ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DEL SOFTWARE

TERCERA PROMOCIÓN

TESIS DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN INGENIERÍA DE SOFTWARE

TEMA: DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO

AUTOR: EDWIN EDISON QUINATOA AREQUIPA

DIRECTOR: ING. MARCELO REA GUAMÁN MSC.

LATACUNGA

2016

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DEL SOFTWARE

CERTIFICACIÓN

Certifico que el trabajo de titulación, "***DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO***", realizado por el señor ***EDWIN QUINATOA AREQUIPA***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***EDWIN QUINATOA AREQUIPA*** para que lo sustente públicamente.

Latacunga, 07 de Enero del 2016

Ing. Marcelo Rea Guamán MSc.

DIRECTOR DE PROYECTO

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DEL SOFTWARE

AUTORIZACIÓN

Yo, ***EDWIN QUINATOA AREQUIPA***, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación "**DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO**", cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 07 de Enero del 2016

Ing. Edwin EdisónQuinatoa Arequipa

C. I. 0502563372

ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

VICERRECTORADO DE INVESTIGACIÓN, INNOVACIÓN Y TRANSFERENCIA DE TECNOLOGÍA

MAESTRÍA EN INGENIERÍA DEL SOFTWARE

AUTORÍA DE RESPONSABILIDAD

Yo, ***EDWIN QUINATOA AREQUIPA***, con cédula de identidad N° **0502563372**, declaro que este trabajo de titulación "***DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO"*** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 07 de Enero del 2016

Ing. Edwin Edisón Quinatoa Arequipa

C. I. 0502563372

DEDICATORIA

A Dios quien supo guiarme por el buen camino y mostrarme día a día que con humildad, paciencia y sabiduría todo es posible, y darme fuerzas para seguir adelante y no desmayar en los problemas que se presentaban, enseñándome a encarar las adversidades sin perder nunca la dignidad ni desfallecer en el intento y sobre todo llenando de bendiciones todos los días.

A mis padres Abelardo y Etelvina por su apoyo, consejos, comprensión y ayuda en los momentos difíciles, y educándome con disciplina, buenos principios y valores para enfrentar en estos días el duro camino de la vida, a mis hermanas Nelly y Jenny por brindarme su cariño y apoyo incondicional, y a la vez estuvieron siempre a lo largo de mi vida estudiantil; a las personas que siempre tuvieron una palabra de aliento en los momentos difíciles y que han sido incentivos y referentes en esta vida.

Edwin

AGRADECIMIENTO

Agradezco en primer lugar a Dios, quien me ha llenado de bendiciones en todo este tiempo, al que con su infinito amor me ha dado la sabiduría suficiente para culminar esta meta.

A mis padres Abelardo y Etelvina, expreso mi más sincero agradecimiento, reconocimiento y cariño por todo su esfuerzo, sacrificio y paciencia demostrada en todos estos años y por sus ejemplos de perseverancia y constancia que me ha infundado siempre, gracias a ellos he llegado a culminar con éxitos una meta más en mi vida.

A mis hermanas, amigos y amigas, por darme ánimos de continuar en los momentos que más lo necesite y a toda mi familia, personas que de una u otra forma me han ayudado directa o indirectamente y por otra parte me han fundamentado a crecer como persona y como profesional he hecho posible la realización de este trabajo.

Agradezco también de manera especial por sus conocimientos, experiencia y colaboración al Ing. Marcelo Rea, quien fue mi Director de Tesis que con su apoyo supo guiarme en el desarrollo de la presente tesis desde su etapa de inicio hasta su culminación. "Ahora se puede expresar que todo lo obtenido es gracias a todos ustedes".

Edwin

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE ANEXOS

RESUMEN

La Dirección Nacional de Tecnología de la Información del IESS de la ciudad de Quito tiene entre sus responsabilidades el de analizar, desarrollar y mantener sus aplicaciones informáticas para lo cual utiliza RUP, pero por ser una Metodología de desarrollo de Software tradicional muy extensa no está acorde a las necesidades que posee la Institución; por tal razón se realizó una revisión de los artículos que abarcan esta área y complementado con un estudio se pudo especificar donde posee las falencias y problemas, en base a ello se desarrolló una Metodología de trabajo basada en la anteriormente mencionada, la cual permitió determinar las fases y disciplinas de la parte de Proceso con sus respectivas iteraciones, roles y artefactos necesarios, ya que está enfocado solo a desarrollo, este trabajo no abarco la parte de Gestión. Esto significa que para validar a la nueva metodología se aplicó a un caso real como fue dentro de la Dirección de Riesgos de Trabajo cuyo problema era la toma de evaluaciones a los afiliados de forma manual, para dar solución fue necesario desarrollar 5 iteraciones con sus respectivos Artefactos como: caso de uso, arquitectura de software, plan de pruebas y manuales; y Roles como: analista de procesos, sistemas, el programador y tester; complementadas con sus respectivas plantillas. En conclusión se especifica que se estableció los requerimientos necesarios permitiendo la óptima disminución de tiempo en desarrollo del prototipo y la aplicación final; ya que la finalidad es otorgar un producto de calidad en el Sector público.

PALABRA CLAVE:

- **METODOLOGÍA DE SOFTWARE**
- **PROCESO UNIFICADO DE RATIONAL**
- **INGENIERÍA DE SOFTWARE**
- **IESS - DIRECCIÓN NACIONAL DE TECNOLOGÍA DE LA INFORMACIÓN**

ABSTRACT

The National Information of Technology at IESS in the city of Quito has among its responsibilities to analyze, develop and maintain their computer applications, for which they use RUP, but because it is a very extensive and traditional software development methodology is not commensurate with the needs that the institution owns; for this reason, a review of articles covering this area and complemented by a study was conducted, and it was possible to say where the shortcomings and problems are, on this basis, a methodology based on the above study was developed, which allowed to determine the phases and disciplines of their respective iteration process, roles and models required, since it is focused only on the development. Hence, this work did not include the part of Management. To validate the new methodology, a real case was applied within the Risk Management Working whose problem was making manual evaluations to the affiliates; to solve this, it was necessary to develop five iterations with their respective models such as: use of cases, software architecture, test plans and handbooks; and roles as analyst processes, systems, programmer and tester; complemented by their respective patterns. In conclusion, we can say that the necessary requirements were established allowing optimal reduction of time in the development of the prototype and its final application since the purpose is to provide a quality product in the Public sector.

KEYWORD:

- **SOFTWARE METHODOLOGY**
- **RATIONAL UNIFIED PROCESS**
- **SOFTWARE ENGINEERING**
- **IESS - NATIONAL INFORMATION OF TECHNOLOGY**

CAPÍTULO I

1. ASPECTOS GENERALES

DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO

Nota: Al inicio del desarrollo del proyecto de tesis el Departamento de desarrollo del IESS se denomina Dirección de Desarrollo Institucional ("DDI"), pero actualmente se denomina Dirección Nacional de Tecnología de la Información ("DNTI"), por lo cual se especificara con este término actual en todo el transcurso del documento.

1.1. Antecedentes

Para la presente investigación se ha tomado como referencia varios estudios de tesis de la Universidad de la Escuela Politécnica del Litoral ("ESPOL") (Tituana Vera, 2009), artículos de la Asociación Ecuatoriana de Software ("AESOFT") (Boni, Caceres, Dueñas, Proaño, & Trelles, 2006), Proyectos del Consejo de Universidades Flamengas de Bélgica ("VLIR") de la ESPOL (Coba Martínez, Macias, Reinoso Espinosa, & Vivanco Andrade), y la publicación de la empresa ProChile de Ecuador (ProChile en Ecuador, 2012) , se determinó que no existe desarrollado una Metodología de Trabajo basado en el Proceso Unificado de Desarrollo de Software ("RUP") para la DNTI del Instituto Ecuatoriano de Seguridad Social ("IESS") de la ciudad de Quito.

En algunos estudios realizados sobre el IESS manifiestan que la Institución tiene como función primordial la de brindar servicios a afiliados y empleadores y que posee diversas Direcciones como son: La Dirección General, Dirección Seguro General de Salud Individual y Familiar, Dirección Sistema de Pensiones, Dirección Seguro General de Riesgo del Trabajo,

Dirección Seguro Social Campesino, Dirección Nacional de Bienes Inmuebles, Dirección de Servicios Corporativos, Dirección Nacional de Tecnología de la Información, Dirección Económico Financiera y Dirección Provincial, pero las funciones que realiza el área de la DNTI es la de modernizar al IESS en todo lo relacionado a Innovación Tecnológica el cual está relacionado con mi campo de estudio. (Castañeda Cadena & Quezada Sarasti, 2007)

Tiene la responsabilidad de la administración de la Información que mantiene el IESS por lo cual está encargado de crear, desarrollar y mantener los aplicativos informáticos necesarios que faciliten el tema comunicacional y transaccional y para ello utilizan la metodología RUP quien está actualmente permitiendo hacer el levantamiento de la Información, diseño del Sistema y la Gestión de Proyectos, pero poseen algunos problemas como el de no estar acorde a las necesidades que posee el IESS y a las particularidades de la Organización, por lo que no está ofreciendo soluciones tan efectivas y está ocasionando una excesiva utilización de tiempo para la solución de los temas de software en la Organización, tomando en cuenta que la DNTI o DDI dentro de la cadena de valor Institucional es una unidad de apoyo. (IESS - Dirección de Desarrollo Institucional, 2013)

Por lo cual analizando varios aspectos y referencias de estos estudios se hace pertinente y necesario el desarrollo de una Metodología de Trabajo basado en RUP, para creación de aplicaciones de software en la DNTI del IESS de la ciudad de Quito ya que esto ayudara a fomentar y resolver problemas detectados en investigaciones anteriores, cuyo proceso definido está orientado a la Ingeniería de Software, y sería un referente de desarrollo de aplicaciones para el resto de Instituciones Estales del gobierno Ecuatoriano.

1.2. Planteamiento del Problema

El Instituto Ecuatoriano de Seguridad Social es una entidad Pública creada con la finalidad de proporcionar servicios a sus afiliados y empleadores, administrar y brindar el servicio de seguridad social. Su organización y funcionamiento se fundamenta en diversos principios de solidaridad,

obligatoriedad, equidad, eficiencia, subsidiariedad y suficiencia, por lo que se crea la DNTI, cuya función principal es la de modernizar al IESS.

La Disponibilidad, integridad y confiabilidad de la Información pueden ser esencial por lo cual es innegable la modernización tecnológica que ha llevado a cabo el IESS para garantizar una mejor y más rápida atención a los afiliados en el trámite de prestaciones y servicios, y que ahora se han simplificado gracias al uso de modernos sistemas informáticos y la puesta en práctica de sistemas administrativos más eficientes, sin embargo, pese a los múltiples logros alcanzados, la labor institucional enfrenta problemas aún no resueltos, dificultades acumuladas y nuevos retos sociales. (Castañeda Cadena & Quezada Sarasti, 2007)

La dependencia de los sistemas y servicios de información implica a que sean más vulnerables ya que no se han diseñado para ser seguros y esto se ha debido a que la DNTI del IESS ubicado de la ciudad de Quito está planificando y gestionando con inconvenientes las aplicaciones debido a lo engorroso de la metodología RUP, existe casos tales como en la gestión y desarrollo de software lo cual en la actualidad está ocasionando que la comunicación sea ambigua e imprecisa y que en un futuro muy cercano tengan sistemas inestables y deteriorados con un sin número de defectos.

El Problema de la creación de aplicaciones de Software se está originando por no estar analizando y estudiando las necesidades reales del producto a elaborar y a su vez por no incluir el tiempo o el esfuerzo especificado de la metodología en las planificaciones, lo que ha ocasionado no poder cumplir los requerimientos de calidad de la metodología, más aun si las necesidades de la Instituciones son de carácter dinámico.

1.3. Justificación

Para el desarrollo de la Metodología de Trabajo se contara con el apoyo pertinente y necesario de los profesionales del área a investigar y que a su vez facilitaran con la fuente de información que servirá de apoyo para el caso de estudio, todo estos beneficios ayudaran a complementar para poder resolver las necesidades y falencias que tiene el IESS de manera especial en

la DNTI, ya que todos estos aspectos a su vez están englobado en el ámbito Tecnológico y el cual permitirá enfocarse de manera más concisa a los problemas, carencias actuales que el usuario y el país genera; y poder determinar los inconvenientes que posee actualmente al utilizar la Metodología RUP, pero todos los beneficios y facilidades que tendrá la nueva metodología está encaminado para el grupo de trabajo los cuales ayudara a reducir tiempo y poder cumplir sus objetivos, y en un futuro evitar problemas complejos de aplicaciones informáticas y a la vez fomentara a gestionar de una forma más concisa los proyectos de software.

Pero todo este proyecto de investigación está encaminado a que se pueda generar un software de calidad ya que de acuerdo a los cambios de gobierno actual el desarrollo tecnológico constituye la base permanente y continúa para el desarrollo del País de manera especial en el sector público admitiendo generar solidez y confianza en el usuario.

Por otro lado la prioridad es que se pueda utilizar en una gran variedad de tipos de sistemas en diferentes áreas y dimensiones de proyectos permitiendo definir de una manera clara, quien debe hacer las cosas, qué debe hacerse, cuándo y cómo; admitiendo acoplarse a las necesidades reales del IESS.

1.4. Objetivos

1.4.1. Objetivo General

- Desarrollar una Metodología de Trabajo basado en RUP que permita la optimización en la elaborar aplicaciones de software en la Dirección Nacional de Tecnología de la Información del IESS de la ciudad de Quito.

1.4.2. Objetivos Específicos

- Elaborar el marco teórico pertinente.
- Especificar los requerimientos para la creación de la Metodología de Trabajo.

- Crear la Metodología de Trabajo para la Dirección Nacional de Tecnología de la Información del IESS.
- Desarrollar un prototipo de software.
- Validar la Metodología de Trabajo que será aplicado al prototipo desarrollado.
- Mejorar las aplicaciones del desarrollo de software.

1.5. Hipótesis

El problema que se desea resolver plantea la siguiente hipótesis:

¿Si se desarrolla una Metodología de Trabajo basado en RUP entonces se optimiza la elaboración de Aplicaciones de Software en la Dirección Nacional de Tecnología de la Información del IESS.?

1.6. Variables de Investigación

Variable Independiente:

- Se desarrolla la Metodología de trabajo basado en RUP.

Variable Dependiente:

- Se optimiza la elaboración de Aplicaciones de Software en la Dirección Nacional de Tecnología de la Información del IESS.

1.7. Metodología de Desarrollo del Proyecto

En el presente proyecto de investigación se utilizaran métodos teóricos y empíricos.

Los Métodos Teóricos se describen a continuación:

- Método Histórico-Lógico: Permite conocer los antecedentes históricos de la creación de aplicaciones de software en la DNTI del IESS de la ciudad de Quito.
- Método Inductivo-Deductivo: En el presente proyecto se aplicara el método inductivo incursionando en el estudio los métodos de desarrollo

tradicionales de software. Adicional se utilizara el método deductivo en la conceptualización de los procesos de desarrollo del software.

- Método Análisis-Síntesis: Para realizar juicios críticos sobre el análisis de los datos adquiridos durante la recopilación de información, se establecerá el marco de referencia de la Metodología RUP estudiada para determinar cuáles son las necesidades de la DNTI del IESS.
- Método Hipotético-Deductivo: Estará presente en toda la investigación desde que nosotros empezamos el estudio de la situación problemática.
- El Método Sistemático: Permite elaborar una adecuada propuesta para el Desarrollo de una Metodología de Trabajo basado en RUP a fin de fortalecer la Ingeniería de Software en la DNTI del IESS.

Los Métodos Empíricos: Estos métodos servirán para recoger información y son: La encuesta, la entrevista, análisis de documentos, preguntas y respuestas. La triangulación servirá para realizar la aseveración de los resultados cuanto la propuesta se encuentre lista para su implementación.

1.8. Descripción de la Metodología a Utilizar

Primeramente para analizar el área de estudio se utilizara el método histórico- lógico el cual permitirá conocer cómo se está llevando a cabo el análisis y desarrollo de las aplicaciones de software en la DNTI y para poder determinar la situación problemática se apoyara del método Hipotético-Deductivo y se utilizara entrevista a los líderes de los proyectos para complementar la investigación del problema.

Al momento de realizar el mantenimiento se posee diversas dificultades, como pérdida de tiempo, por no poseer una adecuada documentación, por no poseer una metodología adecuada que se adapte a las necesidades reales ya sea desde el inicio del desarrollo del software.

Una vez que se allá determinado la problemática utilizando los diversos métodos anteriormente mencionados, se procederá a desarrollar una adecuada metodología de trabajo basado RUP utilizando el Método Sistemático, ya que este permitirá elaborar un marco de trabajo en fase a las

necesidades de la DNTI, el cual se complementara mediante plantillas y artefactos, y para poder determinar el óptimo funcionamiento de la metodología creada se realizaran pruebas en un prototipo creado previamente, permitiendo fortalecer el desarrollo de software en la Dirección del IESS de la ciudad de Quito.

1.9. Área de Estudio

Para la presente investigación la Institución es el IESS el cual:

> "*es una entidad, cuya organización y funcionamiento se fundamenta en los principios de solidaridad, obligatoriedad, universalidad, equidad, eficiencia, subsidiariedad y suficiencia. Se encarga de aplicar el Sistema del Seguro General Obligatorio que forma parte del sistema nacional de Seguridad Social del Ecuador".* (Instituto Ecuatoriano de Seguridad Social, 2013)

Por la cual se describe una breve reseña de cómo evoluciono el IESS, y se especifica que el Seguro Social Ecuatoriano surgió en 1928, como parte del proceso de reforma del Estado y en el cual se creó la Caja de Jubilaciones, Montepío Civil, Retiro y Montepío Militar, Ahorro y Cooperativa, que de conformidad con la Ley se denominó Caja de Pensiones y que protegía a empleados del magisterio público, empleados públicos y bancarios y a los militares y el 2 de octubre de 1935 se dicta la Ley de Seguro Social Obligatorio, fijándose el campo de su aplicación en los trabajadores del sector público y privado y estableciéndose la contribución de aportes bipartita: patronal y personal para la cobertura de los riesgos con beneficios de jubilación, montepío y mortuoria, la Caja de Pensiones se mantiene como institución ejecutora y bajo la dependencia jurídica del creado Instituto Nacional de Previsión y en 1937 con la Ley del Seguro Social Obligatorio se crea la Caja del Seguro de Empleados Privados y Obreros y el Departamento Médico ligado a ella. (Castañeda Cadena & Quezada Sarasti, 2007)

En 1942 se expide la nueva Ley de Seguro Social Obligatorio, se establecen nuevas condiciones de aseguramiento, el financiamiento del Estado con el 40% de todas las pensiones del seguro general y se incorpora

el seguro de enfermedad y maternidad entre los beneficios para los afiliados y en 1963 se fusionaron la Caja de Pensiones y la Caja del Seguro para formar la CAJA NACIONAL DEL SEGURO Y DEL DEPARTAMENTO MÉDICO.

Y en 1970 se transformó la Caja Nacional del Seguro Social en el Instituto Ecuatoriano de Seguridad Social. La Asamblea Nacional, reunida en 1998 definió reformar la Constitución Política de la República, consagró la permanencia del IESS como única Institución Autónoma, responsable de la aplicación del Seguro General Obligatorio.

El Noviembre de 2001, se publica la LEY DE SEGURIDAD SOCIAL, en la cual el IESS tiene la misión de proteger a la población urbana y rural, con relación de dependencia laboral o sin ella, contra las contingencias de enfermedad, maternidad, riesgos del trabajo, discapacidad, cesantía, invalidez, vejez y muerte, en los términos que consagra la Ley de Seguridad Social. El IESS, según lo determina la vigente Ley de Seguridad Social, se mantiene como entidad autónoma, con personería jurídica, recursos propios y distintos de los del Fisco. Así mismo se establecen modificaciones para separar el financiamiento y administración de las contingencias cubiertas por el Seguro General Obligatorio que administra el IESS. El IESS, según lo determina la Constitución de la República del Ecuador, expedida por la Asamblea Nacional Constituyente aprobada en referendo por el pueblo ecuatoriano el 28 de septiembre de 2008, se mantiene como entidad autónoma, con personería jurídica, recursos propios y con la misma estructura orgánica.

En la actualidad a pesar de las necesidades que han surgido, el IESS es el organismo Ecuatoriano único en el País encargado de brindar seguridad social y servicios a afiliados y empleadores cuya sede principal está ubicada en la ciudad de Quito, ya que es un entidad Estatal descentralizada dotada de autonomía normativa, técnica, administrativa, financiera y presupuestaria, está sujeto a las normas de derecho público y rige su organización y funcionamiento por los principios de autonomía, división de negocios, desconcentración geográfica y descentralización operativa, por lo cual su función como Institución es ayudar al Bien Social y posee agencias en casi

todas las capitales de provincia. Entre algunas de sus importantes funciones es: cobertura médica a sus afiliados para lo cual cuenta con hospitales en varias ciudades, otorgar préstamos hipotecarios y quirografarios, el acceso a pensiones de jubilación a los trabajadores. (IESS, s.f.)

El IESS es una entidad cuya organización se fundamenta en proteger a la población urbana y rural en dependencia laboral o no, contra las limitaciones o falta de contingencia en rubros como maternidad, salud integral, riesgos de trabajo, incapacidad, cesantía, vejez, invalidez o muerte, manteniendo actualmente una etapa de transformación estructural considerable desde su base administrativa.

Pueden solicitar afiliación voluntaria los ecuatorianos y ecuatorianas residentes en el país y en el extranjero. Los que viven en el Ecuador, que no sean afiliados al Seguro obligatorio, incluidas las trabajadoras de hogar sin remuneración y los estudiantes. Los residentes en el exterior, cualquiera que sea su ocupación laboral o actividad económica. (Instituto Ecuatoriano de Seguridad Social, 2013)

1.10. Estructura Organizacional de la DNTI

El IESS, para posibilitar el cabal cumplimiento de su Misión debe valerse de herramientas necesarias entre ellas están las de comunicación e informática, estas deben ser de última generación y brindar todas las facilidades a los usuarios internos y externos a la Institución, para garantizar una atención oportuna y eficiente, por lo cual gestiona la administración de su mundo informático a través de su DNTI ya que es una Unidad de apoyo, el cual está encargado de crear, desarrollar y mantener los aplicativos informáticos necesarios que faciliten el tema comunicacional y transaccional en la recaudación de primas de aseguramiento e ingresos en general, y de reconocimiento, entrega de prestaciones económicas y en especial a los asegurados y empleadores, para promocionar la gestión institucional, el IESS mantiene conexión con medios comunicacionales como: radio, prensa escrita y televisiva, de redes sociales y de internet; para la comunicación transaccional directa con los usuarios posee una página WEB, donde constan

todos los temas de obligado conocimiento para empleadores, afiliados y público en general, ya que su función es la de modernizar el área tecnológica y para lo cual cuenta con la siguiente estructura.

El grupo de coordinación del Proyecto de Modernización está conformado de la siguiente manera:

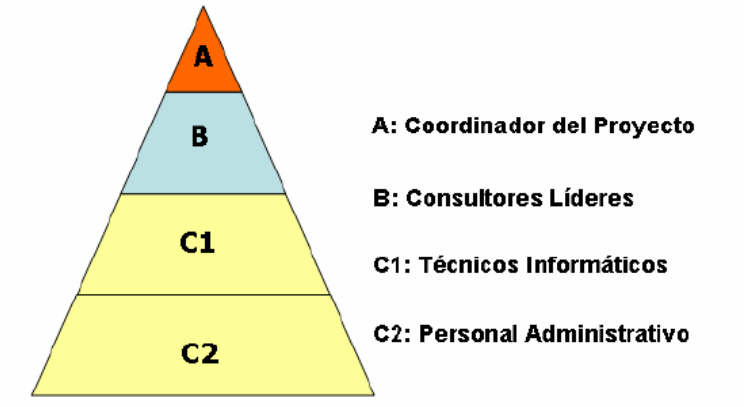


Figura 1: Estructura de Coordinación del Proyecto de Modernización
Fuente: (Castañeda Cadena & Quezada Sarasti, 2007)

En la figura 1, se muestra el esquema de cómo está estructurado la coordinación del proyecto, cuenta con el Coordinador del proyecto cuya función principal es la de dotar de toda la infraestructura que necesita el proyecto como equipo, y personal capacitado para las diferentes áreas.

En segundo plano se encuentra los consultores líderes quienes se encargan del aspecto de desarrollo.

El tercer grupo son técnicos cuya función es la administración, mantenimiento de los equipos y servicios y soporte a usuarios.

El cuarto grupo maneja el aspecto administrativo, son los que se encargan del aspecto del recurso humano.

1.11. Organigrama de la DNTI

DNTI: Dirección Nacional de Tecnología de la Información

Figura 2: Organigrama de DNTI

Fuente: (IESS, s.f.)

La DNTI o DDI es la encargada de la formulación y coordinación de la ejecución de los proyectos y programas de mejoramiento y desarrollo de la Institución, en procura de la eficacia, eficiencia y economía de los procesos del IESS, de conformidad con lo establecido en el Plan Estratégico Institucional y las normas y políticas definidas por el Consejo Directivo, y además es responsable de la administración del sistema informático e infraestructura tecnológica del Instituto. (IESS, s.f.)

La autoridad responsable de la gestión de la Dirección Nacional de Tecnología de la Información es su Director, nombrado por el Director General, de conformidad con las leyes y reglamentos sobre la materia.

1.11.1. Responsabilidades

La DNTI tiene a su cargo las siguientes responsabilidades:

- La ejecución de actividades de apoyo técnico relacionadas con el desarrollo de la Institución.

- La proposición ante la Dirección General de proyectos o programas relacionados con la sistematización de productos, procesos de la Institución, la dirección y supervisión de los proyectos.
- La aplicación estricta de las normas legales y procedimientos vigentes, relacionados con la administración de los recursos humanos.
- La presentación a la Gerencia General, de los informes de rendición de cuentas, sobre el cumplimiento de actividades.

1.11.2. Dependencias de la DNTI

La Dirección Nacional de Tecnología de la Información tiene a su cargo: La Subdirección de Planificación Institucional, La Subdirección de Procesos y Normatividad, La Subdirección de Tecnología y Unidad de Presupuesto, Unidad de Historia Laboral. (Castañeda Cadena & Quezada Sarasti, 2007)

1.11.3. Subdirección de Planificación Institucional

Es la encargada de la Planificación Estratégica Institucional, su evaluación y control apoyado de las herramientas metodológicas, tecnológicas fomentan a la formulación y coordinación de la ejecución de los proyectos, programas de mejoramiento y desarrollo de la Institución.

Tiene a su cargo: La Unidad de Planificación y la Unidad de Gestión de Proyectos.

a. Responsabilidades de la Unidad de Planificación

La formulación del Plan Estratégico Institucional, en coordinación con los órganos y dependencias de la Institución.

La evaluación técnico económica, sistemática y periódica, de la gestión Institucional, en cuanto al cumplimiento del Plan Estratégico Institucional y sus planes operativos.

Análisis y diseño del sistema de control de gestión Institucional.

b. Responsabilidad de la Unidad de Gestión de Proyectos

La evaluación, seguimiento y control del desarrollo de todos los programas y proyectos derivados de la planificación estratégica y operativa de la Institución.

El registro, control y archivo de la documentación sobre el avance de resultados de los proyectos a su cargo.

Construcción, implementación y administración del sistema de control de gestión institucional.

1.11.4. Subdirección de Procesos y Normatividad

Es la encargada de establecer el marco normativo en todos los ámbitos de acción Institucional, así como el esquema de gestión organizacional por procesos, en concordancia con el marco Estratégico Institucional.

Tiene a su cargo: La Unidad de Normatividad y la Unidad de Procesos.

a. Responsabilidades de la Unidad de Normatividad

Elaborar, actualizar y/o derogar las políticas, normas, procedimientos, manuales, instructivos y formularios en general, que regulen el funcionamiento de las Dependencias y Entidades del Instituto.

Realizar de manera adecuada y oportuna actividades de promoción, investigación y divulgación de normatividad y sus respectivos manuales.

Asesorar a las Unidades o departamentos, que lo solicitan en la interpretación y aplicación de procedimientos y técnicas administrativas.

b. Responsabilidades de la Unidad de Procesos

Realización de estudios, diagnóstico y análisis de la estructura, funcionamiento y de costeo de los procesos de la Organización y en especial los de afiliación, aseguramiento y entrega de prestaciones.

Diseño de los procesos mediante los cuales se elaboran y aplican las regulaciones que norman la actividad institucional.

Realizar conjuntamente con el área de tecnología el análisis de procedimientos susceptibles de automatización.

1.11.5. Subdirección de Tecnología

Es la encargada de la planificación, normatividad, programación, organización, gestión, control y evaluación de la plataforma tecnológica y de los servicios que esta brinda, derivados de la planificación estratégica institucional referida.

Tiene a su cargo: La Unidad de Desarrollo, La Unidad de Producción y la Unidad de Implementación Tecnológica.

a. Responsabilidades de la Unidad de Desarrollo

La asistencia técnica para el desarrollo e implementación de sistemas automatizados en los procesos institucionales.

El establecimiento de la normatividad técnica para el desarrollo de las herramientas informáticas que demande la Institución.

La preparación de especificaciones técnicas de los documentos precontractuales y la asistencia técnica a los Titulares o encargados de las Direcciones, para la adquisición y/o desarrollo de herramientas de software.

b. Responsabilidades de la Unidad de Producción

La coordinación de la operación y mantenimiento de los centros de cómputo, las redes de comunicación de datos y las bases de datos, a escala nacional.

La preparación de detalles técnicos, documentos precontractuales y la asistencia técnica a los Titulares encargados de las Direcciones, para la adquisición y/o arrendamiento de hardware, software para la administración y operaciones de la tecnológica, licencias, instalación, mantenimiento y soporte técnico.

El establecimiento y uso de sistemas de información confiables y de sistemas apropiados de documentación y archivos de registro, informes y documentos de las actividades a cargo de esta Unidad.

c. Responsabilidad de Unidad de Infraestructura Tecnológica

La elaboración de los proyectos de implementación de infraestructura tecnológica, de cada una de las dependencias del Instituto.

La implementación de la infraestructura tecnológica, en las dependencias de la Institución.

El análisis de las innovaciones tecnológicas, a fin de incorporar nuevas tendencias al Instituto.

El apoyo a los procesos de adquisición, en la etapa de calificación de ofertas, de equipamiento informático y/o de telecomunicaciones especializado.

1.11.6. Unidad de Presupuesto

Es la encargada de la planificación y ejecución presupuestaria de todos los proyectos y procesos a cargo de la DNTI, así como también la administración del inventario del equipamiento tecnológico institucional.

1.11.7. Unidad de Historia Laboral

Es la encargada de Desarrollar la conceptualización de los productos y servicios institucionales, en coordinación con las subdirecciones de Planificación, Normativa y Procesos, así como la administración de la operación de los mismos, en el ámbito nacional, aplicando criterios de eficacia, eficiencia y altos niveles de calidad.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. INGENIERÍA DE SOFTWARE

2.1.1. Aspectos Históricos

La Ingeniería de Software, en comparación con otras ingenierías aún es muy joven ya que manifiestan que el diseño, la práctica y el mantenimiento del software se puede ponerse en el mismo pedestal de las disciplinas de la ingeniería tradicional, ya que su término se mencionó en una conferencia de Ingeniería de Software de la Organización del Tratado del Atlántico Norte ("OTAN") en 1968, para realizar un análisis sobre la crisis de software de aquel momento, pero hasta la actualidad todavía no se ha salido, ya que la calidad del software es regularmente baja y no cumple los plazos establecidos, ni los presupuestos. (Schach, 2006)

De acuerdo con Rubby Casallas (Casallas), manifiesta que la crisis del software era bastante compleja porque poseía un sin número de problemas por la utilización de métodos Ad Hoc (el ciclo programar – corregir), por tal motivo resalto la importancia de la Ingeniería de Software ya que su finalidad es contar con prácticas más disciplinadas.

A pesar de la evolución de los métodos alto y bajo nivel, paradigmas en donde se ha pasado del modelo procedimental al funcional y de este al Orientado a Objetos y actualmente el orientado a servicios, y en el desarrollo constante del software existe todavía inconvenientes al momento de entregar el producto como envió tardío, exceso del presupuesto establecido e infinidad de fallos de último momento, por lo que se establece que muy poco software se entrega a tiempo, dentro del presupuesto, libre de fallos y cumpliendo con las necesidades del cliente. (Reinoza, 2011)

2.1.2. Definición de Ingeniería de Software

Es una disciplina de la Ingeniería y establece todos los aspectos que conciernen a la producción del Software ya sea desde las etapas iniciales de la especificación del sistema, hasta el mantenimiento del mismo después de que se esté utilizando y lo cual ofrece métodos, técnicas que permiten desarrollar y mantener un software de calidad confiable y eficiente.

Hoy en día es considerada una nueva área de la Ingeniería, ya que trata con áreas diversas de la informática y de la ciencia de la computación tales como construcción de Sistemas Operativos, desarrollo de Internet / Intranet para lo cual aborda todas fases de desarrollo de diversos tipos de sistemas y son aplicables en áreas como: negocios, investigación científica, medicina, producción, logística, banca, control de tráfico, Instituciones Educativas, etc.

Definiciones más precisas de la Ingeniería de Software se cita como:

> *"Ingeniería del Software es una disciplina cuyo objetivo es la producción de software libre de fallas, enviando a tiempo y dentro del presupuesto, que satisfaga las necesidades del cliente".* (Schach, 2006, pág. 4)

a. Capas de la Ingeniería de Software

Independientemente de la dificultad del sistema y del área de aplicación la Ingeniería del software es una tecnología multicapa (ver figura 3), pero donde enfatiza que la ingeniería de software en la primera capa que está orientado hacia la calidad. (Pressman, 2002, pág. 8)

Proceso: Es el fundamento de la ingeniería del software ya que permite generar un marco de trabajo para un conjunto de Áreas Clave de Proceso ("*ACPs*") que se deben establecer para la entrega efectiva de la tecnología. Las áreas claves del proceso forman la base del control de gestión de proyectos del software y establecen el contexto en el que se aplican los métodos técnicos, se obtienen productos del trabajo (modelos, documentos, datos, informes, formularios, etc.), se establecen hitos, se asegura la calidad y el cambio se gestiona adecuadamente.

Métodos: Son los que indican cómo construir técnicamente el software ya que abarcan una gran gama de tareas que incluyen análisis de requerimientos, diseño, construcción de programas, pruebas y mantenimiento. Los métodos de la ingeniería del software dependen de un conjunto de principios básicos que gobiernan cada área de la tecnología e incluyen actividades de modelado y otras técnicas descriptivas.

Herramientas: En Ingeniería del software crean un enfoque automático o semi-automático para el proceso y los métodos. Cuando se integran herramientas para que la información creada por una herramienta la pueda utilizar otra, se establece un sistema de soporte de desarrollo del software denominado Ingeniería del Software Asistida por Computadora *("CASE").*

Figura 3: Capas de la ingeniería del software
Fuente: (Pressman, 2002)

2.1.3. Herramientas

A inicios de 1990, los analistas empezaron a beneficiarse de las herramientas de Ingeniería de Software Asistida por computadora, ya que se crearon explícitamente para su trabajo mediante el apoyo automatizado, el cual colabora desde el principio hasta el fin del ciclo de vida e incrementa la productividad, la comunicación de manera más eficiente con los usuarios e integrar el trabajo que desempeñan en el sistema. (Kendall & Kendall, 2005)

Poseen 2 tipos de herramientas, las analíticas o teóricas se usan para el desarrollo del software y las de software que ayudan al equipo para el desarrollo, mantenimiento de software y se denominan herramientas CASE.

Primero se especificara las herramientas teóricas las cuales son:

a. Depuración Paso a Paso

Es una técnica para resolver los problemas que enfrentan muchas técnicas de ingeniería y permite definir como un medio para posponer, tanto como sean posibles las decisiones sobre el detalle y poder concentrarse en los aspectos importantes y diferir las decisiones que no son esenciales.

b. Análisis Costo – Beneficio

Una forma de comparar si es rentable los beneficios futuros estimados contra los costos futuros proyectados, es poder calcular los costos de hardware, software y la conversión, ya que es complicado medir los beneficios intangibles por lo que se hace suposiciones, estas siempre deben ser establecidas junto con los estimados resultantes de los beneficios, ya que es una técnica fundamental para decidir si un cliente debe computarizar su empresa, y si lo desea definir de qué forma.

c. Métrica de Software

Sin mediciones o métricas es imposible detectar a tiempo los problemas en el proceso de software por lo que existe métricas tales como:

- **El número de líneas de código (“LOC”).-** Es medir el tamaño de un producto, para ver la rapidez con la que avanza el proyecto.
- **Tiempo Medio.-** Es la que permite entre las diversas fallas generadas proporcionar un indicio de su confiabilidad.
- **Esfuerzo persona–mes.-** Se aplica durante el proceso de software.
- **Rotación de Personal.-** Perjudica a los proyectos actuales.
- **Costo.**- Se debe vigilar continuamente a lo largo de todo el proceso.
- **Eficiencia de Detección de fallas.-** Es detectar fallas durante el desarrollo y determinar el número de fallas total durante su vida útil.
- **Métricas de Producto.-** Las cuales miden algún aspecto propio del producto, como su tamaño o confiabilidad.
- **Métricas de Proceso.-** Lo utilizan los desarrolladores para deducir la información acerca del proceso de programación.

Entre las métricas más importantes se tiene: el tamaño (en líneas de código), costo (en dólares), duración (en meses), esfuerzo (persona –mes), calidad (número de fallas detectadas), por lo cual los flujos de trabajo deben medir estas métricas, y generar datos para que la gerencia pueda identificar problemas dentro de la organización del software.

Las herramientas de Programación CASE son las siguientes:

d. CASE

Las computadoras ayudan en cada paso del camino, como el desarrollo de la programación, la creación y organización de artefactos como los planos, contratos, especificaciones, diseños, códigos fuentes, y la documentación es importante en el desarrollo y mantenimiento del software.

e. Taxonomía de CASE

Las Herramientas de Software o CASE es un producto que ayuda en un solo aspecto de la producción del software, ya que se lo utiliza con cada flujo de trabajo del ciclo de vida**,** y por lo cual existe dos tipos de Herramientas:

- **Herramientas extremo Frontal.-** Ayudan al programador durante los primeros flujos de trabajo del proceso, requerimiento, análisis y diseño
- **Herramientas extremo Posterior.-** Permiten ayudar con los flujos de trabajo para la implementación y el mantenimiento de posentrega.
- **Diccionario de datos.-** Es una lista computarizada de todos los datos definidos dentro del producto, y se obtiene de cada entrada una descripción del elemento.
- **Generador de Informes.-** Se utiliza para generar los códigos necesarios para producir un informe.
- **Generador de Pantalla.-** Se utiliza para ayudar al desarrollador de software a producir el código para una pantalla de captura de datos.
- **Verificador de Consistencia.-** Verifica que cada dato de elemento en el documento de especificaciones este reflejado en el diseño.

El uso de una herramienta de representación gráfica, un diccionario de datos, un verificador de la consistencia, un generador de informes y de pantalla constituye una mesa de trabajo.

Mesa de Trabajo.- Es una recopilación de herramientas que juntas dan soporte a una o dos actividades.

- **Entorno.-** Permite dar soporte a todo el proceso de software.

f. El Alcance de CASE

Permite determinar las especificaciones del producto y tendrá solo una copia y la cual es la versión en línea, y puede ser accedida por medio de la herramienta CASE y si se cambia las especificaciones, los integrantes del equipo de desarrollo pueden acceder con facilidad al documentó y estar seguros que ven la versión actual y para lo cual posee.

- La documentación en línea, correo electrónico.
- Herramientas de Codificación.- Se refiere a los editores de texto.

g. Versiones del Software

Cuando se da mantenimiento por lo menos debe haber 2 versiones del producto el cual está compuesto por artefactos del código que a su vez tiene 2 o más versiones de sus componentes que hayan sido cambiados.

g.1. Revisiones

Si en un artefacto se detecta fallos entonces tiene que arreglarse y se genera la nueva versión, pero todas las versiones viejas deben desecharse, dejando solo la correcta, siempre y cuando se haya comprobado y verificado su correcto funcionamiento.

g.2. Variaciones

A diferencia de las revisiones, las variaciones se diseñan para coexistir, el cual permite aportar un producto a una variedad de sistemas operativos y hardware, por lo que se determina que hay múltiples revisiones en cada variación.

h. Control de Configuración

Para cada artefacto de código hay tres formas:

- **El código Fuente**.- Actualmente utilizan lenguajes como C++ o Java.
- **El código Objeto**.- Producido compilado el código fuente o denominado código compilado.
- **El código Compilado**.- Para cada artefacto se combina las rutinas del tiempo de corrido para producir una imagen de carga ejecutable.

También manejan los problemas de desarrollo y mantenimiento.

h.1. Control de configuración durante el mantenimiento de posentrega

Las dificultades pueden surgir cuando más de un programador mantiene al mismo tiempo un producto, ya que la idea de cada programador es realizar copias individualmente de un artefacto por lo que es inadecuado en la parte de mantenimiento trabajar en equipo.

h.2. Líneas Base

El gerente de mantenimiento prepara una línea base, es decir una configuración de los artefactos del producto, y cuando intenta encontrar una falla un programador de mantenimiento pone en su espacio de trabajo privado las copias de los artefactos que necesita y puede cambiar sin afectar a otro programador, y la línea base queda intacta, ya que congela la versión actual del artefacto que está modificando y ningún puede hacer cambios.

h.3. Control de la Configuración durante el desarrollo

Cualquier cambio a un artefacto integrado puede tener un efecto sobre el producto en su totalidad, del mismo modo que un cambio hecho durante el mantenimiento de la posentrega sino también durante la implementación y cuando se aplica adecuadamente el control de la configuración, la gerencia esta consiente del estado de cada artefacto y puede tomar oportunamente la acción correctiva.

i. Herramientas de Construcción

Para una empresa de programación que no desea una herramienta de control de configuración completa por lo menos debe utilizar una herramienta de construcción y de control de versiones, que le ayude en la selección de versión correcta de cada artefacto de código compilado que será posteriormente ligado para formar una nueva versión del producto.

Los beneficios de utilizar la tecnología CASE en una empresa de programación son: desarrollo más rápido, menos fallas, mejor capacidad de uso, mantenimiento más sencillo y mejora de estado de ánimo.

2.2. Ciclo de Vida de Desarrollo de Software

Es la de poder especificar un enfoque definido por diversas fases para la realización del análisis y diseño ya que está encaminado a que los sistemas se desarrollan mejor utilizando un ciclo específico. (Kendall & Kendall, 2005)

Los ciclos de vida también son: “Las actividades básicas del proceso de desarrollo de software”. (Weitzenfeld, 2005, pág. 38)

Por otra parte el (Instituto Nacional de Tecnologías de la Comunicación, 2009), determina que el ciclo de vida es un conjunto de fases por las que pasa el sistema cuando se está desarrollando desde que nace la idea hasta que el software es retirado o remplazado, ya que indica que es lo que hay que obtener a lo largo del desarrollo del proyecto pero no cómo hacerlo y posee fases y entregables, para lo cual se especifica sus funciones:

- **Fases:** Una fase es un conjunto de actividades relacionadas con un objetivo en el desarrollo del proyecto y pueden compartir un tramo determinado del tiempo de vida de un proyecto.
- **Entregables:** Son los productos intermedios que generan las fases. Pueden ser materiales o inmateriales (documentos, software) y los entregables permiten evaluar la marcha del proyecto.

2.2.1. Actividades del Desarrollo de Software

La transición entre las diferentes actividades debe haber continuidad o rastreabilidad de una actividad a la siguiente o la anterior, para ello se describe las actividades más importantes del ciclo de vida.

Tabla 1

Actividades del Desarrollo de Software

Actividad	Descripción
Requerimientos	Se especifican las necesidades del sistema a desarrollar, la especificación de requerimientos sirve como base para la negociación entre los desarrolladores y clientes del sistema y es la plataforma de desarrollo de los demás modelos.
Análisis	Se busca comprender los requerimientos del sistema con el propósito de estructurar la arquitectura del sistema, y se enfoca en que debe hacer el sistema, en lugar de como se supone que lo hará y está relacionado directamente con el problema.
Diseño	Se transforma la arquitectura obtenida durante el análisis en una arquitectura especializada, donde se considera el ambiente de implementación particular del sistema.
Implementación	Se expresa la arquitectura del sistema en una forma aceptable para la computadora o conocida como el código y se apoyan de herramientas CASE, basado en Lenguaje de modelado unificado ("*UML*"), y para el desarrollo total del código completa el desarrollador de manera manual con lenguajes de programación y base de datos.
Integración	Se combina los componentes creados de manera independiente para formar el sistema completo.
Pruebas	Se verifica y valida el sistema a nivel de componentes individuales y su integración, y es uno de los aspectos más críticos del desarrollo, ya que es responsable para obtener calidad de un sistema, en el cual se busca descubrir todos los defectos en los requerimientos, análisis, diseño, implementación e integración.
Documentación	Se describen los aspectos sobresalientes de los requerimientos, análisis, diseño, implementación, integración y pruebas. Esto servirá para usuarios externos e internos, aquellos encargados de mantener el sistema y extenderlo, y entre ellos se genera el manual de usuario, programador, Operador, Administrador.
Mantenimiento	Se corrigen errores no encontrados en el desarrollo y en las pruebas originales del sistema, y en cierto modo es considerado como un nuevo ciclo de actividades de desarrollo, pero a partir de un sistema ya existente.

Fuente: (Weitzenfeld, 2005)

2.2.2. Modelos del Ciclo de Vida

Según (Sommerville, 2005, pág. 8), es un modelo consiste en una descripción simplificada de un proceso determinado de software y el cual presenta una visión de aquel proceso., con sus actividades correspondientes, por lo que la ingeniería de software establece y se vale de una serie de modelos los cuales muestran las distintas etapas y estados por los que pasa un producto de software, desde su concepción inicial, desarrollo, puesta en marcha, mantenimiento y retirada del producto, y describe las fases, y el orden en que ejecutaran sus fases.

A continuación se describe algunos de los modelos tradicionales.

a. Modelo en Cascada

Según (Pressman, 2002, pág. 20), es llamado también “ciclo de vida básico” o “Lineal Secuencial”, es el paradigma más antiguo de ingeniería del software y se cree que fue el primer modelo del proceso introducido, la primera descripción se publicó en 1970 por Winston Royce, por lo cual las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo.

El modelo se inicia con la especificación de requerimientos del cliente y continúa con la planeación, el modelo, la construcción y el despliegue para culminar en el soporte del software terminado, por lo cual en la figura 4, se describe el orden del desarrollo de software.

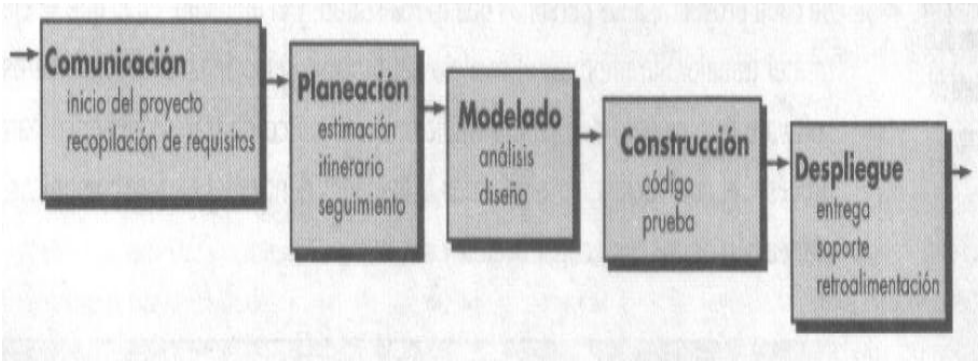


Figura 4: El modelo en cascada
Fuente: (Pressman, 2002)

El modelo es útil en situaciones donde los requerimientos están fijos o se comprendan bien y a su vez se puede encontrar algunos problemas al momento de aplicar como:

- En proyectos reales es raro que sigan el flujo secuencial del modelo.
- Es difícil al cliente establecer todos los requerimientos de manera explícita.
- Un error grave será desastroso si no se detecta antes de la revisión del programa.

b. Modelo Iterativo

Es un modelo derivado del ciclo de vida en cascada, la finalidad es poder reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos en la etapa de recogida de requerimientos, ya que se puede ir refinando en cada una de las iteraciones, por lo que al final de cada iteración se le entrega al cliente una versión mejorada del producto y evalúa y lo corrige o propone mejoras, las iteraciones se repetirán hasta obtener un producto que satisfaga las necesidades del cliente.

Se utiliza en proyectos en los que los requerimientos no están claros por parte del usuario, por lo que se hace necesaria la creación de distintos prototipos para presentarlos y conseguir la conformidad del cliente, y consiste en la iteración de varios ciclos de vida en cascada en la figura 5, se puede observar que el modelo no sigue un sentido lineal ni restrictivo, esto se debe a que es el diseñador junto con los requerimientos de desarrollo son los que marcarán cuantas iteraciones son necesarias. (Lorés & Granollers, 2002)

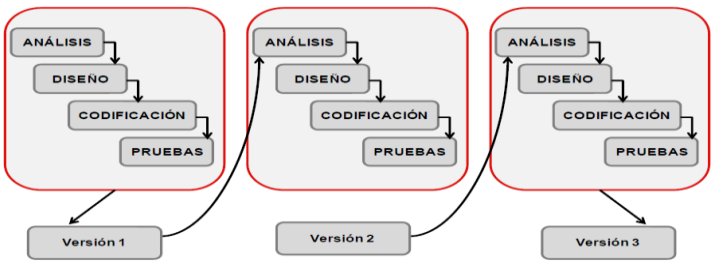


Figura 5: Modelo de ciclo de vida Iterativo
Fuente: (Instituto Nacional de Tecnologías de la Comunicación, 2009)

c. Modelo de Desarrollo Incremental

El modelo es un desarrollo inicial de la arquitectura completa del sistema, y permite incrementos y versiones parciales, ya que cada incremento tiene su propio ciclo de vida y proporciona una funcional adicional o mejorada sobre el sistema y a su vez se evalúa con respecto al desarrollo de versiones futuras. (Weitzenfeld, 2005)

Combina elementos del modelo lineal secuencial con la filosofía interactiva de construcción de prototipos, en la figura 6, el modelo incremental aplica secuencias lineales de forma escalonada mientras progresa el tiempo en el calendario y cada secuencia lineal produce un incremento del software por lo tanto es un modelo más flexible y permite reducir el costo en el cambio de alcance y requerimientos, es más fácil probar y depurar en una iteración más pequeña, este proceso se repite siguiendo la entrega de cada incremento, hasta que se elabore el producto completo.

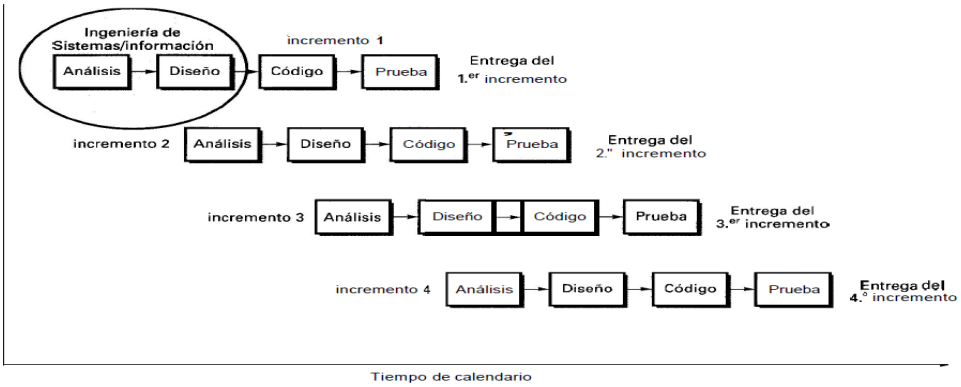


Figura 6: Modelo Incremental
Fuente: (Pressman, 2002)

2.3. Metodologías de Desarrollo de Software

El desarrollo de software no es una tarea fácil, por lo cual existe numerosas propuestas metodológicas que inciden en el proceso de desarrollo y por lo que una de las prioridades de décadas ha sido encontrar procesos y metodologías, que sean sistemáticas, predecibles y repetibles, a fin de mejorar la productividad en el desarrollo y la calidad del producto software. (Instituto Nacional de Tecnologías de la Comunicación, 2009)

2.3.1. Definición de Metodología

Una metodología para el desarrollo de software comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto software desde que surge la necesidad del producto hasta que cumplimos el objetivo por el cual fue creado y no tiene que ser necesariamente adecuada para usarla en todos los proyectos.

Entre los elementos que forman parte de una metodología son: Las fases, los productos de entrada y salida de cada fase y documentos, los procedimientos y herramientas que apoyan a la realización de cada tarea, y los criterios de evaluación para el proceso y producto.

2.3.2. Metodologías Tradicionales y Ágiles

Desarrollar un buen software depende de un gran número de actividades y etapas, donde el impacto de elegir la metodología para un equipo en un determinado proyecto es trascendental para el éxito del producto, por lo que las metodologías poseen dos grupos: los tradicionales y ágiles.

Tabla 2

Comparativa entre metodologías tradicionales y desarrollo ágil

METODOLOGÍA ÁGIL	METODOLOGÍA TRADICIONAL
Basadas en heurísticas provenientes de prácticas de producción de código.	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Especialmente preparados para cambios durante el proyecto.	Cierta resistencia a los cambios.
Impuestas internamente (por el equipo).	Impuestas externamente.
Proceso menos controlado, con pocos principios.	Proceso mucho más controlado, con numerosas políticas/normas.
No existe contrato tradicional o al menos es bastante flexible.	Existe un contrato prefijado.

Continua

El cliente es parte del equipo de desarrollo**.**	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio**.**	Grupos grandes y posiblemente distribuidos.
Pocos artefactos**.**	Más artefactos.
Pocos roles**.**	Más roles.
Menos énfasis en la arquitectura del software**.**	La arquitectura del software es esencial y se expresa mediante modelos.

Fuente: (Instituto Nacional de Tecnologías de la Comunicación, 2009)

En la tabla 2 se muestra una comparativa entre estos dos grupos de metodologías.

2.3.3. Metodologías Tradicionales

Conocida también como metodologías pesadas, se centran en llevar una documentación exhaustiva de todo el proyecto y en cumplir el plan de proyecto, y la fase inicial del desarrollo de proyecto y es utilizado para especificaciones de seguridad y sistemas críticos. (Sommerville, 2005)

Las metodologías tradicionales (formales) podrían dar lugar a programas con menos errores y más adecuados a las necesidades del usuario, ya que se focalizan en la documentación, planificación y procesos (plantillas, técnicas de administración, revisiones, etc.)

a. Microsoft Solution Framework (“MSF”)

Es una flexible e interrelacionada serie de conceptos, modelos y prácticas de uso que controlan la planificación, el desarrollo y la gestión de proyectos tecnológicos, pero se centra en los modelos de proceso y de equipo dejando en un segundo plano las elecciones tecnológicas por otra parte MSF es un framework, no una metodología y Microsoft manifiesta que el éxito de un proyecto de software no pasa a través de un conjunto de listas de control y tareas requeridas. (Taborda, 2012)

Según María Arévalo (Arevalo, s.f.) Manifiesta que MSF le permite dar una orientación de cómo debe organizar un equipo de trabajo, planificar los proyectos, construir e implementar para llegar al éxito con las soluciones

que desarrollan a sus clientes y manifiesta que no puede haber un solo proceso de desarrollo de software para los proyectos que se ejecuten.

En la actualidad está la versión 4.0, definida como “un meta-modelo para describir el ciclo de vida de desarrollo de software” y es un Framework descriptivo similar a la versión 3.0 en muchos aspectos, pero la gran diferencia es que incluye dos metodologías prescriptivas: El MSF para el desarrollo de Aplicaciones Ágiles y MSF para el proceso de mejora CMMI.

a.1. Modelos y Disciplinas en MSF

MSF, una versión 3.0, se compone de modelos y disciplinas encargadas de planificar las diferentes partes implicadas en el desarrollo de un proyecto, y para lo cual se especifica cada uno de ellos:

- **Modelo de Equipo**.- Es diseñado para mejorar el equipo de desarrollo y proporciona una estructura flexible para los equipos de un proyecto, asignando roles y responsabilidades a cada uno.
- **Modelo de Proceso:** Diseñado para mejorar el control del proyecto, minimizando el riesgo, aumentando la calidad, acortando el tiempo de entrega y generando pautas a seguir en el ciclo de vida del proyecto.
- **Disciplina Gestión del Riesgo**.- Ayuda al equipo a identificar las prioridades, tomar las decisiones estratégicas correctas y controlarlas.
- **Disciplina Administración de Proyectos o Gerencia de Proyectos**.- Es una disciplina que describe el rol de la gestión del proyecto que se basa en: Planificar sobre entregas cortas, Incorporar nuevas características sucesivamente e Identificar cambios ajustando el cronograma.
- **Modelo de Arquitectura del Proyecto**.- Permite acortar la planificación del ciclo de vida y definir las pautas para construir proyectos empresariales a través del lanzamiento de versiones.
- **Modelo de Diseño de Proceso**.- Para distinguir entre los objetivos empresariales, las necesidades del usuario y las fases de diseño conceptual, lógico y físico proveen perspectivas diferentes para los usuarios, el equipo y los desarrolladores.

- **Modelo de Aplicación**.- Diseñado para mejorar el desarrollo, el mantenimiento y el soporte, proporciona un modelo de tres niveles para diseñar y desarrollar aplicaciones software.

En la figura 7, se visualiza el Modelo de arquitectura del proyecto.

Figura 7: Modelos y Disciplinas con MSF
Fuente: (Arevalo, s.f.)

a.2. Procesos o Ciclo de Vida de MSF

Todo proyecto esta separado en cinco principales fases:

- **Visión y Alcances.-** El equipo debe tener una visión de lo que quisiera lograr para el cliente e indicarlo y se definen líderes y responsables del proyecto, se identifican las metas y objetivos a alcanzar; y se realiza la evaluación inicial de riesgos del proyecto.
- **Planificación.-** El equipo prepara las especificaciones funcionales, realiza el proceso de diseño de la solución, y los planes de trabajo, estimaciones de costos y cronogramas entregables del proyecto.
- **Desarrollo.-** El equipo realiza la mayor parte de la construcción de los componentes (tanto documentación, código), sin embargo, se puede realizar algún trabajo de desarrollo durante la etapa de estabilización en respuesta a los resultados de las pruebas y la infraestructura también es desarrollada durante esta fase.
- **Estabilización.-** Las pruebas de esta etapa enfatizan el uso y operación bajo condiciones realistas. El equipo se enfoca en priorizar y resolver errores y preparar la solución para el lanzamiento.

- **Implantación.-** Durante esta fase el equipo implanta la tecnología base y los componentes relacionados, estabiliza la instalación.

Figura 8: Ciclo de Vida del Proyecto de MSF

Fuente: (Lamas, s.f.)

a.3. Roles

Se crean equipos de 3 o 4 personas, y en proyectos grandes se requieren 50 personas a más y MSF divide en 7 grupos de apoyo como se representa en la tabla 3, para ayudar a tener un enfoque equilibrado, estos roles incorporan a cada uno, una perspectiva única sobre lo que se necesita.

Tabla 3

Roles, Objetivos y Áreas Funcionales MSF

Rol	Objetivos	Áreas Funcionales
Administración de productos	- Asegurarse de que la solución ofrece un valor empresarial. - Asegurarse de que las necesidades y expectativas de los clientes se satisfacen.	- Comunicaciones de marketing y corporativas. - Análisis de negocio. - Planeación del producto.

Continua

Program Management	- Entregar la solución dentro de las restricciones del proyecto. - Configurar los medios que permiten cumplir las necesidades y expectativas del patrocinador	- Administración de proyectos - Program Management - Administración de recursos - Garantía de procesos - Administración de calidad de proyectos
Arquitectura	- Diseñar una solución para satisfacer los objetivos empresariales en las restricciones del proyecto	- Arquitectura de la solución - Arquitectura técnica
Desarrollo	- Compilar la solución según especificación	- Desarrollo de soluciones - Consultoría tecnología
Experiencia del usuario	- Mejorar la disponibilidad y eficacia de usuarios - Asegurarse de que las necesidades y expectativas de los usuarios se satisfacen	- Accesibilidad - Comunicaciones de soporte técnico - Recursos de aprendizaje - Facilidad de uso - Diseño de la interfaz
Prueba	- Apruebe la solución para su lanzamiento, después de asegurarse que todos los aspectos de solución cumplen o superan sus niveles de calidad.	- Pruebas de regresión - Pruebas funcionales - Prueba de facilidad de uso - Pruebas del sistema
Versión/ operaciones	- Implementación continua y transición a operaciones - Asegurarse que las necesidades de las operaciones empresariales y de TI se satisfacen	- Administración de versiones - Infraestructura de entrega - Administración de compilaciones - Administración de herramientas

Fuente: (Microsoft, 2013)

b. Metodología Rational Unified Process (RUP)

En al año de 1967 nació una metodología la Ericsson Approach diseñada por Ivar Jacobson como guía para el desarrollo de software basado en componentes y luego introdujo el concepto de los Casos de Uso, diez años más tarde Jacobson fundó la compañía Objectory AB y lanza el proceso de

desarrollo Objectory. Posteriormente en el año de 1995 la corporación Rational Software adquiere Objectory y lo fusiona con Rational Approach dando como resultado el desarrollo del Rational Objectory Process ("*ROP*") adoptando el UML como lenguaje de modelado.

Posteriormente la corporación Rational Software implementó modelados de negocio y soluciones empresariales, con el fin de expandir ROP y en el año de 1999 nace el Rational Unified Process (RUP) como una metodología de desarrollo de software compuesta por un conjunto de procesos prácticos que brindan una guía para el desarrollo de actividades en torno al equipo de desarrollo, permitiendo seleccionar un conjunto de componentes de proceso que se ajustan a las necesidades del proyecto. (Sommerville, 2005)

RUP posee un conjunto de actividades que permiten transformar los requerimientos del usuario en un sistema de software y es un marco de trabajo genérico utilizado para una gran variedad de sistemas de software y basado en componentes que se utilizaran para construir el sistema y las interfaces que conectaran los componentes, y UML prepara todos los esquemas de un sistema de software. (Jacobson, Booch, & Rumbaugh, 2000)

RUP es creado por Rational Software Corporation, una división de IBM desde 2003, el producto incluye una base de conocimiento con artefactos de ejemplo y descripciones detalladas para diversos tipos de actividades.

b.1. Características esenciales que definen al RUP

Los aspectos que tiene RUP son: dirigidos por casos de uso, centrado en arquitectura, e iterativo e incremental, y esto lo que le hace único.

- **Proceso Dirigido por los Casos de Uso.-** Son utilizados para el desenvolvimiento y desarrollo de las disciplinas con los artefactos, roles y actividades necesarias; ya que son la base para la implementación de las fases y disciplinas del RUP, y constituye una secuencia de pasos a seguir para la realización de un propósito, y se relaciona con los requerimientos del cliente y su posterior realización.

- **Proceso Iterativo e Incremental.-** Se plantea la implementación del proyecto a realizar en Iteraciones y se define los objetivos por cumplir en cada iteración y así poder ir completando todo el proyecto iteración por iteración, y tiene varias ventajas como, tener pequeños avances del proyectos que son entregables al cliente el cual puede probar mientras se está desarrollando otra iteración del proyecto, con lo cual el proyecto va creciendo hasta completarlo en su totalidad.
- **Proceso Centrado en la Arquitectura.-** La realización de una arquitectura ejecutable constituye un prototipo evolutivo, ya que es la estructura de un sistema de sus partes más relevantes, es utilizado como una implementación parcial del sistema, construida para demostrar algunas funciones y propiedades.

b.2. Ciclo de Vida de RUP

RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema, ya que cada ciclo concluye con una versión del producto para los clientes, se divide en etapas y fases facilitando la administración del proyecto ya que cada fase se subdivide a su vez en iteraciones.

- **Fases del ciclo de vida**

El proceso de RUP es una serie de ciclos que forma la vida de un sistema; cada ciclo forma una versión del sistema, en la siguiente figura 9, se especifica las fases ó (vistas dinámicas) e Hitos de un proyecto.

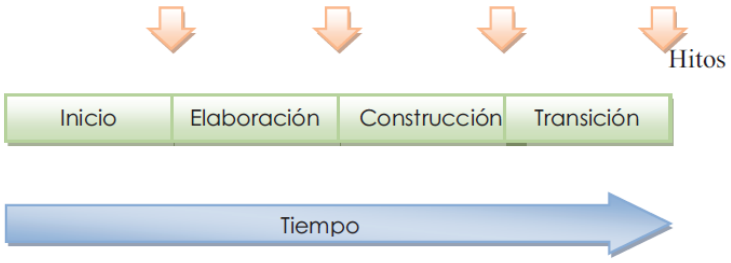


Figura 9: Fases e Hitos de un Proyecto
Fuente: (Jacobson, Booch, & Rumbaugh, 2000)

Las fases que constituyen el ciclo de vida RUP son denominadas:

- **Inicio.-** Es establecer un caso de negocio e identificar todas las entidades externas (personas y sistemas) que interactuarán con el sistema y definir estas interacciones, ya que esta información permite evaluar la aportación que el sistema hace al negocio.
- **Elaboración.-** Es desarrollar una comprensión del dominio de problema, establecer un marco de trabajo arquitectónico para el sistema, desarrollar el plan del proyecto e identificar los riesgos del proyecto, al final de la fase se tendrá un modelo de los requerimientos del sistema (se detallan casos de uso UML), una descripción arquitectónica y un plan de desarrollo del software.
- **Construcción.-** Concierne el diseño del sistema, programación y pruebas, en el lapso de la fase se desarrollan e integran las partes del sistema, al final de la fase se debe tener un sistema software operativo y la documentación para entregar a los usuarios.
- **Transición.-** Se ocupa de mover el sistema desde la comunidad de desarrollo hacia la del usuario y hacerlo trabajar en un entorno real, pero se deja de lado en la mayor parte de los modelos de procesos del software ya que es una actividad de alto costo y a veces problemática y al final, se debe tener un sistema software documentado que funciona bien en su entorno operativo.

- **Etapas del RUP**

Cuenta con dos fases para su desarrollo las cuales son:

- **Etapa de Ingeniería-** Agrupa las fases de Inicio y Elaboración, tiene como objetivo la conceptualización del sistema, el diseño inicial de solución por lo que se inicia el proceso con la administración de requerimientos, la Identificación y especificación de casos de uso, y se asegura la calidad del proyecto mediante casos de prueba e identificación de los riesgos y se determina su plan de manejo para determinar en qué orden y en que iteraciones se desarrollarán los artefactos de software, y se identifica los recursos necesarios económicos y humanos.

- **Etapa de Producción.-** Se realiza un proceso de refinamiento en las estimaciones de tiempo y recursos para las fases de construcción y transición, se determina un plan de mantenimiento para los productos entregados en la etapa anterior y se implementan los casos de uso que falta y se entrega el producto al cliente garantizando la capacitación y el soporte a los usuarios.

- **Iteraciones de Fase**

Cada fase se encuentra dividida en una serie de iteraciones, ofreciendo como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo.

Las disciplinas son un conjunto de actividades relacionadas (flujo de trabajo o vistas estáticas); que se encuentran vinculadas en un área específica dentro del proyecto que se pretende desarrollar, en la figura 10, se especifica estas disciplinas.

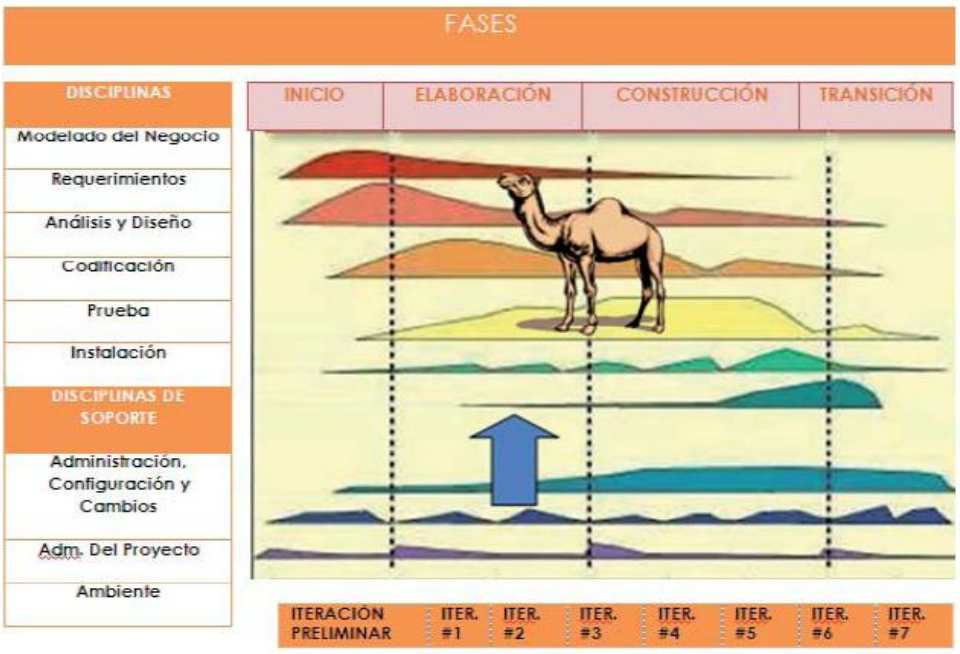


Figura 10: Diagrama de Disciplinas
Fuente: (Sommerville, 2005)

En la figura 11, se detalla como las iteraciones establecidas a su vez se desarrollan en un conjunto de disciplinas o flujos de trabajo.

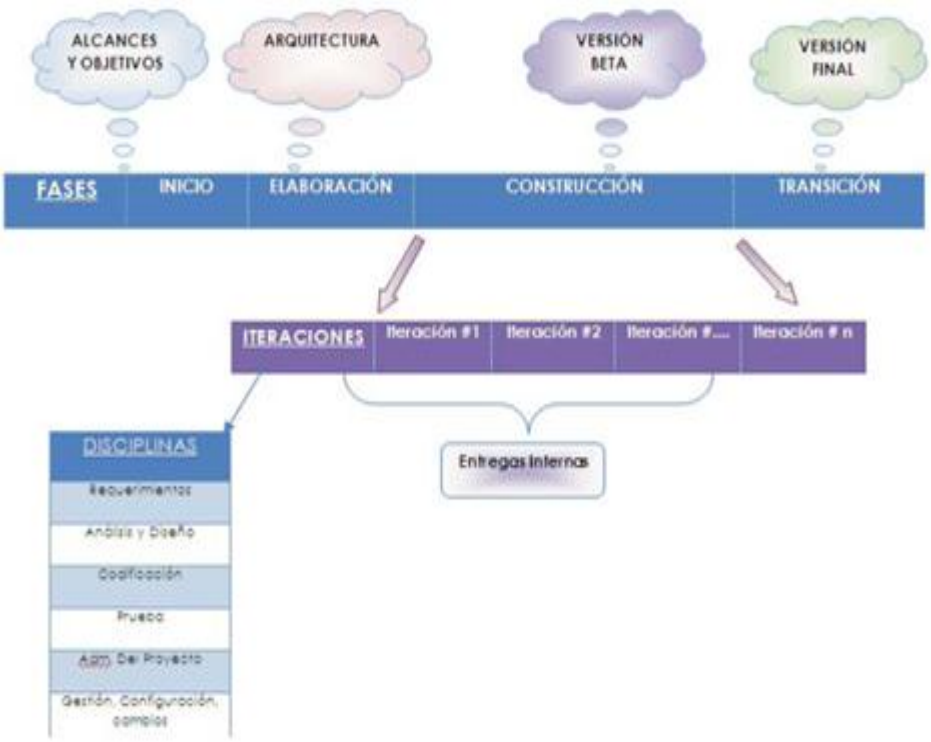


Figura 11: Diagrama de Fases, iteraciones y Disciplina

Fuente: (Jacobson, Booch, & Rumbaugh, 2000)

Dentro de cada iteración, las tareas se categorizan en disciplinas:

Tabla 4

Flujo de Trabajo estáticos de RUP

Disciplinas o Flujos de Trabajo	Responsabilidad
Modelado del Negocio	Los procesos del negocio se modelan utilizando casos de uso del negocio.
Requerimientos	Se definen los actores que interactúan con el sistema y se desarrollan caso de uso para modelar los requerimientos del sistema.
Análisis y Diseño	Se crea y documenta un modelo del diseño utilizando modelos arquitectónicos, modelos de componentes, modelos de objetos y modelos de secuencias.
Implementación	Se implementan y estructuran en subsistemas los componentes del sistema. La generación automática de código de los modelos del diseño ayuda a acelerar este proceso.

Continua

Pruebas	Son un proceso iterativo se ejecutan conjuntamente con la implementación. A la finalización de la implementación tiene lugar las pruebas del sistema.
Despliegue	Se crea un reléase del producto, se distribuye a los usuarios y se instala en su lugar de trabajo.
Configuración y cambio gestión	Este flujo de trabajo de soporte gestiona los cambios del sistema.
Gestión de Proyecto	Este flujo de trabajo de soporte gestiona el desarrollo del sistema.
Entorno	Este flujo de trabajo se refiere a hacer herramientas software apropiadas disponibles para los equipos de desarrollo de software.

Fuente: (Sommerville, 2005)

En la presente tabla 4, se especifica las disciplinas del flujo de trabajo del proceso y el flujo de trabajo del soporte de RUP.

b.3. Roles que se cumplen en el RUP

Un rol define comportamiento y responsabilidades de un individuo, o de un grupo de individuos trabajando juntos como un equipo. (Bermeo, 2010)

- Analistas: Analista de procesos de negocio, Diseñador del negocio, Analista de sistema, Especificador de requerimientos.
- Desarrolladores: Arquitecto de software, Diseñador de interfaz de usuario, de cápsulas y de base de datos, Implementador, Integrador.
- Gestores: Jefe de proyecto, de control de cambios, de configuración, de pruebas y de despliegue, Ingeniero de procesos, Revisor de gestión del proyecto, Gestor de pruebas.
- Apoyo: Documentador técnico, Administrador de sistema, Especialista en herramientas, Desarrollador de cursos, Artista gráfico
- Especialista en pruebas: Especialista, Analista y Diseñador.
- Otros roles: Stakeholders, Revisor, Coordinación de revisiones, Revisor técnico.

2.3.4. Metodologías Ágiles

Permiten el desarrollo rápido de sistemas, ya que específicamente implica al usuario en el equipo de trabajo, donde los requerimientos del sistema

cambian rápidamente durante el proceso de desarrollo, y no se utilizan para desarrollo de sistemas críticos. (Sommerville, 2005)

a. Programación Extrema (“*XP*”)

Es el método ágil más conocido y ampliamente utilizado, fue recalcado por Kent Beck ya que fue desarrollado utilizando buenas prácticas reconocidas, para el desarrollo de software basado en el modelo iterativo e incremental, ya que como primer paso el equipo de desarrollo determinan las diferentes características que va a tener el producto, informa al cliente el tiempo y el costo que tardara en desarrollar aquella característica, ya que todos los requerimientos se expresan como escenarios los cuales se implementan como una serie de tareas. (Schach, 2006)

XP promueve el trabajo en equipo y es adecuado para proyectos con requerimientos imprecisos y muy cambiantes.

La siguiente figura 12, se detalla el proyecto de programación extrema:

Figura 12: Flujo de Proceso del proyecto XP
Fuente: (González, 2008)

a.1. Ciclo de Vida de XP

Enfatiza en el interactivo e incremental del desarrollo, ya que la iteración de desarrollo es un período de tiempo en el que se realiza un conjunto de funcionalidades, más retroalimentación una mejor calidad del producto.

Se describen las fases en las que se subdivide el ciclo de vida XP:

- **Fase de la exploración.-** Los clientes plantean rasgos de las historias de usuario para utilizar en la primera entrega del producto, y el equipo de desarrollo se familiariza con herramientas, tecnologías.
- **Fase del Planeamiento.-** Se priorizan las historias del usuario y se define el alcance del reléase y los programadores estiman el esfuerzo que requieren y de allí se define el primer reléase que no excede normalmente de dos meses, se deben incluir varias iteraciones para lograr un reléase y el cliente decide las historias para cada iteración y las pruebas funcionales las cuales se ejecutan al final de cada iteración, y en la última iteración el sistema está listo para producción.
- **Fase de Producción.-** Requiere pruebas y comprobación extra del funcionamiento del sistema antes de que éste se pueda liberar al cliente y en esta fase, los nuevos cambios son todavía encontrados y debe tomarse la decisión de si se incluyen o no en el release actual.
- **Fase de Mantenimiento.-** Requiere de un mayor esfuerzo para satisfacer también las tareas del cliente y la velocidad del desarrollo puede desacelerar después de que el sistema esté en la producción.
- **Fase de Muerte. -**Es cuando el cliente no tiene más historias para ser incluidas en el sistema y se han satisfecho sus necesidades, se ha cubierto aspectos como rendimiento y confiabilidad del sistema y se genera la documentación final, y no se realizan más cambios en la arquitectura o a su vez ocurre cuando el sistema no genera los beneficios esperados por el cliente o no hay dinero para mantenerlo.

La siguiente figura 13, muestra las fases en las que se subdivide el ciclo de vida Xp:

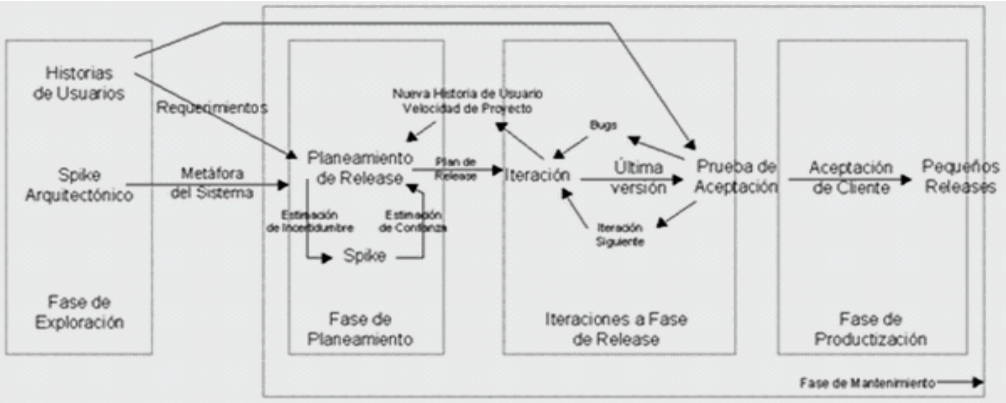


Figura 13: Ciclo de vida de eXtreme Programming
Fuente: (Anaya, s.f.)

a.2. Roles y Responsabilidades

Para implementar el proceso de desarrollo XP, las distintas tareas deben ser cubiertas por diferentes tipos de personas, por lo que a continuación se presentan los roles y responsabilidades del equipo de trabajo.

Tabla 5

Roles y responsabilidades XP

Rol	Responsabilidad
Desarrollador (Programmer)	Escribe las pruebas unitarias y es el encargado de producir el código del sistema de la forma más simple y definida que sea posible.
Cliente (Customer)	Escribe las historias de usuario y las pruebas funcionales para validar su implementación.
Encargado de las pruebas (Tester)	Ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas, difunde los resultados en el equipo y es responsable de las herramientas de soporte para las pruebas.
Encargado de seguimiento (Tracker)	Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, para mejorar futuras estimaciones.
Entrenador (Coach)	Es el responsable del proceso global. Debe proveer guías al equipo de forma tal que se apliquen las prácticas de la programación extrema y se siga el proceso correctamente.
Consultor	Miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto, en el que puedan surgir problemas. El consultor es quien guía al equipo en la resolución de problemas específicos.
Gestor (Big Boss)	Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es la de coordinación.

Fuente: (Canós, Letelier, & Penadés)

a.3. Practicas Básicas de XP

Las tecnologías ayudan en el desarrollo y a la aplicación de lo siguiente.

- **Juego de Planificación**.- Hay una comunicación entre el cliente y programadores, el equipo técnico define una estimación de la implementación y los clientes dicen el tiempo de cada iteración.
- **Versiones Pequeñas.-** Un sistema simple se pone rápidamente en producción y se producen nuevas versiones agregando aquellas funciones consideradas valiosas para el cliente.
- **Metáfora del Sistema.-** Cada proyecto es guiado por una historia simple de cómo funciona el sistema en general, reemplaza a la arquitectura y debe estar en lenguaje común, entendible para todos (Cliente y Desarrolladores), esta puede cambiar permanentemente.
- **Diseño Simple.-** Se diseña la solución más simple para que funcione y sea implementada en el proyecto.
- **Pruebas Continuas.-** Los casos de prueba se escriben antes que el código y los desarrolladores escriben pruebas unitarias y los clientes especifican pruebas funcionales.
- **Refactorización.-** Es una actividad constante de restructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible en posteriores cambios.
- **Programación por parejas.-** El código es escrito por dos personas trabajando en el mismo computador y conlleva ventajas (menor tasa de errores, mejor diseño).
- **Posesión Colectiva del Código.-** Cualquier programador puede cambiar cualquier parte del sistema en cualquier momento y siempre se utilizan estándares.
- **Integración continua.-** Los cambios se integran en el código base varias veces por día y todos los casos de prueba se deben pasar antes y después de la integración.
- **Semana laboral de 40 horas.-** Cada Trabajador trabaja no más de 40 Horas por semana.

- **Cliente en el Sitio.-** El equipo de desarrollo tiene acceso todo el tiempo al cliente, el cual está disponible para responder preguntas, fijar prioridades, etc.
- **Estándares de codificación.-** Todo el código debe estar escrito de acuerdo a un estándar de codificación.

El mayor beneficio de las practicas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras, en la figura 14, esto se ilustra ya que donde una línea entre dos practicas significa que las dos practicas se refuerzan entre sí, por lo que el mérito de XP es integrarlas de una forma efectiva y complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

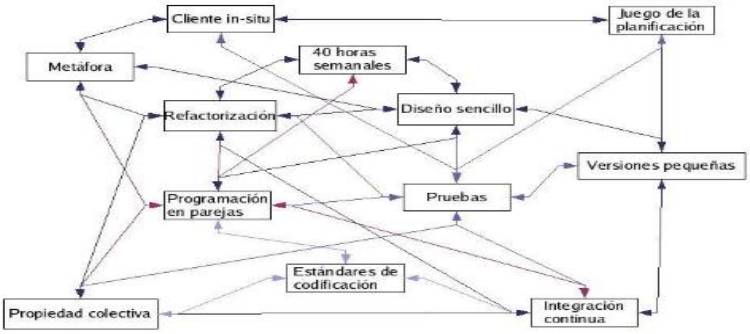


Figura 14: Las practicas Básicas de XP

Fuente: (Canós, Letelier, & Penadés)

b. SCRUM

Scrum ha sido usado para gestionar el desarrollo de productos complejos desde principios de los años 90, ya que no es un proceso o una técnica para construir productos sino un marco de trabajo dentro del cual se pueden emplear varias técnicas y procesos. (Schwaber & Sutherland, 2013)

b.1. Roles

- **El Equipo Scrum**.- Consiste en el dueño del producto, el equipo de desarrollo y un facilitador, los Equipos Scrum son multifuncionales y auto organizado, y no son dirigidos por personas externas al equipo y tienen su grupo de trabajo entre 5 y 9 personas.

- **El Dueño del Producto.-** Representa la voz del cliente define que necesita y desea, es responsable de gestionar la lista del producto y toda la organización debe respetar sus decisiones.
- **El Facilitador.-** Se sitúa como protección del equipo, a modo de barrera ante los obstáculos, no es el líder del equipo pero mantiene los procesos y trabaja de forma similar al director de proyecto.
- **El Equipo de Desarrollo.-** Son profesionales que desempeñan el trabajo de entregar un Incremento del producto "Terminado", al final de cada sprint y participan en la creación del Incremento.

b.2. Eventos de Scrum o Reuniones

En el desarrollo del producto, se lleva a cabo varias reuniones, tanto a nivel de Sprint, como diariamente. El Sprint es el corazón de Scrum es un bloque de tiempo (time-box) de un mes o menos durante el cual se crea un incremento de producto terminado. (Schwaber & Sutherland, 2013, pág. 8), y las reuniones de carácter diario son:

- **Daily Scrum.-** En esta reunión diaria se tratan aspectos relacionados con el estado actual del Sprint en curso del proyecto y la duración suele ser de 15 minutos, independientemente del tamaño del equipo.
- **Scrum de Scrums.-** Se usa cuando los equipos son extensos y se forman grupos de personas que forman equipos individuales.
- **Planificación del Sprint.-** Se realiza antes de empezar con el siguiente Sprint o iteración (cada 15 o 30 días) y se divide en dos partes, ambas con una duración máxima de 4 horas:

    - **Selección de los Requerimientos.-** El equipo recoge los requerimientos priorizados del cliente y se reflejan en el proyecto a crear.
    - **Planificación de la Iteración.-** Una vez obtenidos los requerimientos priorizados, se planifican las tareas que son necesarias desarrollar para el cumplimiento de esos requerimientos.

- **Revisión del Sprint.-** Es una duración máxima de 4 horas, se realiza al finalizar el proceso de desarrollo del Sprint, y en ella el equipo se encarga de transmitir al cliente la finalización de la iteración y le presenta los requerimientos completados, mediante la entrega del incremento funcional del producto (o "demo"). Así, el cliente puede evaluar la iteración y plantear los cambios que considere necesarios, re-planificando de esta manera el proyecto desde la primera iteración.
- **Retrospectiva del Sprint. -** Duración máxima de 4 horas, se realiza después de la revisión del Sprint, el equipo analiza su manera de trabajar en la anterior iteración, y plantea cambios para mejorar el rendimiento del desarrollo del producto.

b.3. Artefactos de Scrum

Es la representación del trabajo o valor en diversas formas, tanto a nivel de proyecto, como de Sprint. (Martínez Cobo, s.f.)

Los más importantes son:

- **Lista de Objetivos/ Requerimientos Priorizada.-** Este documento representa lo que el cliente espera del proyecto ya sea a objetivos y requerimientos, así como entregas o Sprints y detalle de riesgos. El cliente se encarga de crear y gestionar esta lista, con ayuda de un facilitador.
- **Lista de Tareas de la Iteración.-** Refleja la totalidad de las tareas de iteración o Sprint en curso, con el fin de cumplir los objetivos o requerimientos, y entregar algo funcional al cliente acorde con lo esperado.
- **Gráficos de Trabajo Pendiente.**- Reflejan la velocidad con la que avanza el proyecto, permitiendo una vista general de la rapidez con la que el proyecto en general o el Sprint en curso en particular están avanzando.

En la Figura 15, se presenta los aspectos más importantes de SCRUM como roles, eventos y artefactos mediante un flujo de procesos.

Figura 15: Flujo de Proceso de SCRUM
Fuente: (Universidad Union Bolivariana, 2013)

2.4. Ingeniería de Requerimientos

Es el proceso de recopilar, analizar y verificar las necesidades del cliente para un sistema de software se le denomina Ingeniería de Requerimientos, ya que la finalidad es la poder entregar una especificación de requerimientos de software correcta y completa, y permite orientar de mejor manera la forma de comprender y definir sistemas de software complejo. (Sommerville, 2005)

A su vez es un campo de reconocida importancia en el ámbito teórico y práctico de la ingeniería del software, las mayores deficiencias en el proceso de desarrollo de software se encuentran en sus primeras fases. (Ferraro, s.f.)

Abstracción.- Permite facilitar la captación y modelado de los aspectos del problema lo más cercano posible a los conceptos del dominio del problema, con la finalidad de comprender antes de construir.

Declarativo.- Permite postergar decisiones de implementación.

Manifiestan a la ingeniería de requerimientos como el proceso de aplicar un método de análisis estructurado, como el orientado a objetos, ya que algunos problemas surgen en el proceso como resultado de una mala descripción entre los requerimientos de usuario y del sistema. (Sommerville, 2005, pág. 145)

2.4.1. Requerimientos del Usuario

Para un sistema deben describir los requerimientos funcionales y no funcionales de tal forma que sean comprensibles por los usuarios del sistema

sin conocimiento técnico detallado, por lo que si se redacta requerimientos del usuario, no se debe utilizar jerga del software, notaciones estructuradas y se debe utilizar un lenguaje sencillo, con tablas y formularios sencillos y diagramas intuitivos, pero puede producir problemas al redactar frases de lenguaje natural en un documento como falta de claridad, confusión de requerimientos, conjunción de requerimientos.

2.4.2. Requerimientos del Sistema

Son utilizados por los ingenieros de software como punto de partida para el diseño del sistema, y permiten agregar detalles y explicaciones de cómo debe proporcionar los requerimientos del usuario y pueden ser utilizados como parte del contrato para la implementación del sistema, en teoría los requerimientos simplemente deben describir el comportamiento externo del sistema y sus restricciones operativas y no deben tratar de cómo se debe diseñar o implementar el sistema, y poseen los funcionales y no funcionales.

a. Requerimientos Funcionales

Describen servicios o funciones que el sistema debe proporcionar, los cuales están relacionados entre datos de entrada y de salida que debe presentar el sistema, y la completitud significa que todos los servicios solicitados por el usuario deben estar definidos y evitar que los requerimientos tengan definiciones contradictorias, ya que es fácil cometer errores y omisiones cuando se redactan especificaciones para sistemas grandes y complejos y otra es que los stakeholders del sistema tienen necesidades diferentes, y a menudo contradictorias por lo que es posible que los problemas surjan solamente después de un análisis más profundo.

b. Requerimientos no funcionales

Son aquellos requerimientos que hacen referencia a las propiedades emergentes de sistema como es la fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento, estos requerimientos especifican o restringen las propiedades emergentes del sistema.

Los requerimientos no sólo se refieren al sistema de software a desarrollar, sino que pueden restringir el proceso que se debe utilizar para desarrollar el sistema como la especificación de los estándares de calidad que se deben utilizar en el proceso, una especificación que el diseño debe producir con una herramienta CASE, ya que cabe decir que surgen de las necesidades del usuario, debido a las restricciones en el presupuesto, a las políticas de la organización, a la necesidad de interoperabilidad con otros sistemas software o hardware, o a factores externos como regulaciones de seguridad o legislaciones sobre privacidad.

2.4.3. Procesos de Ingeniería de Requerimientos

Tiene la finalidad de crear y mantener un documento de requerimientos del sistema, y realiza la evaluación de si el sistema es útil para el negocio (estudio de viabilidad), el descubrimiento de requerimientos (obtención y análisis), la transformación de estos requerimientos en formularios estándar (especificación), y la verificación de que los requerimientos realmente definen el sistema que quiere el cliente (validación), sin embargo en casi todos los sistemas los requerimientos cambian, y todo esto ayuda a las personas involucradas a desarrollar y a una obtener correctamente los requerimientos y comprensión de lo que quieren que haga el software.

La figura 16, se muestra el esquema de los procesos de ingeniería de requerimientos y la relación entre estas actividades, y también muestra el documento que se elabora en cada etapa del proceso.

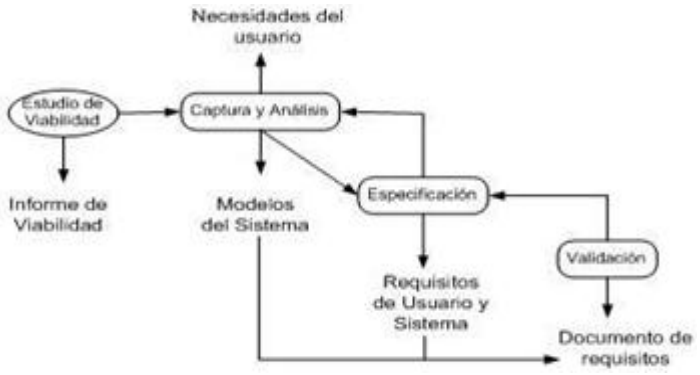


Figura 16: El proceso de Ingeniería de Requerimientos.
Fuente: (Sommerville, 2005)

- **Estudio de viabilidad.-** Permite rendir un informe al equipo de desarrollo del proyecto y al usuario, y se verifica si el proyecto vale la pena desarrollarlo, y es importante para los objetivos del negocio.
- **Captura y Análisis.-** El equipo de desarrollo entra en contacto con el usuario final para determinar el alcance del proyecto a construir, e identifica los servicios que prestará el sistema, su rendimiento, necesidades y restricciones, y cuáles son los objetivos esperados.
- **Especificación.-** Se obtiene un documento de especificación de requerimientos, en cual se llega a definir de una forma completa, precisa y verificable cada uno de los requerimientos o necesidades que debe satisfacer el sistema a desarrollar, además de sus respectivas restricciones de software, hardware.
- **Validación.-** Consiste en mostrar o comprobar que cada uno de los requerimientos obtenidos definen el sistema o proyecto que se va a construir y que desea el cliente. En esta etapa solamente entran aquellos requerimientos que se mencionaron ya en la especificación.
- **Gestión.-** Se realiza la comprensión y control de los cambios de cada una de los requerimientos, sean estos estables para el estado del sistema, o volátiles para eventos en donde el sistema realice una función dada.

2.4.4. Modelos de Diseño

Muestran los objetos o clases en un sistema y poseen diferentes tipos de relaciones entre entidades y son el puente entre los requerimientos y la implementación del sistema, son abstractos para que el detalle innecesario no oculte las relaciones entre ellos y los requerimientos del sistema.

Incluyen detalles para que los programadores tomen las decisiones de implementación ya que un diseño específico se construye durante la implementación del sistema, y entre los aspectos importantes es decidir qué modelos de diseño son necesarios y el nivel de detalle de estos modelos.

Existen 2 tipos de modelos para describir un diseño orientado a objetos:

a. Modelos Estáticos

Describen la estructura estática del sistema en términos de clases del sistema, relaciones y detallan situaciones que no involucra el tiempo.

b. Modelos Dinámicos

Describen la estructura dinámica del sistema y detalla como progresa el sistema, muestra las interacciones entre objetos del sistema y lo que se documenta, incluyen la secuencia de servicios solicitados por los objetos y la forma en que el estado del sistema se relaciona con estas interacciones de objetos a lo largo del tiempo, y para lo cual UML provee diversos modelos para el documento de diseño.

2.4.5. Lenguaje de Modelado Unificado UML

UML es la notación para el desarrollo orientado a objetos, ya que es un lenguaje de modelado visual y utilizado para especificar, visualizar, construir documentar los artefactos, y permite capturar decisiones sobre los sistemas que van a ser construidos y no es un modelo de desarrollo y está respaldado por el OMG.

Tabla 6

Modelos UML

		Análisis	Diseño	Implementación
Estático	Caso de Uso			
	Conceptual			
	Clase			
	Componentes			
	Distribución			
Dinámico	Secuencia			
	Actividad			
	Estado			

Fuente: (Gamboa Rodríguez, 2012)

En la tabla 6, se representan los diferentes tipos de diagramas de acuerdo a los modelos estáticos y dinámicos.

A continuación se describe los modelos especificados:

a. Modelo de Casos de Uso

Según (Booch, Jacobson, & Rumbaugh, 2007, pág. 69), Permiten capturar el comportamiento de un sistema, subsistema, clase o componente tal y como se muestra al usuario externo, y permite definir los límites, relaciones del sistema y el entorno, basado en lenguaje natural e interviene en todo el ciclo de vida, y debe ser simple, claro y conciso de entender, en la figura 17, se especifica los elementos más importantes.

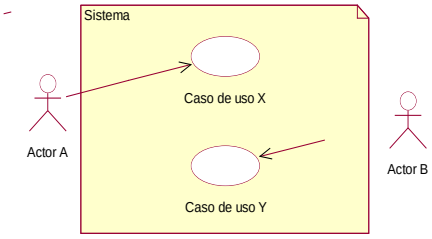
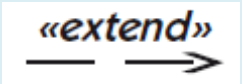
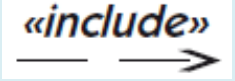


Figura 17: Diagrama de Caso de uso
Fuente: (Letelier Torres & Sánchez Palma, 2002)

La descripción de un gran caso de uso se puede descomponer en otros más sencillos y a su vez puede participar en varias relaciones, además de la asociación con los actores.

Tabla 7
Tipos de relaciones de caso de uso

Relación	Función	Notación
Asociación	La línea de comunicación entre un actor y un caso de uso en el que participa.	
Extensión	La inserción de comportamiento adicional en un caso de uso base que no tiene conocimiento sobre él.	
Generalización de casos de uso	Una relación entre un caso de uso general y un específico que hereda y le añade propiedades.	
Inclusión	La inserción de comportamiento adicional en un caso de uso base que describe explícitamente la inserción.	

Fuente: (Booch, Jacobson, & Rumbaugh, 2007)

En la tabla 7, se especifica las relaciones, la función y la notación que posee el modelo de casos de uso de UML.

b. Modelo Conceptual

Es usado en la fase de análisis para determinar cuál es el dominio de la aplicación lo que permite ver cuáles son los conceptos más importantes de la aplicación, y las funciones que este debe ejecutar ya que ayuda bastante al momento de crear el diagrama de clases, en términos de la Programación Orientada a Objetos, es un objeto del mundo real que representa cosas del mundo real y NO de componentes de software, ya que no se definen operaciones o métodos; en este modelo se pueden mostrar los conceptos, sus atributos y la relación o asociación entre ellos, y se definiría que un concepto es una idea, cosa u objeto el cual permite la descripción del sistema, de los requerimientos y de los Casos de Uso, en la figura 18, se representa los elementos más importantes del modelo.

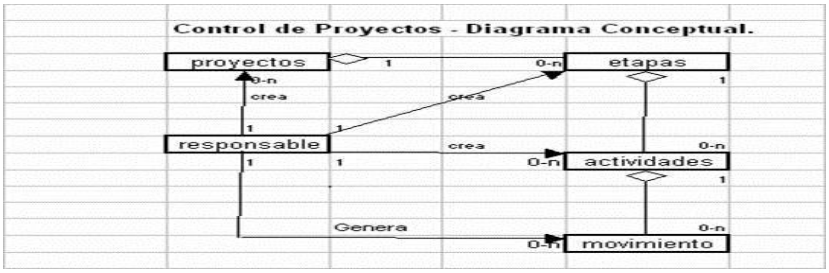


Figura 18: Diagrama Conceptual
Fuente: (Canchala, s.f.)

c. Los Modelos Subsistemas o de Clases

El diagrama muestra la agrupación lógica de los objetos y es usado en la fase de diseño para modelar las clases que se va implementar con los métodos, atributos y eventos, y se visualiza las relaciones entre las clases que involucran el sistema, las cuales pueden ser asociativas, de herencia, y de uso, permitiendo describir los tipos de objetos que hay en el sistema.

Un diagrama de clases está compuesto por los siguientes elementos:

- Clase: atributos, métodos y visibilidad.
- Relaciones: Herencia, Composición, Agregación, Asociación y Uso.

En la figura 19, está definido una clase y representado con 3 compartimientos: el nombre de la clase, atributos y las operaciones.

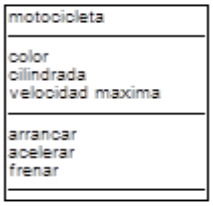


Figura 19: Representación de una clase en UML
Fuente: (Letelier Torres & Sánchez Palma, 2002)

La Encapsulación permite proteger los datos de accesos indebidos, por lo cual posee diversos niveles de encapsulamiento como son:

- **(-)** Privado.- Esta parte es totalmente invisible entre las clases.
- **(#)** Protegidos.- Los atributos y operaciones protegidos están visibles para las clases.
- **(+)** Públicos.- Los atributos yoperaciones públicas son visibles a otras clases.

d. Modelo de Componentes

Describen los elementos físicos del sistema y sus relaciones, asignan la vista lógica de las clases del proyecto a los archivos que contienen el código fuente y ayudan a establecer relaciones entre cada diagrama de clases y los archivos de código fuente, y representa todos los tipos de elementos del software que entran en la fabricación de aplicaciones informáticas.

Las relaciones de dependencia se utilizan en los diagramas de componentes para indicar que un componente utiliza los servicios ofrecidos por otro componente, y pueden agruparse en paquetes, en la Figura 20, se representa que cada clase del modelo lógico se realiza en dos componentes: la especificación y el cuerpo.

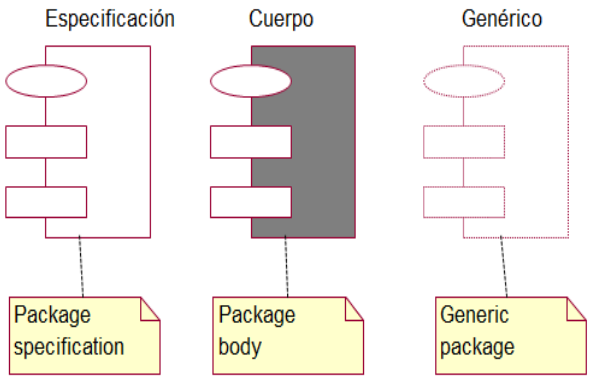


Figura 20: Representación del Modelo de componentes
Fuente: (Letelier Torres & Sánchez Palma, 2002)

e. Modelo de Distribución

Permiten mostrar la disposición física de los distintos nodos que componen un sistema y el reparto de los componentes sobre dichos nodos y a su vez los estereotipos permiten precisar la naturaleza del equipo como: dispositivos, procesadores y memoria.

En la Figura 21, se representa como los nodos se interconectan mediante soportes bidireccionales (en principio) que pueden a su vez estereotiparse.

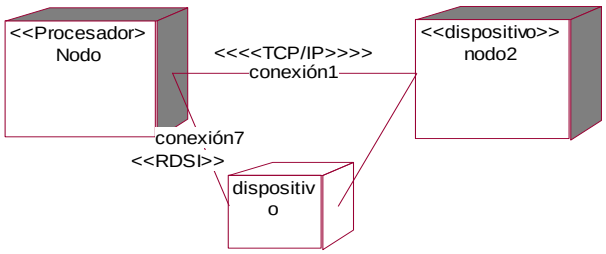


Figura 21: Representación del conexión entre nodos
Fuente: (Letelier Torres & Sánchez Palma, 2002)

f. Modelo de Secuencia

Permite mostrar la secuencia de interacciones de los objetos lo cual permite determinar cómo se comunican los objetos en una interacción, y permite especificar la ejecución, del suceso, fragmento de la interacción, línea de vida, la secuencia de mensajes entre objetos durante un escenario concreto y cada objeto viene dado por una barra vertical, y permite especificar el tiempo el cual transcurre de arriba abajo.

El modelo es útil en la fase exploratoria para identificar objetos y la distribución de los objetos y permite observar adecuadamente la interacción de un objeto con respecto de los demás, por lo que la estructura estática viene dada por los enlaces; la dinámica por el envío de mensajes por los enlaces, en la figura 22, se representa como está estructurado el modelo.

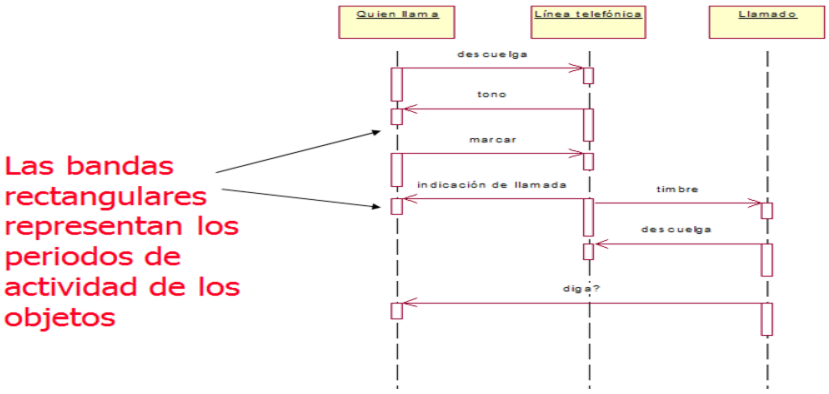


Figura 22: Representación del modelo de Secuencia
Fuente: (Letelier Torres & Sánchez Palma, 2002)

g. Modelo de Actividad

Permite representar los flujos de trabajo paso a paso de un negocio y los componentes operacionales de un sistema, es una variante de los diagramas de Estados, organizado respecto de las acciones y principalmente destinado a representar el comportamiento interno de un método, de un caso de uso o de un proceso de negocio.

El diagrama de actividad es una forma especial de diagrama de estado usado para modelar una secuencia de acciones y condiciones tomadas dentro de un proceso, en la Figura 23, especifica mediante un ejemplo el modelo de actividad.

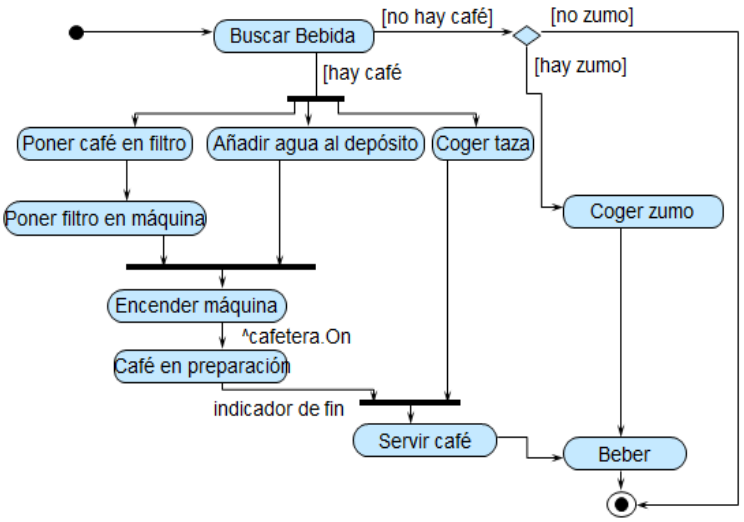


Figura 23: Representación del modelo de Actividad
Fuente: (Letelier Torres & Sánchez Palma, 2002)

h. Modelo de Estado

Muestra cómo los objetos individuales cambian su estado en respuesta a los eventos, permiten describir el comportamiento de un sistema, describe todos los estados posibles en los que puede estar un objeto a lo largo de su ciclo de vida, y se utiliza para verificar los atributos que se ha designado en el diagrama de clases y también son utilizados en los métodos u operaciones para determinar si son los correctos, y sus elementos son:

- **Estados**.- Influye en el comportamiento y evolución del sistema, y pertenece a una clase, y está formado por un Inicio, fin y estado.
- **Eventos**.- Son aquellos que dan lugar a un cambio en el comportamiento del sistema o a un momento significativo en su evolución, por ejemplo un método de una clase.
- **Transiciones**.- Son las líneas de comunicación, lo que une un estado con otro.

En la figura 24, se representa todos los elementos detallados anteriormente mediante un ejemplo:

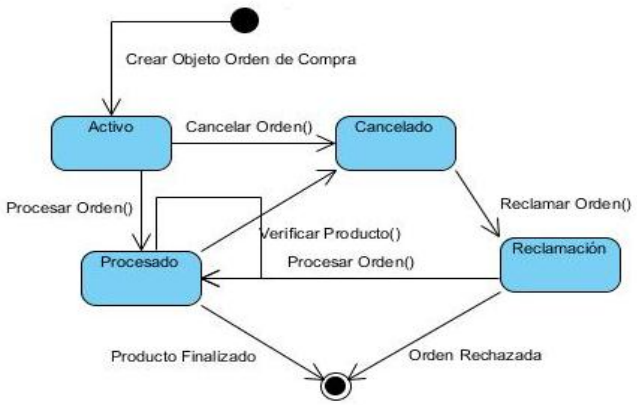


Figura 24: Representación del modelo de Estado
Fuente: (Letelier Torres & Sánchez Palma, 2002)

2.4.6. La Calidad del Software

La calidad del software es el grado con el que un sistema, componente o proceso cumple con los requerimientos especificados y las necesidades o expectativas del cliente o usuario y también con concordancia del software producido con los requerimientos explícitamente establecidos, con los estándares de desarrollo prefijados. (Buades Rubio, 2003)

Cabe recalcar que el software se desarrolla, no se fabrica en el sentido clásico y es inmaterial, y no se deteriora con el uso o el tiempo aunque tiene un ciclo de vida; su fiabilidad es difícil de comprobar; la mayoría del software se construye a medida y necesita de actualización permanente y es dependiente del entorno donde se ejecuta.

La gestión de calidad del software se estructura en tres actividades principales: (Sommerville, 2005)

- **Garantía de la calidad.-** El establecimiento de un marco de trabajo de procedimientos y estándares organizacionales que conduce a software de alta calidad.
- **Planificación de la calidad.-** La selección de procedimientos y estándares adecuados a partir de este marco de trabajo y la adaptación de éstos para un proyecto software específico.

- **Control de la calidad.-** La definición y fomento de los procesos que garanticen que los procedimientos y estándares para la calidad del proyecto son seguidos por el equipo de desarrollo de software.

a. Garantía de la Calidad

Es el proceso que define cómo lograr la calidad del software y cómo la organización de desarrollo conoce el nivel de calidad requerido en el software, el proceso de calidad se ocupa ante todo de definir o seleccionar los estándares que deben de ser aplicados al proceso de desarrollo software o al producto software.

Se puede definir dos tipos de estándares como parte del proceso de garantía de calidad:

- **Estándares de producto**.- Se los aplica sobre el producto software que se comienza a desarrollar, y permite incluir estándares de documentación, cabecera de comentarios estándar para definición de clases, y de codificación, y definen cómo debe utilizarse el lenguaje de programación.
- **Estándares de proceso**.- Permiten definir los procesos que deben seguirse durante el desarrollo del software y permite incluir definiciones de procesos de especificación, diseño y validación, así como una descripción de los documentos que deben escribirse en el transcurso de los procesos.

CAPÍTULO III

3. DESARROLLO DE LA METODOLOGÌA DE TRABAJO

3.1. ESTRUCTURA

Una metodología de desarrollo de software se refiere a un marco de trabajo sobre el cual se va estructurar, definir, controlar y ejecutar el ciclo de vida de un desarrollo de software, en este caso puntual, se estructurara basado en la metodología RUP, tomado como consideración su marco de trabajo pero con las particularidades del IESS, objeto de este estudio.

Por ello es importante señalar que el ciclo de vida a ser aplicado es el iterativo e incremental, tomado como base el concepto de iteraciones para el ciclo de vida de desarrollo de software.

Se ha tomado los roles establecidos en la Institución y las particularidades de gestión de la misma, por ello se van a enumerar los factores críticos que determina RUP para establecer la metodología para el IESS, a continuación se especifica los elementos de la metodología a ser diseñada.

3.1.1. Fases y Disciplinas

La figura 25, bidimensional sobre la cual se apalanca esta metodología es la definida por RUP con fases y disciplinas como lo define en el cuadro siguiente:

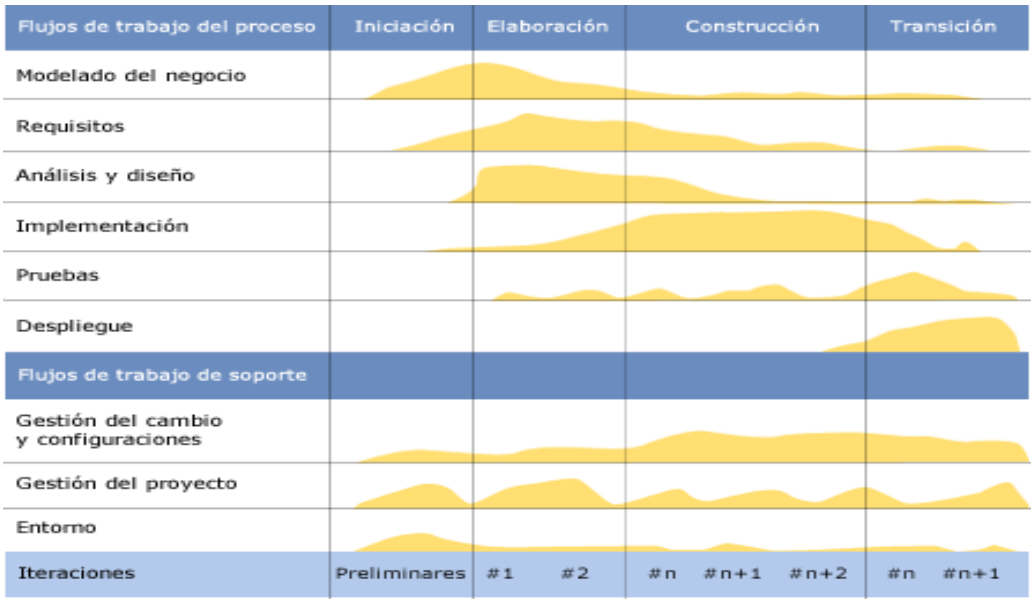


Figura 25: Metodología RUP

Fuente: (Weitzenfeld, 2005)

En base a esos aspectos se determina el diseño de la nueva metodología, en la figura 26, se especifica:

FLUJOS DE TRABAJO (Proceso)	FASES				
	Inicio	Elaboración	Construcción		Transición
Modelado del Negocio					
Requerimientos					
Análisis y Diseño					
Implementación					
Pruebas					
Despliegue					
	I1	I2	I3	I4	I5

Nota: Cuando es opcional la aplicación de la Disciplina

* Es decir cuando hay cambio de arquitectura en base a los requerimientos y los casos de uso

* Cuando se determina otros casos de uso y va hasta el inicio y final

Figura 26: Nueva Metodología de trabajo Basado en RUP

En lo que respecta a los flujos de trabajo o disciplinas de RUP solo se ha especificado netamente los que pertenecen a Procesos para el diseño de la nueva metodología ya que se ha enfocado solo para la parte de Desarrollo, por lo tanto las disciplinas de Soporte no cubren esta investigación.

A continuación se especifica los elementos de la nueva metodología diseñada.

a. Fase de Inicio

En esta fase se realizó un estudio de las necesidades en base a problemática que posee el Departamento de Riesgos de trabajo del IESS, lo cual permitió poder determinar el alcance del proyecto a ser implementado y a su vez comprende los procesos del negocio y de forma preliminar define los requerimientos.

El hito para poder pasar a la siguiente fase se determina cuando el alcance y metas a ser cumplidas con el proyecto están claras y definidas.

b. Fase de Elaboración

En esta fase se realizó un análisis y modelado de los casos de uso del Sistema en un alto porcentaje, basados en los requerimientos establecidos previamente y se definió la línea base de la arquitectura sobre la cual se trabajara para el desarrollo del aplicativo y en base a eso posteriormente se especificó la arquitectura a ser utilizada para todo el proyecto.

Y se desarrolló un prototipo ejecutable del aplicativo utilizando la arquitectura de software definida, y el hito para poder pasar a la siguiente fase se define con arquitecturas definidas inicialmente y con requerimientos funcionales iniciales ya firmados.

c. Fase de Construcción

En esta fase se diseñó el diagrama de clases y se generó un porcentaje del aplicativo mediante el desarrollo del código fuente en una primera

iteración, para su óptimo funcionamiento y ejecución debe estar integrado de forma correcta para posteriormente generar el primer plan de pruebas.

El Hito para poder pasar de esta fase es que el sistema este inicialmente listo para entregar a la comunidad de usuarios para que utilicen.

Pero para el aplicativo de Test de Evaluación se hace necesario desarrollar otra iteración ya que se requiere generar otras versiones de la arquitectura de software y de los casos de uso esto se hace necesario por tener cambios en los requerimientos y a su vez se requiere para el código fuente y el plan de pruebas, cabe especificar que el desarrollo de estos artefactos requirió otra versión por las reglas del negocio especificado.

Cabe recalcar que para generar más de una versión de los artefactos y que dentro de una fase se tenga más de una iteración es opcional, ya que esto depende del aplicativo que se esté desarrollando.

d. Fase de Transición

En esta fase se envió el aplicativo desarrollado a producción pero previamente para garantizar que el producto está listo para su entrega se ejecutó un buen plan de pruebas en su versión final, y para complementar se generó el diagrama de despliegue el cual permite modelar la topología del Hardware a ser utilizada para la ejecución del aplicativo.

Para una óptima utilización del aplicativo se genera el manual técnico y del Usuario.

El Hito para poder pasar de esta fase es el que el sistema esté listo para liberar a la comunidad de usuarios.

Sin embargo dentro del manejo del nuevo diseño se implementa la siguiente particularidad que las iteraciones se lo haría por cada una de las fases a excepción de la fase de Construcción que tienen 2 iteraciones, y las disciplinas serían como apoyo para conseguir los hitos de cada fase.

Cabe especificar que el número de iteraciones que se puede generar depende del tamaño del proyecto a desarrollar ya que cada uno es un escenario distinto, pero para dar validez a la nueva metodología de trabajo se ha especificado que para el proyecto tendrá 5 iteraciones, en las cuales se deben tomar las particularidades de cada una de las disciplinas para poder conseguir el objetivo que son los hitos de cada fase.

Esta modificación de cambio se lo hace debido a las siguientes particularidades:

1. Se tomaron muchos proyectos del IESS, y en el cual se analizó que todos los proyectos tienen como particularidad que para ser definidos como tal deben tener especificados el alcance, el cronograma de trabajo y los recursos asignados.
2. Los interesados del negocio, autoridades de cada uno de las unidades de negocio, solicitan se genere una iteración al proyecto, para determinar el alcance y tiempos, así como los costos a ser utilizados en el desarrollo de este proyecto, para poder continuar con el mismo, según resolución de las autoridades el tiempo dado para esta generación es de 30 días calendario.
3. Otra de las características identificadas en los proyectos analizados se pudo analizar que la definición de la Arquitectura del proyecto, se lo define de forma completa en la segunda iteración, pues solo una vez definida la misma, se puede proceder a elaborar la planificación del desarrollo y puesta en producción.
4. En el desarrollo del aplicativo, en la fase de construcción se elaboran las iteraciones necesarias hasta solventar todo los requerimientos funcionales, y su paso definitivo a producción.

3.1.2. Flujos de trabajo de la nueva Metodología

Se especifica la descripción de las 6 disciplinas que posee la nueva metodología de trabajo.

a. Modelado del Negocio

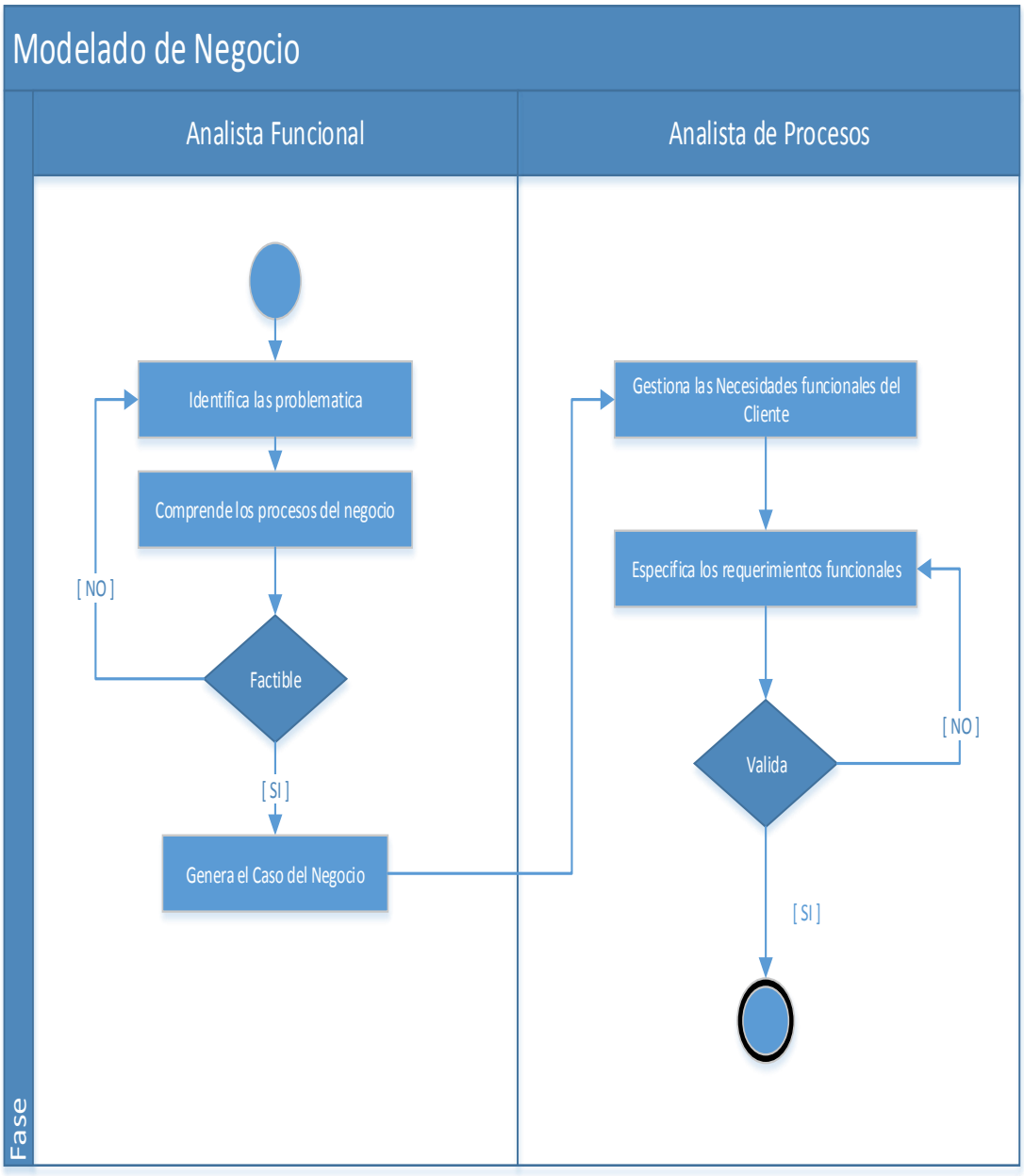


Figura 27: Flujo de Trabajo de Modelado del Negocio

b. Requerimientos

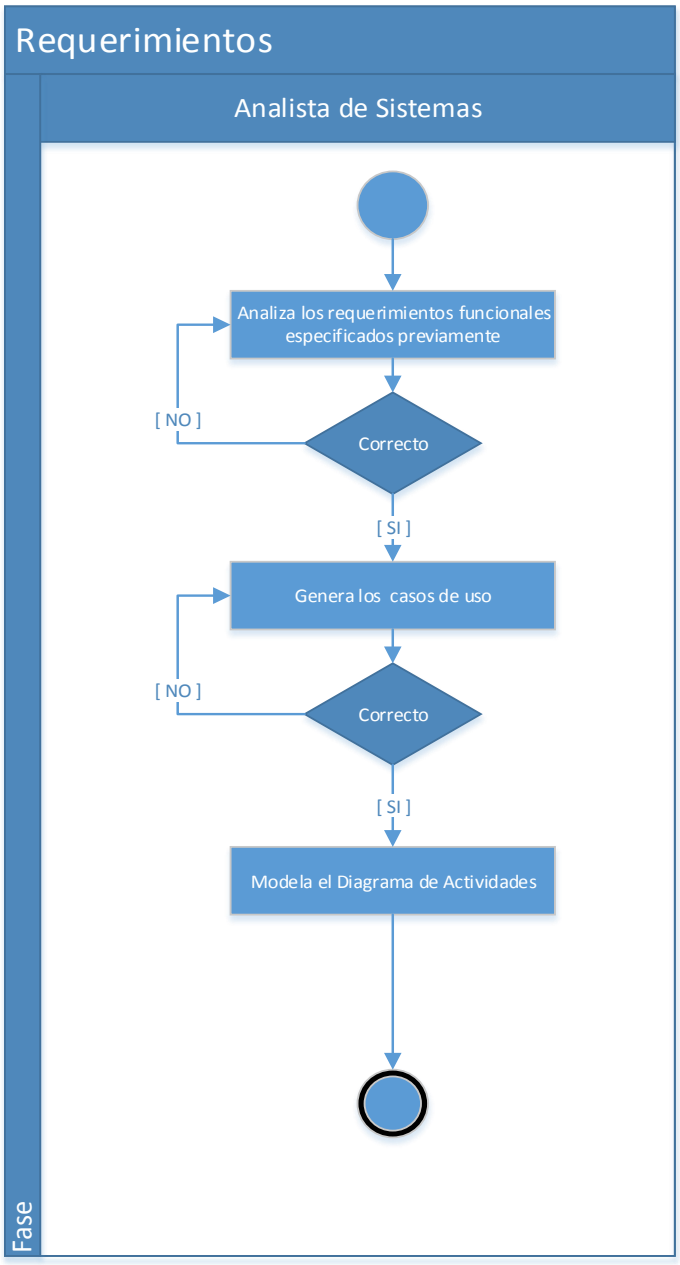


Figura 28: Flujo de Trabajo de Requerimientos

c. Análisis y Diseño

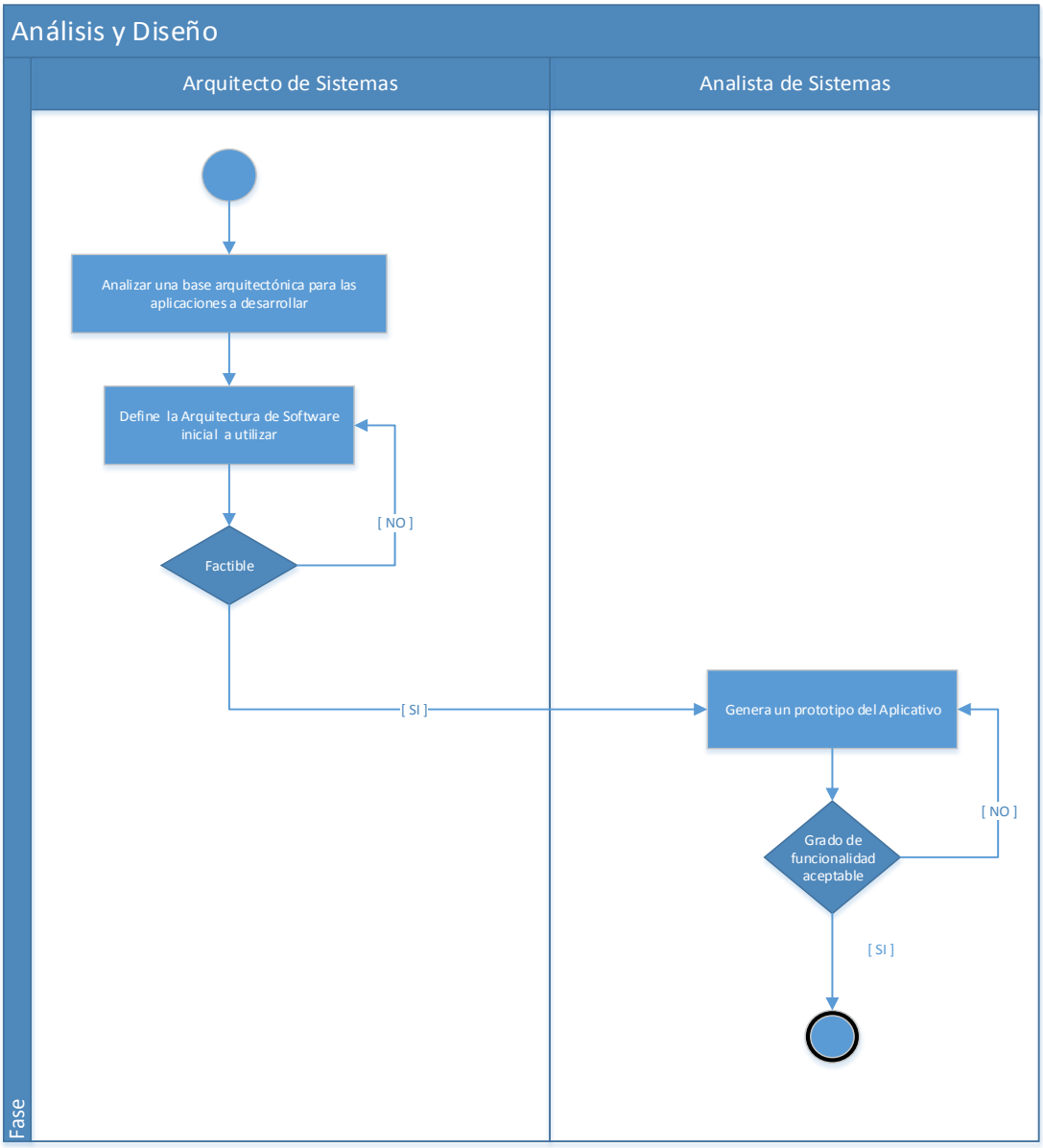


Figura 29: Flujo de Trabajo de Análisis y Diseño

d. Implementación

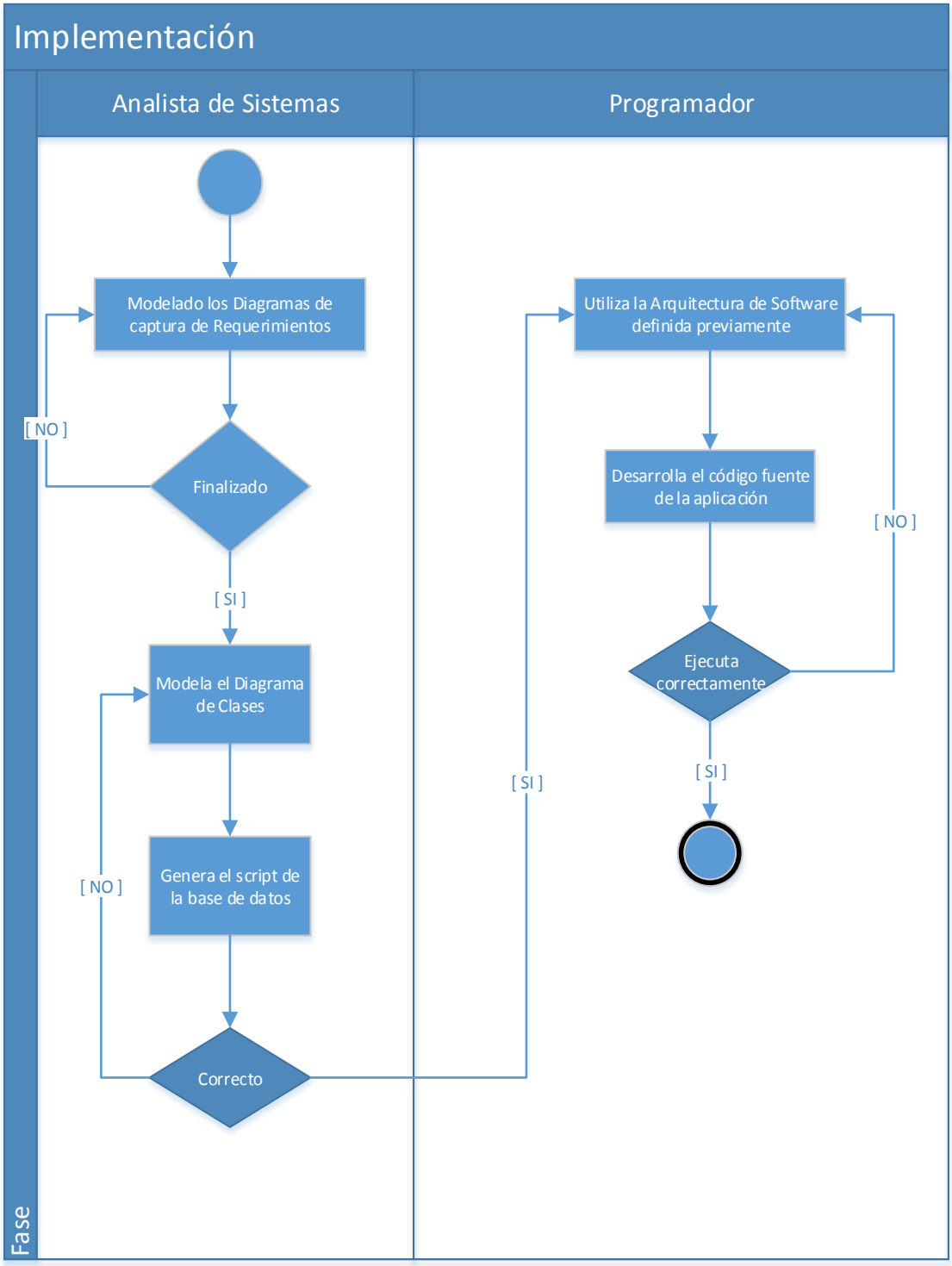


Figura 30: Flujo de Trabajo de Implementación

e. Pruebas

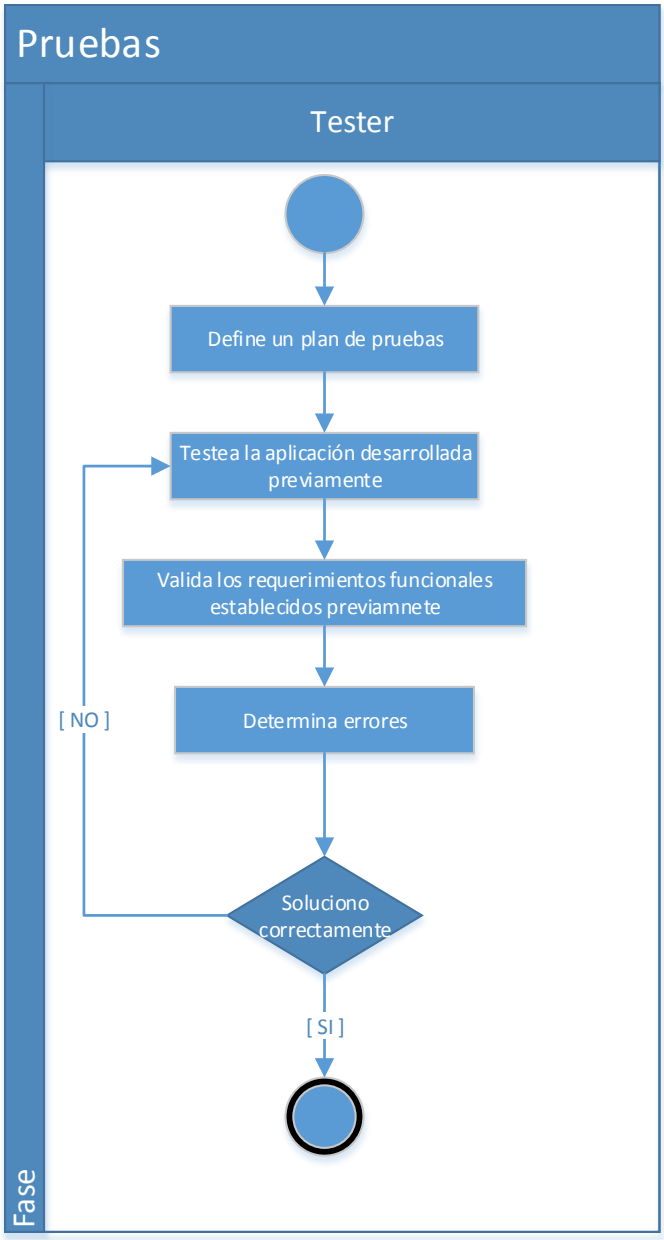


Figura 31: Flujo de Trabajo de Pruebas

f. Despliegue

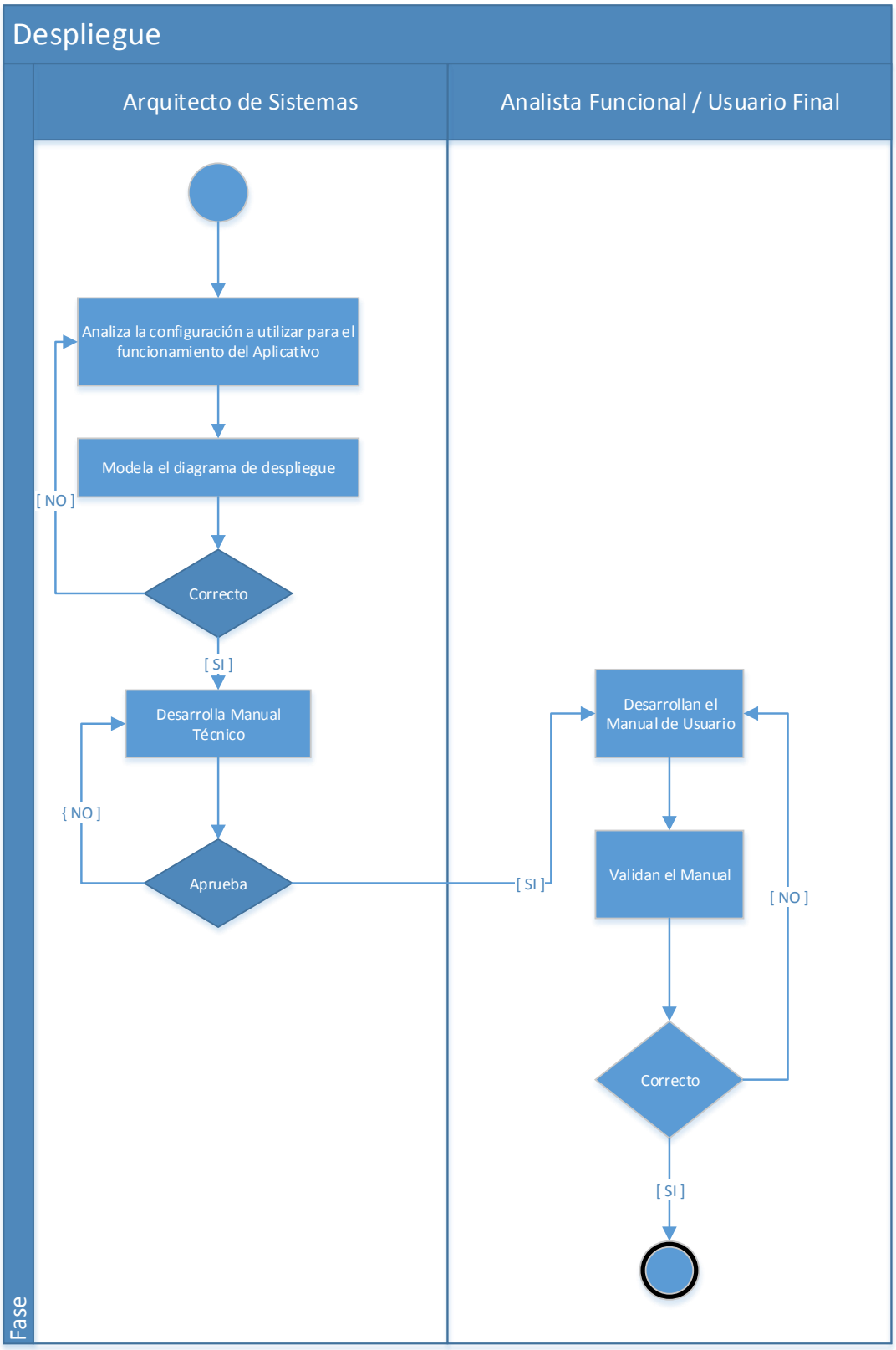


Figura 32: Flujo de Trabajo de Despliegue

3.1.3. Iteraciones

Según (Jacobson, Booch, & Rumbaugh, 2000, pág. 83), el equilibrio correcto entre los Casos de Uso y la arquitectura es algo muy parecido al equilibrio de la forma y la función en el desarrollo del producto, lo cual se consigue con el tiempo. Para esto, la estrategia que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. Cada mini proyecto se puede ver como una iteración (un recorrido más o menos completo a lo largo de todos los flujos de trabajo fundamentales) del cual se obtiene un incremento que produce un crecimiento en el producto.

Una iteración puede realizarse por medio de una cascada como se muestra en la figura 33, Se pasa por los flujos fundamentales (Requisitos o Requerimientos, Análisis, Diseño, Implementación y Pruebas), también existe una planificación de la iteración, un análisis de la iteración y algunas actividades específicas de la iteración. Al finalizar se realiza una integración de los resultados con lo obtenido de las iteraciones anteriores.

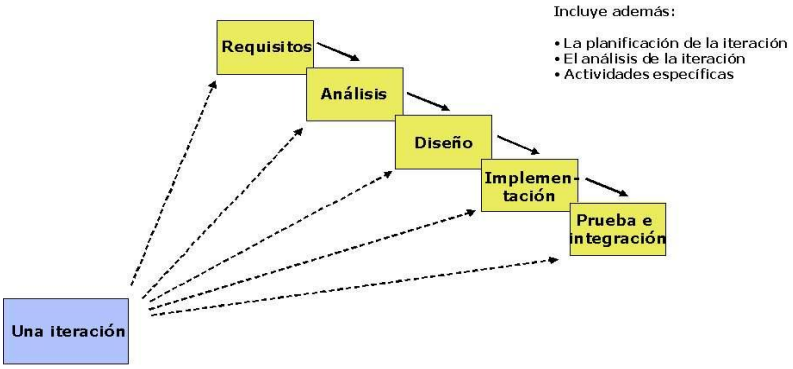


Figura 33: Una iteración RUP
Fuente: (Universidad Distrital Francisco José de Caldas, 2008)

El proceso iterativo e incremental consta de una secuencia de iteraciones. Cada iteración aborda una parte de la funcionalidad total, pasando por todos los flujos de trabajo relevantes y refinando la arquitectura. Cada iteración se analiza cuando termina. Se puede determinar si han aparecido nuevos

requerimientos o han cambiado los existentes, afectando a las iteraciones siguientes. Durante la planificación de los detalles de la siguiente iteración, el equipo también examina cómo afectarán los riesgos que aún quedan al trabajo en curso. Toda la retroalimentación de la iteración pasada permite reajustar los objetivos para las siguientes iteraciones. Se continúa con esta dinámica hasta que se haya finalizado por completo con la versión actual del producto.

Lo que hay que comprender es que en cada iteración comprende:

- Planificar la iteración (estudio de riesgos).
- Análisis de los Casos de Uso y escenarios.
- Diseño de opciones arquitectónicas.
- Codificación y pruebas: la integración del nuevo código con el existente de iteraciones anteriores se hace gradualmente durante la construcción.
- Evaluación de la entrega ejecutable (evaluación del prototipo en función de las pruebas y de los criterios definidos)
- Preparación de la entrega (documentación e instalación del prototipo).

3.1.4. Estructura estática del proceso, roles, actividades, artefactos y flujos de trabajo

Un proceso de desarrollo de software define quién hace qué, cómo y cuándo. RUP define cuatro elementos: los roles que responden a la pregunta ¿Quién?, las actividades que responden a la pregunta ¿Cómo?, los productos, que responden a la pregunta ¿Qué? y los flujos de trabajo de las disciplinas que responde a la pregunta ¿Cuándo?, en la figura 34, se especifica la relación de los elementos especificados.

Figura 34: Relación entre roles, actividades, artefactos
Fuente: (Universidad Politecnica de Valencia, s.f.)

a. Roles

Los Roles que se manejan dentro del IESS y los cuales se utilizara para el desarrollo del proyecto de software con la nueva metodología de trabajo son los siguientes:

- Analista Funcional
- Analista de Procesos
- Analista de Sistemas
- Arquitecto de Sistemas
- Programador
- Tester
- Usuario Final

b. Actividades

Una actividad en concreto es una unidad de trabajo que una persona que desempeñe un rol puede ser solicitado a que realice.

Las actividades tienen un objetivo concreto, normalmente expresado en términos de crear o actualizar algún producto.

c. Artefactos

Un producto o artefacto es un trozo de información que es producido, modificado o usado durante el proceso de desarrollo de software, algunos

artefactos se reutilizaran en algunas iteraciones, ya que los artefactos son resultados del proyecto es decir cosas que se va creando y usando hasta obtener el producto final.

Un artefacto puede ser un documento, un modelo y un elemento del modelo como una clase.

Los artefactos especificados para esta metodología de trabajo son:

- Caso de Negocio
- Requerimientos Funcionales
- Casos del Uso del Sistema
- Diagrama de Actividades
- Documento de Arquitectura de Software
- Prototipo
- Diagrama de Clases
- Código Fuente
- Casos de Uso de Sistema (Plan de Pruebas)
- Diagrama de Despliegue
- Manual Técnico
- Manual de Usuario

d. Disciplinas de Procesos

Una vez especificado los elementos como: roles, actividades y artefactos para la nueva metodología de trabajo no se puede definir un proceso, para ello es necesario contar con una secuencia de actividades que serán realizados por los diferentes roles y artefactos, así como la relación entre cada uno de ellos.

Para darle validez a lo mencionado anteriormente es necesario contar con otro proceso dentro del marco de trabajo de la metodología que se está desarrollando en este caso son las disciplinas, el cual viene a consistir en una relación de actividades para el desarrollo del software.

Ya que la finalidad de cada disciplina es poder adaptar a la realidad que tiene el IESS de manera específica en este caso la Dirección de Riesgos de trabajo el cual va ser el área beneficiada con el aplicativo a desarrollar, para ello es necesario contar con los roles y artefactos permitiendo establecer una relación de diferentes actividades para producir resultados observables.

A continuación se especificara cada disciplina referente a la parte de proceso.

d.1. Modelado del Negocio

La importancia de esta disciplina es poder llegar a entender de mejor manera la problemática que posee el IESS dentro de la Dirección de Riesgos de Trabajo.

La finalidad es poder entender la dinámica que maneja la Institución para poder entender la problemática que posee y en base a ello poder identificar posibles mejoras a realizar y a su vez debe permitir asegurar a los clientes, usuarios y la DNTI el cual tiene el personal de desarrollo puedan tener un entendimiento común.

Para aplicar esta disciplina se enfocara que el problema consiste en que se toma las evaluaciones a los afiliados mediante la utilización de papel y lápiz.

Para ello se definen los requerimientos funcionales el cual es establecido por el Analista de Procesos permitiendo cubrir las necesidades de los clientes, y en base a ello posteriormente ser desarrollado el aplicativo del software.

d.2. Requerimientos

La importancia de esta disciplina es que muy importante ya que permite establecer exactamente lo que desea construir para que los usuarios finales puedan comprender lo que se especifique.

Ya que la finalidad es poder proveer a los desarrolladores un mejor entendimiento de los requerimientos del sistema de acuerdo a nuestra área

de estudio en este caso el IESS, y a su vez permitir definir un ambiente del sistema mediante los casos de uso según el aplicado a desarrollar y a su vez especifica los límites que va a tener el aplicativo.

Para un mejor entendimiento se describe las actividades las cuales deben cumplirse para un determinado proceso del sistema ya que nos permitirá definir de mejor manera el flujo de actividades de los casos de uso.

d.3. Análisis y Diseño

La importancia de esta disciplina es traducir los requerimientos y poder interpretar para el diseño del sistema, y posteriormente definir una arquitectura tomando en cuenta los aspectos más significativos como consistencia y rendimiento ya que se ira refinando hasta llegar a una forma definitiva en cada iteración que se realice, los cuales deben estar enfocados hacia el aplicado a desarrollar ya que será de baja complejidad y de corta duración pero todo esto debe estar apegado a estándares tecnológicos actuales que posee el mercado.

Pero todo esto debe estar acompañado de una buena documentación de la arquitectura de software, dentro de esta disciplina y de acuerdo a nuestra área de estudio es necesario diseñar la interfaz gráfica de usuario para lo cual se construye prototipos permitiendo contrastar con el usuario final y poder determinar cierto grado de funcionalidad del aplicativo en base a la arquitectura definida.

d.4. Implementación

La importancia de esta disciplina es que se puede implementar el diagrama de clases especificando su clase, atributos, métodos con sus respectivas asociaciones de acuerdo a nuestra área de estudio a desarrollar el cual es especificado con el nombre del proyecto Test de Evaluación para el IESS.

Posteriormente se genera un porcentaje del código fuente de la aplicación utilizando el lenguaje de programación PHP e interactuando con el script que

genera el diagrama de clases y con las herramientas definidas en la arquitectura, para luego ser implementado previo las pruebas necesarias que se realicen en las posteriores iteraciones, pero para un óptimo funcionamiento es necesario integrar el sistema según lo especificado a un inicio.

Pero cabe especificar que otra parte del código fuente será desarrollado en otras iteraciones ya que la finalidad es reducir el riesgo y de igual manera se ira integrando y por lo cual se determina que será incremental hasta obtener el aplicativo final.

d.5. Pruebas

La importancia de esta disciplina es que es encargada de detectar errores y posteriormente evaluar la calidad del aplicativo que se está desarrollando ya que la finalidad es poder realizar las pruebas necesarias dentro de cada iteración y poder ir integrando en todo el ciclo de vida

Para dar inicio a esa disciplina es necesario poseer un plan de pruebas el cual contendrá toda la información necesaria como objetivos, estrategias y recursos que se utilizara, ya que la finalidad es poder obtener resultados y esa información tomar como referencia para ir refinando el aplicativo que se desarrolla.

Ya que mientras más temprano se detecte los errores se evitara que se vaya duplicando en las posteriores iteraciones, esto permitirá disminuir tiempo y costo para la entrega del aplicativo final y a su vez permite verificar los requerimientos especificados al inicio.

d.6. Despliegue

La importancia de esta disciplina es poder definir que se requiere para poder implementar el aplicativo y pueda ser utilizado por los usuarios todo esto se especificara en el documento de despliegue.

De acuerdo a nuestra área de estudio se define como se instalara el aplicativo mediante la elaboración de un manual técnico donde se definirá

como se configura la plataforma en este caso VertrigoServ y la aplicación desarrollada.

Para proveer asistencia y ayuda a los usuarios se realiza el manual de usuario donde se especifica paso a paso mediante gráficos y descripciones la funcionalidad del aplicativo de software en este caso el Test de Evaluación para los cursos de la Dirección General de Riesgos del Trabajo del IESS.

Cabe especificar que cada disciplina que se utilizara para dar validez a esta metodología de trabajo en el área de estudio que se ha especificado es necesario poseer un sin número de diagramas los cuales se genera de acuerdo al número de iteraciones y en base a la problemática que se tenga que resolver mediante el aplicativo, los cuales están detallados en el capítulo IV denominado Aplicación de la Nueva Metodología creada para el IESS.

Para dar validez y llevar un seguimiento del plan del aplicativo a desarrollar es necesario contar con plantillas los cuales fueron diseñadas en función del análisis y necesidades que posee la Institución en las cuales se especificara el nombre y la versión del artefacto, fecha de elaboración y modificación, los cuales estarán implementados en los diferentes artefactos definidos en la nueva metodología para el área de desarrollo el cual es la Dirección Nacional de Tecnología de la Información, hay que tomar en cuenta también los roles que participaran, todo esto está detallado en el capítulo IV y en la parte de Anexos.

3.1.5. Descripción de las disciplinas del proceso, roles, actividades y artefactos de la nueva Metodología

Tabla 8

Elementos establecidos para la Metodología de Trabajo

DISCIPLINAS	ARTEFACTOS	DESCRIPCIÓN DEL ARTEFACTOS	ROLES	DESCRIPCIÓN DEL ROL	APTITUDES DEL ROL
Modelado del Negocio	Caso de Negocio	Permitirá comprender los procesos del negocio de la Institución, es una forma preliminar para definir los requerimientos de Sistema	Analista Funcional	Es el responsable de identificar las necesidades del negocio y los objetivos del cliente o usuario, y a su vez ayuda a determinar las soluciones del problema.	Persona conocedora de técnicas de modelado de negocio.
	Requerimientos Funcionales	Permiten describir servicios o funciones que el sistema debe proporcionar, los cuales están relacionados entre datos de entrada y de salida que debe presentar el sistema.	Analista de Procesos	Es el responsable de especificar que se va a construir en base a la problemática generada.	Experto en gestionar las necesidades funcionales del cliente.
Requerimientos	Casos de Uso de Sistema	Permitirá definir los límites del sistema y describir los procesos del negocio.	Analista de Sistemas	Responsable del conjunto de requerimientos no funcionales y funcionales los cuales están modelados en los casos de uso.	Persona con conocimientos técnicos, del negocio y popular; y detalla los casos de uso.
	Diagrama de Actividades	Permitirán para detallar los procesos de las actividades del negocio y describirá como el sistema implementara su funcionalidad, y modela el comportamiento de los casos de uso.		Es el encargado de especificar de una manera simple a la vista del usuario los requerimientos y ayuda a determinar si son esenciales para la funcionalidad del sistema.	

Continua

Análisis y Diseño	**Documento de Arquitectura de Software**	Permitirá controlar el desarrollo del sistema desde una parte técnica y describe las partes del sistema para que comprendan los programadores, y ayuda para incorporar nuevas funciones.	Arquitecto de Sistemas	Posee una visión global del proyecto, para ello participa en la disciplina de los requerimientos permitiéndole elaborar una arquitectura correcta que esté acorde al sistema deseado y permitiéndole dar un buen soporte y evolución.	Experto en dirigir y coordinar las actividades técnicas y artefactos durante el proyecto.
	Prototipo	Especificara cierto grado de funcionalidad de un proceso generando pequeños ejecutables lo cual posteriormente se convertirá en el sistema final.	Analista de Sistemas	Desarrolla una representación limitada del sistema para presentar al cliente y determinar su funcionamiento mediante un análisis y validación de los requerimientos.	Persona responsable del desarrollo de los componentes.
Implementación	**Diagrama de Clase**	Permitirá visualizar las relaciones entre las clases involucradas del sistema (Nombre, Atributos, Métodos)	Analista de Sistemas	Realiza el diagrama en base a los requerimientos y casos de uso especificados para que genere posteriormente el script de la base de datos.	Persona responsable de los estándares de base de datos, tal como nombramiento de campos y tablas, llaves primarias y foráneas e índices.
	Código Fuente	Donde se debe especificar todas las instrucciones necesarias la cual dará origen a la aplicación que se esté desarrollando de un determinado proceso del sistema.	Programador	Es el encargado de convertir los requerimientos especificados en código fuente mediante la utilización de varios lenguajes de programación y está en la capacidad de reducir la complejidad del software permitiendo facilidad al momento de realizar un mantenimiento.	Experto en generar código y cubre aspectos de calidad, corrección, productividad y performance.

Continua

Pruebas	**Casos de Uso de Prueba (Plan de Pruebas)**	Permitirá verificar las diversas funcionalidades del producto de software y para ello se debe especificar el uso de todo tipo de datos de entrada y salida.	Tester	Debe verificar que los requerimientos que se establecieron se cumplan con las tareas realizadas y a su vez detectar errores para mejorar la productividad y llegar a obtener un mejor control de la calidad del software.	Persona con habilidades en el conocimiento del negocio y planifica, diseña, ejecuta y administra las pruebas.
Despliegue	**Documento de Despliegue**	Permitirá mostrar la configuración del funcionamiento del sistema como son los equipos, dispositivos y las interconexiones; y el software que estará en cada máquina.	Arquitecto de Sistemas	Es el encargado de modelar la arquitectura del sistema para que sea robusta y funcione de forma correcta cuando este en ejecución, asegurando que todos los involucrados puedan utilizar.	Persona con habilidades para realizar bosquejos, modelos y generar robustas arquitecturas.
	Manual Técnico	Contendrá una descripción breve del sistema desarrollado y describirá el funcionamiento y configuración completa del sistema.	Arquitecto de Sistemas	Realiza una documentación con aspectos importantes de la funcionalidad del sistema lo cual permitirá dar solución a problemas pero a nivel técnico.	Posee muchos conocimientos técnicos, que permiten establecer una solución técnica pertinente.
	Manual de Usuario	Permitirá que sea entendido por cualquier usuario principiante o avanzado, por lo cual se especificara paso a paso mediante imágenes cómo funciona el sistema.	Analista Funcional/Usuario Final	El primer Rol genera la documentación necesaria con un lenguaje de fácil entendimiento y a su vez cubriendo todos los aspectos importantes para una óptima utilización del sistema. Posteriormente el segundo rol es el encargado de revisar la documentación del manual de usuario ya que esto le permitirá una mayor facilidad de utilización del sistema.	La primera Persona posee el nexo entre el mundo real, cotidiano y un sistema de información; y genera documentación de lo que el sistema provee facilitando la tarea a los usuarios. La segunda Persona es la encargada de utilizar el sistema, responsable de alimentar con datos el sistema.

En la presente tabla 8, se especifica todos los elementos que se han definido para la metodología de trabajo diseñada como son: disciplinas para la parte de proceso, roles y artefactos.

3.1.6. Herramientas Utilizadas

Para el desarrollo del proyecto es necesario contar con un sin número de herramientas de última generación para poder modelar, desarrollar y ejecutar el aplicativo.

Entre las herramientas de Modelado se tiene StarUML que es de licencia gratuita utilizada para modelar diagramas de UML (Lenguaje de Modelado Unificado), el cual permitirá especificar los casos de uso, actividades, diagrama de clases entre otros, para el desarrollo del proyecto se utilizara la versión: 5.0.2.1570

La herramienta de desarrollo que se utilizara es Notepad++ el cual cumple la función de un editor de texto y es de código libre, el cual nos permitirá generar el código fuente del aplicativo de forma sencilla y se utilizara la versión v6.7.8.2

La herramienta que se utilizara como servidor web es VertrigoServ el cual provee un paquete con herramientas como el Servidor Apache, Lenguaje de programación PHP, Base de Datos MYSQL y PhpMyAdmin el cual permite ocuparse de la Administración de MySql el cual permitirá un óptimo funcionamiento del aplicativo y es el componente más importante de la arquitectura especificada y se utilizara la versión v2.22

CAPÍTULO IV

4. APLICACIÓN DE LA METODOLOGÍA DE TRABAJO

4.1. Desarrollo de la Metodología

En base a los problemas señalados que posee la Dirección Nacional de Tecnología de la Información del IESS cuando se desarrolla los proyectos de Software y la inadecuada gestión de la metodología de desarrollo RUP por estos defectos se ha especificado algunos elementos del marco de trabajo de RUP que se adapten de mejor manera para la nueva metodología de trabajo para el IESS objeto de este estudio, la cual permitirá una óptima secuencia entre cada uno como son: fases, roles, actividades, artefactos, disciplinas de la parte de proceso y las iteraciones, los cuales permitirán generar un mejor análisis, diseño y desarrollo de artefactos con las plantillas que se especificara permitiendo que el nuevo aplicativo de software que se desarrolle cubra en su totalidad las necesidades generadas por los usuarios de la Institución.

Por ser proyectos complejos que se desarrollan se ha especificado la importancia de utilizar algunos elementos de RUP, pero cabe especificar que la nueva metodología solo está enfocado a la parte de desarrollo por lo cual toma como referencia la parte de disciplinas de proceso, ya que las disciplinas de soporte están controladas por otros procesos de la DNTI, permitiéndole gestionar los diferentes proyectos del IESS.

Para dar una mayor validez a la nueva metodología de trabajo se encaminó en desarrollar un proyecto de un caso real de la Dirección de Riesgos de Trabajo del IESS, para ello se aplicara en cada una de las fases diferentes iteraciones en las cuales se desarrolla las diferentes disciplinas conjuntamente con cada uno de los artefactos especificados y con todos los elementos descriptos en el capítulo III, para generar un aplicativo de calidad.

A continuación se desarrolla los elementos especificados a la nueva metodología de Trabajo.

4.2. Fase de Inicio

4.2.1. Iteración 1

Es analizar y entender la problemática que posee la Dirección de Riesgos de Trabajo y una de las funciones que realiza es que se dictan cursos a los empleadores o afiliados que se encuentran registrados en el IESS de la ciudad de Quito, pero la forma de evaluar se lo realiza mediante un examen en papel.

En base a lo especificado anteriormente se especifica las necesidades y mejoras a realizar para ello se define los procesos del negocio y se determina los requerimientos funcionales que son la base para el desarrollo de la nueva aplicación del Software, permitiendo a los clientes, usuarios y la DNTI tener un entendimiento en común.

a. Disciplina Modelado del Negocio

El artefacto de CASO DEL NEGOCIO permite determinar la problemática y posteriormente especifica los objetivos, alcance de proyecto, las personas interesadas, los responsables del proyecto, las metas a alcanzar y el tiempo de desarrollo.

Una de las necesidades a ser solventadas con el aplicativo es: Generar Cuestionarios de forma randómica y obtener la calificación de forma inmediata, para una especificación más detallada de este artefacto se incluye en el **Anexo A**: (Estudio – Caso de Negocio).

El artefacto de REQUERIMIENTOS FUNCIONALES se define en base a las necesidades generadas por los interesados del aplicativo y definido la descripción de la funcionalidad por los responsables del proyecto en este caso el Analista de Procesos, para una especificación más detalla se incluye en el **Anexo B:** (Descripción Funcional del Requerimiento).

Una vez establecido estos 2 artefactos dentro de la Disciplina de Modelado de Negocio, se ha culminado la iteración Uno y por ende la Fase de Inicio de la Metodología de trabajo desarrollada.

4.3. Fase de Elaboración

4.3.1. Iteración 2

En base a los requerimientos establecidos se especifica los casos de uso del sistema cuya finalidad es que nos permite definir los límites a desarrollar para el sistema y los actores que interactuaran con los diferentes procesos; y posteriormente se especifica el diagrama de actividades el cual se utiliza para entender de una manera clara el flujo de los diferentes procesos.

Ya que cabe determinar que los actores que van a interactuar en el contexto del sistema son: Administrador el cual genera la necesidad del sistema, el Director el cual validara la aplicación y el Empleado afiliado al IESS el cual rendirá el Test.

Posteriormente se determina la arquitectura de software pero enfocándose que el aplicativo debe desarrollarse con herramientas de software libre y orientado a la web, finalmente se desarrolla un prototipo el cual cubre cierto grado de funcionalidad de acuerdo a los requerimientos establecidos previamente.

a. Disciplina de Requerimientos

La finalidad del artefacto del CASO DEL USO DEL SISTEMA denominado Rendir Test y especificado como versión 1, es que nos permite poder definir como se genera el test y la forma de calificación del examen en línea pero para que tenga éxito debe estar previamente ingresado información de las preguntas y respuestas en la base de datos, pero para ello debe ingresar previamente el usuario y clave el afiliado.

En el caso de uso se especifica aspectos como:

Rendir Test

Actor: Afiliado – Empleado

Flujo principal:

- El Actor desea rendir el Test
- El sistema le genera el Test
- El Sistema posteriormente genera la calificación del Test

En la figura 35, se estable el actor y como interactúa con los procesos.

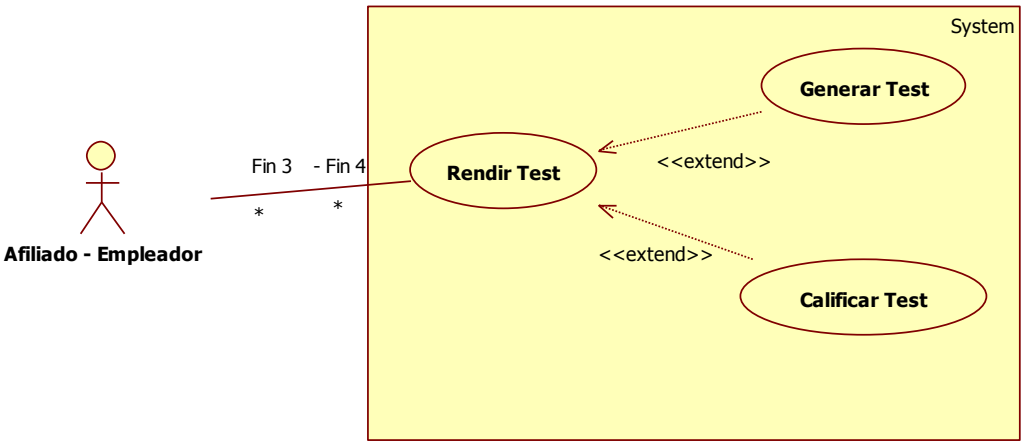


Figura 35: Caso de Uso Rendir Test

Para más información puede encontrarle en el **Anexo C:** (Casos del Uso del Sistema)

La finalidad del artefacto de DIAGRAMA DE ACTIVIDADES es que permite comprender de mejor manera el flujo de las actividades del caso de uso Rendir Test.

En el diagrama de la figura 36, se establece el flujo de procesos que tiene el Usuario y el Sistema permitiendo describir como el sistema implementara su funcionalidad.

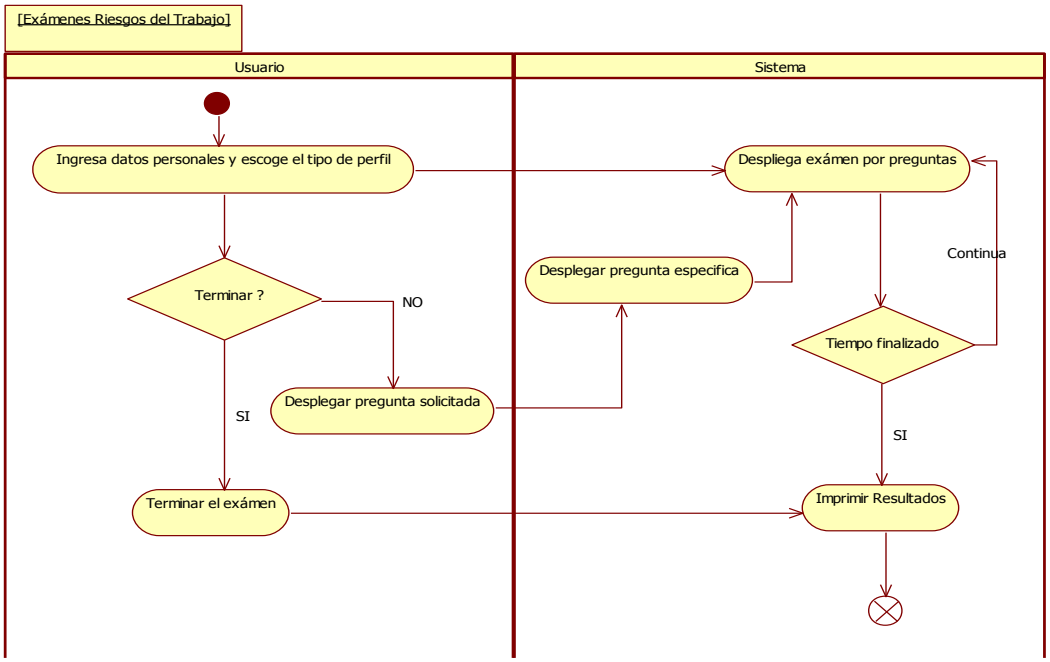


Figura 36: Diagrama de Actividades Rendir Test

b. Disciplina de Análisis y Diseño

La finalidad del artefacto de DOCUMENTO DE ARQUITECTURA DE SOFTWARE es que nos ha permitido proveer una vista de la arquitectura a utilizar cuya definición se estableció según las necesidades, las reglas del negocio y requerimientos del área solicitada del IESS, pero en las próximas iteraciones se ira refinando hasta llegar a su forma definitiva.

La plataforma tecnológica a utilizar es VertrigoServ, ya que los proyectos son de baja complejidad, de corto tiempo de duración y es software libre; el cual permite desarrollar web distribuidas y utilizar la aplicación localmente, admite gestionar base de datos MySQL para ello en la figura 37, se visualiza el escenario de PHPMyAdmin y se accede mediante un navegador WEB.

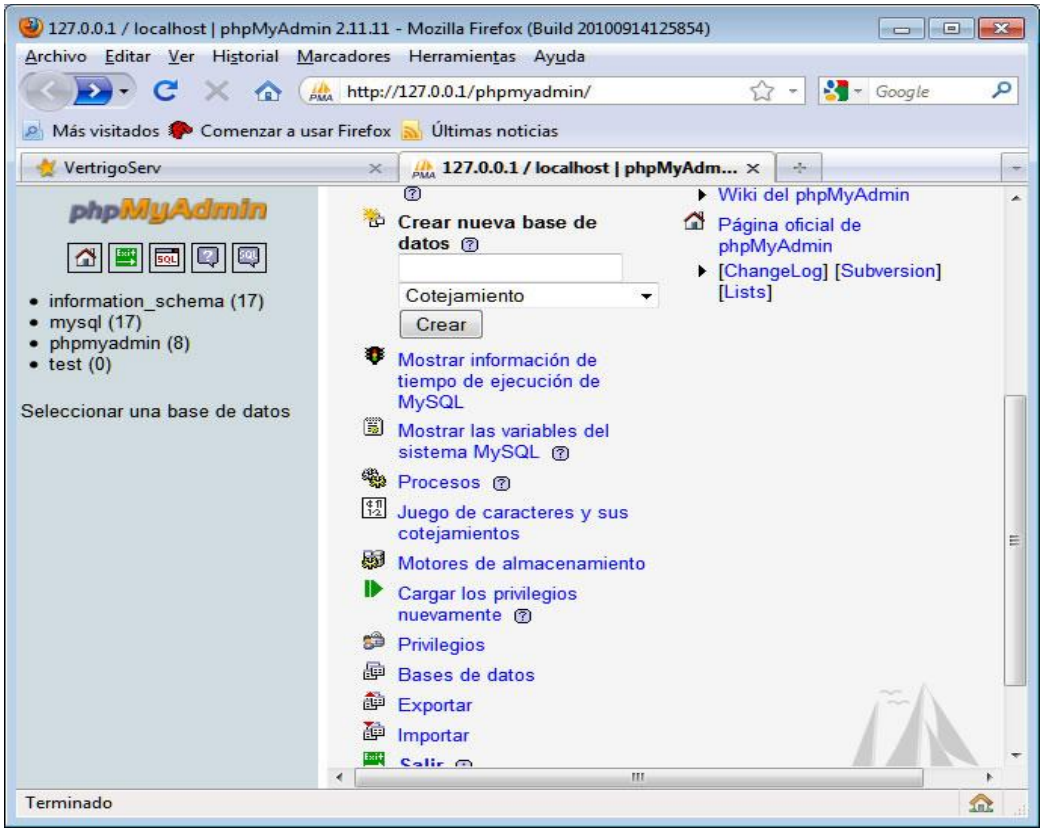


Figura 37: Escenario de PHPMyAdmin

Todos estos aspectos más detallados se encuentra en el **Anexo D:** (Documento de Arquitectura).

La finalidad de desarrollar el artefacto del PROTOTIPO es que permitió poder ya especificar cierto grado de funcionalidad permitiendo presentar al Administrador para que valide el aplicativo, pero para su complemento se utilizó la arquitectura definida, conjuntamente con los requerimientos y el caso de uso especificado.

Se especifica ciertas capturas de pantallas que determinan la funcionalidad del prototipo realizado, en la figura 38, se especifica cómo se despliega el test que va hacer desarrollado por el afiliado que ingreso previamente al sistema.

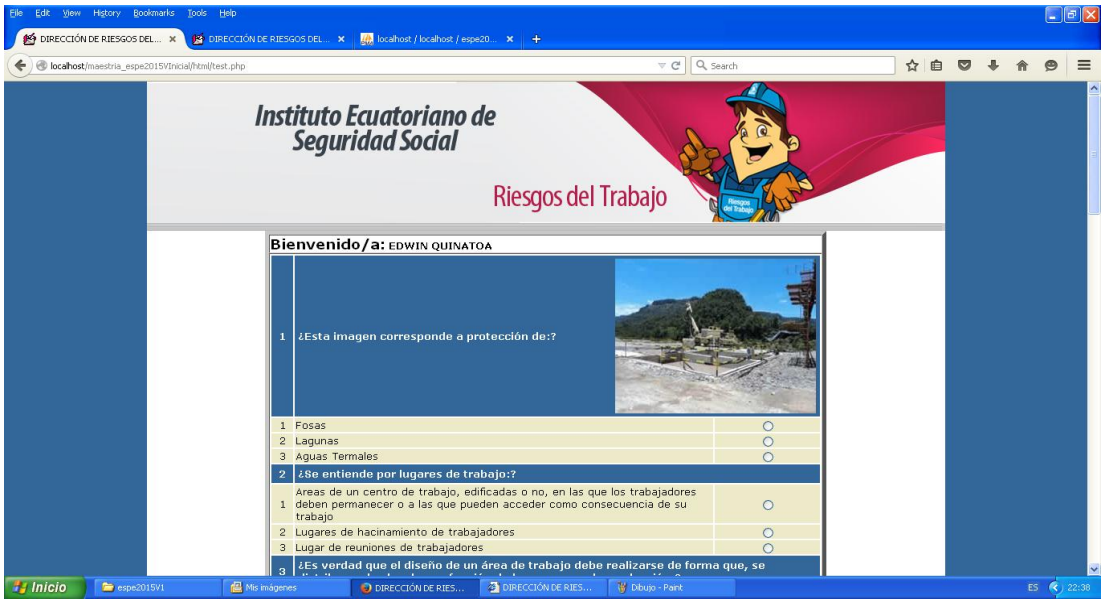


Figura 38: Despliegue de Test

Para más especificaciones de las pantallas del prototipo realizado lo encuentra en el **Anexo C:** (Casos del Uso del Sistema).

Se puede establecer que una vez finalizado estos 4 artefactos dentro de las disciplinas de requerimientos, Análisis y Diseño que ha culminado la Iteración dos y por ende la Fase de Elaboración, los cuales han permitido ir ya cumpliendo con un cierto porcentaje de las reglas del negocio establecido al inicio de este proyecto. Hinchado.

4.4. Fase de Construcción

4.4.1. Iteración 3

Una vez que se ha desarrollado los artefactos de los casos de uso y de arquitectura de software al finalizar la iteración anterior se especifica que es necesario generar una nueva versión de cada uno de ellos dentro de las mismas disciplinas especificadas anteriormente, ya que esto permitirá que exista un equilibrio entre estos dos artefactos ya que en el transcurso del desarrollo se lograra esto, eso se puede ver al final de la iteración y se complementa con el desarrollo de otros artefactos en diferentes disciplinas y todo esto se ira logrando al final de cada mini proyecto.

a. Disciplina de Requerimientos

En esta disciplina se especifica el caso de uso versión dos el cual se especifica Desplegué por pregunta por el nuevo requerimiento generado.

Es necesario realizar el cambio ya que las preguntas del Test se despliegan en un solo formulario lo cual ocasiona que no pueda analizar bien la pregunta el afiliado, para ello es necesario que se desplegué de pregunta en pregunta permitiendo que pueda considerar de mejor manera, y se complementa con un tiempo designado automáticamente por el sistema pero para rendir la evaluación es necesario ingresar un usuario y clave.

En el caso de uso se especifica aspectos como:

Desplegué por pregunta

Actor: Afiliado – Empleado

Flujo principal:

- El Actor desea rendir el Test
- El sistema le genera tiempo de Test
- El Sistema posteriormente genera el desplegué por pregunta

En la figura 39, se estable el actor y como interactúa con los procesos.

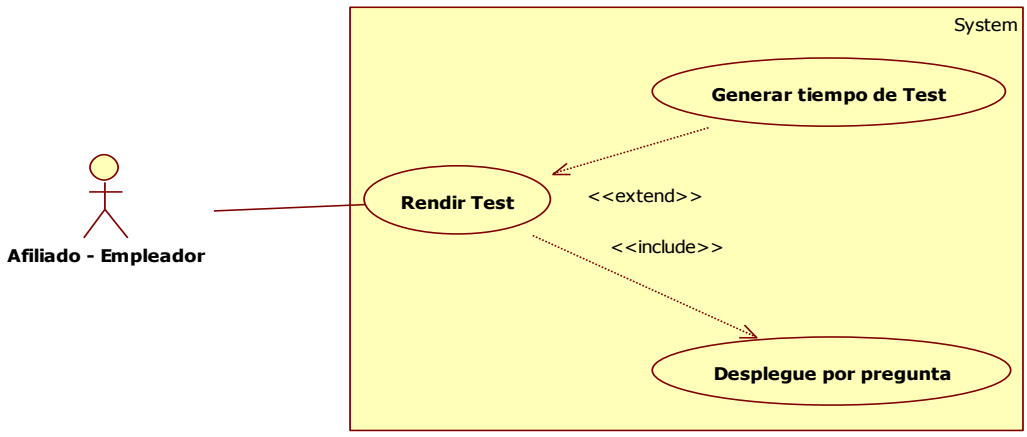


Figura 39: Caso de Uso Desplegué por pregunta

Para más información puede encontrarle en el **Anexo E:** (Casos del Uso del Sistema V2)

b. Disciplina de Análisis y Diseño

Se define que el artefacto de la Arquitectura de Software en su versión dos no es necesario refinar la arquitectura de solución establecida anteriormente ya que cubre las funcionalidades requeridas por el sistema y el cual permitirá desarrollar el código fuente para obtener una aplicación final de calidad después de efectuar las pruebas necesarias.

Por consiguiente se ha determinado que la plataforma tecnológica VertrigoServ es la adecuada para seguir utilizando en esta iteración, todos los aspectos referentes a la arquitectura que se está utilizando en el proyecto se encuentran especificados más detalladamente en el **Anexo D:** (Documento de Arquitectura).

c. Disciplina de Implementación

Una vez especificado en las iteraciones 1 y 2, las disciplinas y los artefactos respectivos se llegó a determinar el artefacto denominado Diagrama de Clases para el proyecto de Test de Evaluación.

En el cual se especifica las clases que involucran al sistema, los atributos, los métodos y las asociaciones que se relacionan con las diferentes clases, ya que la finalidad es poder generar un script para la base de datos para luego unificar con el lenguaje de programación PHP y pueda visualizar los usuarios una vez que se concluya la aplicación ya que hoy en día la información es el elemento más importante dentro de una Organización.

En la figura 40, se especifica el diagrama clases y sus elementos que involucran para el desarrollo del sistema.

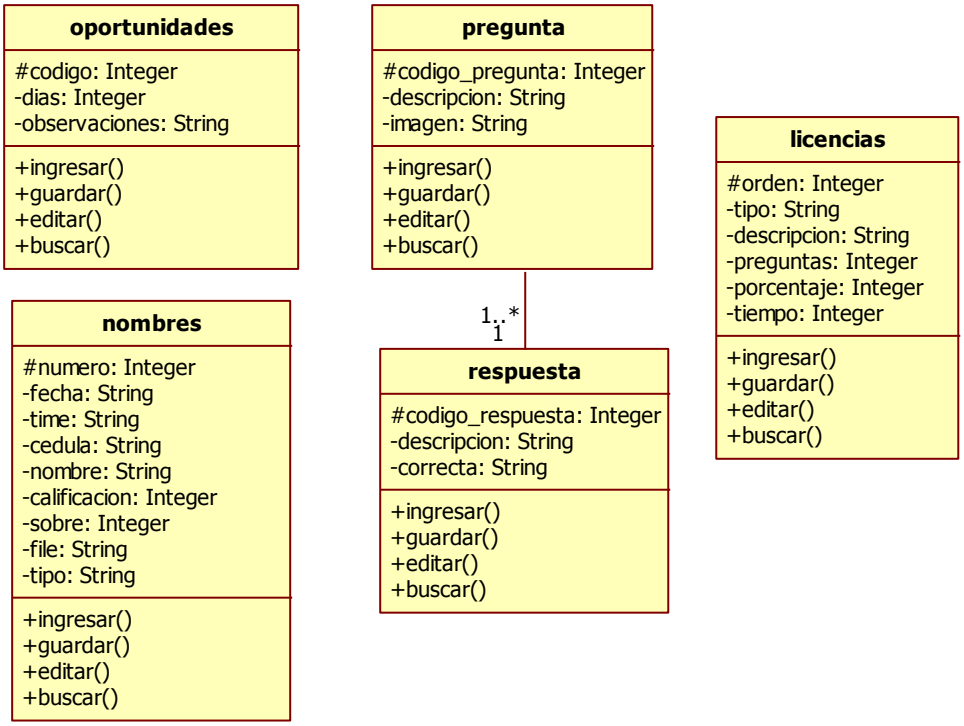


Figura 40: Diagrama de Clases

Los diferentes artefactos desarrollados en las diferentes iteraciones han permitido poder desarrollar una parte del código de fuente el cual es un artefacto más dentro de esta disciplina para el aplicativo para lo cual se ha utilizado el lenguaje de programación PHP, lenguaje de Etiquetas HTML el cual permite mostrar la información por el navegador y para una mejor presentación se utiliza CSS – Hojas de estilo en cascada.

Para el almacenamiento de los datos se utiliza MySLQ y para manipular toda esa información se utiliza PHP, por consiguiente se especifica algunos fragmentos del código de la primera versión de la aplicación tales como:

En el presente código se especifica el diseño para el ingreso de datos del afiliado utilizando HTML y CSS.

```
<form name="form1" method="post"  onSubmit="return ValidaCampos(this)"
action="html/test.php">
        <table width="100%" border="0">
         <tr>
         <td colspan="3"><div class="datos2"><strong>Ingrese los
Datos:</strong></div></td>
```

```
            </tr>
            <tr>
            <td width="25%"><div align="right"
class="datos1"><strong>Nombre</strong>:</div></td>
            <td width="15%"><img src="imagenes/login/logo1.jpg" width="49"
height="46"></td>
            <td width="60%">
             <div align="left">
              <input name="nombre" type="text" id="nombre" size="30" maxlength="50"
onKeyUp="javascript:this.value=this.value.toUpperCase();" style="background-
color:#F0F0F0;border:1px solid #999;">
              </div>
            </td>
            </tr>
            <tr>
            <td><div align="right" class="datos1"><strong>Cédula:</strong></div></td>
            <td width="15%"><img src="imagenes/login/logo1.jpg" width="49"
height="46"></td>
            <td>
             <div align="left">
             <input name="cedula" type="text" id="cedula" onKeyPress="if (event.keyCode
< 47 || event.keyCode > 57) event.returnValue = false;" size="10" maxlength="10"
style="background-color:#F0F0F0;border:1px solid #999;" />
             </div>
            </td>
            </tr>
            </table>
            <p>
            <label></label><label></label></p>
            <p align="center">
              <input type="submit" name="Submit" value="Ingresar" >
            </p>
          </form>
```

Se especifica la función donde se realizó la conexión de la base de datos y se crea algunas consultas desde PHP.

```
<?php
    include "defs.php";
    //echo .$_POST['cedula'];
    $db = mysql_connect($host, "root", "edwin" )or die("NO se pudo realizar la conexión");
```

```
    mysql_select_db($base,$db);

$select="SELECT * from pregunta ORDER BY RAND() limit ".$limite;
    $select1="SELECT * from respuesta where ";
    $selectcedula="SELECT *from nombres where fecha=curdate() and
cedula=".$_POST['cedula'];
?>
```

Para más información parte del código fuente puede encontrarle en el **Anexo J:** (Manual Técnico)

d. Disciplina de Pruebas

Una vez especificado algunos artefactos dentro de la iteración tres, es necesario establecer las pruebas, ya que son procesos para verificar la calidad del software antes de su puesta en marcha, pero para la ejecución de ello es necesario especificar primero un PLAN DE PRUEBAS el cual constituye un artefacto más dentro de esta iteración.

Este plan de pruebas versión uno permitió definir la Misión a desarrollar, los elementos a probar y las pruebas funcionales, para darle cumplimiento a esto se complementó determinando el objetivo a cumplir, las herramientas que se utilizara, y para dar fiabilidad a la aplicación es necesario generar pruebas de carga y desempeño a nivel de infraestructura de hardware y software.

La finalidad de este plan de pruebas es que se especificó todos los elementos necesario que se requiere para poder probar el caso de uso de exámenes en línea, cuyo alcance funcional se basa en la elaboración de los tests, toma del examen y calificación del mismo, incluyendo el ingreso de los datos de los afiliados.

Para una mejor especificación de este artefacto puede encontrarle en el **Anexo F:** (Plan de Pruebas)

4.4.2. Iteración 4

Una vez finalizado la tercera iteración se definió que se requiere de otra iteración el cual permite definir versiones finales de los artefactos como: Caso

de Uso, Arquitectura de Software y Código Fuente de este se lo realiza de acuerdo a los nuevos requerimientos y cambios generados; ya que todos estos ayudaran para complementar y poder generar el artefacto de plan de pruebas en su versión dos.

a. Disciplina de Requerimientos

Dentro de esta disciplina se ha necesario la reutilización del artefacto de Casos del Uso del Sistema versión dos a la cual se especifica como versión final, ya que no ha generado ningún nuevo requerimiento o cambio respectivo, ya que dentro de esta iteración no hace ningún cambio en el artefacto pero ayuda a complementar en el desarrollo del código fuente.

Para más información puede encontrarle en el **Anexo E:** (Casos del Uso del Sistema V2)

b. Disciplina de Análisis y Diseño

Ya especificado el artefacto de la documentación de arquitectura de software en la Iteración 2 y 3 se determina que sigue siendo el mismo por lo cual se define como versión final, por consiguiente se ha determinado que VertrigoServ es el más óptimo, ya que sus herramientas como son: PHP, MYSQL, APACHE permiten generar componentes robustos y han ayudado a incorporar nuevas funciones al aplicativo permitiendo obtener un alto grado de calidad y lo cual ha ocasionado que solo se realice pequeñas modificaciones en el código fuente a lo largo del desarrollo del sistema.

Para más referencias se encuentra detallado en el **Anexo D:** (Documento de Arquitectura).

c. Disciplina de Implementación

En un artefacto anterior ya se especificó una parte del código fuente denominado versión uno, pero una vez que se ha generado incrementos para poder terminar el desarrollo del aplicativo en base a los cambios generados por las pruebas efectuadas, a los casos de uso suscitados y de acuerdo al

caso de negocio que se debe cumplir se desarrolla el artefacto de la versión final para ello se utiliza la arquitectura de software definida anteriormente.

Por consiguiente se detalla algunos fragmentos del código los cuales ayudan a poder terminar la aplicación de Test de Evaluación como son:

En el presente código se especifica cómo definir el tipo de perfil del usuario para que ingrese al Sistema a rendir su test.

```
<tr>
 <td><div align="right"><strong>Tipo de Perfil:</strong></div></td>
 <td><img src="imagenes/login/logo1.jpg" width="49" height="46"></td>
 <td><select name="tipo" style="background-color:#F0F0F0;border:1px solid #999;">
  <?Php
 $temp=1;
 while ($temp<=$num)
 {
  $fila=mysql_fetch_row($res);
  ?>
 <option value="<?Php echo($fila[0]);?>"><?Php echo($fila[1]);?></option>
 <?Php
  $temp=$temp+1;
 }
 ?>
 </select></td>
</tr>
```

Posteriormente se especifica el código que permite seleccionar las respuestas en base a las preguntas que el sistema generé randomicamente para el usuario.

```
<?php
    $select1="SELECT * from espe2015v.respuesta where ";
    $select2=$select1.' fk_pregunta= '.$valor;
     $db1 = mysql_connect($host, "root", "vertrigo" )or die("NO se pudo realizar la
conexión");
    $resul1=mysql_query($select2,$db1);
    $numeroReg1=mysql_num_rows($resul1);
    $temp2=1;
?>
```

Se especifica cómo se determina el proceso para definir si aprobó o reprobó el examen.

```
<tr>
    <td> </td>
    <td width="84%"  align="center" style="font-size:40px; color:#973a1e; " >
   <span class="style7">
   <?php
   if ($porcentaje<=(($calificacion/$limite)*100) ) {
         echo ("APROBADO");
        }
      else{
    echo ("REPROBADO");
      }
    $notafinal= ($calificacion/$limite)*100
?>
 con el <?php echo $notafinal?>%</span></td>
</tr>
```

Para más información parte del código fuente puede encontrarle en el **Anexo J:** (Manual Técnico)

d. Disciplina de Pruebas

Una vez determinado en los artefactos anteriores y desarrollo otra parte del código por otro requerimiento que se generó el cual fue complementado con otro caso de uso especificado anteriormente dentro de esta iteración es necesario determinar otro artefacto denominado PLAN DE PRUEBAS versión dos.

Ya que otro de los aspectos que se genere otra versión es que el aplicativo se va integrando el nuevo con el existe con las diferentes iteraciones y fases de la metodología de trabajo, para lo cual es necesario determinar técnicas para detectar errores.

Esta versión cubre la mayor parte de puntos establecidos en el plan anterior pero con la diferencia que se enfoca en el caso de uso de Desplegué por pregunta, ya que el alcance funcional se basa es que las preguntas del test se visualizan de forma individual, el tiempo generado para la evaluación.

Para una mejor especificación de este artefacto puede encontrarle en el **Anexo G:** (Plan de Pruebas V2)

Se puede especificar que los artefactos de Caso de Uso del Sistema y el Documento de Arquitectura de Software con sus respectivas versiones en la iteración tres y cuatro son opcionales pero para este aplicativo fue necesario para poder lograr un óptimo cumplimento de los requerimientos especificados y por el alcance del caso de negocio.

En esta fase fue necesario realizar dos iteraciones, en algunos artefactos se generó versiones dentro de cada una de las disciplinas especificadas para este modelo, y se puede definir que al finalizar una iteración se abordar una parte de la funcionalidad total del aplicativo, con todo esto desarrollado se culmina la fase de Construcción.

4.5. Fase de Transición

4.5.1. Iteración 5

La finalidad de a ver desarrollado los diferentes artefactos es que en esta iteración la aplicación se puede liberar para producción, para ello se especifica los últimos artefactos para la metodología de trabajo como son: Pruebas con las funcionalidades especificadas en los planes de pruebas especificado anteriormente, el Documento de despliegue para determinar la arquitectura de Hardware a utilizar para la implementación de la aplicación, Manual técnico donde se especifica aspectos técnicos de configuración para la funcionalidad de la aplicación y por último el Manual de Usuario para que utiliza el usuario final.

a. Disciplina de Pruebas

El nuevo artefacto son el PLAN DE PRUEBAS VERSION FINAL o PRUEBAS UNITARIAS donde se determina la información general donde se especifica los casos de uso que están relacionados con los requerimientos desarrollados, en base a ello se determina el registro de pruebas de cada caso de uso donde se determinó la descripción a ejecutar con su respectivo datos

de entrada, los resultados esperados y los resultados obtenidos para complementar esto se complementó las pantallas de Request y Response pero todos estos aspectos se determina en base a los artefactos de planes de pruebas especificados anteriormente.

Para una mejor especificación de este artefacto puede encontrarle en el **Anexo H:** (Pruebas Unitarias)

b. Disciplina de Despliegue

Para complementar la funcionalidad del aplicativo es necesario determinar el artefacto del DIAGRAMA DEL DESPLIEGUE donde se especifica como interactuara el servidor de la aplicación con el usuario o usuarios, para complementar se especifica los requerimientos mínimos que debe tener a nivel de Software y Hardware también.

A continuacion se describe algunas capturas de pantallas:

Se especifica en la figura 41, como interactua el servidor de base de datos, aplicativo y las estaciones de trabajo las cuales seran utilizadas por los afiliados para rendir el Test.

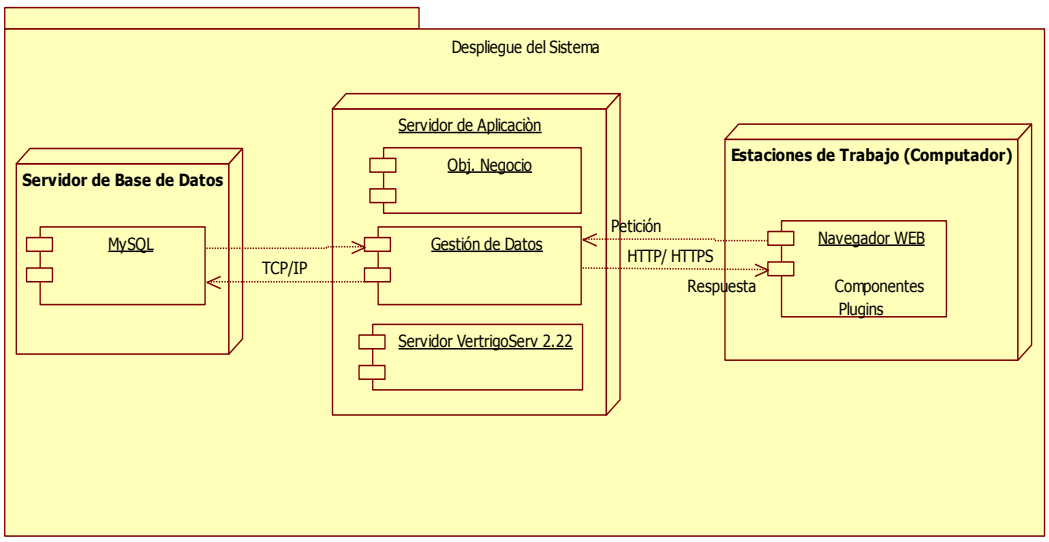


Figura 41: Diagrama de Despliegue del aplicativo

Para una mejor especificación de este artefacto puede encontrarle en el **Anexo I:** (Documentación de Despliegue)

Una vez finalizado los artefactos que determinar el desarrollo del aplicativo según los requerimientos definidos es necesario realizar el artefacto del MANUAL TÉCNICO el cual presente un documento que tiene como finalidad especificar la instalación de la plataforma VertrigoServ y la configuración del sistema el cual permitira dar solucion a los difersos problemas a nivel técnico.

A continuacion se describe algunas capturas de pantallas:

Se especifica en la figura 42, como se visualiza una vez terminado la instalación de VertrigoServ y los programas que se instalaron con la plataforma.

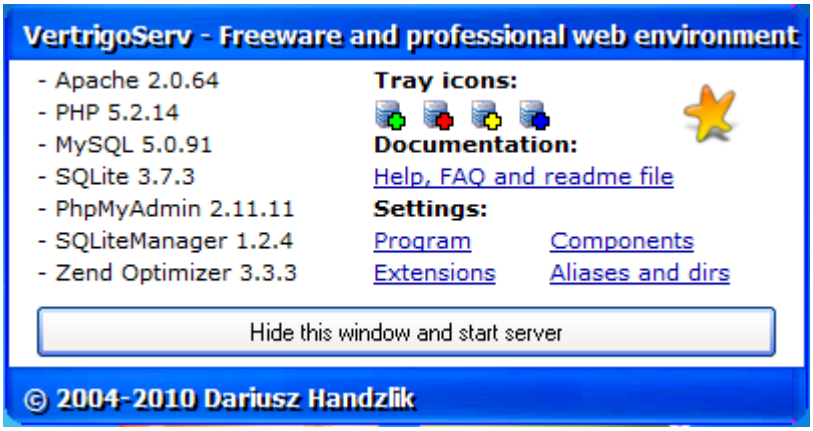


Figura 42: Finalización de Instalación de VertrigoServ

Adentro del PHPMyAdmin se crea la base de datos llamado ***espe2015v*** y dentro de ella se ejecuta el archivo del script para visualizar las tablas generadas, en la figura 43, se especifica todo lo creado.

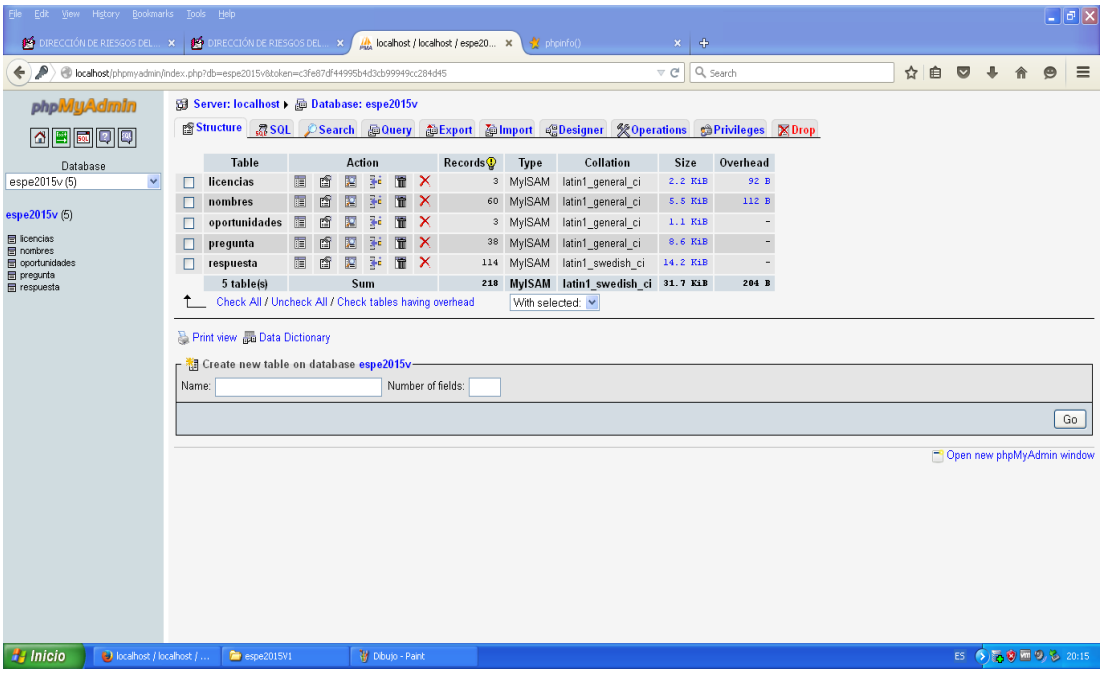


Figura 43: Base de Datos espe2015v

Una vez configurado todos los archivos necesario para el funcionamiento de la aplicación se puede verificar mediante un navegador para el efecto se utilizara Mozilla Firefox, y para ingresar se pondrá ***http://localhost/espe2015v1/index.php,*** en la figura 44, que si todo está bien se visualiza la siguiente pantalla.

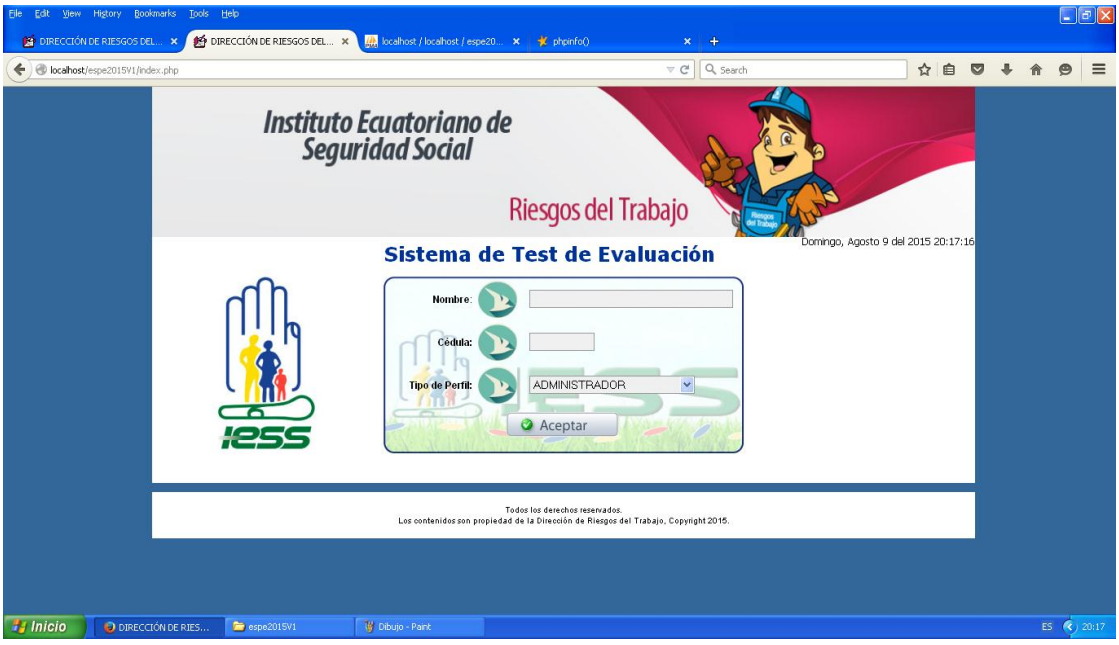


Figura 44: Ingreso al Sistema

Para una descripción más detallada del manual técnico se especifica en el **Anexo J:** (Manual Técnico)

Para facilidad y un mejor entendimiento de uso del aplicativo por parte de los usuarios finales se desarrolla el artefacto de MANUAL DE USUARIO el cual permite especificar paso a paso como funciona es decir como ingresar los datos de entrada para poder obtener resultados, ya que la finalidad es poder determinar los procesos más complejos para su optima facilidad de uso.

A continuacion se describe algunas capturas de pantallas:

Se especifica en la figura 45, cuando ya se termina de resolver el Test y se genera los resultados obtenidos con su respectiva calificación, para su posterior impresión.

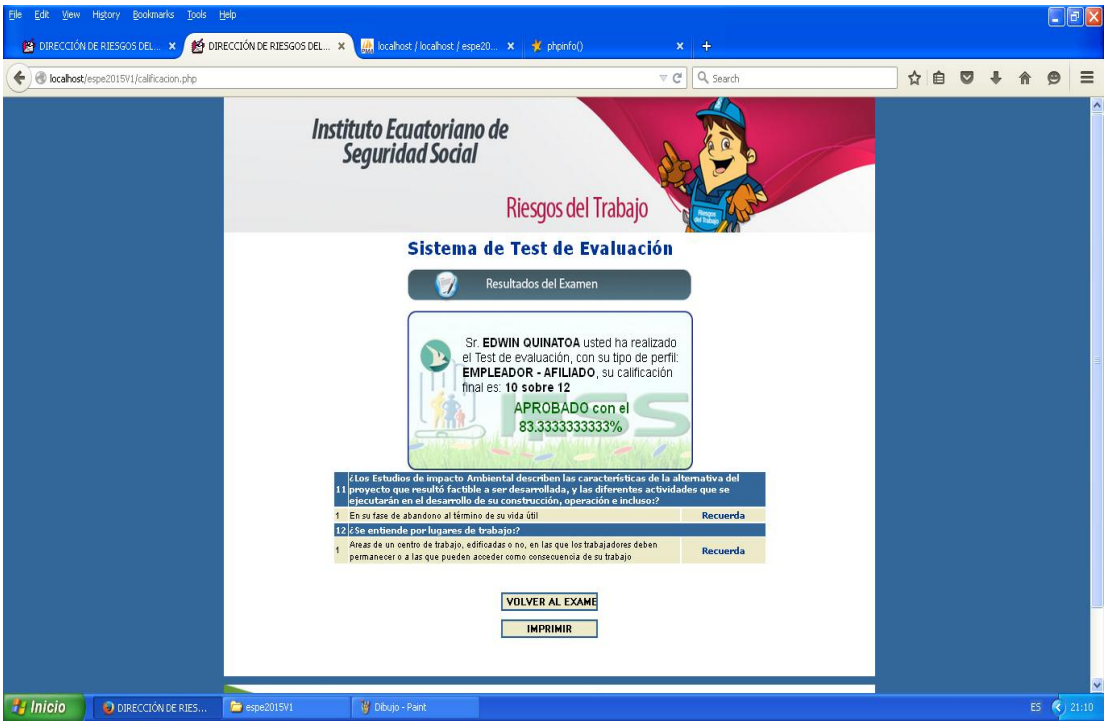


Figura 45: Resultados del Test

Para una descripción más detallada del manual técnico se especifica en el **Anexo K:** (Manual de Usuario)

Con todos estos artefactos desarrollados se pudo aplicar la metodología de trabajo desarrollada para el DNTI del IESS en un caso práctico como lo fue para la Dirección de Riesgos de Trabajo.

Según lo mencionado anteriormente se pudo llegar a generar un mejor entendimiento como en la parte de caso de negocio, especificación de requerimientos funcionales y sobre todo tener un mejor control en la parte de pruebas el cual permitió generar un óptimo código para la aplicación en base a la arquitectura que se estableció para trabajar.

4.6. Artefactos desarrollados con la Nueva Metodología de Trabajo

Tabla 9
Matriz de los artefactos desarrollados para el aplicativo

Fases	Iteraciones	Disciplinas	Artefactos	Roles	Observaciones	Anexo
Inicio	Uno	Modelado de Negocio	Caso del Negocio	Analista Funcional	Especifica el alcance del sistema.	A
			Requerimientos Funcionales	Analista de Procesos	Define la especificación funcional del requerimiento a desarrollar.	B
Elaboración	Dos	Requerimientos	Caso del Uso del Sistema V: <1>	Analista de Sistemas	El escenario es Rendir Tests	C
			Diagrama de Actividades		Se especifica como el sistema implementara su funcionalidad	No tiene
		Análisis y Diseño	Documento de Arquitectura de Software	Arquitecto de Sistemas	Se define VertrigoServ	D
			Prototipo	Analista de Sistemas	Se desarrolla para dar una presentación del aplicativo al cliente.	Apl. Inicial

Continua

Construcción	Tres	Requerimientos	Caso del Uso del Sistema V: <2>	Analista de Sistemas	Es necesario especificar el escenario Desplegué por preg.	E
		Análisis y Diseño	Documento de Arquitectura de Software (el mismo del anterior)	Arquitecto de Sistemas	No hubo cambios en la arquitectura de solución especificada	D
		Implementación	Diagrama de Clases	Analista de Sistemas	Se especificó la clase, atributos y métodos	No tiene
			Código Fuente V: <1>	Programador	En base a los requerimientos definidos	Apl. Final
		Pruebas	Plan de Pruebas V: <1>	Tester	Verificar el caso de uso de rendir test	F
	Cuatro	Requerimientos	Caso de Uso del Sistema V: <f>	Analista de Sistemas	No fue necesario generar otro porque ya no hubo cambios	C y E
		Análisis y Diseño	Documento de Arquitectura de Soft. (el mismo del anterior)	Arquitecto de Sistemas	No hubo cambios en la arquitectura de solución definido	D
		Implementación	Código Fuente V: <2>	Programador	Especificando en base a las pruebas generadas	Apl. Final
		Pruebas	Plan de Pruebas V: <2>	Tester	Verificar el caso de uso de Desplegué por pregunta	G
Transición	Cinco	Pruebas	Plan de Pruebas (pruebas unitarias) V: <f>	Tester	Se especifica las evidencias de las pruebas	H
		Despliegue	Documento de Despliegue	Arquitecto de Sistemas	Elementos para ponerle en ejecución el aplicativo	I
			Manual Técnico	Arquitecto de Sist.	Aspectos técnicos	J
			Manual de Usuario	Analista Funcional/Usuario F.	Aspectos para facilidad del Usuario o afiliado al IESS	K

En la presente tabla 9, se especifica todos los elementos utilizados y desarrollados como son: fases, iteraciones, disciplinas, artefactos y roles con la nueva metodología de trabajo aplicado a un caso real.

CAPÍTULO V

5. VALIDACIÓN DE LA METODOLOGÍA DE TRABAJO

5.1. Análisis de Aplicaciones Realizadas

Para dar validez a la metodología de trabajo realizado, se efectuó un análisis de varias aplicaciones realizadas antes en la Institución dentro de la DNTI las cuales utilizaron RUP tomando como referencia el tiempo de duración que conllevo desarrollar el aplicativo en lo que concierne dentro de las disciplinas de proceso y desde su fase de inicio hasta que el aplicativo salga a producción.

Posteriormente se realizó un análisis del tiempo que conllevo realizar el aplicativo denominado Test de Evaluación utilizando la nueva metodología de trabajo.

Tabla 10
Sistemas anteriores realizados

SISTEMAS ANTERIORES	
Nombre	**Tiempo (días)**
Calificación de derecho a salud	90
Simulador de pensiones	60
Auditoria médica para prestadores	180

Tabla 11
Sistemas Actuales realizados (Nueva Metodología)

SISTEMA ACTUAL		
Nombre	**Tiempo**	
	Fases	**Días**
Sistemas de Test de Evaluación	Inicio	7
	Elaboración	7
	Construcción	18
	Transición	5
Total		**37**

Tabla 12
Análisis de proyectos realizados

ANÁLISIS DE SISTEMAS REALIZADOS			
SISTEMAS ACTUALES		**SISTEMAS ANTERIORES**	
Nombre	**Total de días**	**Nombre**	**Total de días**
Sistemas de Test de Evaluación	37	Calificación de derecho a salud	90
		Simulador de pensiones	60
		Auditoria médica para prestadores	180

Se realiza el análisis de los proyectos que se han desarrollado anteriormente y actualmente (nueva metodología) del IESS, según lo especificado.

Figura 46: Análisis de Sistemas Desarrollados

Una vez realizado el análisis referente al tiempo que conlleva el desarrollo de las aplicaciones se concluye que se efectuó en menos lapso el Sistema de Test de Evaluación para el cual se aplicó la nueva metodología de trabajo.

5.2. Certificaciones de Sustentación de Investigación y Satisfacción

Certificaciones que abalizan y validan el tema de investigación realizada para la Institución del IESS de la ciudad de Quito dentro de la Dirección Nacional de Tecnología de la Información.

- Certificado de Sustentación de la Investigación

INSTITUTO ECUATORIANO DE SEGURIDAD SOCIAL
DIRECCIÓN NACIONAL DE TECNOLOGÍA DE LA INFORMACIÓN

Quito, 06 de Enero de 2016

CERTIFICADO

A quien interese:

A petición verbal del interesado Sr. **Edwin Edisón Quinatoa Arequipa** con número de cédula **0502563372**, certifico que el mencionado señor para complementar su trabajo de investigación **"DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO"**, se le facilito la información necesaria que sustenta los proyectos desarrollados dentro del IESS que fueron la base para el desarrollo de su investigación.

A continuación se describe los proyectos desarrollados utilizando la metodología RUP por la Dirección Nacional de Tecnología de Información.

Nombre	Descripción	Fechas de Desarrollo
Calificación de derecho a salud	Permite validar si un afiliado tiene o no derecho a los servicios del IESS.	Enero/2012 – Marzo/2012
Simulador de pensiones	Trata de Simular según los años de aportaciones y el valor aportado, cuánto va a recibir un afiliado cuando se jubile.	Agosto/2012 – Septiembre/2012
Auditoria médica para prestadores	Permite realizar auditorías a los servicios de salud que entregan los prestadores externos de salud.	Enero/2013 – Junio/2013

El señor Quinatoa, puede hacer uso del presente certificado como a bien tuviere.

INSTITUTO ECUATORIANO DE SEGURIDAD SOCIAL
DIRECCIÓN NACIONAL DE TECNOLOGÍA DE LA INFORMACIÓN

Atentamente,

Ing. Rodney Castro
COORDINADOR DE DESARROLLO

Elaborado por:	Ing. Rodney Castro
Aprobado por:	Ing. Rodney Castro
Fecha:	06 de Enero del 2016

- Certificado de Satisfacción

INSTITUTO ECUATORIANO DE SEGURIDAD SOCIAL
DIRECCIÓN NACIONAL DE TECNOLOGÍA DE LA INFORMACIÓN

Quito, 06 de Enero de 2016

CERTIFICADO

A quien interese:

A petición verbal del interesado Sr. **Edwin Edisón Quinatoa Arequipa** con número de cédula **0502563372**, certifico que el mencionado señor con su trabajo de investigación "**DESARROLLO DE UNA METODOLOGÍA DE TRABAJO BASADO EN RUP PARA LA CREACIÓN DE APLICACIONES DE SOFTWARE EN LA DIRECCIÓN DE DESARROLLO INSTITUCIONAL DEL IESS DE LA CIUDAD DE QUITO**", nos entregó la nueva metodología de trabajo desarrolladacon su respectiva documentación para la Institución, para lo cual se muestra el diseño de la nueva metodología definida a continuación:

FLUJOS DE TRABAJO (Proceso)	FASES				
	Inicio	Elaboración	Construcción		Transición
Modelado del Negocio					
Requerimientos					
Análisis y Diseño					
Implementación					
Pruebas					
Despliegue					
	I1	I2	I3	I4	I5

Nota: Cuando es opcional la aplicación de la Disciplina

* Es decir cuando hay cambio de arquitectura en base a los requerimientos y los casos de uso

* Cuando se determina otros casos de uso y va hasta el inicio y final

INSTITUTO ECUATORIANO DE SEGURIDAD SOCIAL
DIRECCIÓN NACIONAL DE TECNOLOGÍA DE LA INFORMACIÓN

Posteriormente se detalla los elementos definidos para la nueva metodología mediante una matriz.

Flujos de Trabajo	Artefacto	Rol
Modelado del Negocio	Caso del Negocio	Analista Funcional
	Requerimientos Funcionales	Analista de Procesos
Requerimientos	Casos del Uso del Sistema	Analista de Sistemas
	Diagramas de Actividades	
Análisis y Diseño	Documento de Arquitectura de Software	Arquitecto de Sistemas
	Prototipo	Analista de Sistemas
Implementación	Diagrama de Clases	Analista de Sistemas
	Código Fuente	Programador
Pruebas	Plan de Pruebas	Tester
Despliegue	Documento de Despliegue	Arquitecto de Sistemas
	Manual Técnico	
	Manual de Usuario	Analista Funcional / Usuario Final

Por todo lo especificado anteriormente se le extiende comedidamente un agradecimiento muy extenso, ya que esto será a partir de hoy una herramienta primordial para el desarrollo de las aplicaciones de software dentro de la Institución y de manera específica en la Dirección Nacional de Tecnología de Información, lo cual nos permitirá resolver uno de los problemas más engorrosos el tiempo que se tomaba para desarrollar un aplicativo.

El señor Quinatoa, puede hacer uso del presente certificado como a bien tuviere.

Atentamente,

Ing. Rodney Castro
COORDINADOR DE DESARROLLO

Elaborado por:	Ing. Rodney Castro
Aprobado por:	Ing. Rodney Castro
Fecha:	06 de Enero del 2016

5.3. CONCLUSIONES

- En la presente investigación realizada se logró definir una Metodología de Trabajo basado en RUP, en la cual se especificó los roles y artefactos, complementadas con fases, disciplinas enfocadas a la parte de desarrollo y iteraciones, las cuales permitieron dar una mejor solución a los problemas de gestión y desarrollo; y de esa manera acoplándose a las particularidades para el IESS.
- Se concluye que para validar la nueva Metodología de Trabajo fue necesario el desarrollo de un aplicativo de software enfocado a un caso real dentro del IESS de la ciudad de Quito, en esta ocasión en la Dirección de Riesgos de Trabajo a la cual se la denomino Sistema de Test de Evaluación.
- De la aplicación desarrollada se pudo llegar a determinar aspectos positivos y negativos, los cuales servirán de base para poder mejorar la Metodología de trabajo diseñada en un futuro posterior, dentro de la Dirección Nacional de Tecnología de la Información del IESS.

5.4. RECOMENDACIONES

- En base a los resultados obtenidos se recomienda que para poder especificar los requisitos de software se utilice el estándar IEEE 830.
- En futuras investigaciones sería necesario trabajar en la mejora de la Metodología de trabajo establecido para que abarque la parte de disciplinas de soporte que posee RUP, el cual le permitirá generar un mejor complemento a lo especificado.
- En base a los resultados obtenidos se recomienda poder generar más artefactos con sus respectivos roles complementados con plantillas de trabajo, las cuales permitirán lograr obtener una mejor gestión y desarrollo de software en las aplicaciones posteriores y encaminar para poder llegar a la perfección.

GLOSARIO UTILIZADO

ACPs: Áreas clave de proceso / Key Process Areas

Ad Hoc: Una herramienta elaborada específicamente para una determinada ocasión o situación.

AESOFT: Asociación Ecuatoriana de Software

Bipartita: La componen dos grupos, la primera mitad es representante de la empresa y la otra mitad es representante de los trabajadores.

CASE: Ingeniería del Software Asistida por Computadora / Computer-Aided Software Engineering

Coexistir: Es el balance que debe existir entre 2 o más artefactos diferentes.

DDI: Dirección de Desarrollo Institucional

DNTI: Dirección Nacional de Tecnología de la Información

ESPOL: Universidad de la Escuela Politécnica del Litoral

Hito: Es un conjunto de modelos o documentos que han sido desarrollados hasta alcanzar un estado predefinido.

IEEE: Instituto de Ingenieros Eléctricos y Electrónicos **/** Institute of Electrical and Electronics Engineers

IESS: Instituto Ecuatoriano de Seguridad Social

LOC: Líneas de código / Lines of Code

Marco de trabajo: Son múltiples herramientas, modelos y métodos para ayudar en el proceso de desarrollo de software.

MSF: Microsoft Solutions Framework

OMG: Grupo de Gestión de Objetos / Object Management Group, es una corporación sin ánimo de lucro pensada para facilitar la especificación de tecnología del software.

OTAN: Organización del Tratado del Atlántico Norte / North Atlantic Treaty Organization

Rastreabilidad o Trazabilidad: Es la capacidad de seguir una representación del diseño o un componente real del programa hasta los requerimientos.

ROP: Rational Objectory Process

RUP: Proceso Unificado de Desarrollo de Software / Rational Unified Process

Subsidiariedad: Es un principio según el cual el gobierno de un país, ayuda a los estados pequeños.

UML: Lenguaje de Modelado Unificado / Unified Modelling Language

VLIR: Universidades Flamengas de Bélgica

XP: Programación extrema / Extreme Programming

BIBLIOGRAFÍA

Anaya, A. (s.f.). *A propósito de programación extrema XP (eXtreme Programming)*. Recuperado el 26 de Septiembre de 2013, de Monografías: http://www.monografias.com/trabajos51/programacion-extrema/programacion-extrema2.shtml

Arevalo, M. E. (s.f.). *MSF: Microsoft Solution Framework*. Recuperado el 24 de Septiembre de 2013, de Introduccion a la Programacion, ITIL, CMMI, PMBOOK, MOF, MSF, ISO 20000: http://arevalomaria.wordpress.com/2010/01/07/msf-microsoft-solution-framework/

Bermeo, F. (8 de diciembre de 2010). *Metodología RUP*. Obtenido de Desarrollo de software de calidad: http://fabianbermeop.blogspot.com/2010/12/metodologia-rup-desarrollo-de-software.html

Boni, M., Caceres, C., Dueñas, D., Proaño, M., & Trelles, G. (2006). *Industria Local del Software Ecuador*. Recuperado el 18 de Junio de 2013, de Genrencia de sistemas septimo: http://industrialocaldelsoftware.blogspot.com/

Booch, G., Jacobson, I., & Rumbaugh, J. (2007). *El Lenguaje Unificado de Modelado* (Segunda ed.). Madrid, España: Pearson Educación S. A.

Buades Rubio, G. (15 de Mayo de 2003). *Calidad en Ingeniería de Software*. Obtenido de Ingeniería del Software III: http://dmi.uib.es/~bbuades/index.html

Canchala, L. A. (s.f.). *UML, ejemplo sencillo sobre Modelado de un Proyecto*. Obtenido de MSDN: http://msdn.microsoft.com/es-es/library/bb972214.aspx#XSLTsection129121120120

Canós, J., Letelier, P., & Penadés, C. (s.f.). *Metodologías de Ágiles en el Desarrollo de Software.* Recuperado el 12 de Septiembre de 2013, de Universidad Politécnica de Valencia: http://noqualityinside.com.ar/nqi/nqifiles/XP_Agil.pdf

Casallas, R. (s.f.). *Alguno mitos de la Ingeniería de Software.* Recuperado el 30 de Agosto de 2013, de Aún en Crisis: http://www.acis.org.co/fileadmin/Revista_102/columnista.pdf

Castañeda Cadena, L. A., & Quezada Sarasti, W. R. (Febrero de 2007). *Aplicación de la norma técnica ISO 27001:2005, para la gestión de la seguridad de la información en la dirección de desarrollo.* Obtenido de Escuela Politécnica Nacional (Tesis de Grado): http://bibdigital.epn.edu.ec/bitstream/15000/1304/1/CD-0653.pdf

Coba Martínez, C. R., Macias, M. V., Reinoso Espinosa, J. A., & Vivanco Andrade, J. A. (s.f.). *Análisis de los modelos de calidad de software existentes y su Apoyo al cumplimiento de requerimientos en empresas no dedicadas al desarrollo de software.* Recuperado el 22 de junio de 2013, de Universidad de la Escuela Politécnica del Litoral: http://www.dspace.espol.edu.ec/bitstream/123456789/909/1/1660.pdf

EL UNIVERSO. (29 de Noviembre de 2007). *IESS no logra solucionar ni determinar daños en su sistema informático*. Recuperado el 4 de Julio de 2013, de EL UNIVERSO: http://www.eluniverso.com/2007/11/29/0001/9/33D39A3CEABB4B929076DCE89A47CE9D.html

Ferraro, M. (s.f.). *Modelamiento Orientado a Objetos*. Recuperado el 29 de Septiembre de 2013, de Analisis de Sistemas II: http://exa.unne.edu.ar/informatica/anasistem2/public_html/apuntes/maf/cap4.htm

Flórez Fernández, H. A. (15 de Mayo de 2009). *Procesos de ingeniería de software.* Obtenido de Software engineering processes: http://www.konradlorenz.edu.co/images/investigaciones/matematicas/ingenieria_de_software.pdf

Gamboa Rodríguez, P. (30 de Agosto de 2012). *Modelos UML*. Obtenido de UML Modelos de los Datos: http://www.slideshare.net/PattyGamboaRodrguez/uml-modelado-de-datos

González, J. (23 de Septiembre de 2008). *QUE ES LA PROGRAMACION EXTREMA XP*. Obtenido de Gestión de Proyectos: http://proyecto8santotomas.blogspot.com/2008/09/que-es-la-programacion-extrema-xp.html

IESS - Dirección de Desarrollo Institucional. (2013). *Términos de Referencia para la "Contratación de bienes y servicios para la Implantación de Mecanismos de Seguridad y segundo factor transaccional.* Recuperado el 2 de Julio de 2013, de IESS: http://www.iess.gob.ec/documents/10162/0/TR22489_2013-01-18_DDI-GTI-0002-2013.pdf

IESS. (s.f.). *Reglamento Organico Funcional del Instituto Ecuatoriano de Seguridad Social.* Resolución CD. 021.

Ingeniería de Software. (18 de Agosto de 2012). *Microsoft Solutions Framework (MSF)*. Obtenido de Iteraciones de Software: http://codeoptimizations.com/microsoft-solutions-framework-msf/

Instituto Ecuatoriano de Seguridad Social. (2013). *Quienes Somos*. Recuperado el 10 de Julio de 2013, de IESS: https://www.iess.gob.ec/es/inst-quienes-somos

Instituto Nacional de Tecnologías de la Comunicación. (Marzo de 2009). *INGENIERÍA DEL SOFTWARE: METODOLOGÍAS Y CICLOS DE VIDA.* Obtenido de www.inteco.es/file/N85W1ZWFHifRgUc_oY8_Xg

Jacobson, I., Booch, G., & Rumbaugh, J. (2000). *Guia Completa de el Proceso Unificado de Desarrollo de Software* (Primera ed.). Madrid, España: Pearson Education S. A.

Kendall, K. E., & Kendall, J. E. (2005). *Análisis y Diseño de Sistemas* (Sexta ed.). Atlacomulco, México: Pearson Education.

Lamas, L. (s.f.). *Expertos en Colaboración e Integración*. Recuperado el 24 de Septiembre de 2013, de Itsystems: http://www.itsystems.com.uy/Pages/Company/Metodology.aspx?id=2

León Serrano, G. (1996). *Ingeniería de Sistemas de Software* (Primera ed.). Madrid, España: Isdefe.

Letelier Torres, P., & Sánchez Palma, P. (05 de Febrero de 2002). *Análisis y Diseño Orientado a Objetos usando la notación UML.* Obtenido de Universidad Politécnica de Valencia (UPV) - España: http://adimen.si.ehu.es/~rigau/teaching/EHU/ISO/Altres%20Cursos/Montoyo/Curso%20OO%20con%20UML.pdf

Lorés, J., & Granollers, T. (2002). *La Ingeniería de la Usabilidad y de la Accesibilidad aplicada al diseño y desarrollo de sitios web.* Lleida.

Martínez Cobo, R. (s.f.). *Metodologías de desarrollo ágil - Melé (Scrum)*. Recuperado el 27 de Septiembre de 2013, de Monografías: http://www.monografias.com/trabajos91/metodologias-desarrollo-agil-mele-scrum/metodologias-desarrollo-agil-mele-scrum.shtml

Microsoft. (2013). *Descripción general de Microsoft Solutions Framework (MSF)*. Obtenido de MSDN: http://msdn.microsoft.com/es-es/library/jj161047.aspx

Pressman, R. (2002). *Ingeniería del Software: Un Enfoque Práctico* (Quinta ed.). Madrid, España: McGraw-Hill.

ProChile en Ecuador. (2012). *Estudio de Mercado Servicio Desarrollo de Software en Ecuador.* Obtenido de ProChile: http://www.prochile.gob.cl/wp-content/blogs.dir/1/files_mf/documento_11_19_12112936.pdf

Reinoza, M. (9 de Mayo de 2011). *Evolución de la Ingeniería del Software*. Obtenido de Algunas cosas sobre el Internet y el Software Libre: http://internetenlasorganizacionespornela.blogspot.com/2011/05/evolucion-de-la-ingenieria-del-software.html

Schach, S. (2006). *Ingeniería de Software Clásica: Orientada a Objetos* (Sexta ed.). Distrito Federal, México: McGraw-Hill.

Schwaber, K., & Sutherland, J. (Julio de 2013). *La Guía Definitiva de Scrum:Las Reglas del Juego.* Obtenido de La Guía de Scrum:

https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide-ES.pdf#zoom=100

Sommerville, I. (2005). *Ingeniería del software* (Séptima ed.). Madrid, España: PEARSON EDUCACIÓN.

Taborda, D. (12 de Agosto de 2012). *Microsoft Solutions Framework (MSF)*. Obtenido de Ingeniería de software y tecnologías de información.: http://codeoptimizations.com/microsoft-solutions-framework-msf/

Tituana Vera, F. G. (2009). *Análisis de métodos, técnicas y herramientas de verificación y validación de software usados por empresas ecuatorianas desarrolladoras de software.* Obtenido de Escuela Superior Politécnica del Litoral (Tesis de Grado): http://www.dspace.espol.edu.ec/bitstream/123456789/7735/1/D-39464.pdf

Universidad Distrital Francisco José de Caldas. (05 de 2008). *Visión Electrónica*. Obtenido de Software para Gestión y Administración: http://revistas.udistrital.edu.co/ojs/index.php/visele/article/view/795/1086

Universidad Politecnica de Valencia. (s.f.). *Rational Unified Process (RUP)*. Obtenido de http://es.slideshare.net/oscar8711/introduccin-a-rup-presentation

Universidad Union Bolivariana. (2013). *Ingenieria de Software*. Obtenido de El Modelo en V o de Cuatro Niveles: http://ingenieriadesoftware.mex.tl/61885_Modelo-V.html

Weitzenfeld, A. (2005). *Ingeniería de Software Orientado a Objetos con UML, Java e Internet* (Segunda ed.). Distrito Federal, México: International Thomson Editores S. A.

ANEXOS

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por el señor: **EDWIN QUINATOA AREQUIPA**

En la ciudad de Latacunga, a los **07 días del mes de Enero del 2016.**

Ing. Marcelo Rea Guamán MSc.
DIRECTOR DE PROYECTO

Aprobado por:

Ing. Lucas Garcés MSc.
COORDINADOR DE LA MAESTRÍA DE SOFTWARE
"TERCERA PROMOCIÓN"

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO