar que el proceso de desarrollo sea probado en cada fase, además utiliza UML como un lenguaje de modelado, cada fase del modelo puede ser representado mediante diagramas UML.

Las áreas de proceso del modelo son un conjunto de buenas prácticas de desarrollo, cuando se implementan de forma colectiva, satisfacen las necesidades del proyecto. A continuación, se describen el proceso y las buenas practicas del modelo BPSSEM:

Especificación de requisitos: Este proceso proporciona las tareas necesarias para la identificación y la administración de los requisitos de software de los SE.

Análisis de requisitos: Este proceso permite analizar los requisitos identificados para tener una mejor comprensión y visualización del requisito se utiliza la herramienta UML. Los diagramas de modelado a utilizar en esta etapa es el Diagrama de Caso de Uso, como también se puede aplicar las Historias de usuario.

Diseño: Este proceso consta de dos actividades: diseño funcional (diseño de software, lógica) y diseño de la arquitectura (diseño hardware, electrónica). Esta etapa se fundamenta en la especificación de requisitos y el análisis.

Desarrollo de software: Este proceso posee actividades para refinar o extender modelos de la plataforma independiente en el diseño plataforma especifica.

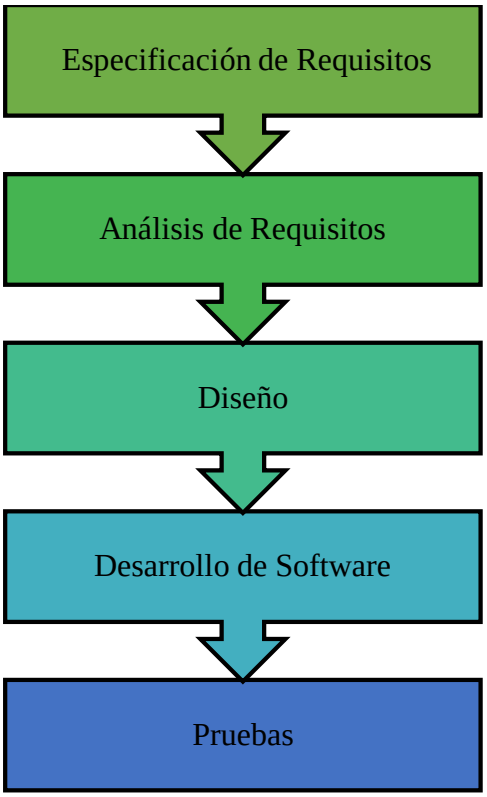
Pruebas: Este proceso realiza las pruebas necesarias para obtener un producto de calidad antes de la entrega al cliente.

Modelo de trabajo basado en las mejores prácticas para el desarrollo SE

En este apartado se procede a especificar el proceso a seguir para desarrollar software para Sistemas Embebidos de calidad. Se describen 5 fases para el proceso de desarrollo de software embebido, estas fases se eligieron de acuerdo a las categorías antes mencionadas para el proceso de selección de las mejores prácticas.

Figura 48

Etapas del Modelo de Trabajo



Nota: Para mejor detalle de las etapas del Modelo de Trabajo, se ha realizado una guía para su aplicación dentro del desarrollo de software para Sistemas Embebidos. (**ver guía:** https://drive.google.com/file/d/1X9f3fswhmw7dcwXR797Iaix4FuokauF5/view?usp=sharing)

Especificación Requisitos

Propósito

El propósito es el recogimiento de las necesidades del cliente y el usuario. En esta fase se establece que es lo que el software va a realizar.

Descripción

Tener una comunicación con los usuarios es importante al momento de crear cualquier tipo de software. Este es el momento que permite capturar lo que el software o sistema debe hacer.

Aquí es donde se define los Requisitos funcionales y no funcionales para el desarrollo software para Sistemas Embebidos, por lo tanto, estas prácticas formaran parte del documento de Especificación de requisitos.

Mejores prácticas

Requisitos funcionales

Requisitos no funcionales

Análisis de Requisitos

Propósito

Extraer los requisitos del producto del software. En esta etapa se debe aplicar la habilidad y experiencia de los procesos de ingeniería de software, así también como técnicas o herramientas que permita visualizar la funcionalidad del software de los SE.

Descripción

Para desarrollar un software para un sistema embebido es averiguar qué es lo que debe hacer el sistema. El análisis corresponde al proceso que se intenta describir las características del software debería poseer. Por lo tanto, es necesario analizar los

requerimientos ya que se puede identificar errores a tiempo, y se ha demostrado que eliminar errores en la etapa de análisis es más económico que subsanarlo en las etapas finales de un proyecto.

En esta etapa se ha identificado como mejores prácticas de análisis a los Diagramas de Casos de Uso e Historias de Usuarios.

Mejores prácticas

Historias de usuario

Casos de uso

Diseño

Propósito:

El propósito de la fase del Diseño del producto es ayudar a la realización de las tareas del desarrollo del producto. Esta etapa consta del diseño de electrónico (hardware) y diseño lógico (software).

Descripción:

Para el diseño de los sistemas embebidos, se debe tomar en cuenta dos arquitecturas. Por un lado, está la arquitectura hardware su construcción se basa en un microprocesador, y por otra parte está la arquitectura software que incluye ciertos sistemas operativos, lenguajes de programación, herramientas de modelado. La integración de estas dos arquitecturas constituye la arquitectura de un sistema embebido.

Mejores prácticas:

La fase de diseño involucra generar un documento de arquitectura que trabaja con las siguientes vistas:

Diagrama de clases

Diagrama de secuencia

Modelo conceptual

Desarrollo de software

Propósito:

El propósito de esta fase es construir o desarrolla el software de un sistema embebido. Esta actividad la realiza por un equipo de programadores, cuyo número de equipo y tiempo depende de los acuerdos especificados en la planificación.

Descripción

Durante esta etapa el equipo realiza gran parte de la construcción de los componentes del software como la documentación y código. Para el desarrollo de software se utiliza los artefactos establecidos en la fase de diseño, así también se establece el paradigma de programación, lenguajes de programación, métodos y técnicas para el proceso de desarrollo de software.

Mejores prácticas:

Documento codificación

Lenguaje POO

Pruebas

Propósito:

El propósito de esta fase es realizar pruebas sobre la solución establecida, las pruebas de esta etapa enfatizan el uso y manipulación del producto bajo condiciones

realistas. El equipo de pruebas y desarrollo se enfoca en priorizar y resolver errores y preparar la solución para la implementación.

Descripción:

La fase de pruebas es esencial dentro del proceso de desarrollo del software. En esta etapa se detecta los errores de software lo más pronto posible. Pues consiste en comprobar que el software realice de forma correcta las tareas indicadas en la especificación de requerimientos. El software se puede probar por separado cada módulo del software, y después probarlo de forma integral.

En esta etapa de pruebas sirve para asegurar de que no existan errores funcionales y que cada requerimiento indicado en la etapa de planificación se cumpla.

Mejores prácticas:

Pruebas funcionales

Pruebas unitarias

Pruebas de aceptación

Resultados experimentales

Para evaluar la aplicabilidad de BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems)*, se ha considerado en realizar Reingeniería de procesos de desarrollo de software en el proyecto de investigación de Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos sobre el módulo de ***"Desarrollo de plataforma de comunicación a través de Red Cedia".***

El proyecto mencionado es un Sistema Embebido desarrollado en el Laboratorio de Investigación del Departamento de Eléctrica y Electrónica de la

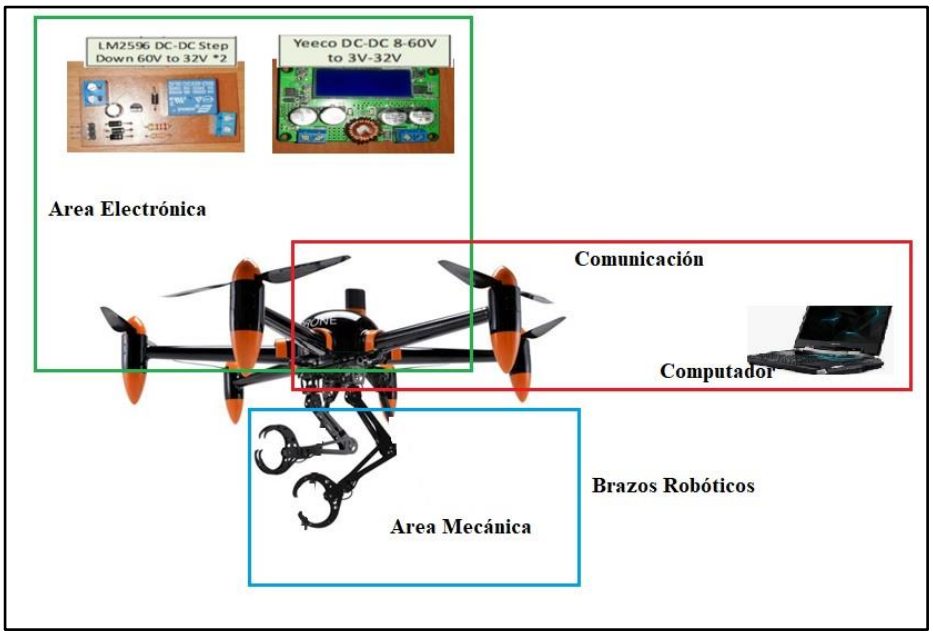
Universidad de las Fuerzas Armadas Espe Sede Latacunga y el apoyo de personal de CEDIA.

El módulo de ***Desarrollo de plataforma de comunicación a través de Red Cedia*** es el primer sistema embebido diseñado aplicando reingeniería de software, y a su vez se utilizó el modelo de trabajo BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems)*. El modelo de trabajo BPSSEM establece las mejores prácticas de desarrollo para el diseño y construcción de SE definidas en ciclo o etapas.

El Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos, es un sistema una plataforma aérea no tripulada capaz de manipular cargas con “óptimo consumo de energía” para lo cual se ha considerado el estudio de diferentes áreas de conocimiento, para la construcción de dos brazos robóticos incorporados a la base del vehículo aéreo no tripulado. El proyecto de investigación hace énfasis en las áreas de la mecánica, la electrónica y comunicación; la primera área corresponde al diseño y construcción de dos brazos robóticos de cinco grados de libertad para una mejor manipulación de la carga y el acoplamiento físico al UAV, por otro lado, el área de la electrónica se encarga de implementar los acondicionamientos de las señales eléctricas para el acoplamiento de los brazos robóticos con el UAV y para finalizar la comunicación que corresponde a la transmisión de datos (ver Figura 49).

Figura 49

Arquitectura: Construcción Mecánica / Eléctrica/Comunicación



Nota: Tomado de (Laboratorio de Investigación del Departamento de Eléctrica y Electrónica ESPE Sede Latacunga,2020)

Especificación de Requisitos

Para la Especificación de Requisitos Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos del módulo *"Desarrollo de una plataforma de comunicación a través de la Red Cedia",* se lo realizo mediante Reingeniería de procesos de software y parte de las especificaciones son proporcionadas por el Proyecto de Investigación CEPRA desarrollado en los Laboratorios de Investigación del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas ESPE sede Latacunga con el aporte de CEDIA. A continuación, se presenta una descripción de las especificaciones proporcionadas.

El enfoque principal del proyecto es implementar una herramienta que permita compartir información entre vehículos no tripulados y controladas por un operador humano que maneja un dispositivo de telecomando, un vehículo no tripulado o UAV (del inglés, unmanned aerial vehicle). El objetivo principal de este trabajo es proponer una arquitectura de comunicación para sistemas de control autónomo y de teleoperación de drones para actividades de acción colaborativa.

El equipo encargado del desarrollo del software para el sistema embebido inicio por definir las características que tendrá el sistema. Para ello, a partir de las especificaciones del sistema se realizó la fase de *Especificación de Requerimientos*. Para esta fase el equipo para el equipo de desarrollo se diseñó un artefacto ERSE (Especificación de Requisitos de Software Embebido), para presentar los requisitos de una manera entendible y consistente para el equipo de desarrollo.

Este artefacto especifica los requisitos, en este artefacto se especifican los requisitos por componente de producto, los requisitos en sí, el tipo de requisito, la fuente y la prioridad. A continuación, se presenta la fase de especificación de requisitos (Artefacto ERSE).

Fecha: _____	Artefacto ERSE- Especificación de requisitos de Software Embebido			
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos			
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA			
Nombre del responsable:	Equipo de desarrollo			
Descripción general Perspectiva del producto				
La parte fundamental de este proyecto es compartir información entre vehículos no tripulados UAV, tomando en cuenta el manipulador humano quien maneja un dispositivo de telecomando. Para establecer comunicación entre los dispositivos UAV incorporado el brazo robótico, se creara una arquitectura de comunicación para sistemas de control autónomo y de teleoperación de drones para actividades de acción colaborativa.				
Funcionalidad del sistema				
1.	Compartir información entre UAV y Brazo Robótico			
2.	Transmitir datos			
3.	Establecer canal de comunicación WebSocket			
4.	Obtener datos de la plataforma móvil o brazo robótico			
Características de los usuarios				
Tipo de usuario: Operador humano				
Formación: Piloto de UAV				
Habilidades: S/N				
Restricciones				
1.	La arquitectura de comunicación permitirá controlar el manipulador aéreo de manera bilateral.			
2.	Utilizar ROS y el ROSBridge Server como plataforma para el intercambio de datos.			
Suposiciones y dependencias				
1.	Exceder el límite del peso en la carga de UAV.			
2.	Exceder número de movimientos del brazo robótico incorporado en el UAV.			
<u>Requisitos específicos</u>				
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos			
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA			
Numero de requisito:	Req 1			
Nombre del requisito:	Compartir información entre UAV y Brazo Robótico			
Descripción del requisito:	El *Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos*, y para compartir información del dispositivo UAV y Brazo Robótico se deberá establecer un canal de comunicación entre las plataformas de UAV y los Controladores, y se crea un canal mediante un protocolo TCP Server. Y los datos para el manejo de la plataforma (UAV y Brazo Robótico) serán enviados por el control remoto. El TCP Server será el encargado de enviar instrucciones del control remoto al controlador y a la plataforma.			
Tipo:	Operacional	Funcional X	Diseño	Restricción
Fuente de requisito:	Funcionalidad del sistema			
Prioridad:	Alta/Esencial X	Media/Deseado	Baja/Opcional	Ninguna

Requisitos específicos				
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos			
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA			
Numero de requisito:	Req 2			
Nombre del requisito:	Transmisión de datos			
Descripción del requisito:	El *Sistema Colaborativo de Robots Aéreos* realizará la transmisión de datos mediante el servidor TCP, y este tendrá dos tareas, en donde la tarea A es el encargado de pedir las instrucciones control y estas deben estar disponibles para cualquier usuario. Mientras que la tarea B debe esperar las instrucciones por parte del usuario y estas instrucciones deben ser reenviadas al UAV.			
Tipo:	Operacional	Funcional X	Diseño	Restricción
Fuente de requisito:	Funcionalidad del sistema			
Prioridad:	Alta/Esencial X	Media/Deseado	Baja/Opcional	Ninguna

Requisitos específicos				
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos			
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA			
Numero de requisito:	Req 3			
Nombre del requisito:	Establecer canal de comunicación WebSocket			
Descripción del requisito:	El *Sistema Colaborativo de Robots Aéreos* permitirá establecer una comunicación entre la plataforma (UAV, Brazo Robótico) y el controlador mediante un canal de comunicación sobre Websocket server. Este canal de comunicación permitirá obtener datos de la plataforma móvil o brazo robótico. Los datos obtenidos pasaran a un servidor externo (Servidor Cedia) mediante el canal de WebSocket.			
Tipo:	Operacional	Funcional X	Diseño	Restricción
Fuente de requisito:	Funcionalidad del sistema			
Prioridad:	Alta/Esencial X	Media/Deseado	Baja/Opcional	Ninguna

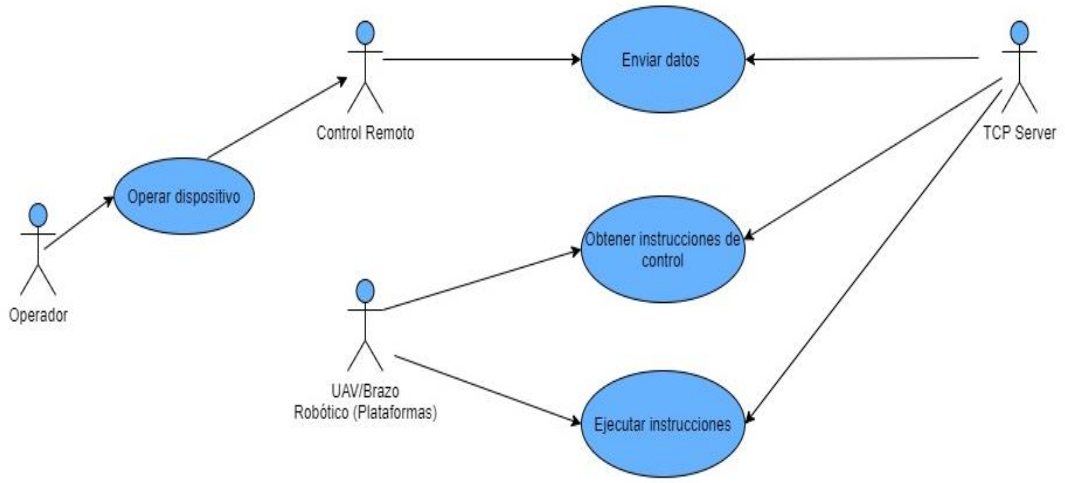
Requisitos específicos				
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos			
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA			
Numero de requisito:	Req 4			
Nombre del requisito:	Obtener datos de la plataforma móvil o brazo robótico			
Descripción del requisito:	El Sistema Colaborativo de Robots Aéreos permitirá obtener datos de la plataforma móvil o brazo robótico se debe utilizar el canal de comunicación, este debe ser construido un nodo WebSocket, este nodo permitirá asignar un identificador al cliente y de esta manera se determinará si es el controlador o plataforma, lo cual permitirá gestionar datos dependiendo el tipo de cliente. Las acciones a realizar por el nodo Websocket son: 4.1 Abrir un puerto del Servidor y este se quedará esperando las conexiones de los clientes. 4.2 Registrará la Ip del cliente, y se asignará a un listado de clientes conectados al WebSocket. 4.3 Se asignará un identificador al cliente e se inicia el hilo (Thread) que corresponda al cliente. 4.4 Los clientes deben estar conectados y autenticados en el websocket enviarán los datos y será almacenados en una base datos y reenviados mediante al cliente. 4.5 Se detiene el hilo de cliente y se desconecta, y se borra del listado de clientes conectados al websocket.			
Tipo:	Operacional	Funcional X	Diseño	Restricción
Fuente de requisito:	Funcionalidad del sistema			
Prioridad:	Alta/Esencial X	Media/Deseado	Baja/Opcional	Ninguna

Análisis de Requisitos

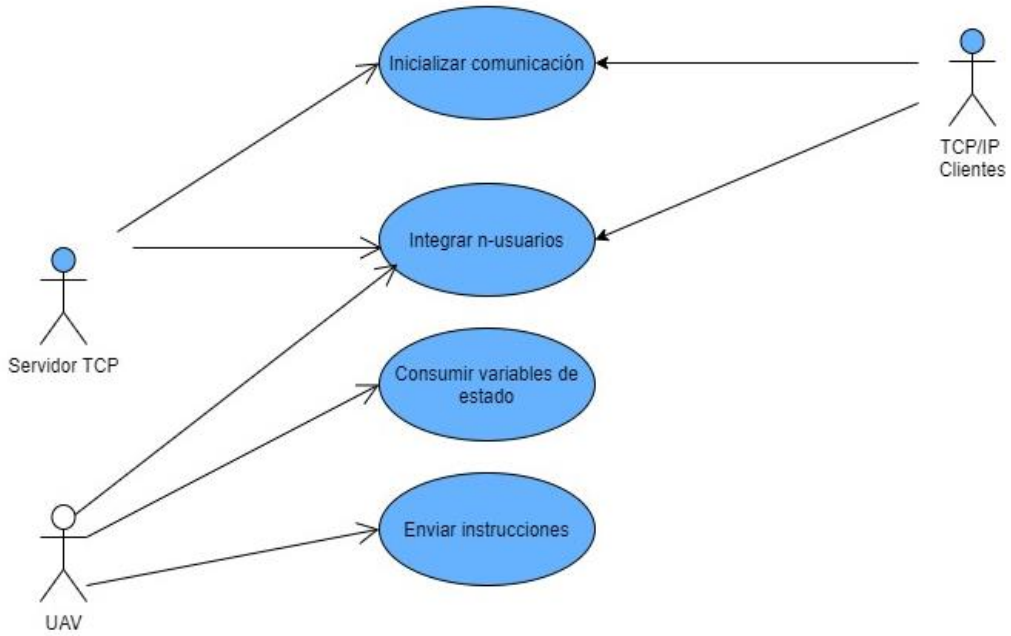
Diagrama de Caso de Uso Funcional Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos.

A continuación, se establece los diagramas de casos de uso (Artefacto DCUF) que describen el sistema en términos de las distintas formas en que se utiliza.

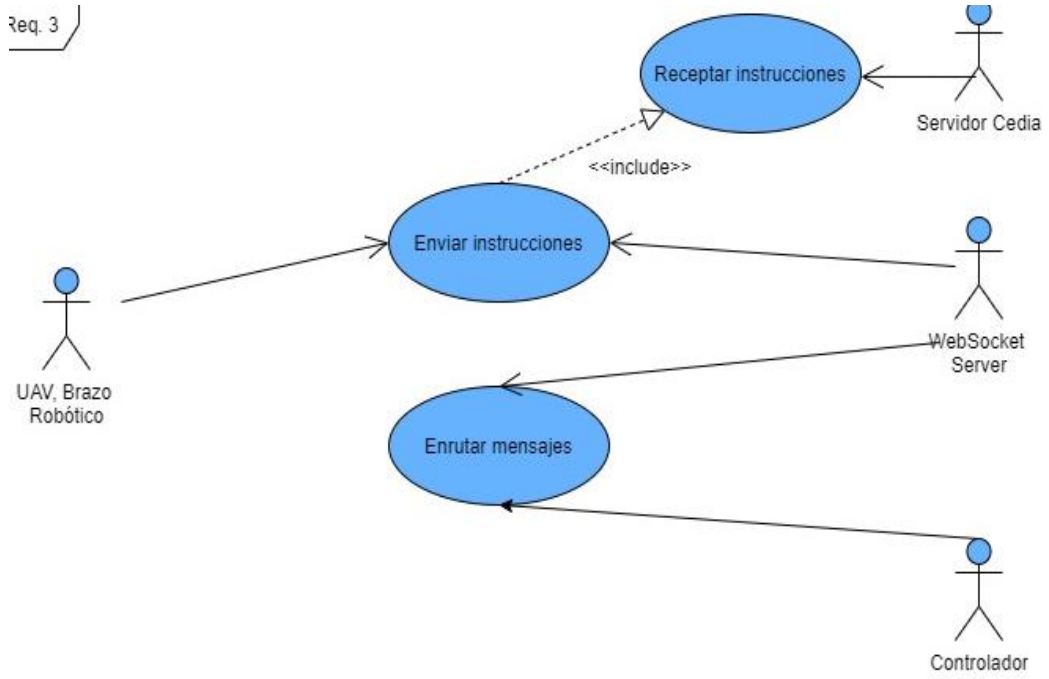
Fecha: ……..	Artefacto DCUF-Diagrama de casos de uso funcional
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Numero Caso de Uso:	N° 1
Nombre del responsable:	Equipo de desarrollo
CU: Compartir información entre UAV y Brazo Robótico	



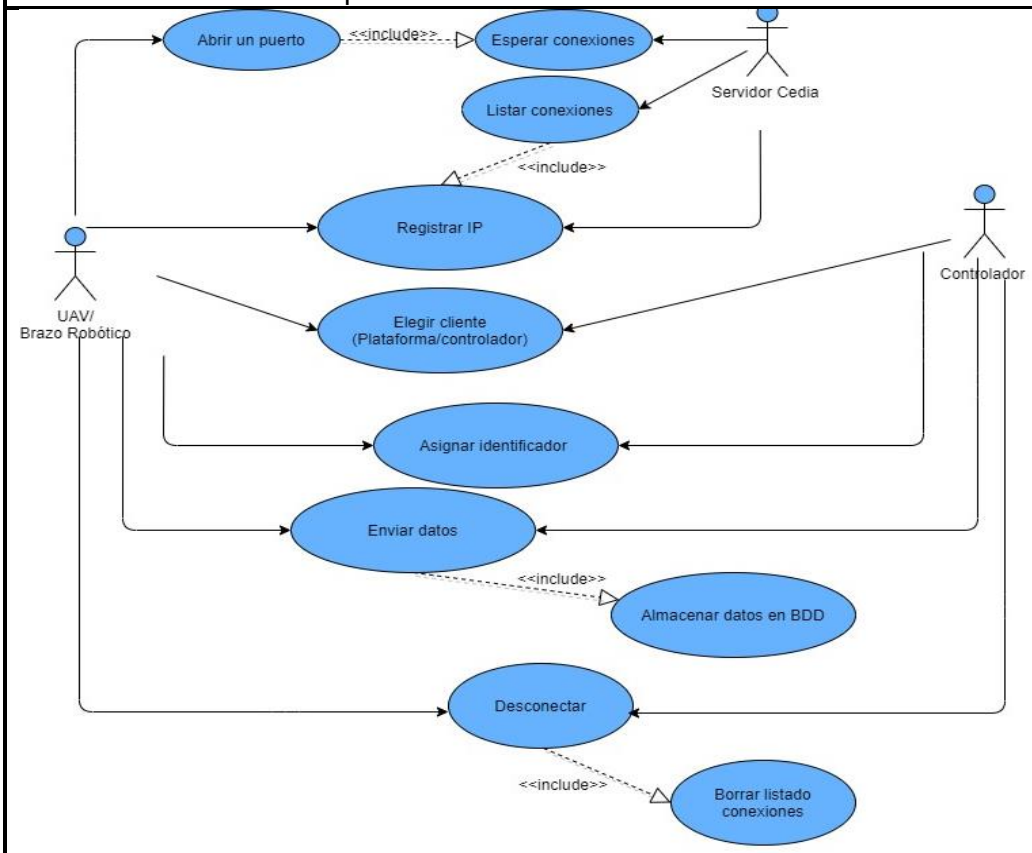
Fecha: ……..	Artefacto DCUF-Diagrama de casos de uso funcional
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Numero Caso de Uso:	N° 2
Nombre del responsable:	Equipo de desarrollo
CU: Transmisión de datos	



Fecha: ……..	Artefacto DCUF-Diagrama de casos de uso funcional
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Numero Caso de Uso:	N°3
Nombre del responsable:	Equipo de desarrollo
CU: Establecer canal de comunicación con WebSocket	



Fecha: ……..	Artefacto DCUF-Diagrama de casos de uso funcional
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Numero Caso de Uso:	N°4
Nombre del responsable:	Equipo de desarrollo
CU: Obtener datos de la plataforma móvil o brazo robótico	

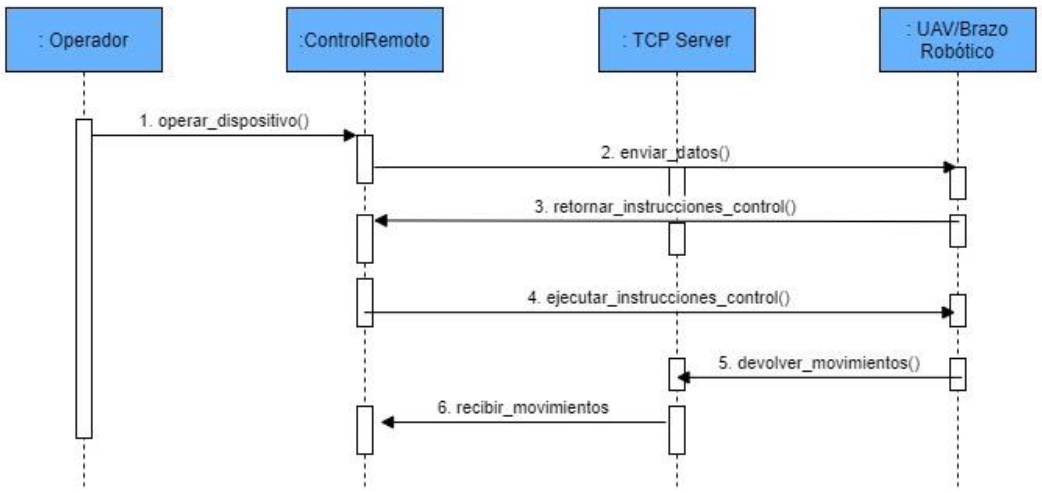


Diseño

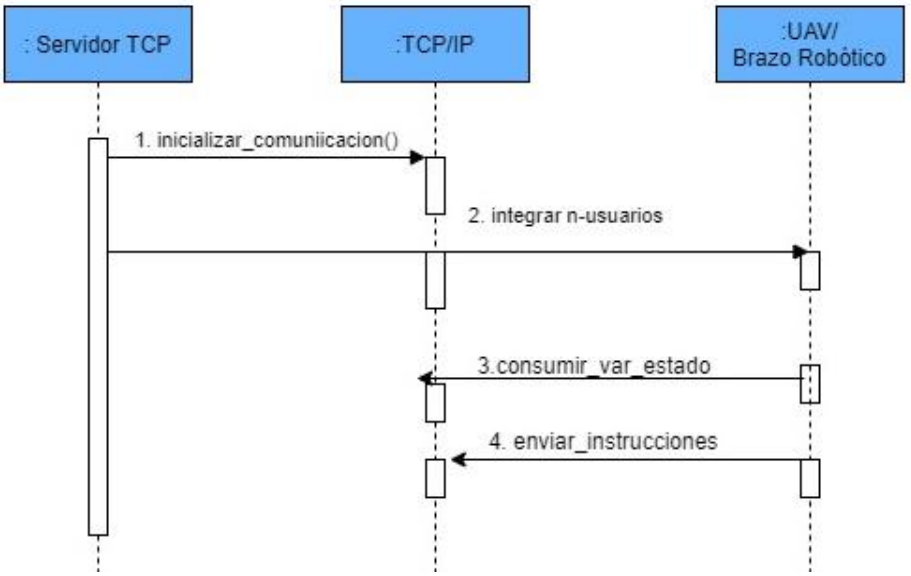
Diagrama de Secuencia Funcional de Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos.

A continuación, se establece los diagramas de secuencia (Artefacto DSEF) que muestra objetos que intervienen en el escenario con líneas discontinuas verticales, y los mensajes pasados entre los objetos como flechas horizontales.

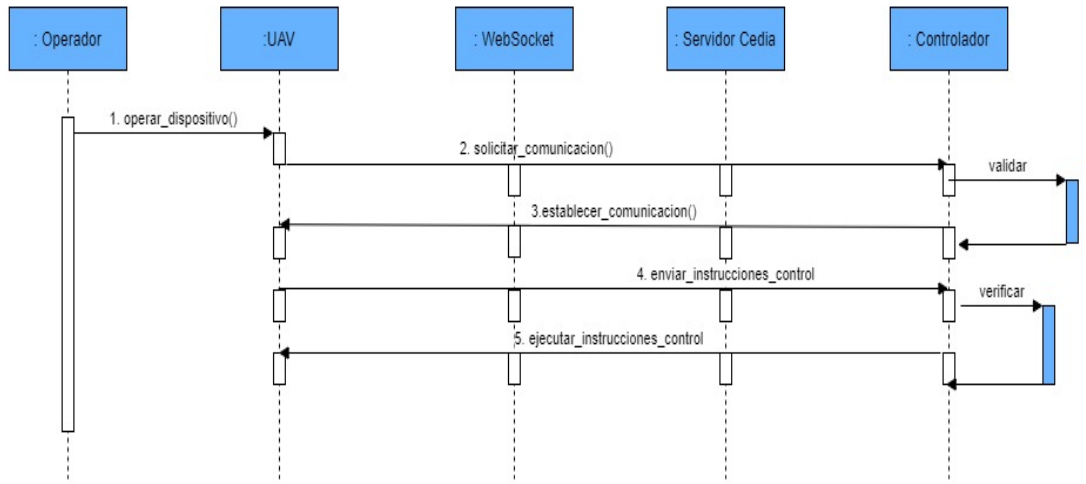
Fecha: ……..	Artefacto DSE-Diagrama de secuencia
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Nombre del responsable:	Equipo de desarrollo



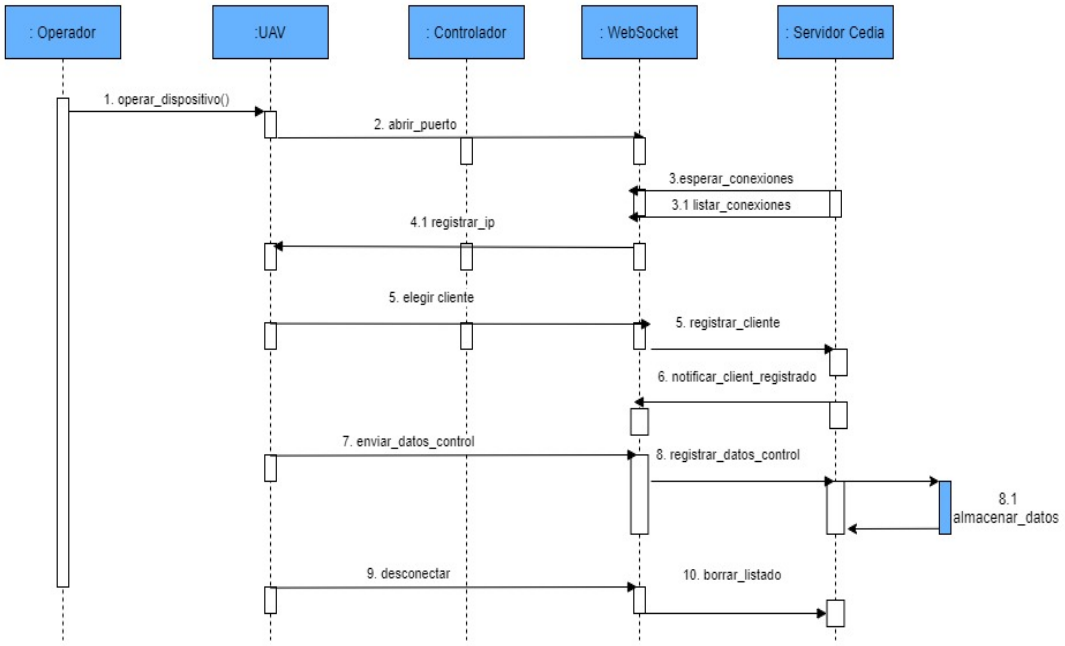
Fecha: ……..	Artefacto DSE-Diagrama de secuencia
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Nombre del responsable:	Equipo de desarrollo



Fecha: ……..	Artefacto DSE-Diagrama de secuencia
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Nombre del responsable:	Equipo de desarrollo



Fecha: ……..	Artefacto DSE-Diagrama de secuencia
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Nombre del responsable:	Equipo de desarrollo



Fecha: ……..	Artefacto DCS-Diagrama de clases
Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Módulo de proyecto:	Desarrollo de plataforma de comunicación a través de Red CEDIA
Nombre del responsable:	Equipo de desarrollo

Documentación y Lenguajes de Programación Orientada a Objetos.

Programación en Matlab es uno de los ambientes de desarrollo integrado usados en este trabajo es Matlab que es un software matemático con su propio lenguaje llamado M, teniendo similitudes al lenguaje C. Matlab es un lenguaje que trabaja con vectores y matrices, funciones, y Programación Orientada a Objetos.2 Este lenguaje es comúnmente usado para la resolución de problemas matemáticos, para el modelado, simulación y control de eventos físicos (Luzuriaga, s. f.).

Matlab al ser un software matemático trabaja con la compilación de scripts para ejecutar diferentes funciones, por lo tanto, el desarrollador pueda programar funciones para un determinado proyecto.

En esta etapa se puede aplicar las mejores prácticas de desarrollo de software identificadas en el capítulo III, como es la *Documentación de Código, y la aplicación de Lenguajes de Programación Orientada a Objetos.*

A continuación, se demuestra la aplicación de las mejores prácticas en el programa principal que llama a las diferentes funciones para el control cinemático de un brazo robot de 5 ejes desarrollado en Matlab:

Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Sección del proyecto:	Brazos robóticos de cinco grados de libertar
Clase:	Principal (Padre)
Código del proyecto:	

```
% Crea el área de trabajo en 3 dimensiones x,y,z
entorno;

% Llama a la tabla de DH del manipulador
TablaDH;

% Declaración de vectores para la posición y velocidad
traytotalp=[];
traytotalv=[];

% Crea el dibujo en 3d por líneas según la tabla de DH
t = 2;   % Tiempo para la ejecución para la resolución del movimiento

opc = 1; % Opcion para que el movimiento del brazo robot sea paso a paso o
         % es continuo, opc=0 paso a paso o opc=1 continuo.
mov = 1; % Activa o deshabilita el movimiento del brazo robot, 0 off y 1 on
numtray = 1; % Numero de trayectorias a controlar.
% Esta opción cuando es 1 habilita la generación de graficas con el
% movimiento total de cada articulación, cuando es 0 la deshabilita
graficas = 1;
pi=[300,0,150,-80,180];
qi = cineinv(pi);
J=J_det(qi);
pi=dh(r,5,qi);

manipulador = makearm(r,qi');
pf = [[0,0,900,90,0];[700,0,200,0,0];[0,-550,200,-45,0]];
i=0;
if(mov==1)
    for i=1:1:numtray

        qf = cineinv(pf(i,1:5));
        J=J_det(qf);
        if (imag(J)~=0)
            fprintf('Esta en una posición singular fuera del espacio de
trabajo \n');
            graficas=0;
            break;
        end
       if (pf(i,4)>120)||(pf(i,4)<-120)
            fprintf('Esta en una posición singular \n');
            graficas=0;
            break;
        end
        tray = controlcine(qi,qf,t);
        trayvel = controlcinevel(qi,qf,t);
        diburobot(r,manipulador,tray,opc);
        qi=qf;
        traytotalp=vertcat(traytotalp,tray);
        traytotalv=vertcat(traytotalv,trayvel);
    end
    if(graficas==1)
            figure(2);
            subplot(2,1,1);
            plot(traytotalp(:,1),'r');
            grid minor;
            title('Grafica de la articulación 1');
            legend('Posición');
            xlabel('Tiempo'); ylabel('\theta');
            subplot(2,1,2);
            plot(traytotalv(:,1),'b');
            grid minor;
            title('Grafica de la articulación 1');
            legend('velocidad');
            xlabel('Tiempo'); ylabel('\theta\prime');

            figure(3);
            subplot(2,1,1);
```

Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Sección del proyecto:	Vehículos no tripulado UAVs
Clase:	Movimiento

Código del proyecto:	%Definir los datos geométricos del céfiro y constantes necesarias.
	`function fsal=movimiento(entrada)`
	`%Datos geométricos céfiro y constantes`
	`S=1.088; %m^2`
	`c=0.39299; %m`
	`us=19; % m/s;`
	`Iy=7.447; %kg/m2`
	`g=9.81;`
	%Definir las variables de entrada de las que se alimenta la función.
	`xa=entrada(1); ya=entrada(2);`
	`Va=entrada(3);gamma=entrada(4);`
	`teta=entrada(5); q=entrada(6);`
	`CXa=entrada(7); CXe=entrada(8);CXf=entrada(9);`
	`CZa=entrada(10);CZap=entrada(11); CZq=entrada(12);`
	`CZe=entrada(13);CZf=entrada(14);`
	`Cma=entrada(15);Cmap=entrada(16); Cmq=entrada(17);`
	`Cme=entrada(18);Cmf=entrada(19);`
	`FT=entrada(20);`
	`ma=entrada(21);`
	`alpha_p=entrada(22);`
	%calcula la densidad del aire para cada valor de la altitud (ya)
	`Data_ATM = get_Atmospheric_Cefiro(ya);`
	`rho=Data_ATM.rho;`
	%función get_Atmospheric_Cefiro(ya) se define
	`function Data_ATM = get_Atmospheric_Cefiro(h)`
	`g_ATM=-9.81;%gravedad de la tierra`
	`R_ATM=287;%constante de los gases`
	`T0_ATM=288;%temperatura en Kelvin`
	`if h<11000%altura de la tropopausa`
	`alpha_ATM=-6.5e-3; %baja la temperatura 6,5° cada km`
	`T0_H=T0_ATM + alpha_ATM*h;`
	`a=sqrt(1.4*R_ATM*T0_H);`
	`rho=1.225*((T0_ATM+...`
	`alpha_ATM*h)/T0_ATM)^(g_ATM/(R_ATM*alpha_ATM)-1);`
	`else`
	`deltaestrato=exp(g_ATM*(h_ATM-11000)/(R_ATM*216.5));`
	`rho=0.3629*deltaestrato;`
	`end`
	`Data_ATM.rho = rho;`
	`Data_ATM.a_speed = a;`
	`rho=1.225*((T0_ATM+...`
	`alpha_ATM*h)/T0_ATM)^(g_ATM/(R_ATM*alpha_ATM)-1);`
	`else`
	`deltaestrato=exp(g_ATM*(h_ATM-11000)/(R_ATM*216.5));`
	`rho=0.3629*deltaestrato;`
	`end`
	`Data_ATM.rho = rho;`
	`Data_ATM.a_speed = a;`

Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Sección del proyecto:	Nodo Websocket
Clase:	Comunicación de Plataformas UAV y Brazo Robótico

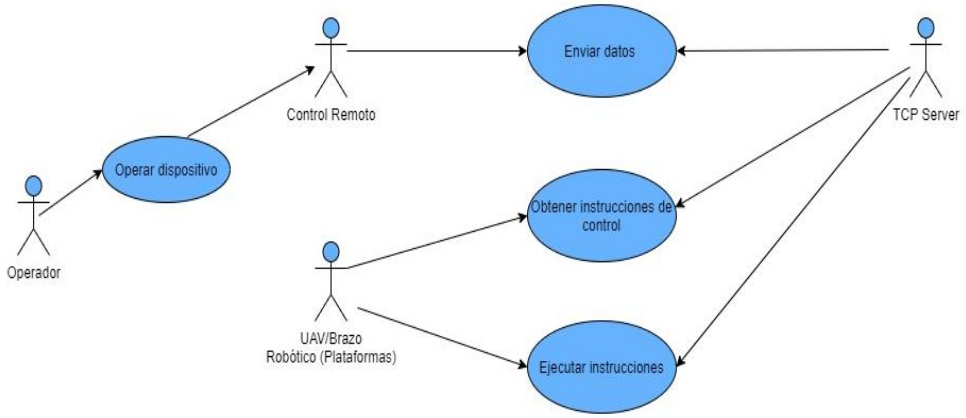
Código del proyecto:	//Abre un puerto en el servidor que se queda esperando las conexiones desde los clientes.
	Inicialización (IP, puerto)
	
	//Registra la IP del cliente y lo asigna al listado de clientes conectados al websocket.
	OnConnect()
	
	//Asignar un identificador al cliente e inicia el Thread.
	Autenticar(controladorPlataforma)
	
	//Clientes conectados y autenticados al websocket.
	OnMessage(Datos)
	
	//detiene el Thread del cliente que se desconectó.
	OnCLose()

Nombre del proyecto:	Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos
Sección del proyecto:	Nodo Websocket-Comunicación
Clase:	Comunicación de Plataformas UAV y Brazo Robótico

Código del proyecto:	//plataforma UAV y Brazo Robótico
	Connect()
	
	//Identificación
	Identificar(“ev”: “id”,
	“tipo”: “platform”
	“cliente”:”drone”)
	
	//Enviar estado de las variables
	Sending_state_var(“ev”: “start”,
	“lat”: “-78.4565”
	“long”:”-1.2455”
	“alt”:” 2500.12”
	“x”:”0.21215”)
	
	//Recibir instrucciones de control
	Receiving_control_inst(“roll”: “-4.4565”,
	“yaw”: “1.2455”
	“pitch”:”2.4312”
	“thurttle”:”0.21215”)

Pruebas

Para la etapa de pruebas se considera a la mejor practica de las *Pruebas Funcionales,* para ello se elaboró un artefacto APFSE (*Artefacto de Pruebas Funcionales de Sistemas Embebidos*).

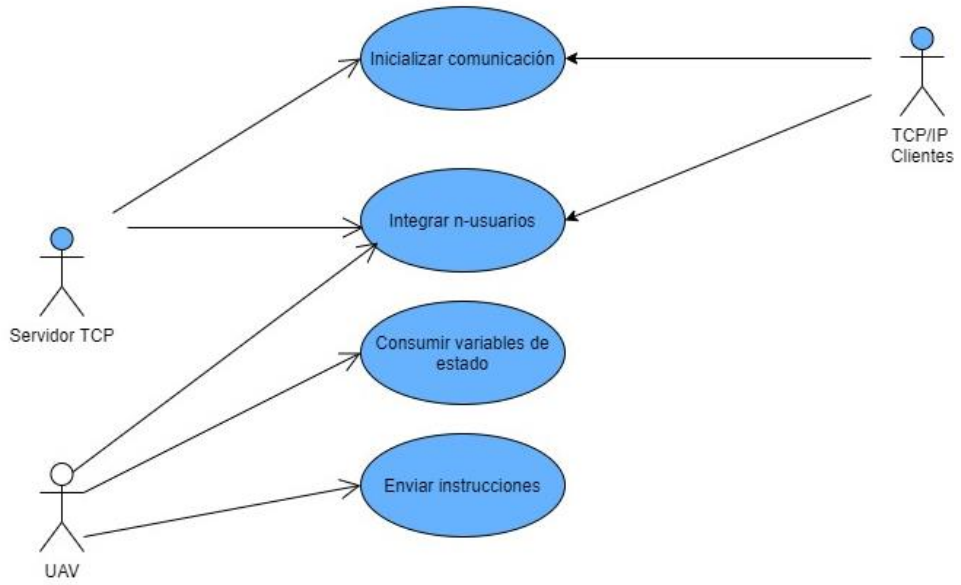
<table>
<tr><td colspan="3">Compartir información y control de vehículos no tripulados</td><td><CP01></td></tr>
<tr><td colspan="4">Objetivo de la prueba:
Asegurar la manera de compartir información entre vehículos no tripulados y brazo robótico.</td></tr>
<tr><td colspan="4">Caso de Pruebas:
Ejecutar cada caso de uso con datos validos e inválidos para verificar lo siguiente:
Compartir información entre UAV (vehículo no tripulado) que será operado por un humano (operador) mediante un control remoto.
El control remoto enviara datos al TCP Server.
El UAV / Brazo robótico obtendrá instrucciones de control desde el TCP Server y de manera bilateral.
El UAV/ Brazo robótico ejecutara instrucciones de control desde el TCP Server y de manera bilateral.</td></tr>
<tr><td colspan="4">Caso de uso involucrado:
 
</td></tr>
<tr><td colspan="4">Resultado CheckList: Si o No</td></tr>
<tr><td>N° Caso de Prueba</td><td>Si</td><td>No</td><td>Observación</td></tr>
<tr><td>1</td><td>✓</td><td></td><td>Ninguna</td></tr>
<tr><td>2</td><td>✓</td><td></td><td>Ninguna</td></tr>
<tr><td>3</td><td>✓</td><td></td><td>Ninguna</td></tr>
<tr><td>4</td><td>✓</td><td></td><td>Ninguna</td></tr>
<tr><td>Resultado:</td><td colspan="3">Todos los ítems de casos de pruebas cumplen con los procesos establecidos.</td></tr>
</table>

Transmitir datos	<CP02>

Objetivo de la prueba:
Transmitir datos mediante el servidor TCP Server y TCP Cliente.

Caso de Pruebas:
Ejecutar cada caso de uso con datos validos e inválidos para verificar lo siguiente:
Para transmitir datos entre vehículos no tripulados se realizará mediante un Servidor TCP y TCP/IP Clientes permite inicializar comunicación.
El servidor TCP Server y TCP/IP Clientes permite integrar n número de usuarios y el UAV/Brazo Robótico.
La plataforma UAV y Brazo Robótico consume variables de estado.
La plataforma UAV y Brazo Robótico envía instrucciones.

Caso de uso involucrado:



Resultado CheckList: Si o No

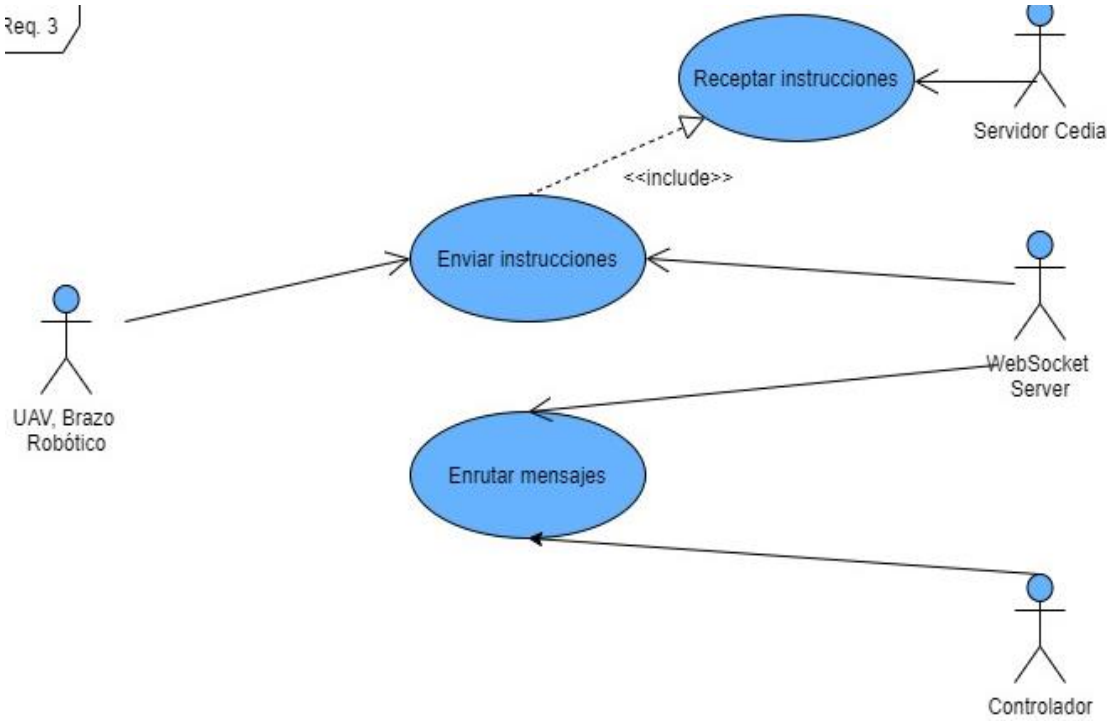
N° Caso de Prueba	Si	No	Observación
1	✓		Ninguna
2	✓		Ninguna
3	✓		Ninguna
4	✓		Ninguna
Resultado:	Todos los ítems de casos de pruebas cumplen con los procesos establecidos.		

Establecer canal de comunicación con Websocket	<CP03>

Objetivo de la prueba:
Establecer canal de comunicación con Websocket para enviar instrucciones de control hacia Servidor Cedia y el vehículo no tripulado.

Caso de Pruebas:
Ejecutar cada caso de uso con datos validos e inválidos para verificar lo siguiente:
El UAV y Brazo Robótico enviara instrucciones de control hacia el servidor CEDIA, estas instrucciones de control son receptadas desde un control remoto manipulado por un operador de forma bilateral.
El UAV y Brazo Robótico enviara instrucciones de control hacia el nodo Websocket Server de forma bilateral.
El nodo Websocket es el responsable de enrutar mensajes hacia el controlador (software).

Caso de uso involucrado:



Resultado CheckList: Si o No

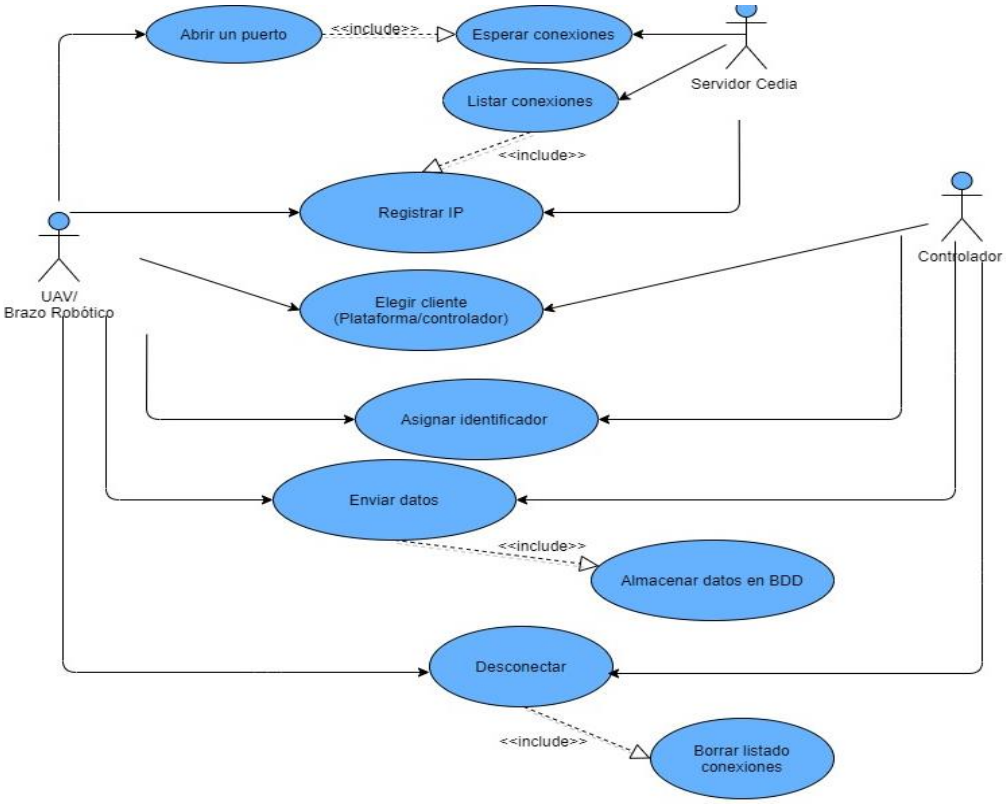
N° Caso de Prueba	Si	No	Observación
1	✓		Ninguna
2	✓		Ninguna
3	✓		Ninguna
Resultado:	Todos los ítems de casos de pruebas cumplen con los procesos establecidos.		

Obtener datos de la plataforma móvil o brazo robótico	<CP03>

Objetivo de la prueba:
Obtener datos de la plataforma móvil o brazo robótico mediante WebSocket que es un canal de comunicación entre plataformas, servidor externo (CEDIA) y controladores.

Caso de Pruebas:
Ejecutar cada caso de uso con datos validos e inválidos para verificar lo siguiente: Abre un puerto en el servidor que se queda esperando las conexiones desde clientes. Registra la IP del cliente y lo asigna al listado de clientes conectados al WebSocket. Definir el tipo de cliente (plataforma, controlador). Asignar al cliente un identificador e inicia un hilo (tarea) de acuerdo al tipo de cliente. Enviar datos (variables de estado, instrucciones de control) servidor externo(CEDIA), vehículos no tripulados y controladores, estos datos serán almacenados. Cuando un cliente termina su proceso (hilo), se detiene y se borra del listado de clientes conectados al websocket.

Caso de uso involucrado:



Resultado CheckList: Si o No

N° Caso de Prueba	Si	No	Observación
1	✓		Ninguna
2	✓		Ninguna
3	✓		Ninguna
4	✓		Ninguna
5	✓		Ninguna
6	✓		Ninguna
Resultado:	Todos los ítems de casos de pruebas cumplen con los procesos establecidos.		

Validación del Modelo de Trabajo de Desarrollo del Software de SE

Introducción

En el presente capítulo se desarrolla la verificación y validación de resultados obtenidos al aplicar Modelo de Trabajo basado en las mejores prácticas de desarrollo de las metodologías tradicionales y no tradicionales en el proceso de desarrollo de software en sistemas embebidos, implementado en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica.

El Modelo de Trabajo, se expuso a los usuarios finales con el objetivo de que validen el proyecto de investigación y puedan exponer sus conclusiones. Se realizó una presentación técnica al jefe del laboratorio de investigación y al equipo de desarrollo de la propuesta y en el transcurso del desarrollo de proyecto se contó con la participación de algunos miembros del equipo de desarrollo, adicional se distribuyó manual de usuarios con la explicación del modelo, etapas y artefactos a generar en la construcción de SE.

Para la validación del modelo de trabajo se utilizó la hipótesis planteada en el Capítulo I, a continuación, se detalla.

La hipótesis que a continuación se describe corresponde a una hipótesis de trabajo, por lo tanto, se procede a validar mediante dos criterios la ***viabilidad,*** y ***aplicabilidad*****.**

Planteamiento de la hipótesis

Hipótesis de trabajo

¿**Si se** diseña un modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales **entonces** ayudará al proceso de desarrollo de software en sistemas embebidos en el Laboratorio de Investigación en

Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas Sede Latacunga?

a. Variable independiente

Diseñar un modelo de trabajo basado en las mejores prácticas de las metodologías tradicionales y no tradicionales.

b. Variable dependiente

Ayudará al proceso de desarrollo de software en sistemas embebidos en el Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas Sede Latacunga.

Verificación de la Hipótesis

Para la validación de los resultados del proyecto de investigación se aplicará la hipótesis de trabajo.

Hipótesis N°1

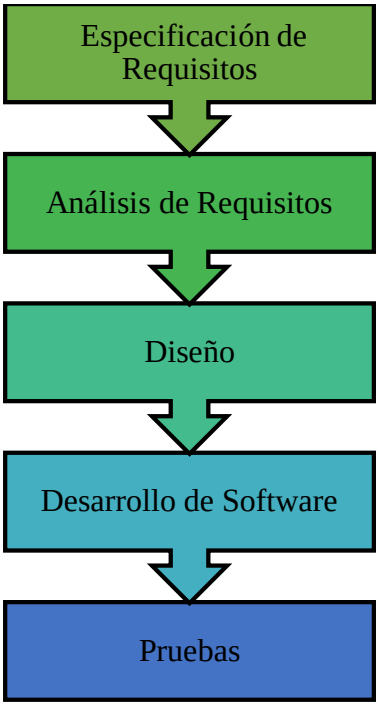
H1 Viabilidad, a partir de un conjunto de publicaciones obtenidas de bases de datos científicas se pudo identificar y seleccionar las mejores prácticas de desarrollo de las metodologías tradicionales y no tradicionales aplicadas a la construcción de sistemas embebidos, por lo tanto, fue posible diseñar un modelo de trabajo en el que se incluyen los artefactos y plantillas que ayudan al proceso de desarrollo de software embebido y servirá de instrumento guía para el equipo de desarrollo del Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica.

Resultado:

Modelo de Trabajo: BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems)***.**

Figura 50

Modelo de Trabajo para Sistemas Embebidos



Nota: Esta figura, muestra el modelo de trabajo definido para la construcción de SE.

Hipótesis N°2

H2 Aplicabilidad, con el Modelo de trabajo diseñado denominado "BPSSEM (*Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems)*", fue posible aplicar el modelo en el caso de estudio de Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos sobre el módulo de *"Desarrollo de plataforma de comunicación a través de Red Cedia".*

Resultado:

Figura 51

Modelo de trabajo BPSSEM y artefactos

Mejores Prácticas	Fases				
	Especificación Requerimientos	Análisis de Requisitos	Diseño	Desarrollo de Software	Pruebas
Requisitos funcionales	✓				
Requisitos no funcionales	✓				
Diagrama de Caso de Uso		✓			
Historia de Usuario		✓			
Diagrama de secuencia			✓		
Diagrama de clases			✓		
Modelo conceptual			✓		
Documentación de código				✓	
P. Orientada a objetos				✓	
Pruebas funcionales					✓
Pruebas de aceptación					✓
Pruebas unitarias					✓

Nota: Esta figura, muestra el modelo de trabajo defino con las fases artefactos y entregables para la construcción de SE.

Conclusiones, Recomendaciones y Trabajos Futuros

Conclusiones

Se construyó un marco teórico orientado a las mejores prácticas que ayuden al proceso de desarrollo de software de sistemas embebidos en el laboratorio de investigación del Departamento de Eléctrica y Electrónica de la Universidad de las Fuerzas Armadas ESPE sede Latacunga, dando como resultado un instrumento con las mejores prácticas de desarrollo de las metodologías tradicionales y no tradicionales para el diseño del modelo de trabajo.

Con la revisión literaria se desarrolló el modelo de trabajo *Mejores Prácticas de Software para Sistemas Embebidos o Best Practice of Software for Embedded Systems,* es un modelo de trabajo diseñado para ayudar al proceso de desarrollo de software de SE. Las características principales es el ámbito de su aplicación en la construcción de SE, utiliza artefactos en cada etapa, el objetivo es ayudar al proceso de desarrollo de software para SE. Para el modelado se aplica la herramienta UML lo cual propone la reutilización de componentes y código. Se relaciona con modelos de sistemas embebidos o empotrados lo cual puede dificultar su aplicación.

Se implementó el modelo de trabajo BPSSEM mediante el proceso de reingeniería en el proyecto de investigación Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos la actividad de Desarrollo de plataforma de comunicación a través de la Red CEDIA. Se procedió a ejecutar las cinco fases establecidas en el modelo con sus respectivos artefactos, evidenciando la aplicación de las mejores prácticas durante el ciclo de desarrollo de un proyecto de SE. Con la implementación del modelo de trabajo BPSSEM se puede evidenciar que aplicable hacia proyectos futuros para la construcción de Sistemas Embebidos, adicional la documentación, artefactos y plantillas generados en el modelo estarán disponibles en el drive para su posterior uso.

La validación de los resultados obtenidos del desarrollo del modelo de trabajo, se lo realizo mediante la implementación de la propuesta, el modelo será una guía para los desarrolladores durante todo el ciclo de vida de los SE a través de la definición de las fases de los procesos y actividades. El modelo de trabajo BPSSEM indicara que usar, como usarlo y cuando usarlo. Por lo tanto, el desarrollo de SE será un proceso planificado, gestionado y controlado. El modelo está estructurado por 5 fases y 11 mejores prácticas.

Para comprobar la hipótesis se tomó en consideración los criterios de **viabilidad y aplicabilidad**. La viabilidad demostró que es posible desarrollar el modelo de trabajo a partir de identificar las mejores prácticas obtenidas de una revisión bibliográfica. Mientras que la aplicabilidad permitió evidenciar que, utilizando el modelo, se diseñó el Sistema Colaborativo de Robots Aéreos para Manipular Cargas con Óptimo Consumo de Recursos sobre el módulo de *"Desarrollo de plataforma de comunicación a través de Red Cedia".* **No obstante, se requerirá de un mayor número de aplicaciones desarrollados con el modelo para consolidar el estudio realizado. En consecuencia, se espera** que el grupo de Laboratorio de Investigación en Automatización, Robótica y Sistemas Inteligentes del Departamento de Eléctrica y Electrónica, utilice el modelo en el mayor número de aplicaciones embebidas.

Recomendaciones

Para crear un modelo de trabajo que ayude al proceso de desarrollo de software de sistemas embebidos, el primer paso es realizar una revisión de la literatura científica considerando bases bibliográficas y motores de búsqueda de tipo académico y científico que se encuentre actualizado en el área. Estos elementos ayudaran a obtener bibliografía de calidad lo cual permitirá definir el modelo de trabajo.

Para el diseño del modelo de trabajo se consideró las mejores prácticas de desarrollo identificadas en las metodologías tradicionales y no tradicionales. Por lo tanto, se recomienda crear un instrumento de análisis y búsqueda de artículos científicos que sustenten la necesidad del diseño un modelo que será una guía en el proceso de desarrollo de sistemas embebidos.

El modelo consta de cinco fases, pero en un futuro es necesario incrementar nuevas etapas como la gestión de la configuración, esta debería estar presente desde la especificación de requerimientos hasta la etapa de pruebas. Aplicar la gestión de la configuración permitiría tener control de versiones de los artefactos de las mejores prácticas y también los diseños por parte del hardware.

Para comprobar la aplicabilidad de un modelo de trabajo para Sistemas Embebidos es necesario aplicar un cierto de tipo de hipótesis, de tal forma que, de credibilidad a las investigaciones, por lo tanto, se recomienda utilizar la hipótesis de trabajo que es una hipótesis acepta provisionalmente como base para futuras investigaciones cuya finalidad es que se produzca una teoría clara sobre la investigación propuesta.

Trabajos Futuros

Para un trabajo futuro se plantea incluir un conjunto de mejores prácticas estén enfocadas al uso de las técnicas del *Lenguaje Unificado de Modelado UML* por cada etapa del modelo de trabajo *BPSSEM* que permitirán el desarrollo y modelado de un software o sistema embebido. *UML* ofrece modelado utilizando diagramas y es la forma más común para expresar un sistema dentro de un equipo de desarrollo. Además, permite modelar cualquier tipo de aplicación en combinación de hardware y software.

Dentro del modelo de trabajo BPSSEM se deberá incluir la *Gestión de la Configuración,* esto permitirá asegurar la calidad de un software o sistema durante

cualquier etapa de construcción de un sistema embebido (SE) mediante el control de cambios y el control de versiones de todos los artefactos de software y hardware, esto facilitará al mantenimiento de un sistema. Para aplicar la *Gestión de la configuración* se debe realizar durante todas las fases del desarrollo o construcción de software o sistema embebido.

Se recomienda considerar para futuras investigaciones refinar el conjunto de buenas prácticas de desarrollo propuesto por el modelo BPSSEM, además de definir y detallar los procesos y plantillas estableciendo nuevas fases como Validación, Entrega y Mantenimiento, esto con la finalidad mejorar el proceso de desarrollo de software.

Referencias Bibliográficas

Areba, J. B. de. (2001). *Metodología del análisis estructurado de sistemas*. Univ Pontifica Comillas.

Arriarán, S. S. (2017). *Todo sobre sistemas embebidos: Arquitectura, programación y diseño de aplicaciones prácticas con el PIC18F*. Universidad Peruana de Ciencias Aplicadas (UPC).

Benchimol, D. (2011). *Proyectos con microcontroladores aprenda a desarrollar sus propias aplicaciones*. USERSHOP.

Cadavid, A. N. (2013). Revisión de metodologías ágiles para el desarrollo de software. *Prospectiva*, *11*(2), 30. https://doi.org/10.15665/rp.v11i2.36

Caiza, L., Chicaiza, D., P. Reyes Ch, R., & Montaluisa, F. (2020). MEAC: An experience to keep the production line active in the software development process. *KnE Engineering*, 22-39. https://doi.org/10.18502/keg.v5i1.5916

Castrillón, E. P. (2011). *Methodology Proposal of Software Development for Virtual Learning Objects—MESOVA –*. *34*, 25.

Cetinkunt, S. (2007). *Mecatronica/ Mecatronics*. Ediciones Larousse Sa De Cv.

Deschamps, J.-P., Imaña, J. L., & Sutter, G. D. (2009). *Hardware implementation of finite-field arithmetic*. McGraw-Hill.

Ebert, C. (2009). *SOFTWARE INTEGRADO: HECHOS, CIFRAS Y*. 11.

Fernández, J. A. M. (2019). *Sistemas programables avanzados*. Ediciones Paraninfo, S.A.

Gabriel, E. (s. f.). *Metodologías de desarrollo de software*. 117.

Gallo, S., Alberto, J., Heredia, C., & Francisco, L. (s. f.). *Desarrollo de un Prototipo Funcional de Bajo Costo para la Medición de la Severidad Vibratoria en Máquinas Rotativas Según las Normas ISO 2372 Y 10816-3*. *2012*, 264.

Gamboa, J. P. Z., & Arreaga, C. A. L. (2018). *Evolución de las Metodologías y Modelos utilizados en el Desarrollo de Software. Evolution of the Methodologies and Models used in Software Development. 3*(10), 14.

Gómez, J. C. (2012). Taxonomía de los modelos y metodologías de desarrollo de software más utilizados. . . *ISSN*, *52*, 11.

Heredero, C. de P., Agius, J. J. L. H., Romero, S. M.-R., & Salgado, S. M. (2019). *Organización y transformación de los sistemas de información en la empresa*. ESIC.

Ignacio, C., & Paola, V. (2015). *Metodologías actuales de desarrollo de software*. *2015*, 7.

Introducción Al Análisis de Sistemas Y la Ingeniería de Software. (s. f.). EUNED.

Lagos, P. S. (s. f.). *CommonKADS y el Lenguaje de Modelado Unificado – UML*. 16.

Leopoldo Pauta Ayabaca, L., & Moscoso Bernal, S. (2018). Verificación y validación de software. *Killkana Técnica*, *1*(3), 25. https://doi.org/10.26871/killkana_tecnica.v1i3.112

Lezcano, L. A., Guzmán, J. A., & Vélez, C. A. (2015). Un Modelo para la Identificación de Elementos KAOS (Especificación Automática de Adquisición de Conocimientos) a partir de la Especificación de Requisitos en Lenguaje Natural. *Información tecnológica*, *26*(6), 129-138. https://doi.org/10.4067/S0718-07642015000600015

Luzuriaga, J. G. V. (s. f.). *Control de trayectoria de la simulación de un brazo robot de 5 grados de libertad, controlado mediante la plataforma C2000 Piccolo LAUNCHXL-F28027F*. 95.

Mejía-Neira, Á., Jabba, D., Caballero, G. C., & Caicedo-Ortiz, J. (2019). Influencia de la Ingeniería de Software en los Procesos de Automatización Industrial. *Información tecnológica*, *30*(5), 221-230. https://doi.org/10.4067/S0718-07642019000500221

Montero, B. M., Cevallos, H. V., & Cuesta, J. D. (s. f.). *Agile methodologies against traditional methods in the software development process.* 10.

Mumphrey, W. S. (s. f.). *Capítulo 7. Ingeniería del Software*. 145.

Oriou, A., Bronca, E., Bouzid, B., Guetta, O., & Guillard, K. (2014). Manage the Automotive Embedded Software Development Cost & Productivity with the Automation of a Functional Size Measurement Method (COSMIC). *2014 Joint Conference of the International Workshop on Software Measurement and the International Conference on Software Process and Product Measurement*, 1-4. https://doi.org/10.1109/IWSM.Mensura.2014.45

Pedre, S. (2017). Sistemas embebidos. *Laboratorio de Robótica y Sistemas Embebidos, Departamento de computación FCEN UBA*.

Perez Abreu, D. (2009). *Sistemas Embebidos y Sistemas Operativos Embebidos* (pp. 1-15).

Piattini, M. (s. f.). *REVISTA INSTITUCIONAL DE LA FACULTAD DE INFORMÁTICA | UNLP*. 3.

Rai, K., Madan, L., & Anand, K. (2014). *Student (B.tech VItIh sem) Department of Computer science Dronacharya College Of Engineering,Gurgaon-122506*. *1*(11), 8.

Sandoval, M. G. G., Torrado, H. D. A., Pinzón, M. L., & Fuentes, A. S. F. (s. f.). *BUENAS PRÁCTICAS APLICADAS A LA IMPLEMENTACION COLABORATIVO DE APLICATIVOS WEB*. 4.

Silvério Miyashiro, M. A., & Ferreira, M. G. V. (2014). One approach to the use of the practices of CMMIDEV V1.3 level 2 in a process of development of embedded systems. *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications*, 268-272. https://doi.org/10.1109/IISA.2014.6878772

Srinivasan, J., Dobrin, R., & Lundqvist, K. (2009). «State of the Art» in Using Agile Methods for Embedded Systems Development. *2009 33rd Annual IEEE International Computer Software and Applications Conference*, 522-527. https://doi.org/10.1109/COMPSAC.2009.186

Vega, J. I. H. (2010). El software embebido y los retos que implica su desarrollo. *ConCiencia Tecnológica*, *40 (Julio-Diciembre)*, 42.

Withers, D. H. (2000). Software engineering best practices applied to the modeling process. *2000 Winter Simulation Conference Proceedings (Cat. No.00CH37165)*, *1*, 432-439. https://doi.org/10.1109/WSC.2000.899749

Anexos