UNIVERSIDAD HISPANOAMERICANA
INGENIERÍA INFORMÁTICA

TESINA PARA OPTAR POR EL GRADO DE BACHILLERATO EN LA CARRERA DE INGENIERÍA INFORMÁTICA

PROPUESTA DE METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE SOFTWARE PARA EL SISTEMA INTEGRAL MÉDICO ADMINISTRATIVO DEL INSTITUTO NACIONAL DE SEGUROS

Sustentante:
Freddy Esquivel Fuentes

Tutor:
Agustín Francesa Alfaro

Septiembre, 2018

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

DECLARACIÓN JURADA

DECLARACIÓN JURADA

Yo Freddy Esquivel Fuentes, mayor de edad, portador de la cédula de identidad número 1-1286-0504 egresado de la carrera de Ingeniería Informática de la Universidad Hispanoamericana, hago constar por medio de éste acto y debidamente apercibido y entendido de las penas y consecuencias con las que se castiga en el Código Penal el delito de perjurio, ante quienes se constituyen en el Tribunal Examinador de mi trabajo de tesis para optar por el título de Bachillerato en Ingeniería Informática, juro solemnemente que mi trabajo de investigación titulado: Propuesta de Metodología de Desarrollo y Mantenimiento de Software para el Sistema Integral Médico Administrativo del Instituto Nacional de Seguros, es una obra original que ha respetado todo lo preceptuado por las Leyes Penales, así como la Ley de Derecho de Autor y Derecho Conexos número 6683 del 14 de octubre de 1982 y sus reformas, publicada en la Gaceta número 226 del 25 de noviembre de 1982; incluyendo el numeral 70 de dicha ley que advierte; artículo 70. Es permitido citar a un autor, transcribiendo los pasajes pertinentes siempre que éstos no sean tantos y seguidos, que puedan considerarse como una producción simulada y sustancial, que redunde en perjuicio del autor de la obra original. Asimismo, quedo advertido que la Universidad se reserva el derecho de protocolizar este documento ante Notario Público.

En fe de lo anterior, firmo en la ciudad de San José, a los diecisiete días del mes de marzo del año dos mil dieciocho.

Firma del estudiante

Cédula: 1-1286-0504

CARTAS DE APROBACIÓN DEL TUTOR

Y CONTRAPARTE

CARTA DEL TUTOR

Cartago, 16 de marzo de 2018

Señora Yenory Rojas Hernández
Directora Ingeniería Informática
Universidad Hispanoamericana

Estimado señora:

El estudiante FREDDY ALBERTO ESQUIVEL FUENTES, cédula de identidad número 112860504, me ha presentado, para efectos de revisión y aprobación, el trabajo de investigación denominado: PROPUESTA DE METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE SOFTWARE PARA EL SISTEMA INTEGRAL MÉDICO ADMINISTRATIVO DEL INSTITUTO NACIONAL DE SEGUROS., el cual ha elaborado para optar por el grado académico de Bachillerato en Ingeniería Informática.

En mi calidad de tutor, he verificado que se han hecho las correcciones indicadas durante el proceso de tutoría y he evaluado los aspectos relativos a la elaboración del problema, objetivos, justificación; antecedentes, marco teórico, marco metodológico, tabulación, análisis de datos; conclusiones y recomendaciones.

De los resultados obtenidos por el postulante, se obtiene la siguiente calificación:

a)	ORIGINAL DEL TEMA	10%	9
b)	CUMPLIMIENTO DE ENTREGA DE AVANCES	20%	19
C)	COHERENCIA ENTRE LOS OBJETIVOS, LOS INSTRUMENTOS APLICADOS Y LOS RESULTADOS DE LA INVESTIGACION	30%	28
d)	RELEVANCIA DE LAS CONCLUSIONES Y RECOMENDACIONES	20%	19
e)	CALIDAD, DETALLE DEL MARCO TEORICO	20%	19
	TOTAL		94

En virtud de la calificación obtenida, se avala el traslado al proceso de lectura.

Atentamente,

Ing. Agustín Francesa Alfaro
113470859
CPIC #6233

CARTA DE LECTOR

Universidad Hispanoamericana
Carrera Ingeniería Informática

Estimado señor

El estudiante Freddy Esquivel Fuentes, cédula de identidad: 1-1286-0504, me ha presentado para efectos de revisión y aprobación, el trabajo de investigación denominado " PROPUESTA DE METODOLOGÍA DE DESARROLLO Y MANTENIMIENTO DE SOFTWARE PARA EL SISTEMA INTEGRAL MÉDICO ADMINISTRATIVO DEL INSTITUTO NACIONAL DE SEGUROS ", el cual ha elaborado para obtener su grado de Bachillerato en Ingeniería Informática.

He revisado lo relativo a la coherencia entre el marco teórico y análisis de datos, la consistencia de los datos recopilados y la coherencia entre éstos y las conclusiones; asimismo, la aplicabilidad y originalidad de las recomendaciones, en términos de aporte de la investigación.

Por consiguiente, este trabajo cuenta con mi aval para ser presentado en la defensa pública.

Atte.

Firma 

Nombre Ing. Luis Navarro Sánchez

Cédula 1-0484-0537

Cartago, 5 de junio de 2018

Señores
Universidad Hispanoamericana

Estimados señores:

Leí y corregí la tesina titulada: *"Propuesta de metodología de desarrollo y mantenimiento de software para el Sistema Integral Médico Administrativo del Instituto Nacional de Seguros"*, elaborado por el estudiante Freddy Esquivel Fuentes, cédula de identidad número 01 1286 0504, para optar por el grado de Bachillerato en la carrera de Ingeniería Informática.

Corregí el trabajo en aspectos tales como: construcción de párrafos, vicios del lenguaje que se trasladan a lo escrito, ortografía, puntuación y otros relacionados con el campo filológico, por lo que considero que desde ese punto de vista, está listo para ser presentado como Trabajo Final de Graduación.

Se suscribe, cordialmente,

Keren Pereira Tencio
Filóloga española
Miembro de la Asociación
Costarricense de Filólogos
Carné ACFIL 0187
Cédula 03 0412 0140

DEDICATORIA

Primeramente a Dios, quien día a día me muestra el camino para salir adelante y me honró con la mejor familia, incondicional, leal y respetuosa; a mis papás y a mis hermanas que son el pilar de mi vida y a todos los que creen en mí, que se sacrifican conmigo, que me aman y que yo amaré hasta el final.

AGRADECIMIENTO

Les agradezco a todas las personas involucradas en este proceso: profesores, compañeros de trabajo y las personas más cercanas a mi vida, quienes me acompañaron y se sacrificaron conmigo para culminar este proyecto.

A mi tutor, Agustín Francesa Alfaro, que con su conocimiento, experiencia y paciencia hizo de este proyecto un éxito.

Finalmente, extiendo un agradecimiento especial a la Subdirección de Informática del Instituto Nacional de Seguros, en el área de Mantenimiento y Desarrollo de Sistemas, por darme la oportunidad de llevar a cabo este proyecto en tan prestigiosa entidad.

CÁPITULO I

PROBLEMA DEL PROYECTO

1.1 ANTECEDENTES Y JUSTIFICACIÓN DEL PROYECTO

1.1.1 Marco de referencia empresarial y contextual

1.1.1.1 Información general de la empresa

1.1.1.1.1 Reseña histórica

El Instituto Nacional de Seguros se fundó el 30 de octubre de 1924, en sus inicios se llamó Banco Nacional de Seguros y en mayo de 1948, cambió su nombre a Instituto Nacional de Seguros (INS), mismo que se mantiene en la actualidad.

El INS tuvo a su cargo la administración del monopolio de los seguros desde su creación hasta el 07 de agosto del 2008, fecha en que entró en vigencia la Ley número 8653 “Ley Reguladora del Mercado de Seguros”, la cual abrió el mercado y permitió la competencia, no obstante, sigue siendo líder indiscutible en el mercado de seguros a nivel nacional y ejemplo de empresas entrantes.

Con la evolución del mercado y la lógica del negocio, la institución ha conformado a lo largo del tiempo un Grupo Financiero debidamente estructurado, que está compuesto por las siguientes entidades: el Hospital del Trauma, INS Valores, INS Inversiones, INS Servicios y el Benemérito Cuerpo de Bomberos de Costa Rica, que juntos comparten una ideología de excelente servicio y colaboración entre sí.

A continuación, se detalla información importante de la institución.

1.1.1.1.1.1 Misión

Es la empresa del Estado que ofrece seguros, reaseguros y servicios complementarios en el mercado nacional e internacional, que promueve la inversión, la cultura de seguros y la prevención de riesgos e impulsa el bienestar económico y social del país.

1.1.1.1.1.2 Visión

Lograr en el periodo del 2015 al 2018 un crecimiento no menor al del mercado en primaje de seguros y reaseguros, en el mercado nacional e internacional, fortaleciendo el liderazgo en el mercado, siendo reconocida como el modelo a seguir entre las empresas aseguradoras y las del sector público y como promotor de la cultura de seguros y prevención en todo el país, mediante una gestión eficiente de alta calidad, con tecnología de información y conectividad que simplifique las operaciones y fidelice las relaciones con los asegurados y los intermediarios.

1.1.1.1.2 Propuesta de valor.

1.1.1.1.2.1 Para el cliente.

Nuestros productos y servicios atienden las necesidades de protección del cliente, se ofrecen mediante trámites ágiles, con una calidad superior y en menor tiempo.

1.1.1.1.2.2 Para la sociedad costarricense.

Nuestro accionar tendrá un componente claro de creación de valor público, para la sociedad costarricense, estamos comprometidos con la prevención de riesgos, la salud, el bienestar colectivo y la responsabilidad social empresarial.

1.1.1.1.2.3 Establecimiento de valores

- Ética: actuaremos con integridad, transparencia, rectitud, coherencia, cumplimiento de la legalidad, honestidad y los valores del ser costarricense.
- Calidad: haremos que los productos y servicios se presten bajo estándares de calidad definidos y validados por los clientes.
- Espíritu de servicio: brindaremos servicio a nuestros clientes aportando conocimiento y experiencia, velando por satisfacer sus necesidades y expectativas, aplicando los principios de mejora continua y eficiencia esperada.

- Comunicación: nos comunicaremos de forma continua, respetuosa, transparente y oportuna con nuestros clientes, intermediarios, colaboradores y con la sociedad costarricense.
- Proactividad: tomaremos acciones sobre las oportunidades que se presentan a diario; a través del trabajo en equipo lograremos prever, intuir y actuar de manera positiva sobre todas las oportunidades de mejora.

1.1.1.1.3 Organigrama institucional

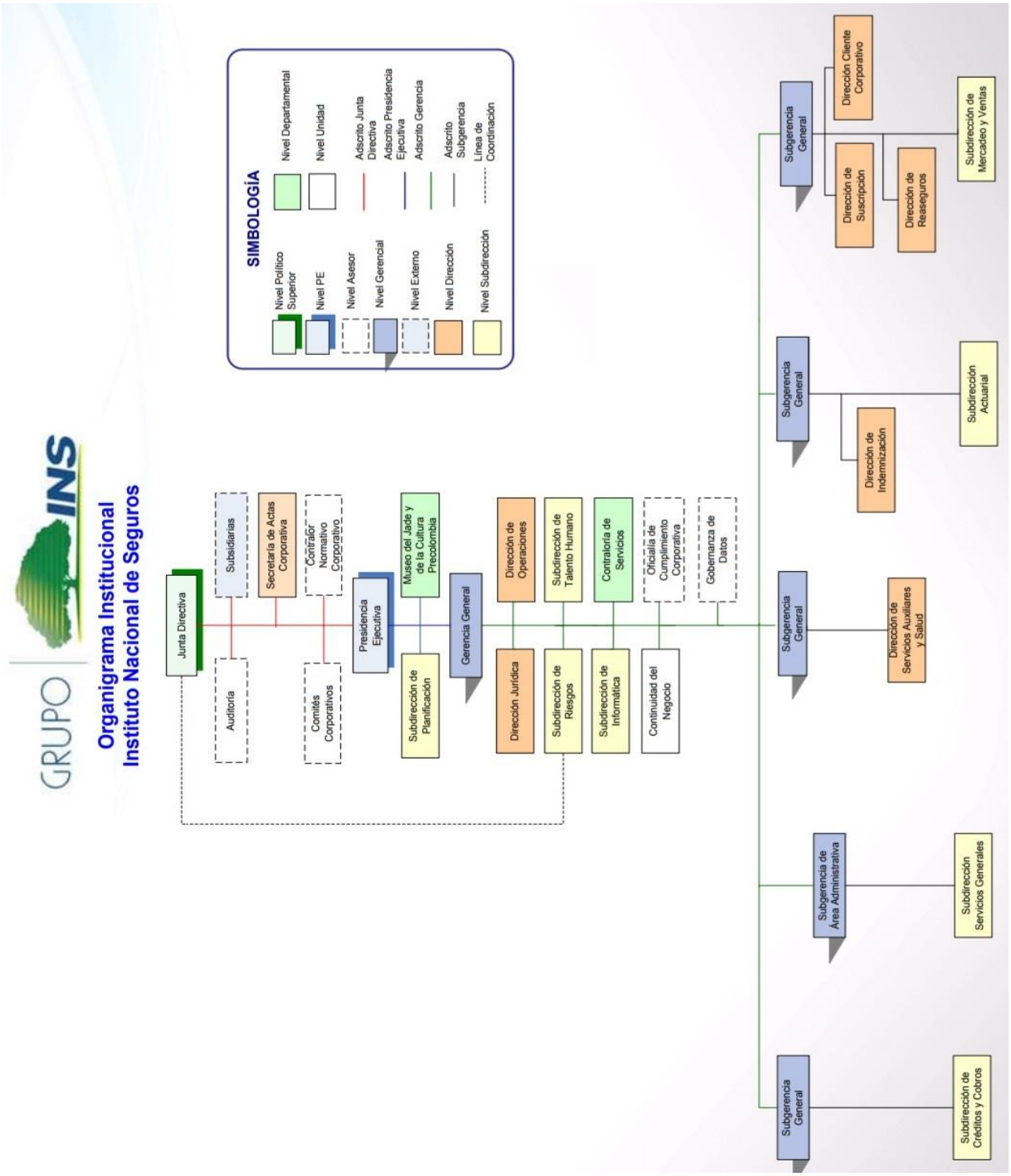


Figura 1
Organigrama Instituto Nacional de Seguros.
Fuente: Disponible en: “http://portal.ins-cr.com”

1.1.1.1.4 Subdirección de Informática y departamento de Mantenimiento y Desarrollo de Sistemas

La Subdirección de Informática es, dentro del proceso de Mantenimiento y Desarrollo de Sistemas (MSI-DSI), el escalón más alto de la jerarquía, es quien rige las normas, recursos y manera de proceder para cada uno de los proyectos que el Instituto Nacional de Seguros tenga la necesidad de abarcar, tanto a nivel de la Unidad Funcional como para el departamento de MSI-DSI.

El departamento de MSI-DSI para SIMA (Sistema Integral Médico Administrativo) está compuesto actualmente de un Administrador de Proyectos, que es la representación frente a la subdirección, y este, a su vez, cuenta con un líder de producto que es el encargado de canalizar cada una de las tareas que se remitan para el departamento.

MSI-DSI para SIMA cuenta con un total de 10 recursos para los procesos de análisis, mantenimiento y desarrollo de software, actualmente estos procesos incluyen toda la parte correspondiente a la atención de incidentes, manejo de versiones, desarrollo, pruebas, puestas en producción y seguimiento. Cabe aclarar que el mantenimiento de software se considera como un desarrollo evolutivo del sistema, por lo tanto, las prácticas utilizadas en el mantenimiento de software son exactamente las mismas que se utilizan en la creación de nuevas soluciones.

.

1.1.1.1.5 Sistema Integral Médico Administrativo

Siendo el INS un proveedor de servicios de salud, desde sus inicios promueve la creación de Centros Médicos Regionales (CRM), capaces de atender en un principio necesidades básicas de sus usuarios y valoraciones correspondientes, siendo en el año 2013 cuando se da la apertura el Hospital del Trauma que es capaz de solventar todas las necesidades de sus asegurados, en su mayoría consecuencia de accidentes de tránsito.

Desde junio del año 2005 se puso en producción el Sistema Integral Médico Administrativo (SIMA), que permite la integración del área de salud en toda su cobertura nacional; desde ese momento, se establecen las unidades de Desarrollo y Mantenimiento para este *software*.

El SIMA está desarrollado bajo la plataforma ORACLE, tanto a nivel de base de datos como de aplicativos, entre los cuales se encuentran formularios, reportes y librerías.

1.1.1.1.6 Unidad Funcional de Seguros Solidarios e INS Salud

Esta unidad o departamento es parte vital de la investigación, la misma está compuesta por usuarios expertos en cada uno de los módulos del SIMA.

Es por medio de esta unidad que se realiza el proceso de manejo de incidencias y es el primer nivel posterior a la solicitud realizada por el usuario final. Tiene la responsabilidad de descartar, inicialmente, toda posibilidad de solución a nivel de sistema como parametrización, prácticas erróneas del usuario, entre otros.

Posterior a su análisis, la Unidad Funcional tiene como responsabilidad canalizar la incidencia a quien corresponda, llevando a cabo el seguimiento, aprobación, pruebas y cualquier procedimiento correspondiente para dar por finalizada la atención del caso de manera correcta, hasta llegar al usuario que hizo la solicitud. Dentro del Instituto Nacional de Seguros, el proceso anterior se maneja con una herramienta propia, desarrollada para la necesidad específica y bajo la lógica del negocio, los detalles de esta herramienta se detallan en el siguiente apartado.

1.1.1.2 Tendencia del mercado

Si bien es cierto, el INS es una compañía dedicada a la venta de seguros, el enfoque de este proyecto se basa en la propuesta de una metodología de desarrollo de software, por lo tanto, es indispensable analizar la tendencia del mercado en relación con las metodologías de desarrollo mayormente utilizadas.

En la actualidad, existen modelos de madurez, estándares y metodologías que funcionan como guía para ayudar a las organizaciones a mejorar el modo en que realizan los procesos que conforman el negocio (Software Enginnering Institute, 2010).

Dichos estándares o metodologías pueden definirse como la base necesaria para la ejecución de cualquier proyecto de desarrollo de s*oftware* que se considere serio y que necesite sustentarse en algo más que la experiencia y capacidades de los desarrolladores de una empresa.

Conforme pasan los años, más empresas a nivel mundial implementan metodologías de desarrollo de *software* que les permitan mejorar la calidad del producto que se desarrolla, acelerar los tiempos de entrega y aumentar la productividad.

Desde el año 2005, la empresa VersionOne realiza una encuesta anual a una gran cantidad de empresas donde se demuestra cuáles son las metodologías ágiles más utilizadas a nivel mundial, el último informe fue arrojado en abril del 2016 y demostró que el método más utilizado actualmente es Scrum con un 58%, un porcentaje bastante amplio sobre el segundo lugar que le pertenece al híbrido entre Scrum y XP; además, un dato bastante curioso y a la vez preocupante es que el 2% de los encuestados asegura no saber qué metodología están utilizando en sus empresas.

En la siguiente figura se visualiza el gráfico donde se muestra el total de los resultados de la encuesta antes mencionada:

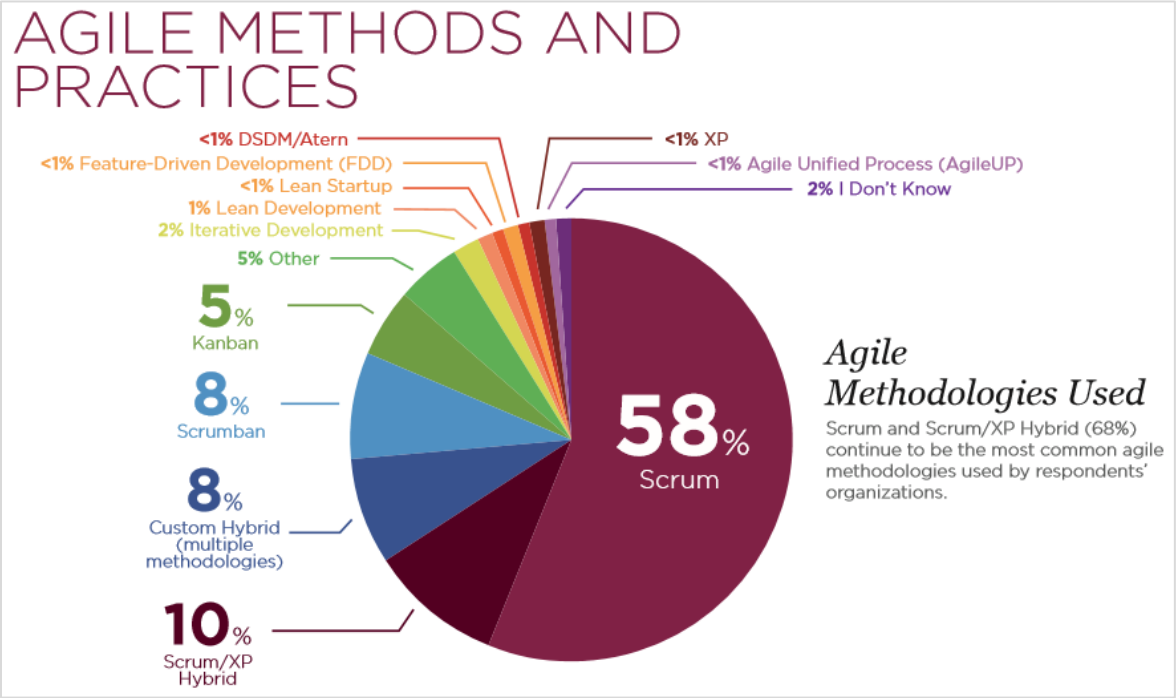


Figura 2
Ranking metodologías de desarrollo ágiles más utilizadas.
Fuente: disponible en: “https://www.versionone.com”

En Costa Rica, las metodologías ágiles también han adquirido gran relevancia y varias empresas han decidido ir por ese camino con el fin de crear *software* de alta calidad y poder incursionar en un mercado altamente competitivo, empresas como: Ciris Informatic Solutions, GTS Expertos en *Software*, Rossmon Technology, Avantica, entre otras, difunden con gran empeño la utilización de metodologías ágiles en los productos que ofrecen, ya que son conscientes de que en la actualidad la diferencia entre una empresa y otra es marcada por la calidad, productividad y precio de sus productos.

Adicionalmente, se ha identificado que el tema sobre propuestas de metodologías de desarrollo de *software*, es recurrente en trabajos de investigación tales como:

- Tesis realizada para mejorar la gestión de requerimientos de sistemas en el Instituto de Desarrollo Rural, cuya principal propuesta se basaba en:

  “crear un modelo de calidad para la gestión de requerimientos en desarrollo por seguir en el Área de Sistemas, para determinar todos los procedimientos necesarios para atacar el problema desde su origen y hasta la entrega final de cada proyecto, con el fin de asegurar los resultados esperados por la Institución”. (Ocampo, 2016).

- Tesina en la cual el objetivo principal consiste en: “Diseñar una metodología de actualización de software en el departamento de Tecnologías de Información que optimice los períodos de actualización en los servidores de la empresa Clinverse S.A. durante el primer semestre del 2016”. (Chanto, 2016).

- Tesina realizada para la estandarización de procesos del Ministerio de Ambiente y Energía, cuyo objetivo principal fue: “Analizar, mejorar y estandarizar los procesos principales del Departamento de Control y Protección de Vi da Silvestre del Minae [sic]”. (Marchena, 2013).

Según lo expuesto anteriormente, es posible demostrar que el uso de Metodologías de Desarrollo va en aumento y que el hecho de contar con una de ellas aporta gran valor a la calidad y productividad general del *software* que se desarrolla.

1.1.1.3 Justificación del proyecto

Es necesario realizar este proyecto debido a que el área que brinda soporte al SIMA no cuenta con una metodología de desarrollo clara y documentada, donde se establezcan las directrices, criterios y normas necesarias para brindar a todo el grupo de trabajo un modelo enfocado a la productividad y calidad de los productos desarrollados.

Place (2009) define la utilización de estándares o modelos de desarrollo de la siguiente forma:

> Peor que usar un mal estándar o un estándar incorrecto es no seguir ninguno, de la misma forma, lo peor que podemos hacer es no tener ningún criterio para enfrentar los desarrollos. Para aumentar nuestra productividad, tanto individualmente como en equipo, debemos siempre seguir estándares y fijar criterios de desarrollo. (Place, 2009, p. 25)

La competencia, evolución del mercado de desarrollo de *software*, además de los conceptos básicos de calidad, coste, eficiencia y flexibilidad, ha hecho que la empresa requiera la utilización de una metodología de desarrollo para poder ofrecer un mejor desempeño en la creación de proyectos para poder alcanzar los objetivos planeados y satisfacer las necesidades del usuario.

Para reforzar lo indicado anteriormente, se ha demostrado como una gran cantidad de compañías a nivel mundial, se han interesado en establecer modelos de estandarización en sus procesos de desarrollo, por ejemplo: el modelo CMMI-DEV, que es utilizado para la mejora de las distintas áreas de proceso en los proyectos de desarrollo y mantenimiento de *software*.

En la siguiente tabla se muestra el incremento de empresas que a nivel mundial han sido certificadas en el modelo CMMI-DEV:

Tabla 1

Ranking mundial en certificaciones CMMI-DEV ver1.3

País	Nivel 2	Nivel 3	Total
China	20	1843	**1.863**
Estado Unidos	267	639	**906**
India	14	381	**395**
México	110	84	**194**
España	58	62	**120**
República de Corea	29	85	**114**
Brasil	48	52	**100**
Colombia	5	64	**69**
Japón	13	42	**55**
Francia	32	21	**53**
TOTAL	**596**	**3273**	**3.869**

Fuente: León, Gonzales, Hernández & Medina (2013), disponible en: https://sas.cmmiinstitute.com.

En la tabla 1 se puede observar un incremento del 254% en certificaciones de nivel 2 y 3 entre el año 2013 y 2015, lo cual indica que cada día crece más el número de empresas preocupadas por mejorar los procesos y realizar productos de calidad.

Según el estudio Mapeo Sectorial de Tecnologías Digitales (2015), realizado por la Cámara de Tecnologías de Información y Comunicación (Camtic) y la Promotora de Comercio Exterior de Costa Rica (Procomer), se puede observar que incluso a nivel nacional varias empresas han optado por certificaciones, modelos o estándares de control y gestión de la calidad como mecanismos para fortalecer los procesos y asegurar la entrega de productos de calidad a los usuarios.

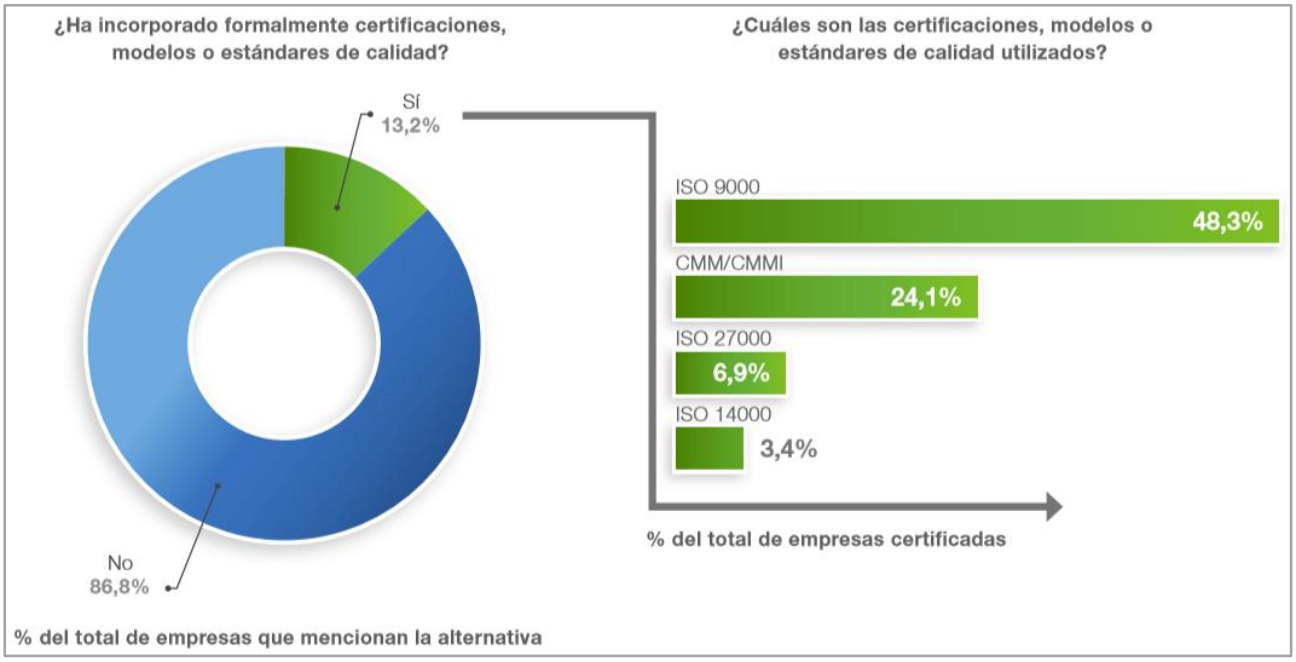


Figura 3
Certificaciones empresariales, modelos y estándares de calidad.
Fuente: Cámara de Tecnologías de Información y Comunicación (2015). Disponible en: "https://www.camtic.org"

El panel de empresas participantes incluyó la colaboración de 219 compañías costarricenses del sector de tecnologías digitales, de las cuales el 13% ha incorporado formalmente certificaciones, modelos o estándares de control y gestión de calidad.

Según lo anterior, se puede asegurar que si se enfocan los esfuerzos en mejorar la calidad y productividad, a través del uso de una metodología o modelo de desarrollo seleccionado correctamente, se puede lograr un gran avance en cada uno de los procesos involucrados en el desarrollo de *software*, tales como: mantenimiento correctivo, el mantenimiento evolutivo y la creación de nuevas soluciones, por lo cual, es necesario el estudio de las distintas metodologías de desarrollo más utilizadas en la actualidad, para determinar cuál es la que más se adapta a la necesidades estratégicas del área, mejorando así la calidad, productividad, eficiencia y costos de desarrollo y mantenimiento de *software*.

Adicionalmente y en vista de la necesidad que tiene la organización de contar con la propuesta de una metodología de desarrollo para el área que brinda soporte al SIMA, se conversó con el líder de producto, quien expresó que el proyecto se adapta en gran parte a la proyección del departamento en busca de calidad y optimización del flujo involucrado, tanto para el mantenimiento como para el desarrollo de *software*. De esta manera, el proyecto toma gran relevancia al ser justificado con las carencias del proceso actual; como confirmación de lo mencionado, en el Anexo A se puede visualizar la carta de aprobación por parte de la empresa.

1.2 DEFINICIÓN DEL PROBLEMA

En la actualidad, se ha podido observar que a nivel del SIMA se genera una gran cantidad de incidencias, tanto a nivel de las soluciones ya existentes como de los desarrollos creados desde cero; este problema se presenta debido a que el área que se encarga del mantenimiento y desarrollo del sistema antes mencionado, no cuenta con una metodología de desarrollo que le sirva de guía para mejorar los procesos involucrados, de manera tal que las incidencias reportadas sean resueltas de forma correcta y en el menor tiempo posible.

Asimismo, el proceso de desarrollo de *software* no cuenta con estándares de calidad que le permitan disminuir la cantidad de correcciones realizadas posterior a la implementación, esto aunado a entrega de productos que cumplan con todas las especificaciones requeridas por el usuario final.

El INS cuenta con una herramienta de incidencias llamada GESTIC, desde donde se han obtenido estadísticas que indican que la cantidad promedio de incidencias reportadas por mes consta de 197, un número bastante alarmante considerando que esta cifra se ha mantenido desde hace aproximadamente cinco años.

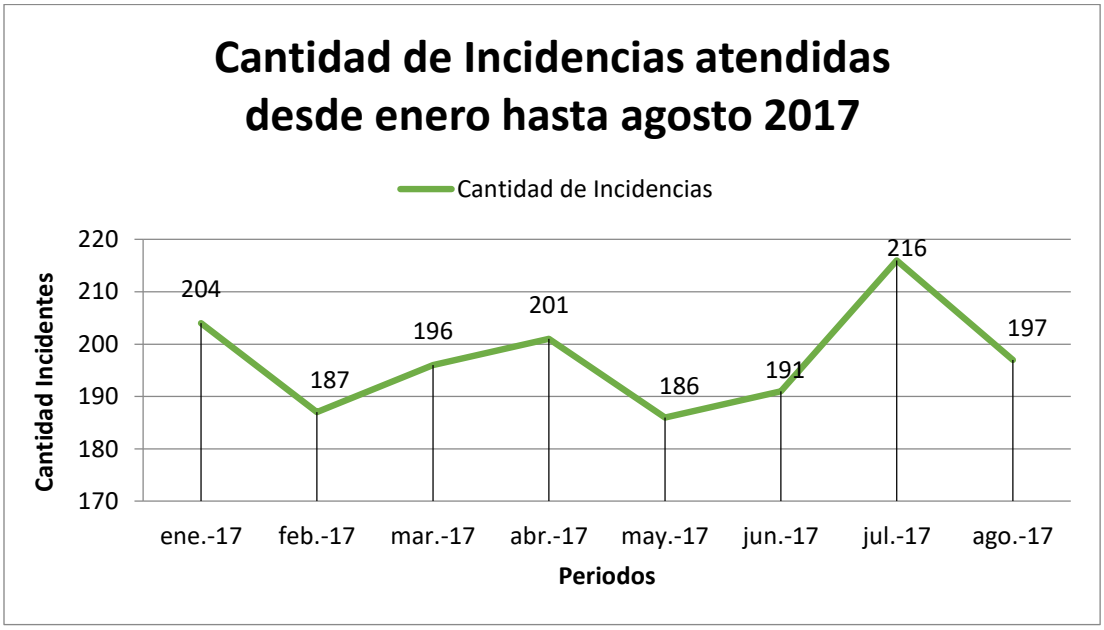


Figura 4
Incidencias atendidas en el año 2017.
Fuente: elaboración propia.

Adicionalmente, se ha realizado una pequeña recopilación de datos para obtener la cantidad de errores que se han reportado posterior a la implementación de los últimos tres desarrollos creados a nivel del SIMA, todos desarrollados en el año 2017.

Tabla 2

Últimos desarrollos creados en el SIMA

Número Requerimiento	Descripción	Cantidad Incidencias
2017-200-00	Proceso de cierre masivo de casos y cálculo de la provisión	47
2017-200-00	Creación de reportes estadísticos a nivel de estancia de pacientes en los distintos Centros Médicos	17
2017-200-00	Manejo de inventario de medicamentos en quirófanos del Hospital de Trauma	28
	TOTAL	**92**

Fuente: elaboración propia.

Se puede observar que la cantidad de incidencias reportadas para requerimientos desarrollados recientemente, es bastante elevada y se han logrado identificar algunos de los factores más relevantes para que se presenten este tipo de resultados:

- Estimación de tiempo incorrecta
- Asignación de recursos inexactos
- Tendencia de modificación del requerimiento original
- Falta de comunicación entre el desarrollador y los usuarios funcionales
- Mala calidad de la solución creada

El problema expuesto anteriormente trae consigo una serie de consecuencias tales como: pérdidas económicas a nivel de la empresa, ya que los tiempos en atención de incidencias son bastante altos e incluso, en muchas ocasiones, es necesario el pago de horas extra para poder abarcar la gran cantidad de trabajo; además, los usuarios que utilizan el sistema se ven afectados ante la gran cantidad de errores que este presenta, pero el más perjudicado ante esta problemática es el cliente final, que paga un seguro con la finalidad de ser atendido de la mejor manera, pero que en algunas ocasiones por los errores indicados anteriormente, se ve afectado por largos tiempos de espera para realizar algún trámite en específico.

1.2.1 Diagrama causa y efecto

A continuación, se muestra un diagrama de causa y efecto, donde se visualizan las causas que originan el problema expuesto y las consecuencias que el mismo trae consigo.

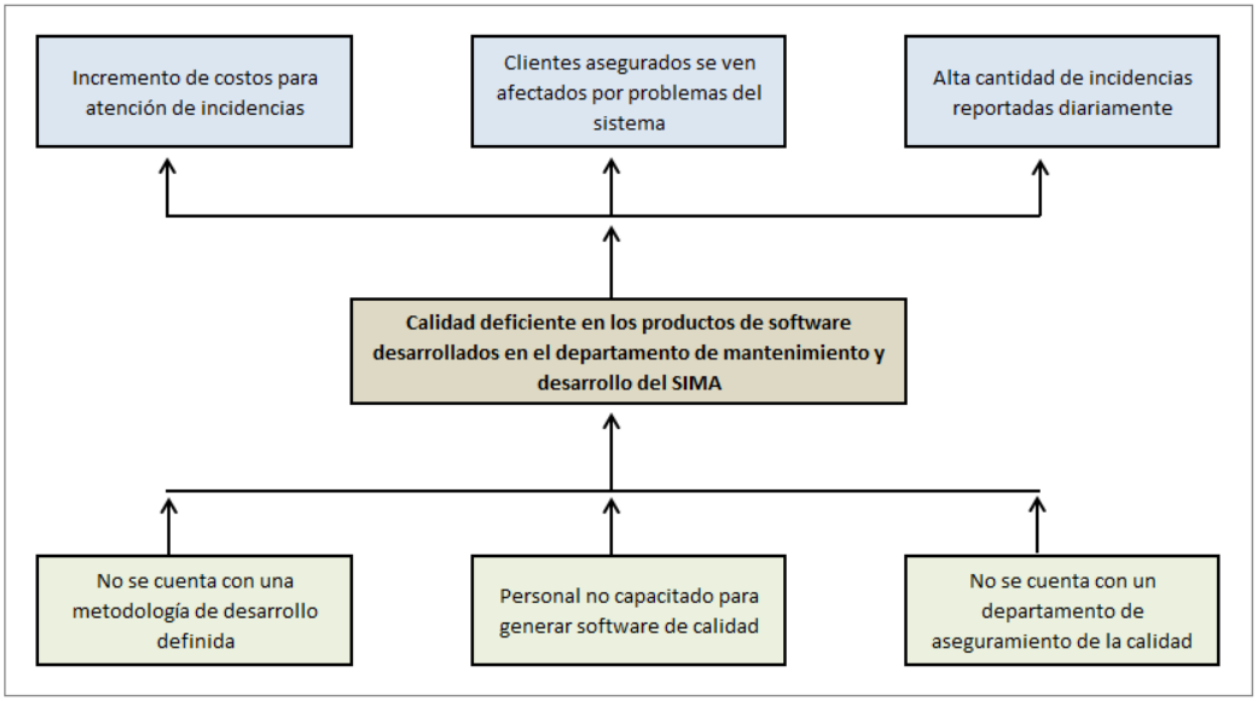


Figura 5
Diagrama causa y efecto.
Fuente: elaboración propia.

1.3 OBJETIVOS DE LA INVESTIGACIÓN

1.3.1 Objetivo general

Proponer una metodología de desarrollo y mantenimiento de software para el Sistema Integral Médico Administrativo del Instituto Nacional de Seguros.

1.3.2 Objetivos específicos

- Identificar la situación del proceso actual de mantenimiento y desarrollo de *software* del Sistema Integral Médico Administrativo del INS.

- Realizar un estudio comparativo entre las metodologías de desarrollo de *software* más utilizadas en la actualidad.

- Identificar las necesidades y la brecha existente entre lo que se tiene actualmente y lo deseado para mejorar el proceso de Mantenimiento y Desarrollo del Sistema Integral Médico Administrativo del INS.

- Diseñar una propuesta de la metodología de desarrollo que más se ajuste a las necesidades requeridas en el proceso de Mantenimiento y Desarrollo del Sistema Integral Médico Administrativo del INS.

1.4 ALCANCE Y LIMITACIONES

1.4.1 Alcances

El primer entregable del proyecto consiste en reconocer todos los procesos involucrados en el proceso de mantenimiento y desarrollo de *software* del área, con el fin de determinar cuáles son las deficiencias o carencias generales que deben ser consideradas para la propuesta de la metodología de desarrollo.

En el segundo entregable se pretende conocer a profundidad las metodologías de desarrollo más utilizadas en la actualidad. Para lograr el cometido, se utilizará material bibliográfico y audiovisual con el fin de conocer las características que componen y diferencian a cada una de las metodologías estudiadas.

En el tercer entregable se pretende identificar las necesidades y la brecha en relación con lo que existe actualmente y lo que se desea, para la ejecución de este objetivo se requiere conocer la forma en que se realiza el desarrollo y mantenimiento de *software* en la compañía, así como las metodologías de desarrollo ágiles más utilizadas en la actualidad.

Tomando como insumo principal el conocimiento adquirido con el estudio de las deficiencias del proceso de mantenimiento y desarrollo de software del área y la comprensión general de las metodologías de desarrollo, en el cuarto entregable se

pretende diseñar la propuesta de una metodología que se adapte a las necesidades del SIMA.

1.4.2 Limitaciones

A pesar del establecimiento de un estándar en el proceso de mantenimiento y desarrollo de *software* mediante el uso de una metodología de desarrollo, existe la posibilidad de que algunos desarrolladores no cumplan con todas las normas definidas en el mismo.

Este proyecto es una propuesta, donde se demostrará que el uso de una metodología de desarrollo en el área de mantenimiento y desarrollo es de vital importancia, sin embargo, queda a discreción de la empresa su posible implementación.

CAPÍTULO II

MARCO TEÓRICO

2.1 INGENIERÍA DE *SOFTWARE*

Para entender de mejor manera la definición de Ingeniería de *Software*, es necesario definir la palabra *Software*, que según el Diccionario de la Real Academia Española (RAE, 2018) se refiere a un conjunto de programas, instrucciones y reglas informáticas para ejecutar ciertas tareas en una computadora.

Partiendo de esa base, es posible definir a la Ingeniería de *Software* como una disciplina conformada por un conjunto de métodos, herramientas y técnicas que son utilizadas en la elaboración de programas informáticos.

Sin embargo, la Ingeniería de *Software* no se limita solamente a la creación de *software*, sino que: "se interesa por todos los aspectos de la producción de *software*, desde las primeras etapas de la especificación del sistema hasta el mantenimiento del sistema después de que se pone en operación" (Somerville, 2011, p.7).

2.2 MANTENIMIENTO DE *SOFTWARE*

Consiste en la modificación de un producto de *software* después de haber sido entregado al usuario final con el fin de corregir defectos, mejorar el rendimiento o realizar cambios solicitados por el negocio.

Según Pressman (2010), el mantenimiento de *software* inicia casi de inmediato después de liberar el producto a los usuarios finales e indica lo siguiente:

> En cuestión de días, los reportes de errores se filtran de vuelta hacia la organización de ingeniería de *software*. En semanas, una clase de usuarios indica que el *software* debe cambiarse de modo que pueda ajustarse a las necesidades especiales de su entorno. Y en meses, otro grupo corporativo, que no quería saber nada del *software* cuando se liberó, ahora reconoce que puede ofrecerle beneficios inesperados. Necesitará algunas mejoras para hacer que funcione en su mundo. (p.656)

2.2.1 Tipos de mantenimiento de *software*

A lo largo de su vida útil, una solución de *software* puede requerir modificaciones por distintas razones, que se dividen en distintos tipos de mantenimiento tales como:

- Mantenimiento correctivo: consiste en la corrección de un problema específico del *software* ante un funcionamiento incorrecto, deficiente o incompleto.

- Mantenimiento preventivo: corrección de un problema antes de que se presente. También ayuda a facilitar el mantenimiento futuro del sistema.

- Mantenimiento perfectivo: acciones llevadas a cabo para mejorar la calidad interna de los sistemas en cualquiera de sus aspectos: reestructuración del código, definición más clara del sistema y optimización del rendimiento y eficiencia.

- Mantenimiento adaptativo: modificaciones que afectan a los entornos en los que el sistema opera, por ejemplo, cambios de configuración del *hardware*, *software* de base, gestores de base de datos, comunicaciones, etc. (Martínez Torres, 2010).

2.3 METODOLOGÍAS DE DESARROLLO DE *SOFTWARE*

Una metodología de *software* es un marco de trabajo que permite realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito, comprende los procesos a seguir sistemáticamente para idear, implementar y mantener un producto *software* desde que surge la necesidad del mismo hasta que se cumple con el objetivo por el cual fue creado. Bahit (2012) indica: "Implementar una metodología de gestión, básicamente nos permite organizar mejor un proyecto y obtener mejores resultados del *software* entregado al cliente, evitando los fracasos". (p.13), por lo tanto, evitando el tan molesto mantenimiento correctivo.

2.4 METODOLOGÍAS TRADICIONALES

Las metodologías tradicionales fueron propuestas originalmente para que el desarrollo de *software* fuera ejecutado de forma ordenada, la historia indica que este tipo de modelos ha dotado de cierta estructura útil al trabajo de ingeniería de *software*, sin embargo, los productos de *software* desarrollados siguen “al borde del caos”. (Pressman, 2010).

Este tipo de metodologías establecen un conjunto de elementos involucrados en el proceso, tales como: actividades estructurales, acciones de ingeniería de *software*, tareas, productos del trabajo, aseguramiento de la calidad y mecanismos de control de cambios para cada proyecto. Cada modelo también prescribe un flujo de trabajo de manera tal que todos los elementos de proceso se relacionen entre sí (Pressman, 2010).

A continuación, se explican algunas de las metodologías tradicionales más utilizadas en ingeniería de *software*.

2.4.1 Modelo de la cascada

También llamado ciclo de vida clásico, es un enfoque sistemático y secuencial para el desarrollo de *software* que inicia con la especificación de los requerimientos por parte

del usuario y avanza a través de la planeación, modelado, construcción y despliegue (Pressman, 2010).

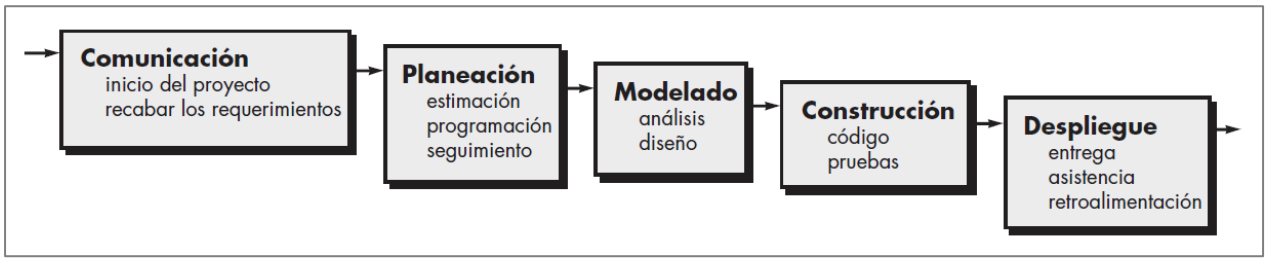


Figura 6
Etapas del modelo de la cascada.
Fuente: Pressman R. (2010)

Como se puede visualizar en la figura 6, en la primera etapa se genera la especificación del requerimiento, luego en la planeación es donde se estiman los tiempos del proyecto y se crea el cronograma de trabajo, la etapa de modelado contiene el análisis y diseño del requerimiento, en la construcción es cuando el desarrollador confecciona el *software* especificado en el requerimiento, además se ejecutan pruebas técnicas y funcionales, y, por último, en la etapa de despliegue se entrega todo el proyecto y se realizan los ajustes respectivos.

Entre las ventajas del modelo de cascada se pueden mencionar las siguientes:

1. El tiempo que se pasa en diseñar el producto en las primeras fases del proceso puede evitar problemas que serían más costosos cuando el proyecto ya estuviese en fase de desarrollo.

2. La documentación es muy exhaustiva y si se une al equipo un nuevo desarrollador, podrá comprender el proyecto leyendo la documentación.

3. Al ser un proyecto muy estructurado, con fases bien definidas, es fácil entenderlo.

4. Ideal para proyectos estables, donde los requisitos son claros y no van a cambiar a lo largo del proceso de desarrollo. (Domínguez, 2017)

Pero la metodología de la cascada también posee una serie de desventajas, las cuales se describen a continuación:

1. Es raro que los proyectos reales sigan el flujo secuencial propuesto por el modelo. Aunque el modelo lineal acepta repeticiones, lo hace en forma indirecta; como resultado, los cambios generan confusión conforme el equipo del proyecto avanza.

2. A menudo, es difícil para el cliente enunciar en forma explícita todos los requerimientos. El modelo de la cascada necesita que se haga y tiene dificultades para aceptar la incertidumbre natural que existe al principio de muchos proyectos.

3. El cliente debe tener paciencia. No se dispondrá de una versión funcional del programa o programas solicitados hasta que el proyecto esté muy avanzado. Un error grande sería desastroso si se detectara hasta revisar el programa en funcionamiento (Pressman, 2010).

2.4.2 Modelo espiral de Boehm

Fue propuesto en 1998 por Barry Boehm y se define como un modelo evolutivo del proceso del *software* y se ajusta con la naturaleza de hacer prototipos basados en el modelo de cascada, pero con el potencial para realizar un desarrollo rápido de versiones cada vez más completas.

Con el uso del modelo espiral, el *software* es desarrollado en una serie de entregas evolutivas, en las primeras iteraciones lo que se entrega puede ser un modelo o prototipo y en las iteraciones posteriores se producen versiones que se acercan más a los que requiere el usuario o cliente (Somerville, 2011).

El modelo espiral se divide en un número de actividades de marco de trabajo, también llamadas regiones de tareas y, generalmente, existen entre tres y seis regiones de tareas, las cuales se muestran en la siguiente figura:

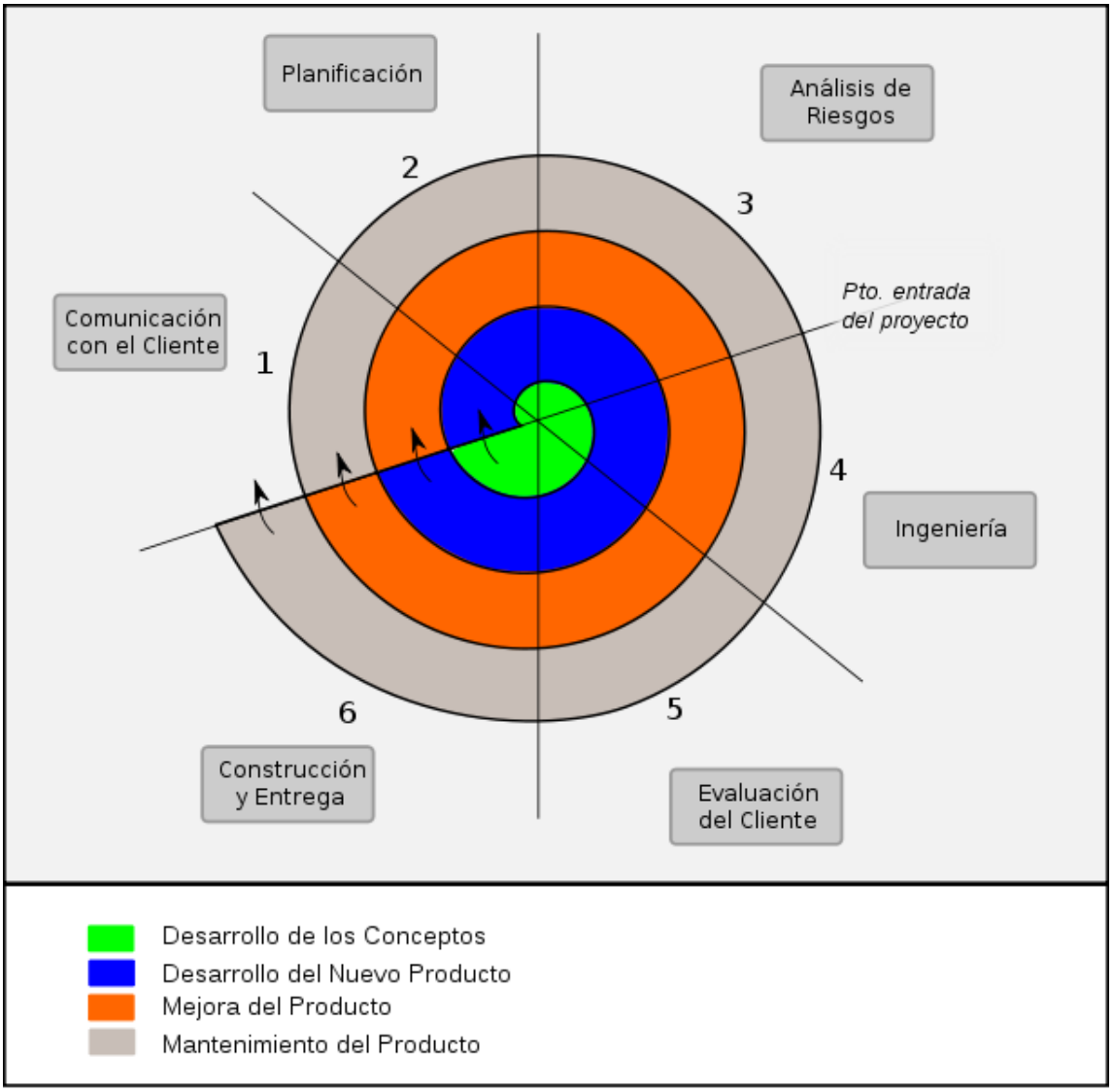


Figura 7
Modelo espiral.
Fuente: Disponible en: “http://sofware1nathalygrijalva.blogspot.com”

Comunicación con el cliente: se refiere a las tareas requeridas para establecer una comunicación entre el equipo de desarrollo y el usuario o cliente.

Planificación: tareas requeridas para definir los recursos, el tiempo y otro tipo de información relacionada con el proyecto.

Análisis de riesgos: son las tareas requeridas para la evaluación de posibles riesgos técnicos y de gestión del proyecto.

Ingeniería: las tareas requeridas para la construcción de una o más partes o representaciones de la aplicación.

Evaluación del cliente: se refiere a las tareas requeridas para obtener la reacción del cliente según la presentación del *software* creado durante la etapa de ingeniería.

Construcción y entrega: tareas requeridas para construir, probar, instalar y proporcionar soporte al usuario. (Grijalva, 2012)

La diferencia principal entre el modelo espiral con otros modelos de desarrollo de *software* consiste en el reconocimiento explícito del riesgo. El ciclo inicia por elaborar los objetivos, luego se numeran formas alternativas de alcanzar dichos objetivos y de lidiar con las restricciones en cada uno de ellos. Cada alternativa es valorada contra el objetivo con el fin de identificar las fuentes de riesgo del proyecto.

El siguiente paso consiste en resolver dichos riesgos, mediante actividades de recopilación de información, como análisis más detallados, creación de prototipos y simulación. Una vez valorados los riesgos, se realiza cierto desarrollo, seguido por una actividad de planeación para la siguiente fase del proceso. (Somerville, 2011)

2.4.3 Modelo de desarrollo incremental

Se fundamenta en la idea de diseñar una propuesta inicial, presentar la misma al usuario para la validación respectiva y, luego, basado en las observaciones del usuario, ir desarrollando las distintas versiones hasta producir un sistema adecuado. (Somerville, 2011). A continuación, se muestran las actividades de este modelo:

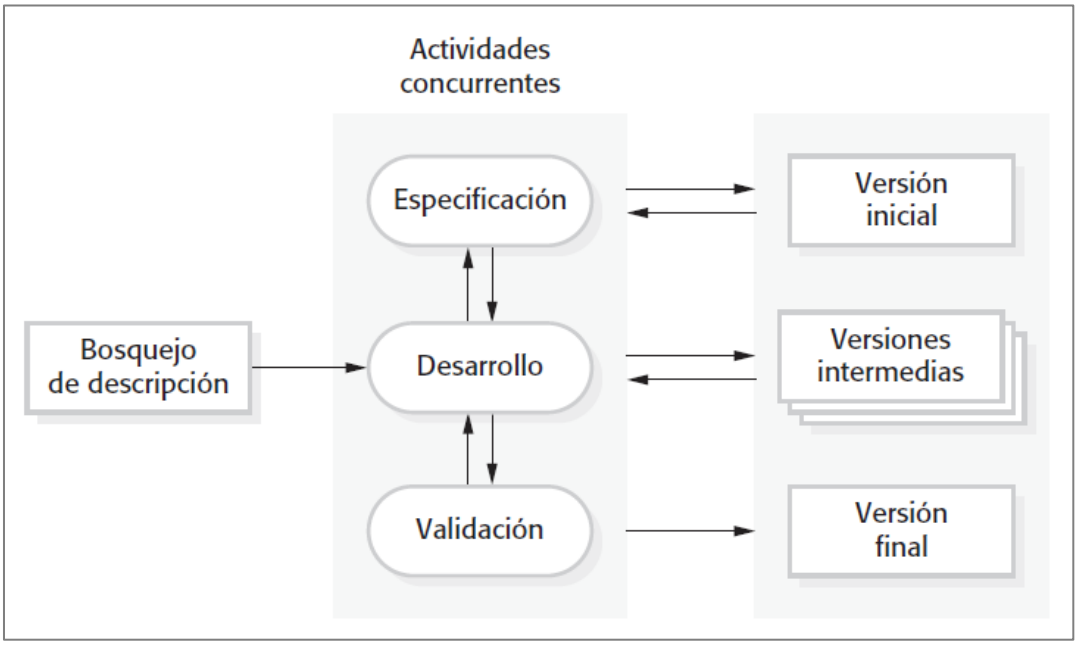


Figura 8
Actividades modelo de desarrollo incremental.
Fuente: Somerville I. (2010).

Como se puede observar en la figura 8, las actividades de especificación, desarrollo y validación están entrelazadas en vez de separadas, lo cual busca una rápida retroalimentación a través de las actividades.

El desarrollo de *software* incremental se considera como una parte fundamental de los enfoques ágiles, se considera mejor que un enfoque en cascada para la mayoría de los sistemas empresariales, de comercio electrónico y personales. (Somerville, 2011).

> “El desarrollo incremental refleja la forma en que se resuelven problemas, rara vez se trabaja por adelantado una solución completa del problema, sino más bien se avanza en una serie de pasos hacia una solución y se retrocede cuando se detecta que se cometieron errores”. (Somerville, 2011, p.33)

Cada incremento o versión contiene algunas de las necesidades que tiene el usuario, por lo general estos primeros incrementos incluyen la función más importante o más urgente, esto con el fin de que el usuario tenga la posibilidad de evaluar el desarrollo en una etapa relativamente temprana, para validar si el desarrollo entregado en el incremento cumple con las necesidades, en caso contrario, el mismo debe cambiarse y posiblemente definir una nueva función para incrementos posteriores. (Somerville, 2011)

Algunos de los beneficios que brinda el desarrollo incremental son los siguientes:

1. Se reduce el costo de adaptar los requerimientos cambiantes del cliente. La cantidad de análisis y la documentación que tienen que reelaborarse son mucho menores de lo requerido con el modelo en cascada.

2. Es más sencillo obtener retroalimentación del cliente sobre el trabajo de desarrollo que se realizó. Los clientes pueden comentar las demostraciones del *software* y darse cuenta de cuánto se ha implementado. Además, los clientes encuentran difícil juzgar el avance a partir de documentos de diseño de *software*.

3. Es posible que sea más rápida la entrega e implementación de *software* útil al cliente, aun si no se ha incluido toda la funcionalidad. Los clientes tienen posibilidad de usar y ganar valor del *software* más temprano de lo que sería posible con un proceso en cascada. (Somerville, 2011)

En relación con los problemas del modelo de desarrollo incremental, se pueden mencionar los siguientes:

1. El proceso no es visible y los administradores de proyecto requieren entregas regulares para medir el avance. Si los sistemas se desarrollan rápidamente, resulta poco efectivo en términos de costos producir documentos que reflejen cada versión o incremento del sistema.

2. La estructura del sistema tiende a degradarse conforme se tienen nuevos incrementos, a menos que se gaste tiempo y dinero en la refactorización para mejorar el *software*, el cambio regular tiende a corromper su estructura. La incorporación de más cambios de *software* se vuelve cada vez más difícil y costosa. (Somerville, 2011)

2.5 METODOLOGÍAS ÁGILES

Las metodologías ágiles son sistemas de gestión de proyectos que ayudan a usar el tiempo de manera efectiva y creativa, son muy útiles para visualizar y organizar las tareas a realizar, mejorar el rendimiento y el trabajo en equipo, además permiten tener un seguimiento detallado del cada una de las etapas de un proyecto, tanto a nivel personal como grupal.

Bahit (2012) define las metodologías ágiles como: “Gestión de proyectos **adaptativa**, que permite llevar a cabo, proyectos de desarrollo de *software*, **adaptándote a los cambios** y evolucionando en forma conjunta con el *software*” (p.13). Es decir, a diferencia de las metodologías tradicionales, éstas sí son flexibles y permiten cambios en el proceso de desarrollo. En la siguiente tabla se muestran algunos de los aspectos más relevantes de las metodologías tradicionales y ágiles.

Tabla 3

Metodologías tradicionales vs. metodologías ágiles

Metodologías Tradicionales	Metodologías Ágiles
Predictivos	Adaptativos
Orientados a procesos	Orientados a personas
Proceso rígido	Proceso flexible
Se concibe como un proyecto	Un proyecto es subdividido en varios proyectos más pequeños
Poca comunicación con el cliente	Comunicación constante con el cliente
Entrega de Software al finalizar el proyecto	Entregas constantes de software
Documentación extensa	Poca documentación

Fuente: recuperado de http://www.redalyc.org.

Como se puede observar en la tabla 3, se especifica con precisión las grandes diferencias que existen entre las metodologías tradicionales y ágiles.

2.5.1 Manifiesto Ágil

En febrero de 2001, se realizó una reunión en Utah, Estados Unidos, donde participó un grupo de diecisiete profesionales reconocidos del desarrollo de *software* y referentes de las metodologías livianas existentes al momento, con el objetivo de definir los valores y principios que les permitirían a los equipos desarrollar *software* de forma más acertada con las necesidades del cliente y responder mejor a los cambios que pudieran surgir a lo largo de un proyecto de desarrollo. Se pretendía ofrecer una alternativa a los procesos de desarrollo de *software* tradicionales, caracterizados por la rigidez y dominados por la documentación.

En esta reunión fue creada la *Agile Alliance*, una organización sin fines de lucro cuyo objetivo es el de promover los valores y principios de la filosofía ágil y ayudar a las organizaciones en su adopción. También, se declaró la piedra angular del movimiento ágil, conocida como Manifiesto Ágil (Alaimo, 2013).

El Manifiesto Ágil está compuesto por cuatro valores y doce principios, los cuales se describen a continuación:

2.5.1.1 Valores

- Valorar a las personas y las interacciones entre ellas por sobre los procesos y las herramientas.
- Valorar el *software* funcionando por sobre la documentación detallada.
- Valorar la colaboración con el cliente por sobre la negociación de contratos.
- Valorar la respuesta a los cambios por sobre el seguimiento estricto de planes.

.5.1.2 Principios

1. La mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de *software* con valor.

2. Se acepta que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Se entrega *software* funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajan juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El *software* funcionando es la medida principal de progreso.

8. Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.

10. La simplicidad o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.

12. A intervalos regulares, el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia (Beck et al., 2001).

2.5.2 Metodología Scrum

Scrum es un marco de trabajo en el que equipos interdisciplinarios y autoorganizados (Cross-funcional) pueden crear productos o desarrollar proyectos de una forma iterativa e incremental. El desarrollo se estructura en ciclos de trabajo llamados *Sprints* (también conocidos como iteraciones), dichas iteraciones no deben durar más de cuatro semanas cada una y tienen lugar una tras otra sin pausa entre ellas. Los *Sprints o* iteraciones deben finalizar en una fecha determinada, independientemente de si el trabajo ha finalizado por completo o no, y jamás se pueden demorar más del tiempo establecido. Normalmente, el equipo interdisciplinario define una duración estándar para el *Sprint* y esta es la que mantienen para todas las iteraciones.

Adicionalmente, el equipo acuerda un objetivo colectivo respecto a lo que creen que sea posible entregar como un trabajo terminado al final del *Sprint*. Durante el *Sprint* no es posible añadir nuevos elementos, SCRUM se adapta a los cambios en el siguiente *Sprint*.

Todos los días, el equipo se reúne brevemente para inspeccionar su progreso y ajustar los siguientes pasos necesarios para completar el trabajo pendiente. Al final del *Sprint*, el equipo lo revisa con las personas involucradas en el producto (*Stakeholders*) y realiza una demostración de lo que han desarrollado, de la cual se obtiene la retroalimentación necesaria para ser incorporada en el siguiente *Sprint*.

Scrum enfatiza un producto “funcionando” al final del *Sprint* que esté realmente “terminado”. En el caso del *software*, esto significa un sistema que está integrado, testado, con la documentación de usuario generada y potencialmente entregable. (Deemer et al., 2012)

Para comprender de manera resumida todo lo explicado en relación con Scrum, en la siguiente figura se pueden observar los principales roles, artefactos y eventos involucrados en el proceso.

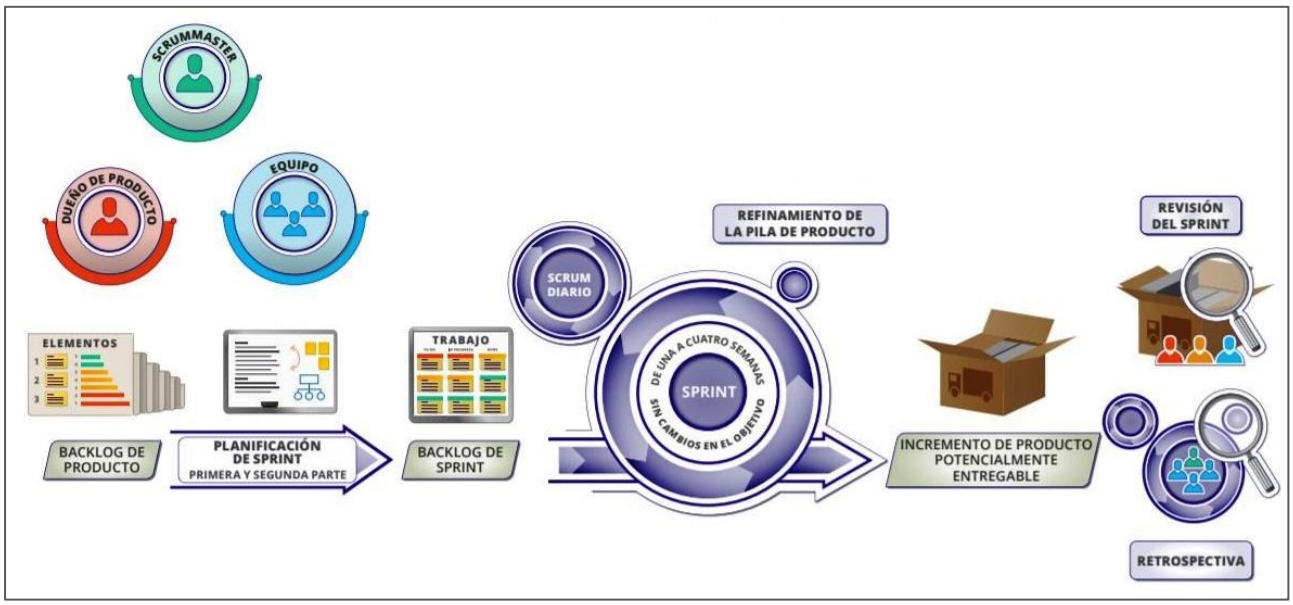


Figura 9
Visión general de Scrum.
Fuente: Disponible en: “http://scrumprimer.org”

2.5.2.1 Principios

Alaimo (2013) define los cinco valores que componen la metodología Scrum de la siguiente manera:

Foco: los Equipos Scrum se enfocan en un conjunto acotado de características por vez. Esto permite que al final de cada *Sprint* se entregue un producto de alta calidad.

Coraje: debido a que los Equipos Scrum trabajan como verdaderos equipos, tienen la posibilidad de apoyarse entre compañeros y así tener el valor de asumir compromisos desafiantes que les permitan crecer como profesionales.

Apertura: los Equipos Scrum privilegian la transparencia y la discusión abierta de los posibles problemas que se pueden presentar. No existen agendas ocultas ni triangulación de conflictos. La sinceridad se gratifica y la información está disponible para todos, durante todo el tiempo.

Compromiso: los Equipos Scrum tienen mayor control sobre sus actividades, por eso se espera de su parte el compromiso profesional para el logro del éxito.

Respeto: debido a que los miembros de un Equipo Scrum trabajan de forma conjunta, compartiendo éxitos y fracasos, se fomenta el respeto mutuo.

2.5.2.2 Roles

En un Equipo Scrum deben intervenir tres tipos de roles:

- Dueño del producto
- Equipo
- ScrumMaster

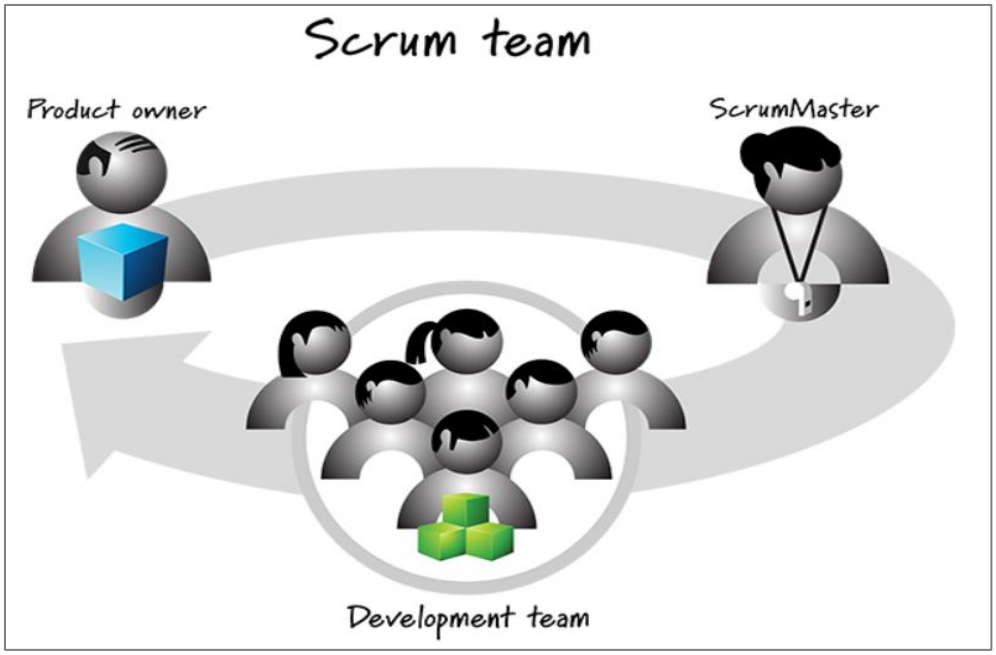


Figura 10
Roles de Scrum.
Fuente: Disponible en: “http://www.innolution.com”

2.5.2.2.1 Dueño del producto

Es el responsable de maximizar el retorno de inversión (ROI), a base de identificar las funcionalidades del producto, trasladarlas a una lista priorizada, decidir cuáles deberían estar al principio de la lista para el siguiente *Sprint*, priorizar y refinar continuamente dicha lista.

Es responsable a nivel de ganancias y pérdidas del producto, asumiendo que se trata de un producto comercial. En el caso de aplicaciones internas, el dueño de producto no es responsable del ROI en el mismo sentido que en un producto comercial (que generaría ingresos), pero aun sería responsable de maximizar el ROI en el sentido de escoger en cada *Sprint* los elementos que más valor aportan. En la práctica, “valor” es un término muy difuso y la priorización puede verse influenciada por el deseo de satisfacer a los clientes clave, la alineación con los objetivos estratégicos, atacar riesgos, mejorar y otros factores.

En algunos casos, el dueño de producto y el cliente son la misma persona; esto es frecuente en el caso de desarrollos internos. En otros, el cliente pueden ser millones de personas con necesidades muy variadas, en cuyo caso, en muchas organizaciones el rol del *Product Owner* (dueño de producto) es similar al de *Product Manager* (director de producto) o *Product Marketing Manager* (director de *marketing* de producto).

Sin embargo, el rol del dueño de producto es algo distinto del director de producto tradicional, ya que interactúa de forma activa y regular con el equipo, prioriza trabajando con todos los *stakeholders* y revisa los resultados de cada *Sprint*, en lugar de delegar las decisiones de desarrollo en un jefe de proyecto. Es importante hacer notar que en Scrum, hay una y sólo una persona que sirve como dueño de producto, ejerce la autoridad final como tal y él o ella es responsable del valor del trabajo realizado, aunque dicha persona no tiene por qué trabajar sola. (Deemer et al. 2012)

2.5.2.2.2 Equipo

También llamado Equipo de Desarrollo, construye lo que el dueño de producto establece, por ejemplo, una aplicación o un sitio Web. El equipo en Scrum es *cross-funcional*, engloba toda la experiencia y conocimiento necesarios para desarrollar un producto potencialmente entregable en cada uno de los *Sprint* y es autoorganizado, con un amplio margen de autonomía y responsabilidad.

El equipo decide cuántos elementos (del conjunto que ofrece el dueño de producto) va a desarrollar durante el *Sprint* y cuál es la mejor manera de lograr dicho objetivo.

Cada miembro del equipo es simplemente un "*miembro de equipo"*. Nótese que no hay títulos fijos especializados en un grupo que adopta Scrum: no hay analistas de negocio, administradores de bases de datos, arquitectos, líderes de equipo, diseñadores/especialistas en UX (Experiencia del usuario), programadores u otros.

Los miembros del equipo trabajan juntos durante cada *Sprint* de cualquier manera que sea apropiada para lograr el objetivo que se han marcado ellos mismos. Dado que solo hay miembros de equipo, el equipo no es sólo *cross-funcional*, sino que además muestra aprendizaje múltiple: cada persona indudablemente tendrá ciertas fortalezas, pero también continuará aprendiendo otras especialidades. Cada persona tendrá habilidades principales, secundarias e incluso terciarias y se espera de ellos que “vayan por el trabajo”, que emprendan tareas con las que se sienten menos familiarizados, con el objetivo de ayudar a completar un elemento.

Por ejemplo, una persona cuya habilidad principal es el diseño de interacción de usuario, podría tener habilidades secundarias en automatización de pruebas; una persona con habilidad principal en redacción técnica podría también ayudar con el análisis y la programación.

El Equipo Scrum tiene siete más/menos dos personas y, en el caso del *software*, el equipo podría incluir a personas con habilidades en análisis, desarrollo, pruebas, diseño de interfaz, diseño de bases de datos, arquitectura, documentación y demás. El equipo desarrolla el producto y proporciona ideas al dueño de producto, sobre cómo hacer que este sea un éxito.

En Scrum, los equipos son más productivos y efectivos si todos los miembros están dedicados al 100% a trabajar en un producto durante el *Sprint*. El equipo evita la multitarea sobre varios productos o proyectos, para así evitar el gran costo que tienen la

pérdida de concentración y el cambio de contexto (*context switching*). Los equipos estables se asocian con mayor productividad, por lo que hay que evitar cambiar miembros del equipo.

Las áreas de producto con muchas personas se organizan mediante múltiples equipos, cada uno de ellos concentrado en diferentes funcionalidades del producto y con alta coordinación de sus esfuerzos. Dado que cada equipo realiza todo el trabajo necesario para una funcionalidad desde la perspectiva del cliente (planificación, análisis, desarrollo y pruebas), los equipos se denominan también "*feature teams*" (equipos de funcionalidad). (Deemer et al. 2012)

2.5.2.2.3 ScrumMaster

El ScrumMaster ayuda al área de producto a aprender y aplicar Scrum para obtener valor de negocio, hace todo lo que esté en su mano para ayudar al equipo, al dueño de producto y a la organización a tener éxito. Sirve al equipo, ayuda a eliminar impedimentos, protege de interferencias externas y les ayuda a adoptar prácticas de desarrollo modernas. Entrena y guía al dueño de producto, al equipo y al resto de la organización en el uso correcto de Scrum.

El ScrumMaster es un *coach* y un formador, se asegura de que todos los involucrados (incluidos el dueño de producto y los gerentes) comprendan los principios y las prácticas de Scrum y ayuda a guiar a la organización a través de los

habitualmente difíciles cambios que son necesarios para lograr el éxito en el desarrollo ágil. Como Scrum hace visibles múltiples impedimentos y amenazas a la efectividad del equipo y del dueño de producto, es importante contar con un ScrumMaster comprometido, trabajando de forma enérgica en ayudar a resolverlos; de lo contrario, será difícil que el equipo o el dueño de producto tengan éxito.

El ScrumMaster debería tener un rol dedicado al 100%, aunque un equipo pequeño podría tener un miembro ejerciendo dicho rol (y con una menor carga de trabajo regular en dicho caso). Los mejores ScrumMasters surgen de todo tipo de experiencias y disciplinas: ingeniería, diseño, pruebas, dirección de producto, gestión de proyectos o gestión de la calidad.

El ScrumMaster y el dueño de producto no pueden ser la misma persona, ya que su enfoque es muy diferente y combinar ambos roles, normalmente conduce a confusión y conflictos. Uno de los desafortunados resultados de combinar dichos roles es un dueño de producto haciendo microgestión, lo cual es justo lo opuesto al equipo autogestionado que requiere Scrum.

Al contrario de los gerentes tradicionales, el ScrumMaster no le dice a la gente lo que tiene que hacer o asigna tareas, sino que facilita el proceso dando soporte al equipo mientras que este se organiza y se gestiona por su cuenta. Si el ScrumMaster se encontraba anteriormente en una posición de gerencia sobre el equipo, tendrá que realizar un cambio profundo en su forma de pensar y estilo de interacción para tener éxito con Scrum.

Es importante mencionar que no existe ningún tipo de rol de jefe de proyecto en Scrum. Esto se debe a que no es necesario en absoluto, pues las responsabilidades tradicionales del jefe de proyecto han sido divididas y reasignadas entre los tres roles de Scrum, sobre todo entre el equipo y el dueño de producto, más que en el ScrumMaster. La práctica de Scrum con jefes de proyecto añadidos demuestra un desconocimiento fundamental del proceso y típicamente desemboca en conflictos de responsabilidad, confusión respecto a la autoridad y resultados por debajo de lo esperado. (Deemer et al., 2012)

2.5.2.3 Elementos o artefactos de Scrum

Scrum propone tres herramientas o “artefactos” para mantener organizados los proyectos, dichos artefactos ayudan a planificar y revisar cada uno de los *Sprints*. A continuación, se explica cada una de estas herramientas.

2.5.2.3.1 Pila del producto

Es el primero de los elementos y el principal de Scrum, también conocido como *Product Backlog*, consiste básicamente en un listado de ítems o características del producto a consumir, el mismo es visible para todos los involucrados de proyecto y es mantenido y priorizado por el Dueño del Producto. Es muy importante que exista una clara priorización, ya que es ésta la que determinará el orden en el cual el equipo de desarrollo transformará los ítems en un producto funcional acabado. (Alaimo, 2013)

Cada uno de los ítems que conforman la pila del producto debe cumplir con un formato establecido que será necesario especificar, la información mínima que se suele incluir es la siguiente:

- Descripción de la funcionalidad/requisito, denominado “historia de usuario”.
- Prioridad
- Estimación del esfuerzo necesario. (Menzinsky; López & Palacio, 2016).

A continuación, en la figura 11 se muestra un ejemplo del formato que podría tener una pila de producto:

Id	Prioridad	Descripción	Est.
1	Muy alta	Plataforma tecnológica	30
2	Muy Alta	Interfaz de usuario	40
3	Muy Alta	Un usuario se registra en el sistema	40
4	Alta	El operador define el flujo y textos de un expediente	60
5	Alta	xxx	999

Figura 11
Ejemplo de pila de producto.
Fuente: Menzinsky, A.; López, G. & Palacio, J. (2016); Alaimo D. (2013)

En la figura anterior se puede ver como cada una de las tareas son priorizadas para definir cuáles son las más importantes y las que tienen que ser finalizadas más rápidamente, según la estrategia del negocio.

2.5.2.3.2 Pila del *sprint*

Es un conjunto de ítems de la pila del producto (*Product Backlog Items* (PBI)) que fueron seleccionados para trabajar en ellos durante un cierto *Sprint*, junto con las tareas que el equipo de desarrollo ha identificado que debe cumplir para poder crear un incremento funcional potencialmente entregable al final del *Sprint*. (Alaimo, 2013)

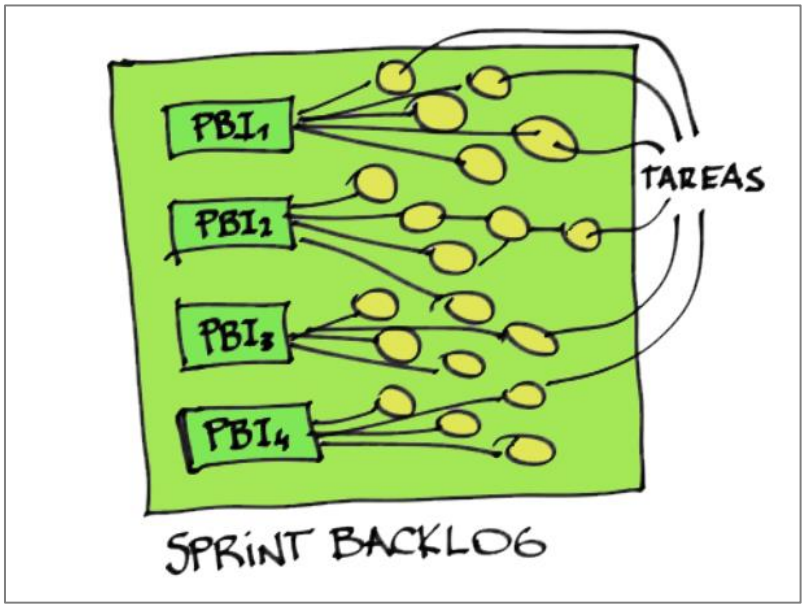


Figura 12
Ejemplo pila del Sprint.
Fuente: Alaimo D. (2013)

Esta lista se genera al inicio de cada uno de los *Sprints* y representa aquellas características que el equipo se compromete a desarrollar durante la iteración actual, los ítems incluidos se dividen en tareas las cuales generalmente, no deben demandar una duración superior a un día de trabajo del miembro del equipo asignado a la tarea (Bahit, 2012).

La pila del *Sprint* es actualizada diariamente por el equipo y muestra la siguiente información:

- Las tareas pendientes, en curso y terminadas.
- La estimación del esfuerzo pendiente de cada una de las tareas inconclusas.
- El nombre del miembro que ha sido asignado a la tarea.

El *Backlog Sprint* se visualiza mediante un tablero físico como el siguiente:

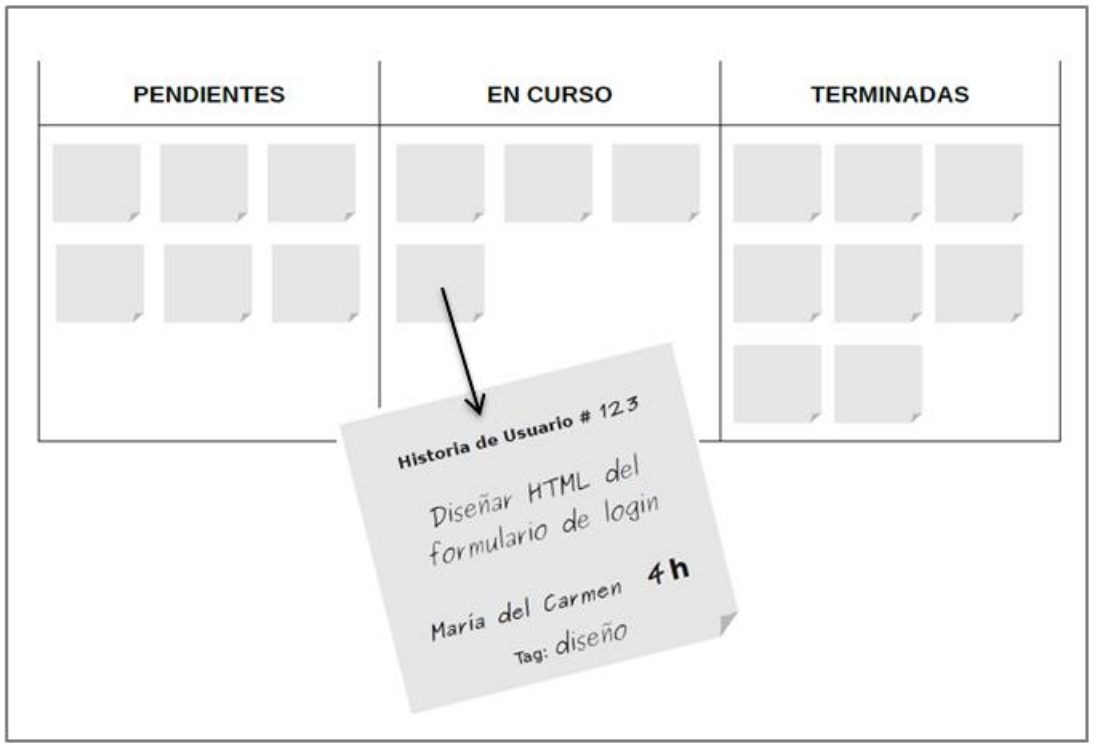


Figura 13
Tablero físico (Scrum Taskboard).
Fuente: Bahit E. (2012).

Generalmente, este tablero es ubicado en un lugar visible para que todos los miembros del equipo puedan tener claro el estado actual de todas las tareas que componen el proceso.

2.5.2.3.3 Incremento

Es la parte del producto producida en un *Sprint* y tiene como característica el estar completamente terminada y operativa, en condiciones óptimas para ser entregada al

usuario final, es importante aclarar que los prototipos, módulos, submódulos, partes pendientes de pruebas o integración no se deben considerar como incremento. (Deemer et al. 2012).

2.5.2.4 Eventos de Scrum

Es de vital importancia para el equipo de Scrum asegurarse que las tareas que se están realizando son correctas y de no ser así, poder tomar decisiones oportunas que permitan mantener un proceso que aporte valor. Por eso, en cada uno de los eventos de Scrum se aprovecha para realizar la inspección y adaptación de cualquier aspecto que se presente. A continuación, se explicará de forma breve cada uno de estos eventos:

2.5.2.4.1 *Sprint*

También conocidas como iteraciones, son el elemento clave de Scrum para mantener un ritmo de avance continuo; la duración máxima debe ser de cuatro semanas, durante las cuales se construye un incremento del producto, dicho incremento debe ser completamente operativo y útil para el usuario final (Deemer et al. 2012).

2.5.2.4.2 Planificación del *Sprint*

Es una reunión en la cual se toman como base las prioridades y necesidades de negocio, y se determinan cuáles y cómo van a ser las funcionalidades que se

incorporarán al producto en el siguiente *sprint*. La misma es conducida por el Scrum Master y deben estar presentes todos los implicados en el proyecto. Dicha reunión puede tardar hasta una jornada de trabajo completa, según el volumen o complejidad de las historias de usuario que se desean incluir en el próximo incremento (Deemer et al. 2012).

Básicamente en la reunión se debe dar respuesta a dos cuestiones:

- ¿Qué se entregará al terminar el Sprint?
- ¿Cuál es el trabajo necesario para realizar el incremento previsto o como lo llevará a cabo el equipo? (Deemer et al. 2012)

2.5.2.4.3 Scrum diario

Consiste en una reunión diaria la cual debe ser breve, no más de quince minutos, en la que el equipo sincroniza el trabajo y establece el plan a seguir en las siguientes 24 horas (Deemer et al. 2012).

2.5.2.4.4 Revisión del *Sprint*

Reunión realizada al final del *Sprint* con el fin de comprobar el incremento generado, no debe tener una duración superior a las cuatro horas, en el caso de *Sprints* extensos, y

lo habitual es que con una o dos horas de duración suele ser suficiente para revisar todo lo necesario (Deemer et al., 2012).

2.5.2.4.5 Retrospectiva del *Sprint*

Reunión realizada tras la revisión de cada *Sprint* y antes de la reunión de planificación del siguiente, tiene una duración recomendada no mayor a las tres horas; en ella el equipo realiza un autoanálisis de su forma de trabajar e identifica fortalezas y debilidades, esto con el objetivo de consolidar y afianzar las primeras, y planificar acciones de mejora sobre las segundas.

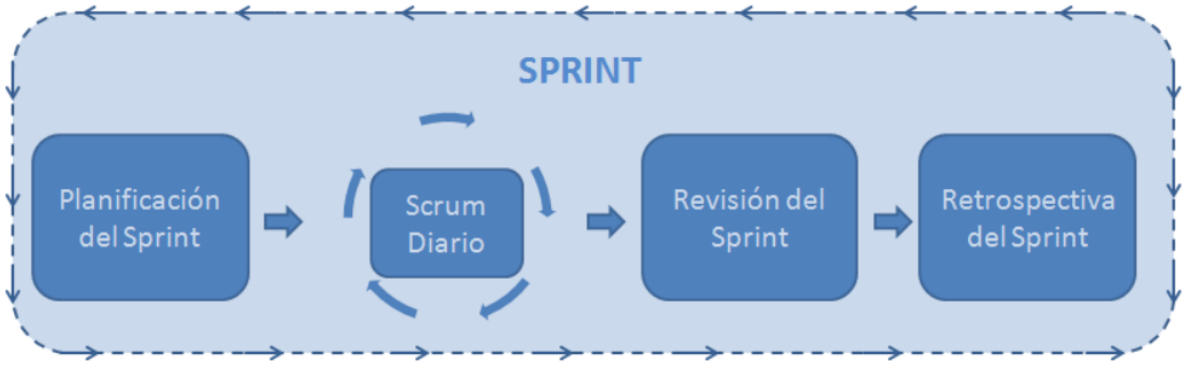


Figura 14
Eventos de Scrum.
Fuente: disponible en: http://managementplaza.es.

2.5.3 Metodología Kanban

Kanban es un término japonés que puede ser traducido al español como “tarjetas visuales” (Kan: visual, Ban: tarjeta). De todas las metodologías ágiles, Kanban es la

más reciente, ya que comenzó a implementarse prácticamente diez años después que otras metodologías.

Fue descubierta por el ingeniero Taiichi Ohno, de la empresa automotriz Toyota, y llegó a la industria del *software* en el año 2004, de la mano de David Anderson (pionero del método Kanban para su aplicación en Tecnologías de Información) en la Unidad de Negocio XIT de Microsoft, arrojando resultados muy alentadores. Según David Anderson, Kanban permitió en Microsoft, producir cambios incrementales en la forma de trabajo con una mínima resistencia al cambio, generando un incremento de productividad superior al 300% y, en promedio, una reducción del ciclo de desarrollo en un 90%.

Básicamente, Kanban se centra en una producción a demanda, es decir, se produce únicamente lo necesario, de manera tal que el ritmo de la demanda sea quien controle el ritmo de la producción (se requieren ene partes, entonces solo se producen ene partes), limitando el trabajo en curso a una cantidad predefinida (Bahit, 2012).

Es posible resumir Kanban como un sistema o proceso diseñado para disparar trabajo cuando hay capacidad para procesarlo (Bahit, 2012, citado por David Anderson, Microsoft).

En un sistema Kanban, este disparador es representado por tarjetas, que son distribuidas en cantidades limitadas (esto es, lo que limitará el trabajo en curso). Cada

ítem de trabajo es acompañado por una de estas tarjetas, por lo cual, un nuevo ítem solo podrá ser iniciado si se dispone de una tarjeta Kanban libre. En caso de que no existan tarjetas libres, no se pueden iniciar nuevos trabajos. Y cuando un ítem de trabajo es finalizado, una nueva tarjeta es liberada, permitiendo el comienzo de un nuevo ítem de trabajo (Bahit, 2012).

2.5.3.1 Las tres reglas de Kanban

Según Bahit (2012), la esencia de Kanban reposa sobre tres reglas principales, las cuales son descritas a continuación:

I. Mostrar el proceso
II. Limitar el trabajo en curso
III. Optimizar el flujo

2.5.3.1.1 Mostrar el proceso

Esta regla busca hacer visibles los ítems de trabajo permitiendo conocer de manera explícita el proceso de trabajo actual, así como los impedimentos que vayan surgiendo. Dicha visualización, se realiza a través de tableros físicos, al igual que en Scrum, solo que con diferentes columnas.

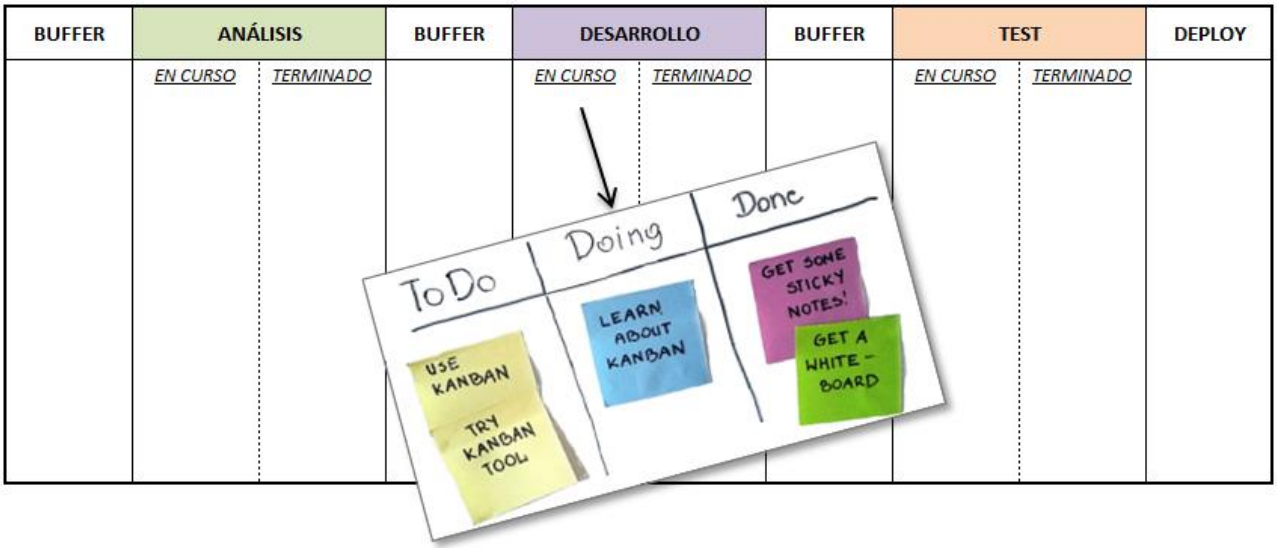


Figura 15
Ejemplo tablero Kanban.
Fuente: Bahit, E., (2012).

Los tableros físicos son utilizados en Kanban para representar el flujo de trabajo, éste se verá reflejado de izquierda a derecha en el tablero, mediante columnas que representen cada una de las etapas (análisis, diseño, desarrollo, pruebas). Cada una de esas columnas pude dividirse, a su vez, en dos: en curso y terminado. Frecuentemente, en cada una de las columnas representativas de un proceso, suele colocarse una columna intermedia para los ítems en espera (Bahit, 2012).

En el caso de Kanban, la implementación de tableros físicos es fundamental a la hora de comprender la capacidad de proceso del equipo de desarrollo. Para cumplir con la regla de mostrar el proceso, será necesario definir con precisión:

- Punto de inicio y finalización de la visibilidad del proceso
- Tipos de ítems de trabajo
- Diseño de tarjetas de acompañarán a cada ítem

2.5.3.1.2 Limitar el trabajo en curso

En Kanban el límite de trabajo en curso está dado por el *Work in Progress* (WIP) o Trabajo de Curso. El límite WIP consiste en definir la cantidad de ítems simultáneos que pueden ser ejecutados en un mismo proceso. Cada uno de estos procesos puede tener distintos límites WIP, como se muestra a continuación:

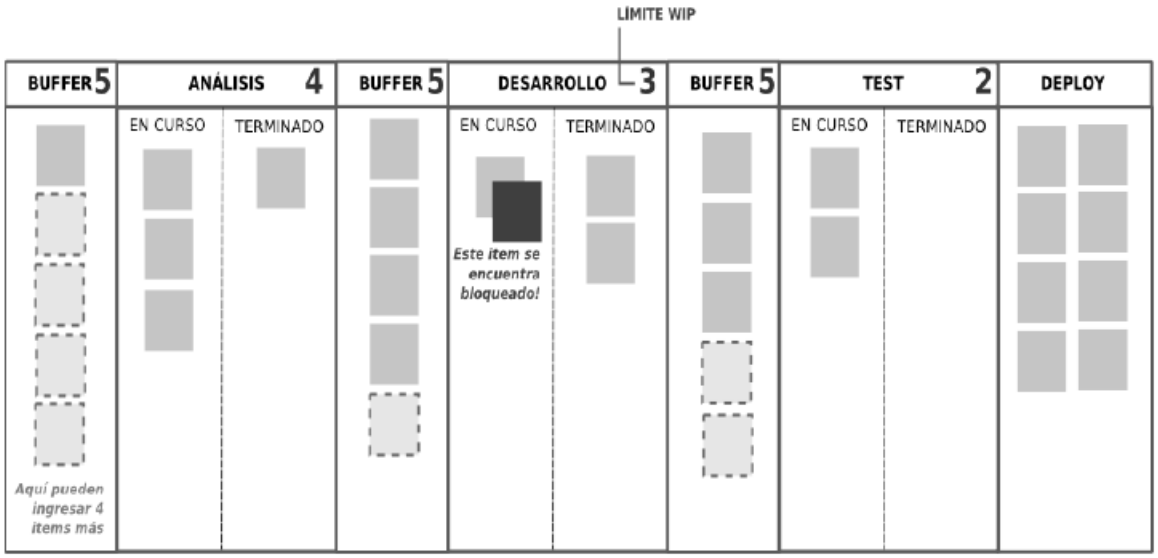


Figura 16
Límite work in progress.
Fuente: Bahit E. (2012).

Los límites WIP permiten detectar cuellos de botella rápidamente. Con solo observar el tablero prestando atención en las columnas bloqueadas y su proceso anterior y posterior inmediato, se puede deducir dónde se encuentran los puntos de conflicto y trabajar para remediarlos (Bahit, 2012).

2.5.3.1.2 Optimizar el flujo

Es el aumento visible de los ítems de cada uno de los procesos en un sistema Kanban. El objetivo de optimizar dicho flujo de trabajo es alcanzar un proceso de desarrollo estable, previsible y acorde a las necesidades del proyecto, en el cual se distinga un ritmo de trabajo parejo (Bahit, 2012).

Para Bahit (2012, citado por Mary y Tom Poppendieck, Lean software development): “Un flujo regular, establece la capacidad de un equipo para entregar *software* funcionando con una velocidad fiable. Una organización que entrega con un flujo regular logra establecer claramente las posibilidades de su proceso y puede medir fácilmente su capacidad”.

Si bien no existen reglas definidas para la optimización del flujo de trabajo en Kanban, las principales actividades sobre las cuales suele enfocarse esta optimización, en la práctica, son:

- El trabajo sobre cuellos de botella
- Análisis sobre las colas de entrada (buffer de ítems de trabajo)
- Mejoras que impliquen modificaciones en el proceso de creación de valor.

2.5.4 Metodología *eXtreme Programming* (XP)

También conocida como XP, es una metodología creada por Kent Beck, Ward Cunninghamn y Ron Jeffries a finales de los noventa, propone un conjunto de prácticas que, al ser aplicadas de manera simultánea, pretenden enfatizar los efectos positivos en un proyecto de desarrollo de *software* (Bahit, 2012).

Para alcanzar el objetivo de *software* como solución ágil, la metodología XP se estructura en tres capas que agrupan las doce prácticas básicas de XP:

1) **Metodología de programación:** agrupa las prácticas: diseño sencillo, test, refactorización y codificación con estándares.

2) **Metodología de equipo:** agrupa las prácticas: propiedad colectiva de código, programación en parejas, integración continua, cuarenta horas semanales y metáfora del negocio.

3) **Metodología de procesos:** agrupa las prácticas: cliente *in situ*, entregas frecuentes y planificación del juego. (Fernández, 2013)

La metodología XP es algo más que una simple guía de buenas prácticas ya que convierte la programación en algo mucho más humanizado, que permite a las personas relacionarse y comunicarse para encontrar soluciones, sin jerarquías ni enfrentamientos. Los analistas y programadores trabajan en equipo con el usuario final, todos comprometidos con un mismo objetivo. (Fernández, 2013)

2.5.4.1 Las variables XP: coste, tiempo, calidad y alcance

El punto de partida de la metodología XP son las variables que utiliza para cada proyecto: coste (la inversión económica y en recursos), tiempo (el tiempo empleado, determinado por la fecha de entrega final), calidad (del código y del aplicativo desarrollado) y alcance (conjunto de funcionalidades). (Fernández, 2013)

2.5.4.2 Valores de XP

Los creadores de esta metodología quisieron medir su utilidad a través de cuatro valores, que representan aquellos aspectos cuyo cumplimiento garantiza el éxito en el proyecto: comunicación, simplicidad, realimentación y coraje. A continuación, se muestra el significado de cada uno de ellos:

- **Comunicación:** debe ser fluida entre todos los participantes involucrados en el proyecto; además, el entorno tiene que favorecer la comunicación espontánea, ubicando a todos los miembros en un mismo lugar. La comunicación directa aporta mucho más valor que la escrita, es posible observar los gestos del usuario final o la expresión de cansancio de algún miembro del equipo.

- **Simplicidad:** cuanto más sencilla sea la solución, más fácilmente puede ser adaptada a cambios futuros. Las complejidades aumentan el coste del cambio y disminuyen la calidad del *software*. En la metodología XP se deben olvidar frases como: “haremos un sistema genérico que (…)”, o “esta parte de código se agrega por si acaso algún día se necesita”.

- **Realimentación:** el usuario final debe utilizar desde la primera entrega el aplicativo desarrollado, brindando sus impresiones y sus posibles necesidades no satisfechas, de manera tal que esas historias vuelvan a formar parte de los requisitos del sistema.

- **Coraje:** “Si funciona no lo toques”, la frase más típica de los desarrolladores de software. Con XP se deben manipular continuamente las cosas que ya funcionan, para mejorarlas, por lo que se debe cambiar esta frase por la de: “Si funciona, puedes mejorarlo” y para ello, se requiere de mucho valor y coraje. (Fernández, 2013)

2.5.4.3 Las doce prácticas básicas de XP

Kent Beck, Ward Cunninghamn y Ron Jeffries tenían muy claro las prácticas que les habían dado los mejores resultados en sus proyectos, así que intentaron aplicarlas todas juntas, para ello crearon las doce prácticas que se refuerzan entre sí para obtener los mejores resultados. En la siguiente figura se pueden observar las relaciones que existen entre ellas:

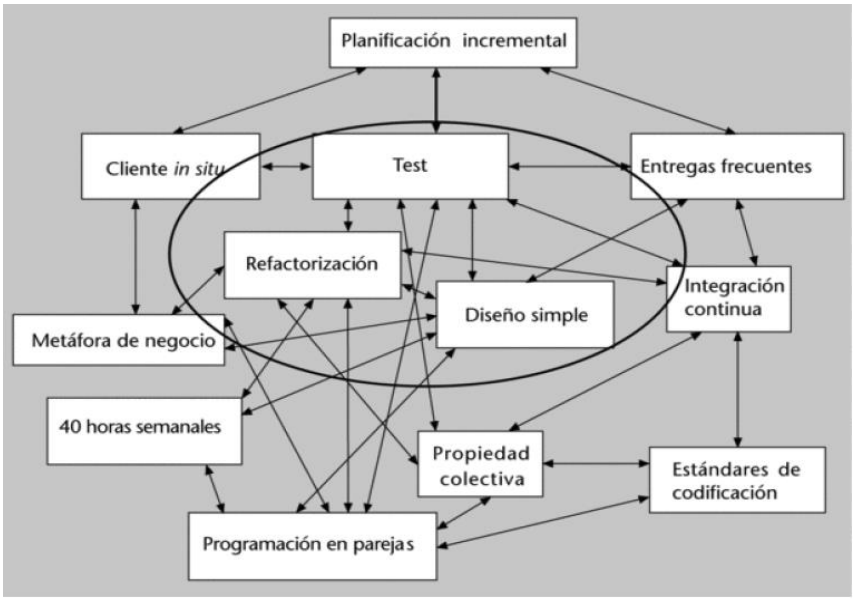


Figura 17
Relación de las 12 prácticas básicas de XP.
Fuente: Fernández, J. (2013).

En la figura 17 se puede observar la relación que existe entre las 12 prácticas de XP, Fernández (2013) explica dicha relación de la siguiente manera:

> En el centro hemos situado las prácticas que más resultados nos pueden dar al adaptarlas; no son otras que el diseño simple, el test y la refactorización.
> Incluso si no queremos tomar la totalidad de las prácticas de XP adoptando estas tres a nuestra metodología habitual, podemos tener una sustancial mejora en los resultados obtenidos. (p.24)

2.5.4.3.1 Diseño simple

El principio se basa en utilizar el diseño más sencillo que consiga que todo funcione, de esta manera se facilita el mantenimiento y se minimiza el riesgo de modificaciones que sean realizadas sin comprender el código.

Fernández (2013) indica que la metodología XP establece como código sencillo aquel que cuente con las siguientes características:

- No posee código redundante, ni duplicado.
- Supera todos los test de funcionalidad, integridad y aceptación.
- No utiliza sintaxis complejas, es decir, que queda clara la intención de los programadores en cada línea de código.
- Contiene el menor número posible de clases y métodos. (Fernández, 2013)

2.5.4.3.2 Refactorización

Así como en el Génesis se explica que en un principio todo era caos y luego todo fue orden, en el ámbito del desarrollo de *software* generalmente pasa lo contrario. Inicialmente todo son líneas de código bien ordenadas y comentadas, pero conforme se van introduciendo cambios, el orden se va perdiendo hasta llegar a una serie de líneas de código caóticas.

Para mantener la curva del coste de cambio tan plana como sea posible, en la metodología XP se aplica la refactorización, que consiste en modificar el código para dejarlo en buen estado, volviendo a escribir las partes que sean necesarias, pero siempre desde un punto de vista global a la funcionalidad, independientemente del cambio que sea realizado (Fernández, 2013).

2.5.4.3.3 Test

El objetivo principal de los test no consiste en detectar errores, sino en evitarlos, y tampoco se trata de corregirlos, sino prevenirlos. Si una funcionalidad no ha sido testeada, sólo funciona en apariencia, los test deben ser aplicados tras cada cambio realizado, si no se realizan es posible incurrir en fallos humanos lo cual puede resultar fatal (Fernández, 2013).

Existen tres tipos de test, los cuales se describen a continuación:

- **Test de aceptación:** es creado conjuntamente con el cliente final y debe reflejar las necesidades funcionales solicitadas.
- **Test unitario:** es creado por el programador para ver que los cambios realizados funcionan correctamente.
- **Test de integridad:** es creado por el equipo de desarrollo para probar que todo el conjunto funciona correctamente después de la nueva modificación.

2.5.4.3.4 Estándares de codificación

El equipo de desarrollo debe estandarizar el modo en que se escribe el código y utilizar nomenclaturas propias que todos los miembros del equipo puedan entender. Por ejemplo, si el programa ha de escribirse en Java, lo mejor es que todos utilicen las convenciones definidas a nivel internacional por Java.

El hecho de utilizar una nomenclatura común permite que cualquier persona del equipo entienda con mayor facilidad el código desarrollado por otro miembro; de esta manera, se facilitan las modificaciones futuras y la refactorización (Fernández, 2013).

2.5.4.3.5 Propiedad colectiva del código

Permite aplicar la refactorización y asegurar que el diseño es simple y que se codifica según los estándares definidos; además, se debe eliminar otra de las ideas que están muy arraigadas en el mundo del desarrollo de aplicaciones: la "propiedad individual" del código.

Frases como "que lo modifique quien lo hizo que seguro que lo entiende mejor" o "¿quién ha tocado mi función?" dejan de tener sentido en la metodología XP, ya que el diseño simple garantiza que será fácilmente entendible; la refactorización permite que cualquier miembro del equipo rehaga el código para devolverle su sencillez en el caso el código escrito sea muy complejo; los test garantizan que los cambios realizados no alteren el comportamiento de la aplicación y la codificación con estándares aporta el grado de compresión adicional.

En XP el código es propiedad de todo el equipo y cualquiera de los miembros tiene el derecho y la obligación de modificarlo, para hacerlo más eficiente o comprensible, sin que nadie se sienta ofendido (Fernández, 2013).

2.5.4.3.6 Programación por parejas

Pensar a nivel general y en detalle a la misma vez, es muy complicado para una sola persona, entonces, ¿por qué no poner dos cerebros? Esto es exactamente lo que propone XP con la programación en parejas, uno de los miembros debe estar pensando a nivel táctico y el otro a nivel estratégico, de manera que esos dos procesos siempre estén activos reduciendo así los errores y mejorando la calidad del programa. El nivel de los miembros de la pareja debe ser equivalente, no sirve que uno sepa mucho y otro no tenga ni idea, deben de estar equilibrados y obviamente llevarse bien para que tenga éxito.

El hecho de asignar las tareas por parejas hace que el tiempo de aprendizaje se reduzca casi a cero, simplemente sustituyendo uno de los miembros por otro nuevo. Así pues, la rotación de áreas y de parejas garantiza que se puede hacer un reparto más equitativo del trabajo sin tener que depender de una sola persona para una labor específica.

Otro efecto muy importante que produce la programación en parejas es el psicológico, ya que disminuye la frustración de la programación en solitario, cada desarrollador tiene a alguien que entiende el problema justo al lado y con el que puede compartir los problemas para darle una solución más expedita (Fernández, 2013).

2.5.4.3.7 Integración continua

En XP no se espera a que todas las partes de código estén desarrolladas para ser integradas en el sistema, sino que a medida que se van creando las primeras funcionalidades ya se van ensamblando en el sistema, de manera que éste puede ser construido varias veces durante un mismo día. Esto se hace para que las pruebas de integración vayan detectando los errores desde el primer momento y no al final de todo.

Es responsabilidad de cada equipo publicar lo antes posible cada funcionalidad o cada modificación. La idea es que todos los miembros del equipo trabajen con la última versión del código (Fernández, 2013).

2.5.4.3.8 Cuarenta horas semanales

No se puede trabajar durante 14 horas seguidas y hacerlo con calidad y las semanas de 70 horas de trabajo son contraproducentes. Los equipos de XP están diseñados para ganar, no para morir en el intento. Al final de la semana se tiene que llegar cansado pero satisfecho, nunca exhausto ni desmotivado. Trabajar horas extra mina la moral y el espíritu del equipo. Si durante dos semanas hay que hacer horas extras, entonces es que el proyecto va mal y se debe replantear (Fernández, 2013).

2.5.4.3.9 Metáfora del negocio

Para que dos o más personas se puedan comunicar de forma eficiente, deben tener el mismo vocabulario y compartir el mismo significado. El modelo de negocio que entiende el usuario final seguramente no se corresponderá con el que cree entender el desarrollador de *software*. Es por esto por lo que en los equipos de XP deben crear metáforas con la que el usuario final se encuentre cómodo y que le sirva al equipo de desarrollo a la hora de definir cómo debe ser el sistema (Fernández, 2013).

Bahit (2012) define la metáfora de negocio como: “La forma de ser didácticos para explicar a nuestro receptor, un concepto técnico y que éste, lo comprenda con facilidad” (p.75).

2.5.4.3.10 Cliente *in situ*

En XP es necesario que el usuario final esté dispuesto a participar activamente del proyecto, contando con la disponibilidad suficiente, para interactuar con el equipo en todas las fases de su desarrollo.

La comunicación cara a cara con el usuario final es fundamental, ya que a partir de esta, el equipo avanzará más rápidamente en el proyecto, puesto que:

- Evacuará todas sus dudas sobre el proyecto, con el cliente y en el momento en que estas surjan.
- Se establecerán las prioridades a desarrollar, en tiempo real, así como la resolución de conflictos. (Bahit, 2012)

2.5.4.3.11 Entregas frecuentes

Se deben desarrollar lo antes posible versiones pequeñas del sistema, que aunque no tengan toda la funcionalidad, puedan aportar una idea general de cómo ha de ser la entrega final y que sirvan para que el usuario final se vaya familiarizando con el entorno, así como para que el equipo de desarrollo pueda ejecutar las pruebas de integridad. (Fernández, 2013)

2.5.4.3.12 Planificación incremental

La planificación nunca será perfecta, variará en función de cómo se presenten las necesidades del negocio y en cada ciclo de replanificación se volverán a establecer las cuatro variables de la metodología XP.

Asumir una planificación estática no corresponde con la agilidad que se pretende dar, ya que las necesidades del negocio pueden cambiar drásticamente mientras el proyecto se encuentra en la etapa de desarrollo. En XP la planificación se va revisando

continuamente, de forma incremental, priorizando aquellas necesidades de negocio que aporten mayor valor (Fernández, 2013).

2.5.4.4 El ciclo de vida de la metodología XP

Cuando Kent Beck y compañía aplicaban las doce prácticas de XP por separado, obtuvieron muy buenos resultados, así que unificarlas en el ciclo de vida de una metodología es lo que dio origen a XP (Fernández, 2013).

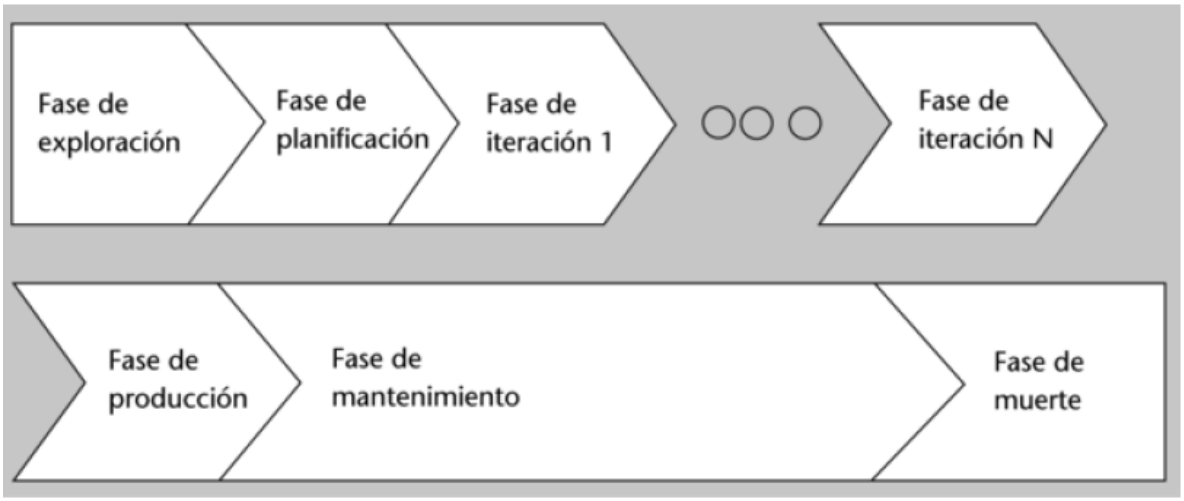


Figura 18
Ciclo de vida de XP.
Fuente: Fernández J. (2013).

2.5.4.4.1 La fase de exploración

Es la primera fase del ciclo de vida de la metodología XP y consta de tres procesos:

1) Las historias de usuario
2) El *spike* arquitectónico
3) La metáfora del negocio

Todo inicia con las **historias de usuario,** en esta fase los usuarios plantean a grandes rasgos las funcionalidades que requieren, dichas historias tienen el mismo propósito que los casos de uso, salvo que son escritas por los usuarios y no por el analista, deben ser descripciones cortas y plasmadas en el lenguaje del usuario sin terminología técnica. No se deben confundir las historias de usuario con el análisis de requisitos, la principal diferencia radica en la profundidad de análisis, con los requisitos se pretende llegar al último detalle. (Fernández, 2013)

En el ***spike* arquitectónico**, en este proceso el equipo de desarrollo:

- Prueba la tecnología.
- Se familiariza con la metodología.
- Se familiariza con las posibilidades de la arquitectura.
- Realiza un prototipo que demuestre que la arquitectura es válida para el proyecto. (Fernández, 2013)

Una vez finalizadas las historias de usuario y el *spike* arquitectónico, se pasa a realizar conjuntamente la **metáfora del negocio**, la misma debe cumplir las siguientes características:

- Debe ser una historia común compartida por el usuario y el equipo de desarrollo.
- Debe servir para que el usuario se sienta a gusto refiriéndose al sistema con los términos que conoce.
- Debe servir a los desarrolladores para implementar las clases y objetos del sistema. (Fernández, 2013).

2.5.4.4.2 La fase de planificación

La planificación se compone de un procedimiento establecido, el cual se muestra a continuación:

- El cliente entrega al equipo de desarrollo las historias de usuario que ha confeccionado, pero priorizándolas de mayor a menor importancia.

- El equipo de desarrollo las estudia y estima el coste de implementarlas:
    - Si el equipo de desarrollo considera que la historia de usuario es demasiado compleja, entonces el usuario final debe descomponerla en varias historias independientes más sencillas.

  - Si el equipo de desarrollo no ve claro cómo implementar una parte de la historia, el usuario puede realizar un *spike* tecnológico para ver cómo se podría implantar y así poder evaluar el coste.

- Una vez que se tiene disponible la lista de historias priorizadas junto con su coste de implementación, se procede a convocar la reunión del plan de entregas.
  - El plan de entregas se compone de una serie de planes de iteración en el que se especifica qué funcionalidades se van a implementar en cada vuelta de la fase de iteraciones.
  - Participan de esta reunión tanto los usuarios como el equipo técnico y cada uno debe aportar su visión del negocio, de manera que se obtengan más rápidamente aquellas funcionalidades que den el mayor beneficio posible para el negocio.
  - A cada iteración se le asigna un tiempo intentando que todas sean más o menos idénticas (aunque no es necesario).
  - Se determina el alcance del proyecto. (Fernández, 2013).

2.5.4.4.3 La fase de iteraciones

Debido a que el proyecto debe estar dividido en iteraciones, esta fase de repetirá tantas veces como la cantidad de iteraciones existentes. Generalmente, cada iteración suele ser de dos a tres semanas.

Lo más importante es que en cada iteración se realicen primeramente las tareas que aportan más valor al negocio, de manera que, si por algún motivo se debe reducir el alcance del proyecto, sólo afecte a las funcionalidades secundarias del aplicativo. (Fernández, 2013).

2.5.4.4.4 La fase de producción

Esta fase es alcanzada desde la primera versión que logre las funcionalidades mínimas que aporten un valor real al negocio y una operativa estable. Es decir, no se debe esperar a tener todas las funcionalidades implementadas, sino que en cuando se tenga algo que los usuarios puedan utilizar y que ayude al negocio, entonces se pone en producción esa versión.

Durante esta fase, el ritmo de desarrollo tiende a decaer debido a que el equipo debe solventar las posibles incidencias que reporten los usuarios. Es por esto que en algunas ocasiones es necesario incorporar nuevo personal al grupo de trabajo. (Fernández, 2013).

2.5.4.4.5 La fase de mantenimiento

Una vez que el alcance del proyecto se ha conseguido y que todas las funcionalidades se encuentren en producción, se revisan en conjunto con el usuario final aquellas nuevas historias de usuario que se han producido tras la puesta en producción del

proyecto. Estas nuevas funcionalidades se van incorporando según su valor de negocio y el presupuesto adicional del que se disponga.

El equipo de desarrollo se reduce a la mínima expresión, dejando algunos miembros para el mantenimiento (Fernández, 2013).

2.5.4.4.5 La fase de muerte del proyecto

Cuando no existen más historias de usuario para introducir en el sistema o cuando se reduce progresivamente valor de las historias de usuario implementadas en él, el proyecto entra en la fase de muerte. Se irá disminuyendo el tiempo invertido en él, hasta abandonarlo totalmente cuando no aporte valor al negocio o cuando sus historias de usuario hayan sido absorbidas por otro sistema de información (Fernández, 2013).

2.5.4.5 Roles de la metodología XP

Cada rol tiene funciones claras dentro de la metodología XP. Cada miembro del equipo puede ejecutar uno o varios roles, o incluso cambiar de rol durante las diferentes fases del proyecto. A continuación, se describen los roles y sus principales funciones:

Programador:

- Escribe las pruebas unitarias.
- Produce el código del programa.

Cliente:

- Escribe las historias de usuario.
- Diseña las pruebas de aceptación.
- Prioriza las historias de usuario.
- Representa al colectivo de usuarios finales.
- Está siempre disponible para consultas.

Encargado de pruebas (*tester*):

- Ayuda al cliente a diseñar pruebas de aceptación.
- Ejecuta las pruebas de aceptación.
- Ejecuta las pruebas de integración.
- Difunde los resultados entre el equipo de desarrollo y el cliente.

Encargado de seguimiento (*tracker*):

- Se encarga de realimentar todo el proceso de XP, midiendo las desviaciones con respecto a las estimaciones y comunicando los resultados para mejorar las siguientes estimaciones.
- Realiza el seguimiento de cada iteración del proceso de XP tanto en la etapa de iteraciones como en la de producción.

- Revalúa la posibilidad de incorporar o eliminar historias de usuario.

Entrenador (*coach*):

- Se encarga del proceso global.
- Garantiza que se sigue la filosofía de XP.
- Conoce a fondo la metodología.
- Provee guías y ayudas a los miembros del equipo a la hora de aplicar las prácticas básicas de XP.

Consultor:

- No forma parte del equipo.
- Tiene un conocimiento específico de un área en concreto.
- Ayuda a resolver un problema puntual, ya sea de *spike* tecnológico o de valor de negocio.

Gestor (*boss*):

- Es el máximo responsable del proyecto.
- Hace de enlace con los clientes.
- Se encarga de coordinar y de garantizar las condiciones necesarias para el desarrollo del trabajo.

CAPÍTULO III

MARCO METODOLÓGICO

3.1 TIPO Y ENFOQUE DE LA INVESTIGACIÓN

3.1.1 Tipo de investigación

Según Barrantes (2014, citado por Pineda, M.), la finalidad de una investigación puede ser: básica orientada a la búsqueda de nuevos conocimientos y aplicada, donde su propósito es la resolución de problemas prácticos.

Sáez (2017) indica que la investigación aplicada “trata de determinar la aplicabilidad de una teoría y sus principios.” (párr. 3.1.1), exactamente lo que se intenta realizar en este proyecto, que es proponer la implementación de una metodología de desarrollo en la compañía, con el fin de solucionar una problemática existente.

3.1.2 Enfoque de la investigación

En relación con los tipos de enfoque de una investigación, Hernández (2014) expresa lo siguiente:

> Mientras que un estudio cuantitativo se basa en investigaciones previas, el estudio cualitativo se fundamenta primordialmente en sí mismo. El cuantitativo se utiliza para consolidar las creencias (formuladas de manera lógica en una teoría o un esquema teórico) (…) y el cualitativo, para que el investigador se forme creencias propias sobre el fenómeno estudiado”. (p.10)

Considerado las afirmaciones expuestas por Hernández (2014) sobre los tipos de enfoque, la naturaleza de este proyecto será mixta: cualitativa porque se obtendrá información por medio de la observación y entrevistas a colaboradores del departamento y cuantitativa, debido a que la literatura representa un papel muy importante en el conocimiento de las metodologías de desarrollo, que es uno de los temas de mayor relevancia en esta investigación.

3.2 FUENTES Y SUJETOS DE INFORMACIÓN

Según Hernández (2014), “es vital identificar fuentes de información (líderes, redes, grupos, organizaciones) e investigadores potenciales (socios).” (p.501). Para la presente investigación, los sujetos de información serán los colaboradores del área de mantenimiento y desarrollo del SIMA, los usuarios expertos del sistema (usuarios finales), la unidad funcional y jefes de área, los cuales mediante la aplicación de entrevistas, cuestionarios y comunicación verbal, brindarán la información requerida para encontrar la solución a la problemática planteada.

Tabla 4

Definición de sujetos de información

Puesto Laboral	Profesión u oficio	Experiencia	Relación con el tema
Jefatura	Administrador	Alta	Conocimiento general
Analistas	Ingenieros	Alta	Conocimiento general
Técnicos	Desarrolladores	Alta	Experto a nivel técnico
Usuarios funcionales	Ingenieros	Media	Experto en comportamiento del sistema
Usuarios finales	Profesionales	Media	Experto en comportamiento del sistema

Fuente: elaboración propia

En la tabla 4, se pueden ver los colaboradores que brindarán información importante para la presente investigación.

3.2.1 Fuentes primarias

Las fuentes primarias

> (…) son todas aquellas de las cuales se obtiene información directa, es decir, de donde se origina la información. Es también conocida como información de primera mano o desde el lugar de los hechos. Estas fuentes son las personas, las organizaciones, los acontecimientos, el ambiente natural, etcétera". (Bernal, 2010, p.191)

En esta investigación, se utilizarán como fuentes primarias: tesis, entrevistas y los cuestionarios que se les realizarán a los involucrados en el proceso de desarrollo y mantenimiento del SIMA.

3.2.2 Fuentes secundarias

Según Bernal (2010) se definen las fuentes secundarias como

> (…) todas aquellas que ofrecen información sobre el tema que se va a investigar, pero que no son la fuente original de los hechos o las situaciones, sino que sólo los referencian. Las principales fuentes secundarias para la obtención de la información son los libros, las revistas, los documentos escritos (en general, todo

> medio impreso), los documentales, los noticieros y los medios de información. (p.192).

En este caso las fuentes secundarias a utilizar serán todo tipo de libros que contengan la teoría requerida para el tema de investigación, la documentación brindada por el área de desarrollo y mantenimiento del SIMA y toda aquella información valiosa y fidedigna consultada en internet.

3.3 TÉCNICAS Y HERRAMIENTAS DE RECOLECCIÓN DE DATOS

Actualmente, existen una gran cantidad de técnicas o instrumentos para la recolección de información en el trabajo de campo de una determinada investigación; según el método y el tipo de investigación se utilizan unas u otras técnicas, a continuación, se definen las técnicas utilizadas en este proyecto:

Encuesta. Según Bernal (2010) la encuesta “se fundamenta en un cuestionario o conjunto de preguntas que se preparan con el propósito de obtener información de las personas.” (p.194). Basado en esto, se aplicará un cuestionario con preguntas abiertas sobre el proceso de desarrollo y mantenimiento, el cual estará dirigido a los programadores, implementadores, usuarios técnicos y usuarios finales. Los resultados obtenidos serán analizados y estudiados mediante gráficos estadísticos que muestren la información del problema.

Entrevista. De acuerdo con Bernal (2010), se define como la

> Técnica orientada a establecer contacto directo con las personas que se consideren fuente de información. A diferencia de la encuesta, que se ciñe a un cuestionario, la entrevista, si bien puede soportarse en un cuestionario muy flexible, tiene como propósito obtener información más espontánea y abierta. (Bernal, 2010, p.194).

Para el proyecto se someterá a una entrevista a los principales encargados del proceso de desarrollo y mantenimiento, con una serie de preguntas, a partir de las cuales se pretende establecer un diálogo, del cual se pueda conseguir información relevante sobre el manejo de las incidencias que se presentan cotidianamente.

Observación directa. La observación directa

> (…) permite obtener información directa y confiable, siempre y cuando se haga mediante un procedimiento sistematizado y muy controlado, para lo cual hoy están utilizándose medios audiovisuales muy completos, especialmente en estudios del comportamiento de las personas en sus sitios de trabajo. (Bernal, 2010, p.194)

En este caso, se utilizará la observación participante ya que el investigador es parte de la situación observada, lo que permite tener acceso a información a la que no tendría acceso un observador externo.

Análisis de documentos. “Técnica basada en fichas bibliográficas que tienen como propósito analizar material impreso. Se usa en la elaboración del marco teórico del estudio.” (Bernal, 2010, p.194). Con el análisis de documentos se pretende estudiar y conocer las metodologías de desarrollo existentes, para identificar cuál es la que más se adapta al área de desarrollo y mantenimiento del SIMA.

3.4 VARIABLES DE INVESTIGACIÓN

A continuación, se muestra el detalle de variables definidas para esta investigación:

Tabla 5

Definición de variables

Objetivos específicos	Variables asociadas	Descripción
Identificar las deficiencias existentes en el proceso actual de Mantenimiento y Desarrollo del SIMA	Tiempo de atención de requerimientos e incidencias	Extracción de estadísticas mediante la herramienta GESTIC para obtener los tiempos de atención
	Etapas de desarrollo utilizadas	Mediante la observación identificar las etapas de desarrollo que se utilizan actualmente
	Problemáticas visualizadas por el usuario funcional y final	Obtener los resultado obtenidos por las encuestas y entrevistas
Realizar un estudio comparativo entre las Metodologías de Desarrollo de Software más utilizadas en la actualidad	Metodologías de desarrollo	Estudiar mediante la bibliografía disponible las metodologías más utilizadas en el mercado actualmente
	Elementos comparativos	Identificar las características de cada una de las metodologías para realizar una comparación entre ellas
Identificar las necesidades y la brecha existente entre lo que se tiene actualmente y lo deseado para mejorar el proceso de mantenimiento y desarrollo del SIMA	Necesidades de la compañía	Mediante la observación y entrevistas se identifican las necesidades que tiene existen en el área
	Brechas existentes	Realizando una comparación entre el proceso actual y las necesidades se obtienen las brechas
Establecer la metodología de desarrollo que más se ajuste a las necesidades del área que brinda soporte al SIMA	Propuesta de la metodología	Analizar los resultados de las encuestas y entrevistas, aunado al conocimiento adquirido con el estudio de las metodologías para validar cual es la que más se adapte al área

Fuente: elaboración propia

En la tabla 5, se pueden observar las variables que intervienen en cada uno de los objetivos específicos que conforman la investigación.

3.5 DISEÑO DE LA INVESTIGACIÓN

En este apartado se definen las fases que componen el proyecto para lograr los objetivos propuestos:

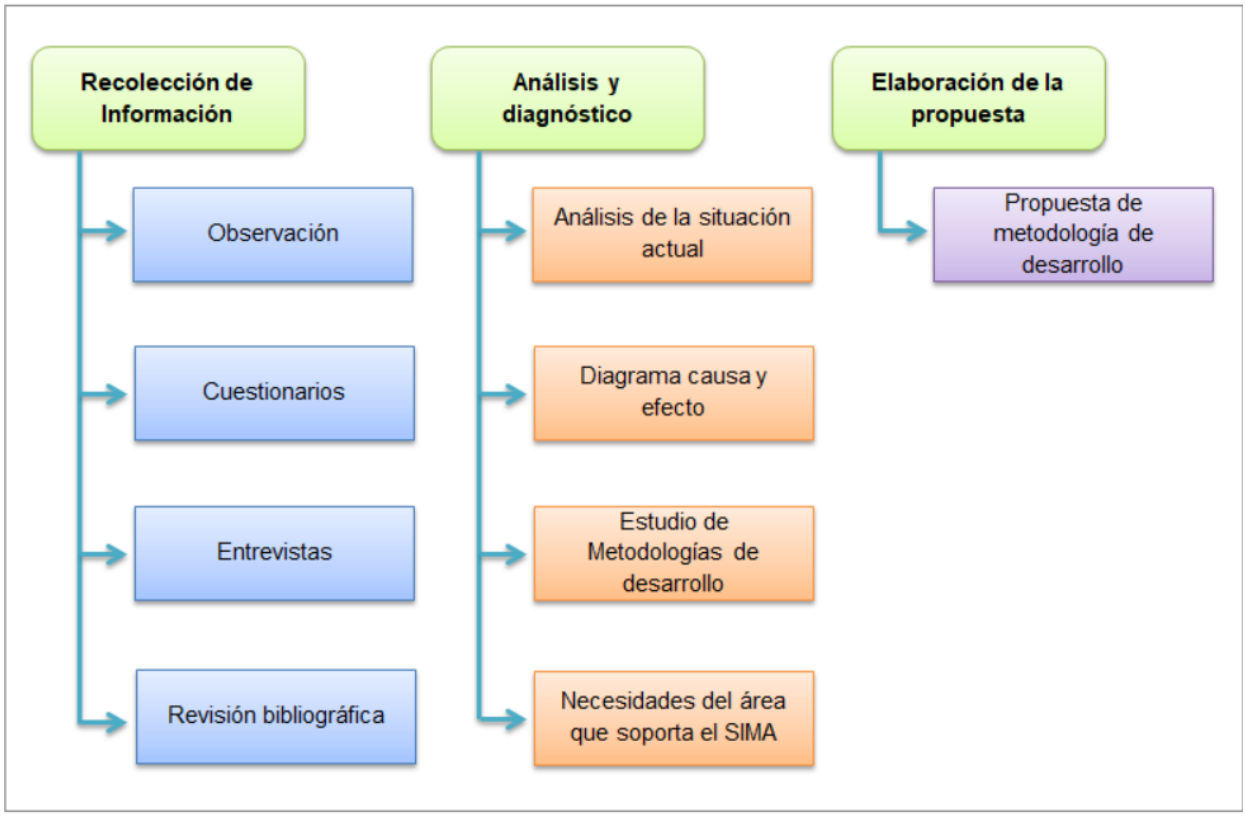


Figura 19
Mapa conceptual del diseño de la investigación.
Fuente: elaboración propia

En la figura 16 se muestran las distintas fases a realizar para la elaboración del proyecto y, a continuación, se expone cada una de ellas:

Recopilación de información: a través de la observación y entrevistas se determinarán las razones por las cuales es necesario plantear una propuesta para la mejora de procesos de desarrollo y mantenimiento de *software* del SIMA. Además, mediante la revisión de bibliografía se obtendrán los conocimientos relacionados a las metodologías de desarrollo más utilizadas en la actualidad.

Análisis y diagnóstico: una vez recopilada la información en la fase anterior, se lleva a cabo el análisis de la situación actual y necesidades del proceso de desarrollo y mantenimiento, además se definen las causas y efectos de la problemática, y se realiza el estudio y comparación de las metodologías de desarrollo.

Elaboración de la propuesta: con la información recolectada y el análisis de la misma, se procede a establecer la metodología de desarrollo que más se adapte al proceso de desarrollo y mantenimiento del SIMA.

CAPÍTULO IV

DIAGNÓSTICO DE LA SITUACIÓN ACTUAL

4.1 DESCRIPCIÓN DE LA SITUACIÓN ACTUAL

Para conocer la situación actual de la empresa se realizó una entrevista al Líder de Producto del SIMA, quien es la persona que conoce con exactitud el modo de operar en el área de mantenimiento y desarrollo del sistema, dicha entrevista provee una serie de aportes de gran importancia para tener un panorama más claro de cómo se están llevando a cabo las actividades del área.

A continuación, se muestra el detalle de las respuestas suministradas posterior a la entrevista aplicada:

Pregunta 1: Según su conocimiento, ¿se cuenta con una metodología de desarrollo de *software* para llevar a cabo las actividades de mantenimiento y desarrollo requeridas por el SIMA? (Si la respuesta es negativa pase a la pregunta número 3).

En la respuesta suministrada para la pregunta número 1 se indica lo siguiente: “No, sin embargo, se cuenta con un modo de trabajo definido, pero no documentado” (Solano Díaz, 2018, p.1). Esto indica que en el grupo de mantenimiento y desarrollo del SIMA no se cuenta con una metodología de desarrollo definida y documentada, pero sí se sigue un patrón de trabajo que ha sido adquirido por el grupo de manera informal.

Pregunta 2: ¿Cuál es la metodología de desarrollo de *software* que se utiliza actualmente para llevar a cabo las actividades de mantenimiento y desarrollo del SIMA? (En caso de recibir una respuesta pase a la pregunta número 4).

La respuesta a la pregunta número 2 no ha sido suministrada debido a que no se cuenta con una metodología de desarrollo de *software* formal.

Pregunta 3: ¿De qué modo se realizan las tareas de mantenimiento y desarrollo del SIMA al no contar con una metodología de desarrollo definida?

En la respuesta suministrada para la pregunta número 3 se indica lo siguiente:

> Mantenimiento: se llevan a cabo mediante el uso de una herramienta llamada Gestic, la cual está disponible para que los usuarios autorizados del SIMA puedan reportar cualquier tipo de incidente, este a su vez llega a una bandeja, donde los desarrolladores pueden reclamarlo (reservarlo para ser atendido), realizar el análisis respectivo y, posteriormente, efectuar los ajustes requeridos. El incidente reportado puede ser devuelto al usuario si no aplica y además, la persona que lo está atendiendo puede solicitar ampliaciones de información en caso de ser necesario. En cuanto a las incidencias, es frecuente que el usuario devuelva soluciones debido a que no era lo que se solicitaba o bien, que la corrección realizada sigue presentando el mismo error e incluso en ocasiones se generan nuevos problemas.
>
> Desarrollo: se lleva a cabo por fases, primero se realiza el levantamiento del requerimiento realizado por un analista de sistema en conjunto con el usuario (cabe aclarar que desarrollador que realiza el requerimiento no participa en esta actividad), en esta fase el resultado final es un documento donde se explica lo que

> se debe realizar, ERS (Especificación del Requerimiento). Luego, este documento es enviado al desarrollador para que realice el análisis respectivo (se puede apoyar en el analista para cualquier tipo de duda) y con base en esto, pueda generar un cronograma de actividades para conocer el tiempo requerido para la finalización del requerimiento. Si el cronograma es aprobado, el desarrollador inicia con la etapa de desarrollo (no existe etapa de diseño actualmente); Cuando toda la solución es desarrollada, el requerimiento está listo para pruebas funcionales (realizadas por la Unidad Funcional de Informática), posteriormente se realizan pruebas de usuario final, si todo es correcto entonces se realiza la implementación en ambiente de producción. (Solano Díaz, 2018, p.2).

La respuesta anterior se puede resumir de la siguiente forma:

- **Desarrollo**

    - El desarrollador no participa en el proceso del levantamiento de requerimientos.
    - Los desarrollos son rígidos, basados en la especificación del requerimiento.
    - No se cuenta con una etapa de diseño, muy importante para definir de qué manera se realizará la solución.

- La solución es desarrollada de principio a fin, sin ningún tipo de iteraciones o incrementos, esto quiere decir que la implementación en ambiente de producción es realizada al final con toda la solución.
- Se cuenta con un proceso de aseguramiento de la calidad (QA), el cual es realizado por la unidad funcional.

- **Mantenimiento**

  - Se cuenta con una herramienta de gestión de incidentes llamada Gestic.
  - La comunicación con el usuario puede no ser lo suficientemente expedita al tener que realizar solicitudes de aplicación (aclaración de dudas) por medio de la herramienta Gestic.
  - En algunas oportunidades, la calidad de la solución realizada no es la esperada.
  - Al ser los usuarios finales quienes reportan las incidencias, se presentan casos en que lo solicitado no es lo suficientemente claro para el desarrollador o simplemente el reporte no es suficiente.

Pregunta 4: Según su criterio, ¿la calidad del mantenimiento y desarrollo de *software* implementado en el SIMA es la esperada?

La respuesta suministrada para la pregunta número 4 indica lo siguiente:

> Se podría decir que no es la peor calidad, pero, sinceramente, sí le hace falta mucha calidad, actualmente la cantidad de incidentes reportados es muy elevada, incluso se reportan varios después de implementar un nuevo requerimiento e incluso existen varias quejas por parte de los usuarios finales. Basados en eso, se puede decir que aún no se cumple con el estándar de calidad esperado y queda mucho por mejorar. (Solano Díaz, 2018, p.3).

Esto indica que, con el proceso actual, el sistema se logra mantener, sin embargo, se requiere aplicar acciones para fortalecer la calidad de los productos desarrollados.

Pregunta 5: ¿Considera usted que el uso de una metodología de desarrollo de *software* puede mejorar la calidad del *software* creado o modificado?

La respuesta suministrada para la pregunta 5 indica lo siguiente: “Me parece que sí podría mejorar en gran medida, ya que este tipo de alternativas de desarrollo buscan precisamente eso: mejor calidad y eficiencia en las soluciones creadas” (Solano Díaz, 2018, p.3). Como se puede observar, sí se requiere mejorar la calidad de *software* del SIMA y una metodología de desarrollo de *software* podría ayudar a tal fin.

Pregunta 6: ¿Cree usted que metodologías de desarrollo ágiles son más eficientes que las metodologías tradicionales?

La respuesta suministrada en la pregunta número 6 indica lo siguiente: “Totalmente de acuerdo, fueron creadas con ese propósito para mayor eficiencia en los procesos”. (Solano Díaz, 2018, p.3). Tal y como se indica en la respuesta anterior este tipo de metodologías busca crear soluciones de forma ágil y que se adapten a lo realmente requiere el usuario final.

Pregunta 7: ¿Considera usted que el método utilizado para llevar a cabo las actividades de mantenimiento y desarrollo del SIMA cubre todas las necesidades de la compañía?

La respuesta obtenida indica:

> Definitivamente, falta mucho por mejorar, por ejemplo: la calidad de los productos realizados, los tiempos en atención de incidentes, la comunicación con el usuario final y, en general, se requiere contar con un modelo documentado donde esté claramente definida la forma de trabajar en el SIMA. (Solano Díaz, 2018, p.3).

Como se puede observar, es evidente que el método actualmente utilizado en el proceso de mantenimiento y desarrollo del SIMA no cumple con las expectativas del grupo y existen muchas carencias de deben ser mejoradas.

Pregunta 8: ¿Estaría dispuesto a impulsar la implementación de una metodología de desarrollo de *software* para optimizar las tareas de mantenimiento y desarrollo ejecutadas en el SIMA?

La respuesta suministrada en la pregunta número 8 expresa lo siguiente: “Totalmente de acuerdo, es una de las cosas que queremos proponer para este año para así mejorar día con día las actividades que realizamos”. (Solano Díaz, 2018, p.3). Basados en la respuesta anterior se puede ver el interés que existe en la implementación de una metodología de desarrollo que subsane las carencias actuales en el proceso de mantenimiento y desarrollo del SIMA.

Pregunta 9: ¿Estaría de acuerdo en que se realice un estudio de las distintas metodologías de desarrollo de *software* para identificar cuál puede proporcionar una mejora significativa en el proceso actual de mantenimiento y desarrollo del SIMA?

La respuesta para la pregunta número 9 fue la siguiente: “Exactamente, es lo que se desea, elegir una metodología que se adecúe de buena forma al proceso actual y que proporcione una mejora visible en las actividades que hoy en día se realizan”. (Solano Díaz, 2018, p.3). Como se puede observar en la respuesta anterior se cuenta con todo el apoyo para para poder identificar cuál es la metodología de desarrollo de *software* que se adapte de mejor forma al proceso de mantenimiento y desarrollo de *software* del SIMA.

Pregunta 10: ¿Cuál cree que sea el nivel de aceptación por parte del grupo de analistas y programadores en caso de implementar una nueva metodología de desarrollo de *software*?

La respuesta suministrada en la pregunta número 10 indica lo siguiente: “Para el SIMA contamos con un excelente grupo de colaboradores que estoy seguro aceptarán de buena forma cualquier metodología que pretenda mejorar los procesos actuales de mantenimiento y desarrollo” (Solano Díaz, 2018, p.4). Esto indica que la resistencia al cambio por parte del grupo para utilizar una nueva metodología de desarrollo de *software* debería ser bastante positiva.

Pregunta 11: ¿Qué aspectos considera que se pueden mejorar del método utilizado actualmente en el proceso de mantenimiento y desarrollo del SIMA?

La respuesta suministrada para la pregunta número 11 indica lo siguiente:

> Se espera que los desarrollos sean un poco más flexibles, actualmente, lo que se establece en el requerimiento, es como si estuviera "escrito en piedra", además creemos que la comunicación con el usuario final debe ser más directa, al fin y al cabo, ellos son los que utilizan el sistema y son ellos los que deben estar satisfechos con la solución realizada, definitivamente la Unidad Funcional que es la que actualmente realiza pruebas debe ser más rigurosa para buscar una mejora en la calidad del *software* creado. Los desarrollos de gran tamaño deben ser

> realizados en pequeños entregables y no todo junto como se realiza actualmente. Además, se requiere que el método de trabajo que se utilice esté claramente documentado y que sea conocido por todo el grupo actual y por otros miembros que se puedan unir en un futuro. Evidentemente, se debe reducir la cantidad elevadísima de incidentes reportados diariamente y para lograr esto, se requiere gran calidad del trabajo realizado por todo el grupo de mantenimiento y desarrollo del SIMA. (Solano Díaz, 2018, p.2).

La respuesta anterior se puede resumir señalando los siguientes aspectos a mejorar:

- Mayor flexibilidad en el proceso de desarrollo para que se puedan realizar ajustes necesarios, aunque no estén definidos en la especificación del requerimiento.
- Mejorar la comunicación con el usuario final, para que el desarrollador pueda ir validado si la solución en proceso va en buena línea.
- Que la Unidad Funcional sea más rigurosa en las pruebas de calidad, para así reducir la cantidad de incidencias reportadas por el usuario final.
- Que los proyectos grandes sean divididos en interacciones o entregables completamente funcionales para que se puedan ir implementando poco a poco en ambiente de producción y así disminuir el impacto.
- Definir una metodología desarrollo de *software* que sea conocida por todo el grupo y que sea difundida a toda persona que se incorpore en un futuro.

4.2 RECOLECCIÓN DE DATOS

Se ha realizado una encuesta a todo el grupo de trabajo del SIMA, con el fin de medir el conocimiento que tiene el grupo en relación con las metodologías de desarrollo de *software*, así como conocer la opinión sobre distintos aspectos involucrados en las actividades que se llevan a cabo en el área.

A continuación, se muestran los resultados obtenidos de la encuesta realizada:

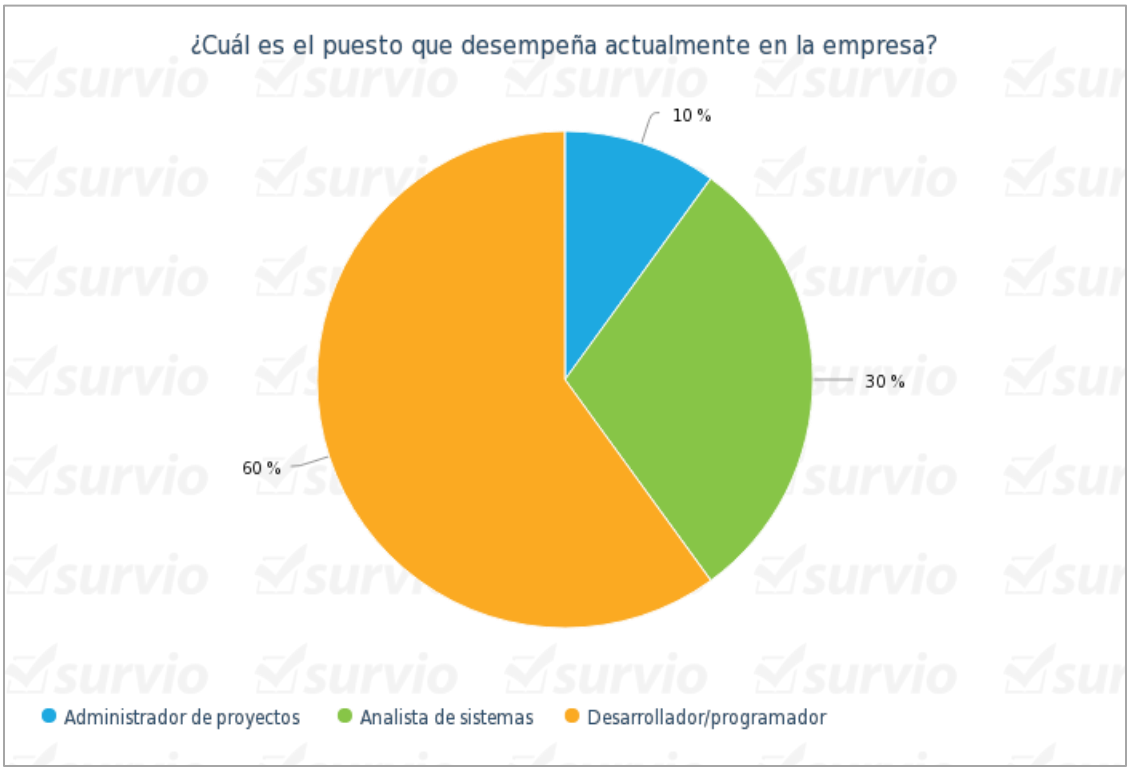


Figura 20
Gráfico de puestos desempeñados.
Fuente: elaboración propia

Como se puede observar en la figura 20, el 60% de los encuestados son desarrolladores, esto quiere decir que los datos obtenidos en la encuesta representan en su gran mayoría el criterio de los desarrolladores del SIMA.

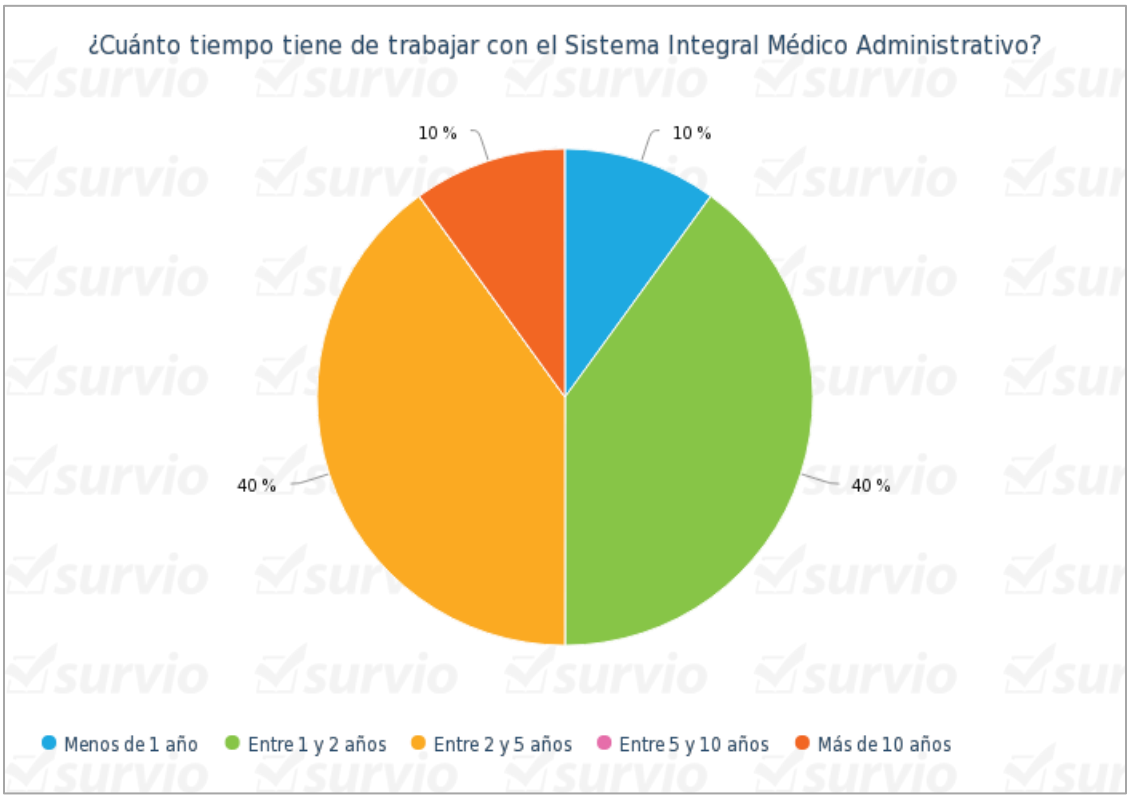


Figura 21
Gráfico sobre años de antigüedad del colaborador.
Fuente: elaboración propia.

Como se puede observar en la figura 21, el 80% de colaboradores tiene entre uno y 5 cinco años de trabajar con el SIMA (dividido de la siguiente forma: un 40% de colaboradores que han laborado entre uno y dos años y el otro 40% han laborado entre dos y cinco años). Por lo tanto, los resultados obtenidos representan en su gran mayoría a colaboradores con esa cantidad de años de antigüedad.

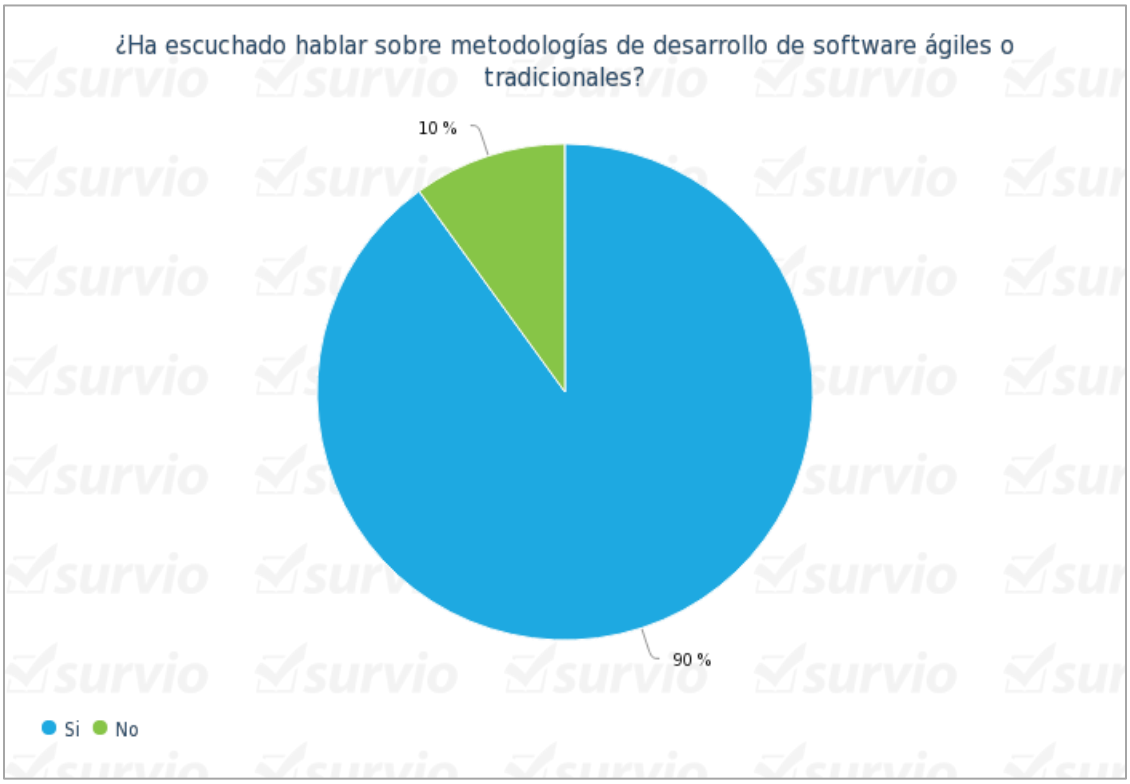


Figura 22
Gráfico sobre conocimiento de metodologías en el grupo de colaboradores de SIMA.
Fuente: elaboración propia.

Como se puede ver en la figura 22, el 90% de los encuestados asegura haber escuchado sobre metodologías de desarrollo de *software*, ya sean tradicionales o ágiles. De esta manera, se deduce que la mayor parte del grupo de colaboradores del SIMA comprende del tema.

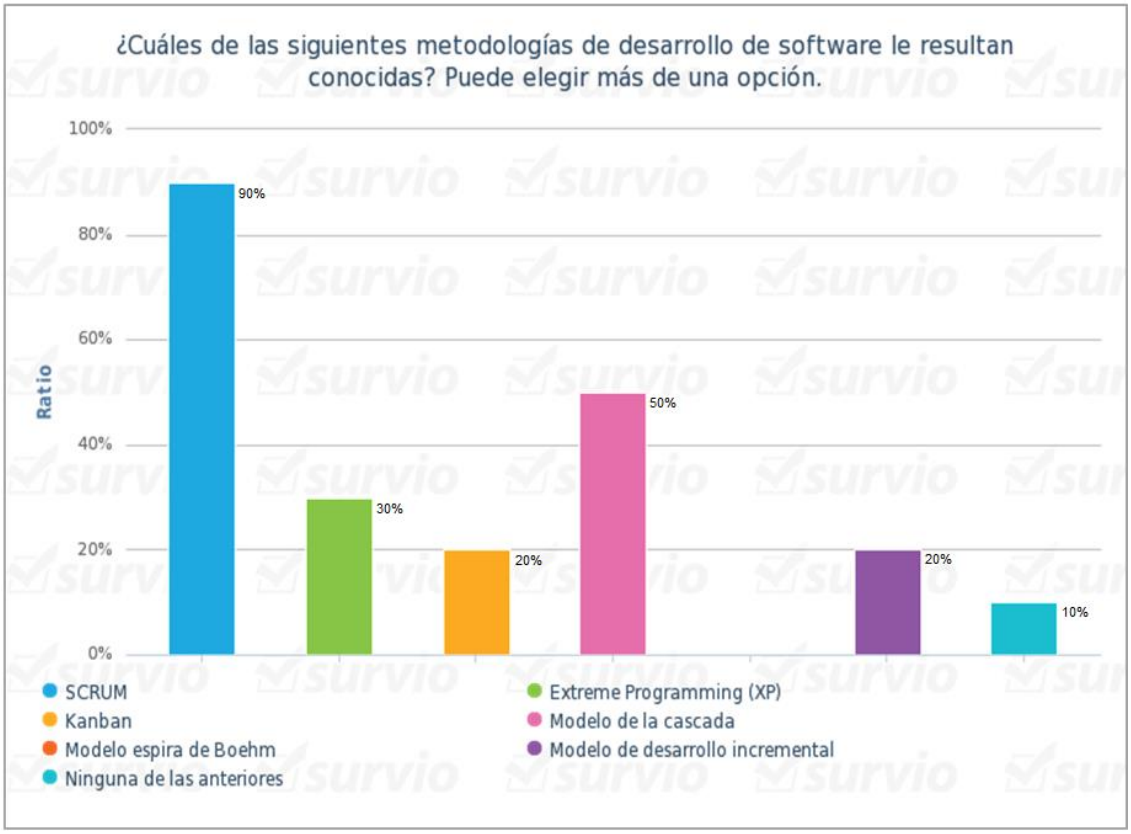


Figura 23
Gráfico sobre metodologías más conocidas por el grupo de colaboradores del SIMA.
Fuente: elaboración propia.

En relación con el conocimiento de los distintos tipos de metodologías de desarrollo de *software*, se puede observar en la figura 23, que un 90% de los encuestados tiene algún conocimiento sobre SCRUM, un 50% sobre el modelo de cascada, mientras que el 30% indica conocer la Programación Extrema (XP). Esto demuestra que existe un conocimiento variado sobre ciertos tipos de metodologías.

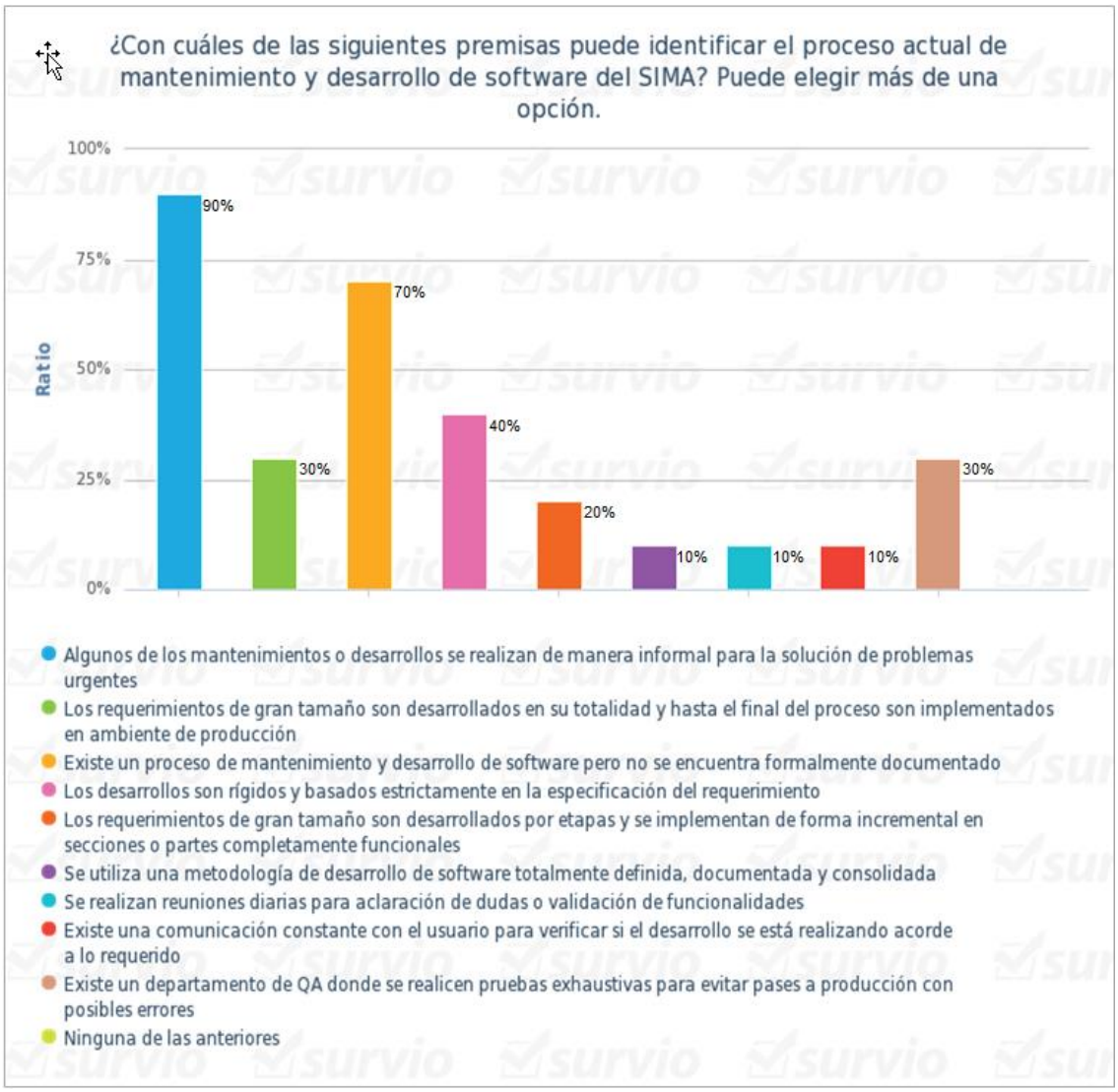


Figura 24
Gráfico sobre proceso actual de mantenimiento y desarrollo del SIMA.
Fuente: elaboración propia.

En la figura 24 se obtiene un panorama general de cómo se percibe por parte del grupo de mantenimiento y desarrollo del SIMA el modo actual de trabajar.

A continuación, se muestran las cinco características que según el grupo de encuestados identifican de manera más acertada el proceso actual de mantenimiento y desarrollo del SIMA:

1- Algunos de los mantenimientos o desarrollos se realizan de manera informal para la solución de problemas urgentes, (90% de los encuestados).

2- Existe un proceso de mantenimiento y desarrollo de *software* pero no se encuentra formalmente documentado, (70% de los encuestados).

3- Los desarrollos son rígidos y basados estrictamente en la especificación del requerimiento, (40% de los encuestados).

4- Los requerimientos de gran tamaño son desarrollados en su totalidad y hasta el final del proceso son implementados en ambiente de producción, (30% de los encuestados).

5- Existe un departamento de QA donde se realicen pruebas exhaustivas para evitar pases a producción con posibles errores, (30% de los encuestados).

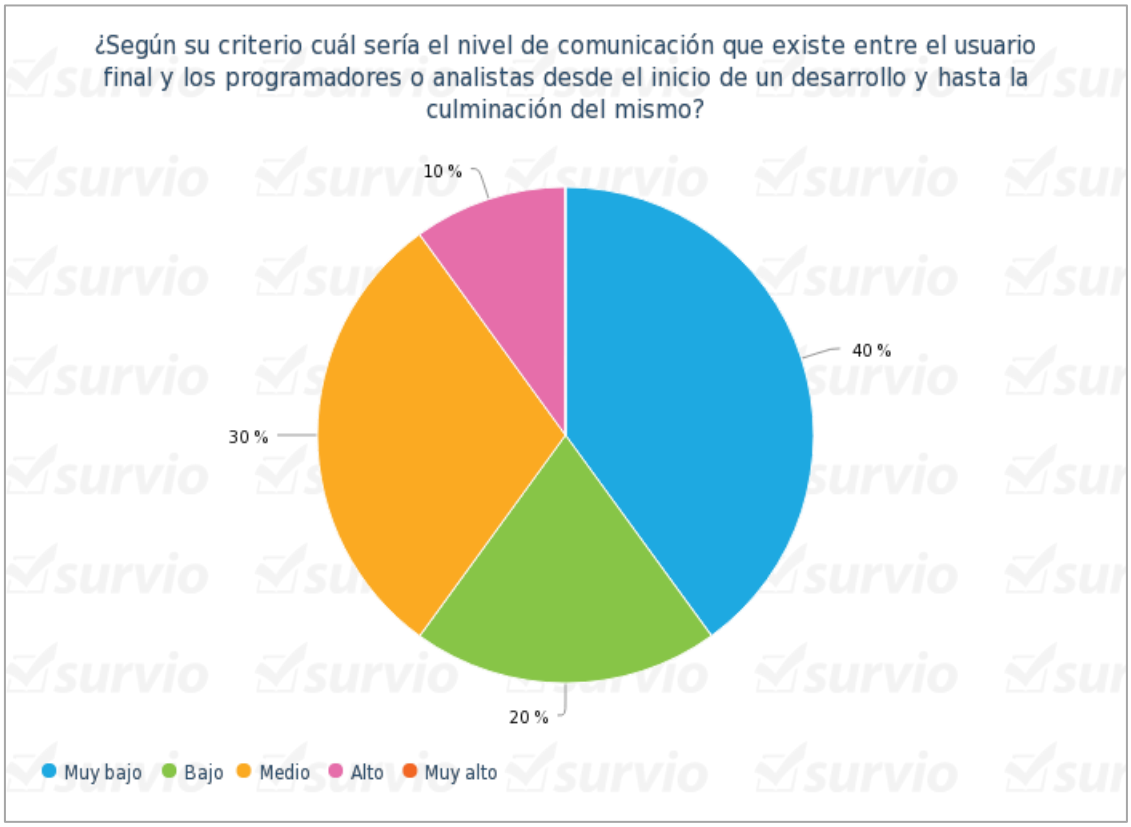


Figura 25
Gráfico sobre nivel de comunicación entre usuario final y programadores o analistas.
Fuente: elaboración propia.

De acuerdo con la figura 25, se puede observar que la comunicación existente entre el grupo de mantenimiento y desarrollo del SIMA con el usuario final tiende a tener un nivel bastante bajo, de hecho, los rubros “Muy bajo” y “Bajo” representan un 60% del total de los encuestados, siendo “Muy bajo” el de mayor valor con un 40%. De esta manera se deduce que el método utilizado actualmente tiende a dejar de lado la comunicación con el usuario final.

Figura 26
Gráfico sobre unión de grupo entre los colaboradores del SIMA.
Fuente: elaboración propia.

En relación con la unión de grupo se puede observar en la figura 26 que el 50% de los encuestados considera que el nivel de este tipo de iniciativas es de nivel “Medio”, mientras que el 20% opina que el nivel es “Bajo”. Esto indica que aún se puede mejorar el trabajo en equipo para mejorar la eficiencia en los procesos de mantenimiento y desarrollo del SIMA.

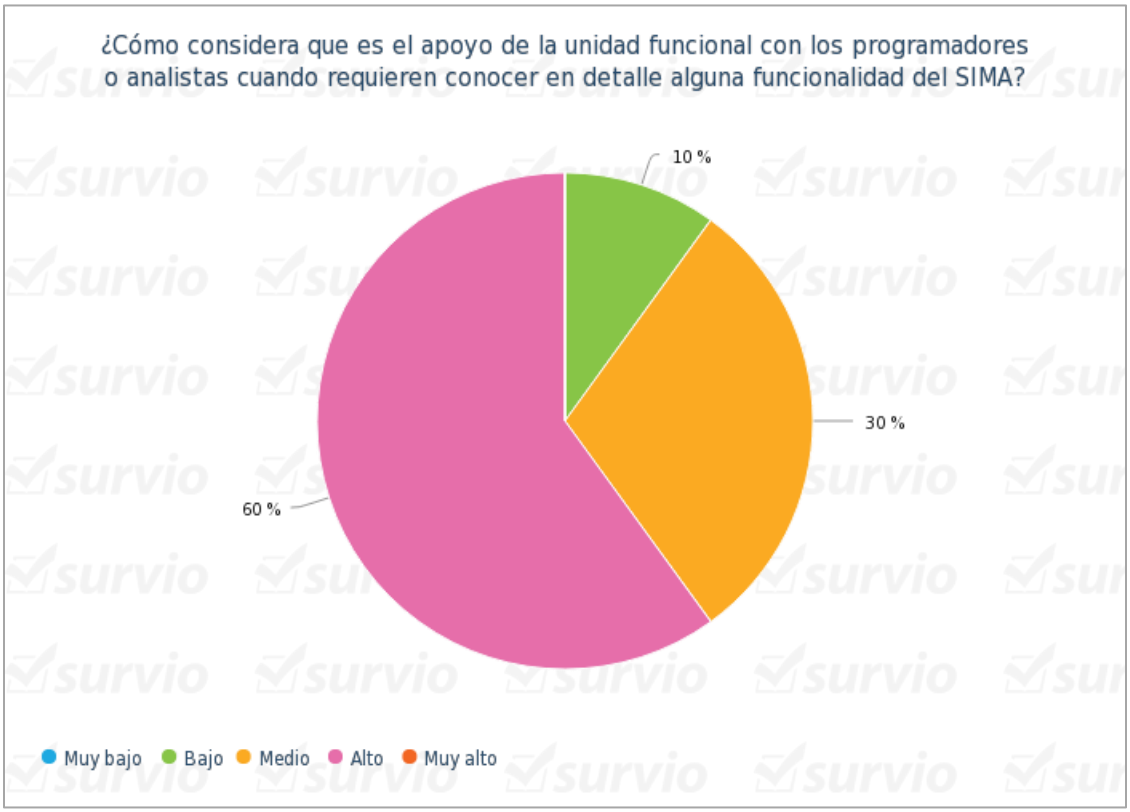


Figura 27
Gráfico sobre apoyo de la Unidad Funcional con los programadores y analistas.
Fuente: elaboración propia.

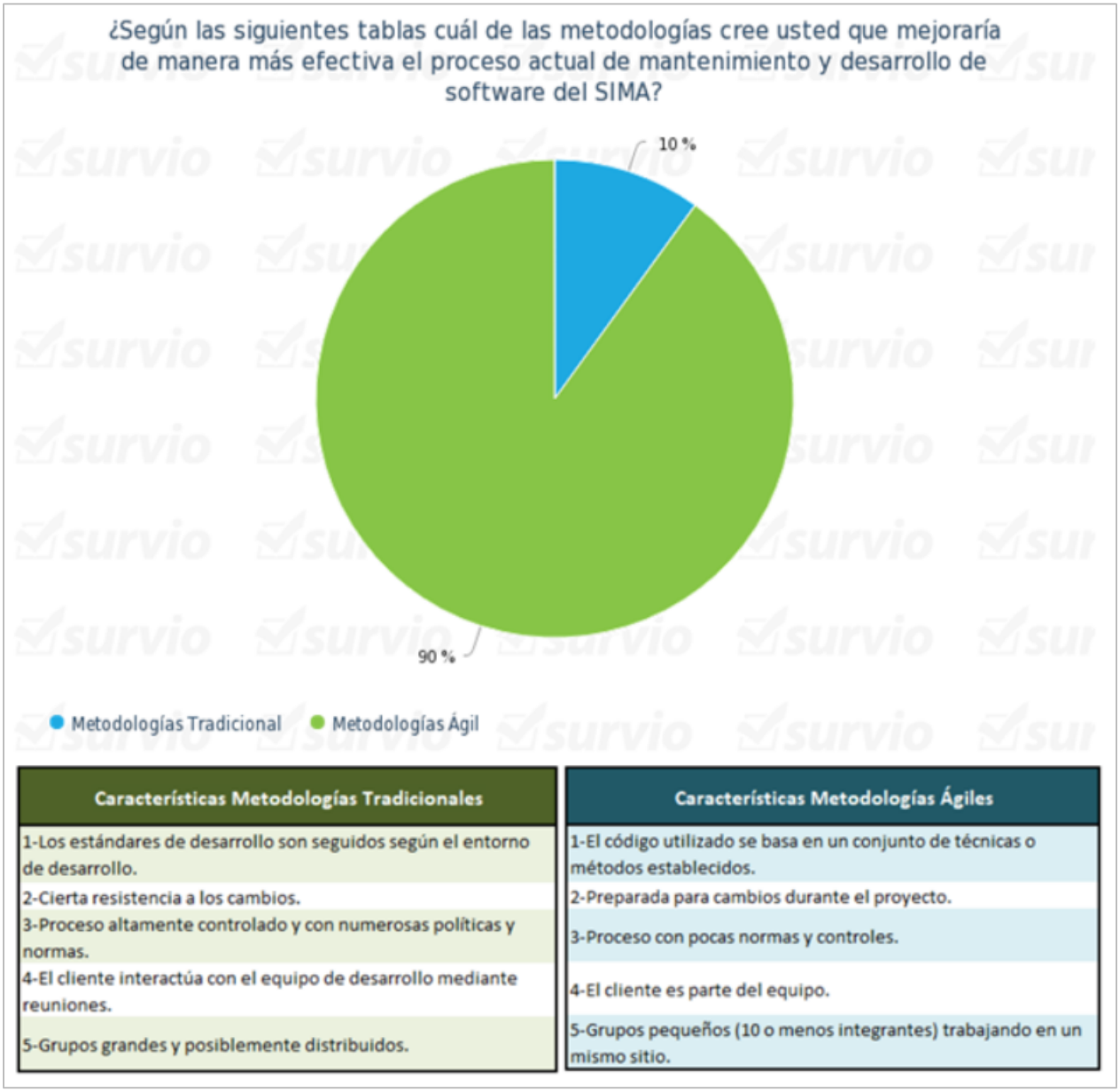
Según el gráfico representado en la figura 27, se puede observar que el apoyo que brinda Unidad Funcional al equipo de mantenimiento y desarrollo del SIMA, es bastante bueno, de hecho un 60% de los encuestados opina que el nivel de apoyo es “Alto”, mientras que un 30% cree que es de nivel “Medio”.

Figura 28
Gráfico que indica si las metodologías tiene un impacto positivo en SIMA.
Fuente: elaboración propia.

Según el gráfico representado en la figura 28, el 100% del grupo de mantenimiento y desarrollo del SIMA opina que el uso de una metodología de desarrollo de *software* puede tener un impacto positivo en la calidad y eficiencia de los productos desarrollados.

Figura 29
Gráfico de aceptación sobre el uso de metodologías de desarrollo en el SIMA.
Fuente: elaboración propia.

De acuerdo con el gráfico representado en la figura 29, se puede observar que el 100% del grupo de mantenimiento y desarrollo del SIMA está de acuerdo con adaptarse a una nueva metodología de desarrollo de *software*.



Características Metodologías Tradicionales	Características Metodologías Ágiles
1-Los estándares de desarrollo son seguidos según el entorno de desarrollo.	1-El código utilizado se basa en un conjunto de técnicas o métodos establecidos.
2-Cierta resistencia a los cambios.	2-Preparada para cambios durante el proyecto.
3-Proceso altamente controlado y con numerosas políticas y normas.	3-Proceso con pocas normas y controles.
4-El cliente interactúa con el equipo de desarrollo mediante reuniones.	4-El cliente es parte del equipo.
5-Grupos grandes y posiblemente distribuidos.	5-Grupos pequeños (10 o menos integrantes) trabajando en un mismo sitio.

Figura 30
Gráfico de tipo de metodología más aceptada por los colaboradores del SIMA.
Fuente: elaboración propia.

Como se puede observar en la figura 30, al grupo encuestado se le han brindado algunas de las características de las metodologías tradicionales y ágiles, basados en esto, el 90% de los encuestados opina que las metodologías ágiles son la mejor opción para mejorar el proceso de mantenimiento y desarrollo del SIMA.

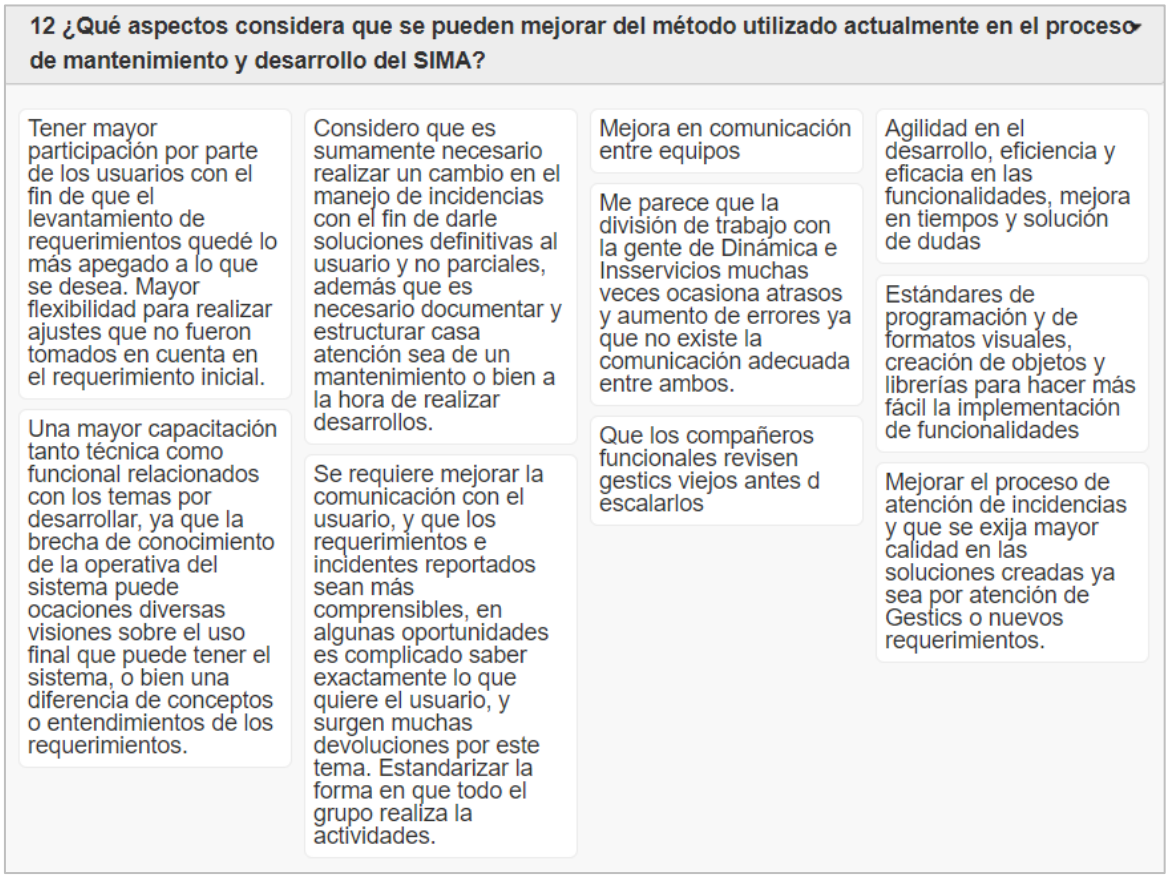


Figura 31.
Opiniones sobre aspectos me mejora en el proceso de mantenimiento y desarrollo del SIMA.
Fuente: elaboración propia.

Como se puede observar en la figura 31, los encuestados han brindado una serie de aspectos a mejorar en el proceso actual de mantenimiento y desarrollo de SIMA, los cuales se muestran a continuación:

1. Mayor participación por parte del usuario final en los procesos de mantenimiento y desarrollo del SIMA.
2. Mayor flexibilidad en el proceso de mantenimiento y desarrollo del SIMA, con el fin de que se puedan realizar los ajustes adicionales requeridos por el usuario.

3. Brindar capacitaciones técnicas y funcionales a los miembros del área de mantenimiento y desarrollo del SIMA.
4. Mejorar el manejo de incidencias con el fin de darle soluciones definitivas al usuario final.
5. Documentar detalladamente los cambios realizados tanto en incidencias como en nuevos requerimientos con el fin de que la solución esté disponible para todo el grupo.
6. Establecer un estándar donde se defina la forma que se realizan todas las actividades de mantenimiento y desarrollo del SIMA.
7. Mejorar la comunicación que existe en el equipo de mantenimiento y desarrollo del SIMA.
8. Que la Unidad Funcional realice mejores filtros antes enviar una incidencia o error al grupo de mantenimiento y desarrollo del SIMA.
9. Mejorar la eficiencia y eficacia en el proceso de mantenimiento y desarrollo del SIMA, esto con el fin de mejorar el servicio que se brinda al usuario final y su vez al cliente.
10. Mejorar la calidad en las soluciones creadas ya sea por atención de incidencias o por nuevos requerimientos.

4.3 DETERMINACIÓN DE BRECHAS

Mediante el uso de herramientas de recolección de datos tales como la entrevista, la encuesta y la observación, se ha logrado obtener información suficiente para definir la brecha que existe entre lo que se tiene actualmente y lo que se desea para mejorar el proceso de mantenimiento y desarrollo del SIMA.

En la siguiente tabla se muestran, de manera más simple, las brechas existentes en el proceso de mantenimiento y desarrollo del SIMA:

Tabla 6

Brechas en el proceso de mantenimiento y desarrollo del SIMA

Rubro	Escenario actual	Escenario deseado
Especificación de los requerimientos	Los requerimientos son realizados de forma rígida, es decir, solo se realiza lo que se ha especificado en el requerimiento.	Flexibilidad para poder agregar funcionalidades necesarias requeridas por el usuario final.
Estándar en el proceso de mantenimiento y desarrollo del SIMA.	Se sigue un patrón similar en toda el área, pero no está definido ni documentado.	Definir una metodología de desarrollo de software que quede debidamente documentada.
Aseguramiento de la calidad	Se cuenta con la Unidad Funcional que se encarga de realizar pruebas y velar por la calidad del producto, sin embargo, muchos mantenimientos y desarrollos son devueltos por mala calidad.	Mejorar la calidad en el producto realizado por los desarrolladores, así como mayor exigencia en las pruebas de calidad realizadas por la Unidad Funcional.
Comunicación con el usuario final	La comunicación se presenta únicamente con los analistas al elaborar la especificación del requerimiento, los desarrolladores no participan en reuniones con el usuario final, y el usuario no se involucra en el proceso de mantenimiento o desarrollo.	Mayor participación por parte del usuario final en los requerimientos.
Trabajo en equipo	Los requerimientos de gran tamaño son atendidos por una única persona quedando todo el conocimiento en la misma.	Se espera que, sobre todo, los requerimientos de gran tamaño puedan ser atendidos por más de una persona, esto para que exista el apoyo mutuo y que el conocimiento adquirido sea compartido; Esto es muy útil en casos de enfermedad o renuncia de alguno de los miembros del equipo.
Implementación en ambiente de producción	Las soluciones son puestas en producción al final de todo el desarrollo o mantenimiento.	Que las puestas en producción de un requerimiento sean puestas en interacciones completamente funcionales (entregables parciales).
Cantidad de incidentes	La cantidad promedio de incidentes reportados por mes ronda los 200, una cantidad bastante elevada.	Con el uso de una metodología de desarrollo de software se espera ir bajando la cantidad de incidencias poco a poco.

Fuente: elaboración propia.

CAPÍTULO V

DISEÑO Y DESARROLLO DEL PROYECTO

5.1 ELECCIÓN DE LA METODOLOGÍA

Para seleccionar la metodología, se ha llegado a un acuerdo con todo el grupo de mantenimiento y desarrollo del SIMA para proponer una metodología ágil, ya que las tradicionales son muy similares a lo que se tiene actualmente.

Por decisión unánime del grupo, se han seleccionado para el estudio únicamente Scrum y eXtreme Programming y, para conocer cuál de las dos metodologías se adapta de mejor manera a lo esperado por el área de mantenimiento y desarrollo del SIMA, se ha realizado una comparación de las características de ambas metodologías.

A continuación, se muestra la tabla 7, con la comparación mencionada anteriormente:

Tabla 7

Características entre metodologías Scrum y XP

Rubro	Scrum	eXtreme Programming (XP)	Selección
Aumento de la productividad	Alto	Alto	Ambas
Documentación	Simple	Simple	Ambas
Comunicación con el cliente	Alta	Muy alta	Xp
Calidad del producto	Alto	Muy alto	Xp
Flexibilidad sobre cambios en los requisitos	No siempre	Siempre	Xp
Código fuente	No se especifica	Sencillo y comprensible para todos	Xp
Pruebas unitarias	Poco frecuentes	Muy frecuentes	Xp
Iteraciones	De 1 a 4 semanas	De 2 a 3 semanas	Ambas
Compañerismo	Alto	Muy alto	Xp
Cantidad de roles	Bajo	Muy alto	Scrum
Satisfacción del usuario final	Alta	Alta	Ambas

Fuente: elaboración propia.

En la tabla número 7 se definió una columna llamada selección, donde se indica cuál de las dos metodologías se adapta más a lo que desea el área de mantenimiento y desarrollo del SIMA, los resultados demuestran que el 73% de la escogencia se inclina a favor de la metodología eXtreme Programming, sobre un 27% para la metodología Scrum, por lo tanto la metodología seleccionada es precisamente XP.

Adicionalmente, para reforzar la selección realizada se ha consultado un artículo de la Universidad Politécnica de Valencia, donde se indica que metodología eXtreme Programming tiende a ser un poco más ágil que Scrum.

Tabla 8

Ranking de agilidad

Características	Scrum	eXtreme Programming
Sistema como algo cambiante	5	5
Colaboración	5	5
Resultados	5	5
Simplicidad	5	5
Adaptabilidad	4	3
Excelencia técnica	3	4
Prácticas de colaboración	4	5
Media total	**4,43**	**4,57**

Fuente: disponible en: http://www.cyta.com.ar.

Tal y como se observa en la tabla 8, se presenta una leve diferencia en la agilidad de ambas metodologías, pero con mejor rendimiento para XP.

Después de todo este estudio, se ha convocado otra reunión con el grupo de mantenimiento y desarrollo de SIMA para dejar en firme la metodología propuesta, según minuta M-TI-007 del anexo II, se puede observar que se ha seleccionado la metodología eXtreme Programming ya que cumple de mejor manera los objetivos del área.

5.2 NECESIDADES DEL ÁREA DE MANTENIMIENTO Y DESARROLLO DEL SIMA

Se han identificado una serie de necesidades básicas en el proceso de mantenimiento y desarrollo del SIMA, las mismas se describen a continuación:

- Mejorar la flexibilidad en los requerimientos de *software*.
- Mejorar la calidad en los productos de *software* generados.
- Mayor exigencia en las pruebas realizadas por la unidad funcional.
- Mayor participación por parte del usuario final.
- Implementar la atención de requerimientos en parejas.
- Incorporar el uso de puestas en producción frecuentes.
- Bajar la gran cantidad de incidencias reportadas diariamente.
- Definir un estándar de desarrollo que quede debidamente documentado.

Gracias a los valores, herramientas, el ciclo de vida y las 12 prácticas que posee la metodología eXtreme Programming, todas las necesidades mencionadas anteriormente pueden ser afrontadas de muy buena manera.

Seguidamente, se muestra una tabla donde se indica la forma de abordar cada una de las necesidades:

Tabla 9

Necesidades soportadas por la metodología XP

Necesidad SIMA	eXtreme Programming (XP)
Mejorar la flexibilidad en los requerimientos de *software*	Para tener requerimientos más flexibles se hace uso de la práctica "Planificación incremental", que indica que la planificación estática no es congruente con la agilidad y que más bien las necesidades del negocio pueden cambiar drásticamente mientras el proyecto se encuentra en la etapa de desarrollo.
Mejorar la calidad en los productos de *software* generados	En XP, la calidad tiene una importancia muy elevada por lo tanto, 3 de las 12 prácticas tienen relación directa con la calidad del producto generado: diseño simple indica que se debe generar código simple y funcional; refactorización, modificar código para dejarlo en buen estado y estándares de codificación, estandarizar el modo en que se escribe el código.
Mayor exigencia en las pruebas realizadas por la unidad funcional	Existe la práctica "Test", que para asegurar el correcto funcionamiento del producto propone 3 tipos de pruebas distintas: test unitario (realizado por el desarrollador), test de integridad (realizado por todo el grupo para asegurarse que todo el sistema funciona correctamente) y test de aceptación (en conjunto con el usuario final para asegurarse que cumple con la funcionalidad solicitada).
Mayor participación por parte del usuario final	La práctica "Cliente *in situ*" establece que es necesario que le usuario final esté dispuesto a participar activamente del proyecto.
Implementar la atención de requerimientos en parejas	La práctica "Programación en parejas" sugiere que es mucho más eficiente y eficaz que la programación sea realizada en parejas.
Incorporar el uso de puestas frecuentes en ambiente de producción	La práctica "Entregas frecuentes" indica que se deben desarrollar lo antes posible versiones pequeñas del sistema, que aunque no tengan toda la funcionalidad puedan aportar una idea general de cómo ha de ser la entrega final y que sirvan para que el usuario final se vaya familiarizando con el entorno.
Bajar la gran cantidad de incidencias reportadas diariamente	Gracias a la refactorización que pretende modificar todo código en mal estado para optimizarlo, se pueden mejorar la gran cantidad de aplicativos que no funcionan correctamente.
Definir un estándar de desarrollo que quede debidamente documentado	El hecho de utilizar la propia metodología XP ya define un estándar del modo en que se realizan todos los procesos de mantenimiento y desarrollo del SIMA.

Fuente: elaboración propia.

5.3 DEFINICIÓN DE ROLES XP

La metodología XP propone una serie de roles importantes que le permiten ser implementada en cualquier compañía, sin embargo, para esta propuesta se pretende que el rol de Gestor no sea un enlace con el usuario final, sino más bien se desea aprovechar la práctica de cliente *in situ* para que el grupo tenga una comunicación más directa con los usuarios, sin necesidad de intermediarios. Para el área de mantenimiento y desarrollo del SIMA los roles son definidos de la siguiente forma:

Tabla 10

Roles de la metodología XP para SIMA

Rol	Cantidad funcionarios	Descripción de rol	Funcionario
Gestor	1	Máximo responsable del proyecto, se encarga de coordinar y de garantizar las condiciones necesarias para llevar a cabo el proyecto.	Líder de producto DSI-MSI ORACLE.
Entrenador	1	Encargado del proceso global, garantiza que se sigue la filosofía XP y ayuda al equipo a la hora de aplicar las prácticas básicas de XP.	Analista primer nivel encargado del proyecto.
Cliente	n	Escribe historias de usuario, prioriza las historias de usuario, disponible para consultas del grupo de desarrollo y realiza pruebas de aceptación.	Usuario final del proyecto pueden ser médicos, farmacéuticos, administrativos, entre otros.
Encargado de seguimiento	3	Recoge, analiza y publica información sobre la marcha del proyecto, supervisa el cumplimiento de las estimaciones en cada iteración.	Analista segundo nivel.
Programador	5	Produce el código fuente del *software* y realiza pruebas unitarias.	Desarrolladores del área.
Encargado de pruebas	4	Realiza pruebas de aceptación en conjunto con el cliente, ejecuta pruebas de integración.	Unidad funcional del SIMA.
Consultor	n	Es externo no forma parte del equipo pero puede apoyar al equipo en situaciones puntuales.	Cualquier funcionario del INS que pueda apoyar en alguna situación, ejemplo: Área de arquitectura.

Fuente: elaboración propia.

En la tabla anterior, debido a que los usuarios finales y los posibles consultores varían dependiendo del requerimiento, entonces la cantidad de recursos es “*n*”.

5.4 ADAPTACIÓN DE LA METODOLOGÍA XP

En este apartado se muestra de qué forma se puede adaptar la metodología XP en el proceso de mantenimiento y desarrollo del SIMA, basada en los valores, prácticas y el ciclo de vida de la propia metodología.

Es importante destacar que el mantenimiento de software es en el fondo un proceso desarrollo de menor tamaño para la corrección de errores o mejora de las capacidades de un sistema informático, por lo tanto, los valores y prácticas de la metodología aplican del mismo modo para ambos procesos.

5.4.2 Valores de XP

Comunicación: se debe fomentar la comunicación entre los miembros del grupo y con los usuarios finales; con el fin de agilizar el proceso, es indispensable contar con el apoyo del grupo cuando alguno de los miembros se encuentre ante una situación que no pueda resolver de manera individual.

Simplicidad: crear la cultura en todo el grupo de mantenimiento y desarrollo del SIMA para que el código confeccionado sea simple y funcional, logrando así aplicaciones más sencillas de mantener.

Retroalimentación: valor ligado a la comunicación, donde el usuario final expresa las necesidades no satisfechas, la idea es crear confianza entre el grupo de mantenimiento y desarrollo del SIMA con los usuarios finales, para que ambas partes puedan tener una estrecha comunicación y lograr acuerdos que aporten beneficios.

Coraje: incentivar a los desarrolladores para que manipulen continuamente todo aquel código que pueda ser mejorado y simplificado, la intención es que todas las aplicaciones que componen el SIMA contengan la esencia de la metodología XP.

5.4.3 Prácticas de XP

1-Diseño simple: práctica que debe ser asumida por todos los desarrolladores del área con la intención de que el código creado sea lo más sencillo posible y consiga que todo funcione.

2-Refactorización: práctica que debe ser realizada por los desarrolladores, pero además impulsada por el gestor y el entrenador, la intención es crear la cultura en el grupo de mantenimiento y desarrollo del SIMA para que cualquiera de los miembros del grupo, al identificar código fuente que pueda ser mejorado, tenga el coraje de realizar el cambio utilizando la práctica de diseño simple y procurando que la funcionalidad del aplicativo se mantenga o se optimice.

Como se puede observar, el diseño simple y la refactorización están estrechamente relacionadas, ya que toda modificación realizada debe ser lo más sencilla posible, con estas dos prácticas se logra que las aplicaciones funcionen cada vez mejor y que el código que las compone sea fácil de mantener.

Un ejemplo sencillo para demostrar la refactorización y el diseño simple es mediante el siguiente código:

Diseño original

```
declare
 v_fecha date := sysdate;
 v_descripcion_mes varchar2(100) := null;
begin
 if to_char(v_fecha, 'mm') = '01' then
  v_descripcion_mes := 'Enero';
 elsif to_char(v_fecha, 'mm') = '02' then
  v_descripcion_mes := 'Febrero';
 elsif to_char(v_fecha, 'mm') = '03' then
  v_descripcion_mes := 'Marzo';
 elsif to_char(v_fecha, 'mm') = '04' then
  v_descripcion_mes := 'Abril';
 elsif to_char(v_fecha, 'mm') = '05' then
  v_descripcion_mes := 'Mayo';
 elsif to_char(v_fecha, 'mm') = '06' then
  v_descripcion_mes := 'Junio';
 elsif to_char(v_fecha, 'mm') = '07' then
  v_descripcion_mes := 'Julio';
 elsif to_char(v_fecha, 'mm') = '08' then
  v_descripcion_mes := 'Agosto';
 elsif to_char(v_fecha, 'mm') = '09' then
  v_descripcion_mes := 'Septiembre';
 elsif to_char(v_fecha, 'mm') = '10' then
  v_descripcion_mes := 'Octubre';
 elsif to_char(v_fecha, 'mm') = '11' then
  v_descripcion_mes := 'Noviembre';
 elsif to_char(v_fecha, 'mm') = '12' then
  v_descripcion_mes := 'Diciembre';
 end if;
 dbms_output.put_line('Descripción: '||v_descripcion_mes);
end;
```

Diseño modificado

```
declare
 v_fecha        date := sysdate;
 v_descripcion_mes varchar2(100) := null;
begin
 case to_char(v_fecha, 'mm')
  when '01' then v_descripcion_mes := 'Enero';
  when '02' then v_descripcion_mes := 'Febrero';
  when '03' then v_descripcion_mes := 'Marzo';
  when '04' then v_descripcion_mes := 'Abril';
  when '05' then v_descripcion_mes := 'Mayo';
  when '06' then v_descripcion_mes := 'Junio';
  when '07' then v_descripcion_mes := 'Julio';
  when '08' then v_descripcion_mes := 'Agosto';
  when '09' then v_descripcion_mes := 'Septiembre';
  when '10' then v_descripcion_mes := 'Octubre';
  when '11' then v_descripcion_mes := 'Noviembre';
  when '12' then v_descripcion_mes := 'Diciembre';
 end case;

 dbms_output.put_line('el mes actual es: '||v_descripcion_mes);
end;
```

Figura 32
Ejemplo refactorización y diseño simple.
Fuente: elaboración propia.

Como se puede observar en la figura 32, se ha realizado una modificación (refactorización) para mejorar el código original que utilizaba la sentencia *IF* reiteradas

veces y en lugar de esto, hacer uso de la cláusula *CASE* que es mucho más eficiente y además, como se puede visualizar, el diseño es más sencillo.

3-Test: se refiere a las pruebas realizadas para comprobar la funcionalidad adecuada de la aplicación, esta práctica se divide en tres tipos:

- Test unitario: prueba realizada específicamente por el programador para ir probando todos los cambios realizados y así asegurar su adecuado funcionamiento.

- Test de integridad: pruebas realizadas con todo el grupo para asegurarse de que todo funciona correctamente, este tipo de pruebas deben ser realizadas sobre todo en requerimientos que tengan afectación global en todo el sistema o en una parte del mismo, por ejemplo, cuando se modifica el tipo de dato sobre un campo de una tabla de base de datos.

- Pruebas de aceptación: el usuario final, con ayuda del encargado de pruebas (Unidad Funcional en SIMA), realiza las pruebas de aceptación para cada una de las historias de usuario definidas al inicio de cada iteración. Estas pruebas son utilizadas para validar que cada requerimiento a implementar funciona tal y como se había especificado.

Para la documentación formal de las pruebas de aceptación se debe utilizar la siguiente plantilla:

Prueba de aceptación	
Código	1
Nombre de historia:	99
Nombre de historia:	Registro de citas
Descripción: Permite ingresar los datos de la cita y almacenarlos en la base de datos.	
Condiciones de ejecución: Este proceso será ejecutado siempre que exista la necesidad de darle una cita a un paciente.	
Entrada/Pasos de ejecución **Parámetros de entrada:** Identificación del paciente -Identificación del paciente -Código de especialidad que refiere -Código de médico que refiere -Código de especialidad de la cita -Fecha propuesta -Indicador de urgencia **Pasos de ejecución:** 1-Ingresar al módulo de enfermería. Ruta: Módulos Médicos/Identificación/Casos/Actualiza Casos. 2-Consultar un paciente y presionar el botón "Registrar cita". 3-Ingresar los datos de entrada. 4-Guardar la información.	
Resultado esperado: Mensaje que indique que los datos fueron ingresados exitosamente.	
Evaluación de la prueba: Ejecutadas correctamente.	

Figura 33
Plantilla de prueba de aceptación.
Fuente: elaboración propia.

Como se puede ver en la figura 33, cada prueba de aceptación está relacionada a una historia de usuario, por lo que las pruebas realizadas son exclusivas para esa funcionalidad, para el caso ejemplificado únicamente se debe validar que la cita sea registrada correctamente.

4-Estándares de codificación: XP promueve la programación basada en estándares, de manera que sea fácilmente comprensible por todo el equipo de desarrollo y así facilitar la refactorización.

Es importante resaltar que programar empleando estándares es una buena práctica que debe ser seguida en cualquier metodología de desarrollo, ya que pretende facilitar la propiedad colectiva de código.

Código original	Código estandarizado
declare v_nombre varchar2(100); cedula varchar2(15); v1 date; ed number; begin v_nombre := null; ... ... end;	declare v_nombre varchar2(100); v_cedula varchar2(15); v_fecha_nacimiento date; v_edad number; begin v_nombre := null; ... ... end;

Figura 34
Ejemplo sobre estándares de codificación.
Fuente: elaboración propia.

Algo tan simple como la definición de una variable debe ser estandarizado, ya que esto permite al desarrollador identificar fácilmente cuando se hace uso de las mismas; como se puede observar en la figura 34, en el código original se declaran variables de cualquier forma y sin nombres significativos, para mejorar eso se utiliza un estándar muy simple que consiste en agregar el prefijo “v_” y luego un nombre significativo que represente el valor almacenado en la variable.

5-Propiedad colectiva de código: estrategias tales como: la rotación de personal, la utilización de estándares de codificación y la programación en parejas son destinadas para lograr la propiedad colectiva de código.

Se debe procurar dar rotación a los programadores para que puedan trabajar en distintos proyectos, la intención es que cualquier desarrollador tenga la posibilidad de continuar la codificación que alguien más empezó sin tener muchas dificultades.

6-Programación en parejas: XP propone que se desarrolle en parejas, ambos trabajando en un mismo ordenador.

- El código es permanentemente revisado por dos personas, por lo que se pueden identificar errores con mayor facilidad.
- Se codifica de manera conjunta, haciéndolo lo más simple posible.
- En caso de que se presente algún problema, se resuelve de forma más rápida.
- El requerimiento finaliza con más personas que conocen los detalles de cada parte del código.

Sin embargo, para el área de mantenimiento y desarrollo del SIMA se implementará esta práctica únicamente para requerimientos grandes o complejos, además, con la variación de que algunas actividades pueden ser divididas y creadas en conjunto, pero en distintas computadoras.

7-Integración continua: la integración continua es una práctica realizada exclusivamente por los desarrolladores, la idea es que los programadores mantengan los objetos y aplicaciones del SIMA actualizados para evitar que otros miembros del grupo tengan problemas por la existencia de componentes descompilados.

8-Cuarenta horas semanales: la metodología XP promueve trabajar 40 horas semanales, lo importante es planificar el trabajo de manera tal que se lleve un ritmo constante y razonable sin sobrecarga del trabajo al equipo.

9-Metáfora del negocio: gracias a la comunicación entre el grupo técnico y funcional de XP con el usuario final, es posible establecer la metáfora del negocio, la cual consiste en eliminar tecnicismos informáticos o funcionales para que la definición de las necesidades del sistema sea comprensible por cualquier persona independientemente de su especialidad.

10-Cliente *in situ*: uno de los requerimientos de la metodología XP es tener al usuario final disponible durante todo el proyecto, no solamente como apoyo a los desarrolladores, sino formando parte del grupo.

- Los desarrolladores requieren especificaciones y detalles que solo el usuario final puede proporcionar.
- No se requieren largos documentos de especificaciones, sino más bien solo detalles.

- El cliente puede ayudar a los desarrolladores a prevenir a tiempo situaciones no deseables y corregirlas.

Es importante aclarar que gran parte de los usuarios del SIMA son especialistas del área de salud tales como: médicos, farmacéuticos, nutricionistas, terapeutas, entre otros y está claro que no es posible que permanezcan todo el día con los desarrolladores, sin embargo, tienen toda la disponibilidad para realizar reuniones y evacuar todas las dudas que vayan surgiendo en el proceso.

11-Entregas frecuentes: las entregas deben ser lo más pequeñas posibles, conteniendo siempre los requerimientos del negocio más importantes para el usuario en ese momento dado. De esta manera, el cliente va obteniendo funcionalidades del sistema en forma gradual hasta finalizar el proyecto.

En cada entrega los programadores obtienen retroalimentación del usuario final, determinando si lo implementado es lo que en realidad necesita. Este proceso debe ser coordinado entre los programadores y el encargado de seguimiento (analista de segundo nivel en SIMA).

12-Planeación incremental: consiste en la comunicación entre el usuario final y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y el usuario final decide sobre el ámbito y tiempo de las entregas y de cada iteración. Esta práctica se puede ilustrar como un

juego, donde existen dos tipos de jugadores: usuario final y programador. El usuario establece la prioridad de cada historia de usuario, de acuerdo con el valor que aporta para el negocio. Los desarrolladores o programadores estiman el esfuerzo asociado a cada historia de usuario.

5.4.4 Ciclo de vida de XP aplicado al SIMA

La metodología XP nace realmente cuando las 12 prácticas fueron unificadas en un ciclo de vida que consta de cinco fases esenciales, por lo tanto, a continuación, se explicará la aplicación de cada una de estas fases en área de mantenimiento y desarrollo del SIMA.

5.4.4.1 Fase de exploración

Es la primera fase del ciclo de vida de la metodología XP y consta de tres procesos:

5.4.4.1.1 Las historias de usuario

Es la especificación de funcionalidades deseadas para el sistema, que son escritas por los mismos usuarios y no por el analista, aunque éste debe brindar todo el apoyo necesario al usuario final, las descripciones deben ser cortas y escritas en el lenguaje del usuario evitando utilizar terminología técnica.

A nivel documental, una historia de usuario será como una ficha, a la que se le asigna una numeración (sería como el ID de la ficha), un nombre, un usuario, una prioridad, una estimación de tiempo de desarrollo, el número de iteración que se le asigna, una descripción y unas observaciones. A continuación, se muestra un ejemplo de cómo sería una historia de usuario.

Historia de usuario	
Número:	99
Nombre de historia:	Registro de citas
Usuario:	Administrativo/Médico
Prioridad en negocio:	**Alta**
Programador responsable:	Equipo de XP
Iteración:	1
Tiempo de desarrollo:	32 horas
Descripción	
Se mostrará por pantalla una interfaz que permitirá a los usuarios con perfil de administrador o médico registrar citas médicas.	
Observaciones	
El médico de la cita es asignado automáticamente por el usuario según la disponibilidad de horarios. CONFIRMADO con el usuario final.	

Figura 35
Ejemplo historia de usuario.
Fuente: elaboración propia.

Como se puede apreciar en la figura 35, existen diversos aspectos que complementan los técnicos del proyecto, tales como: el número de iteraciones, tiempo

estimado de desarrollo y algunos otros aspectos técnicos, pero la mayor parte de la ficha es responsabilidad del usuario final.

5.4.4.1.2 El *spike* arquitectónico

Los *spikes* arquitectónicos suelen ser de dos tipos: técnicos o funcionales. Los funcionales se utilizan para analizar funcionalidad, su riesgo y complejidad. Los técnicos se utilizan para determinar la viabilidad.

Cuando en SIMA se requiere un cambio de alto impacto como por ejemplo el cambio de alguna estructura principal de base de datos que afecte otros objetos o aplicaciones, es indispensable realizar el *spike* para analizar qué tan viable es el cambio y el impacto que pueda tener en el funcionamiento general del sistema.

5.4.4.1.3 La metáfora del negocio

Se trata de plasmar la estructura, diseño y funcionamiento del sistema en un lenguaje con el cual se le dé al grupo de desarrollo y los usuarios una misma visión sobre el proyecto, además de brindarles un primer vistazo muy completo a posibles nuevos integrantes del grupo para hacer su adaptación más rápida.

Es muy importante dentro del desarrollo de la metáfora de negocio, darle nombres adecuados a todos los elementos del sistema constantemente y que estos

correspondan a un sistema de nombres consistente. Esto será de mucha utilidad en fases posteriores del desarrollo, para identificar aspectos importantes del sistema.

Algunos aspectos a destacar sobre la metáfora del negocio son los siguientes:

- La metáfora ayuda a entender los elementos básicos y sus relaciones mediante ejemplos, es posible completar la definición funcional sin entrar en compleja documentación.

- Proporciona una visión de la arquitectura, sin entrar en el detalle.

- Debe servir para que el usuario se sienta a gusto refiriéndose al sistema con los términos que conoce.

- Establece un mecanismo sencillo de elaborar el producto y comunicar los objetivos.

En la figura 36 se muestra la interacción que existe entre las distintas actividades que componen la fase de exploración y los roles involucrados en esta etapa.

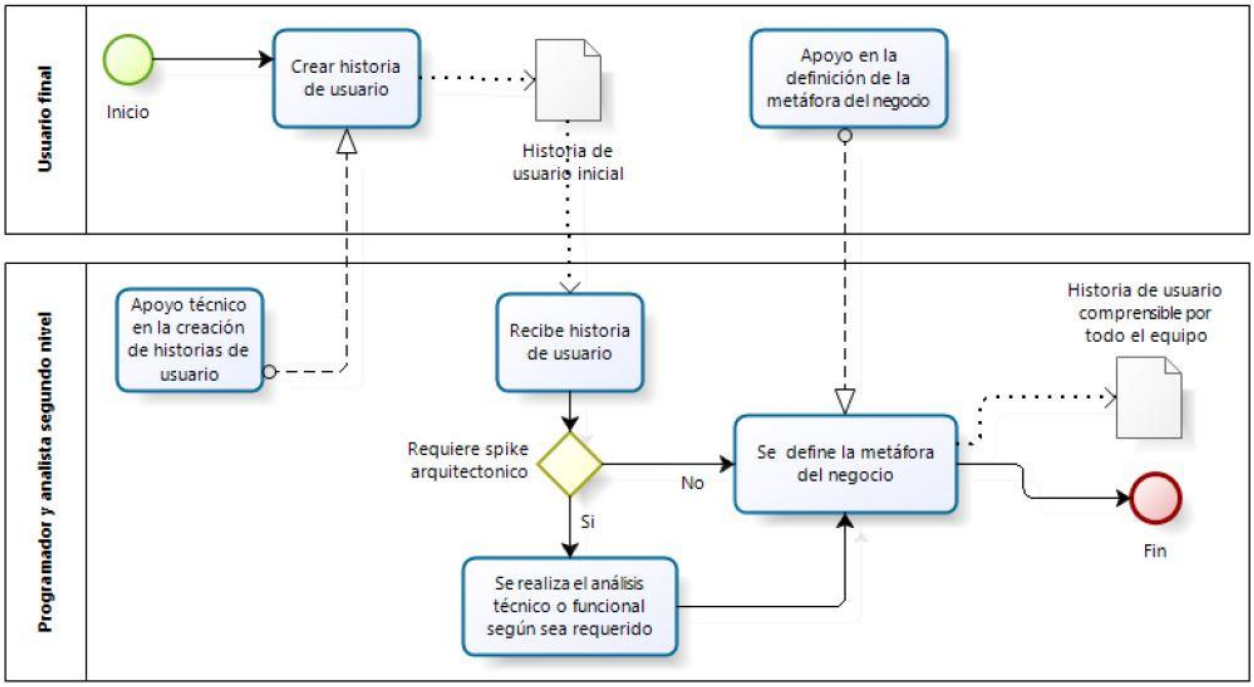


Figura 36
Interacción de los roles en la fase de exploración.
Fuente: elaboración propia.

5.4.4.2 Fase de planificación

Esta fase se compone de un procedimiento ya establecido el cual se explica a continuación:

1- Se debe realizar una reunión con todos los integrantes involucrados en el proyecto para analizar cada una de las historias de usuario escritas anteriormente y establecer la prioridad.

- El usuario final establece la prioridad de cada historia de usuario de acuerdo con el valor que aporta para el requerimiento.

- El equipo técnico puede realizar preguntas e incluso aportar ideas sobre la prioridad de las historias de usuario.
- La iteración se formará de acuerdo a como el cliente agrupe las historias.

2- El equipo técnico (analistas y desarrolladores) debe estudiar y analizar las historias de usuario ya priorizadas y estimar el coste de implementación.

- Si el equipo de desarrollo considera que la historia de usuario es muy compleja, entonces debe solicitar al usuario final que se descomponga en varias historias independientes más sencillas.

- Si el equipo técnico no tiene claro cómo implementar alguna historia de usuario, debe realizar un *spike* tecnológico que le ayude a identificar de qué modo se podría solucionar y así tener más claro el coste de implementación.

3- Una vez disponibles las historias de usuario priorizadas junto con el coste de implementación respectivo, se realiza una reunión para definir el plan de entregas.

- El plan de entregas se compone de una serie de iteraciones en las que se especifica cuáles historias de usuario se van a implementar en cada vuelta de la fase de iteraciones.

- En esta reunión deben participar tanto los usuarios finales como el equipo técnico y cada uno puede aportar su visión del negocio de manera que se obtengan más rápidamente las funcionalidades que mayor beneficio aportan al negocio.

- A cada iteración se le debe asignar un tiempo de implantación, intentando que todas sean los más parecidas posibles.

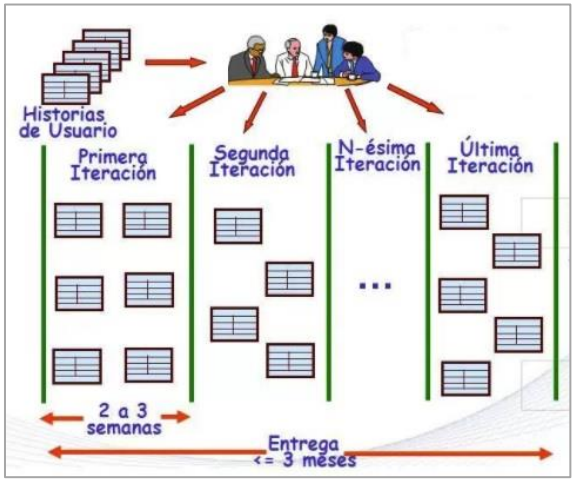


Figura 37
Ejemplo de plan de entregas o iteraciones.
Fuente: Disponible en: https://modulopoo.wordpress.com

Tal y como se observa en la figura 37, cada iteración está compuesta por una o varias historias de usuario que deben ser finalizadas en el tiempo establecido.

En la figura 38 se muestra la interacción que existe entre las distintas actividades que componen la fase de planificación y los roles involucrados en esta etapa.

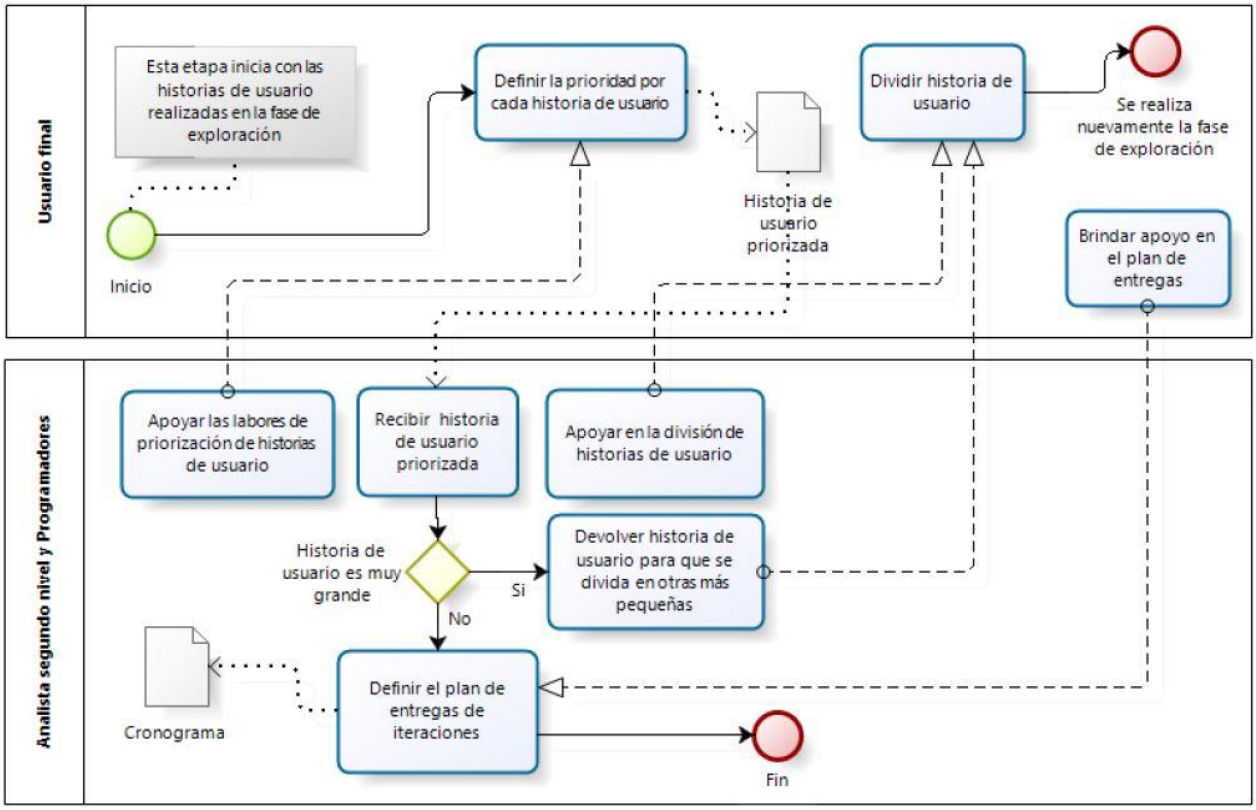


Figura 38
Interacción de los roles en la fase de planificación.
Fuente: elaboración propia.

5.4.4.3 Fase de iteraciones

Debido a que el proyecto debe estar dividido en iteraciones, esta fase se repetirá tantas veces como cantidad de iteraciones existentes. Generalmente, cada iteración suele ser de dos a tres semanas.

Al inicio de cada iteración, el equipo técnico realiza una reunión de planificación de la iteración, en donde la labor se centra específicamente en asignar tareas y distribuir el trabajo a desarrollarse en la iteración.

- El tiempo máximo de reunión es de dos horas por cada semana que dure la iteración.

- Cada historia de usuario se traduce en tareas específicas de programación.

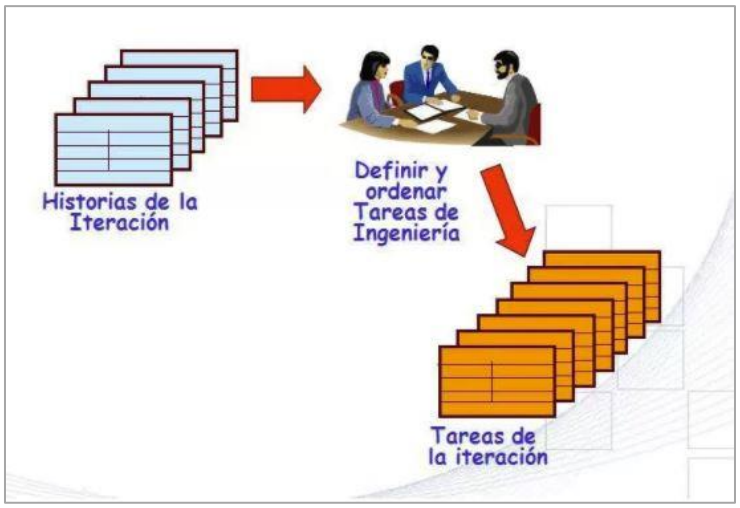


Figura 39
Ejemplo división de historias de usuario en tareas.
Fuente: disponible en: https://modulopoo.wordpress.com

Como se observa en la figura 39, las historias de usuario son descompuestas en tareas más simples, las cuales deben ser analizadas y desarrolladas por uno o dos programadores, según la complejidad del requerimiento, teniendo presente siempre que la codificación debe ser simple y estandarizada.

- Los programadores deben realizar las pruebas unitarias necesarias para asegurarse que las historias usuarias que componen la iteración cumplen con lo solicitado por el usuario sin ningún tipo de errores. Además, el desarrollador crea el documento de pruebas de aceptación.

- En esta etapa, la unidad funcional (encargado de pruebas) interviene para verificar que a nivel funcional se cumple con lo solicitado en las historias de usuario, además se realizan pruebas de integridad para asegurarse que la solución creada no afecte otras funcionalidades del módulo o sistema.

- Cuando todo está debidamente probado, el encargado de pruebas envía y coordina las pruebas de aceptación con el usuario final.

- Entre el encargado de pruebas y el usuario final realizan las pruebas de aceptación respectivas, según los pasos indicados en las mismas por cada historia de usuario; si todo funciona correctamente, el usuario da la aprobación y la iteración queda lista para ser implementada en producción. En caso de que alguna de las historias de usuario presente problemas o errores, el usuario lo debe indicar para que la corrección sea realizada por parte de los programadores.

Una estrategia muy importante es que por cada iteración se deben realizar primeramente las tareas que aportan más valor al negocio, de manera que, si por algún motivo se reduce el alcance del proyecto, solo sean afectadas las funcionalidades secundarias de la aplicación.

A continuación, en la figura 40 se muestra la interacción que existe entre las distintas actividades que componen la fase de iteraciones y los roles involucrados en esta etapa.

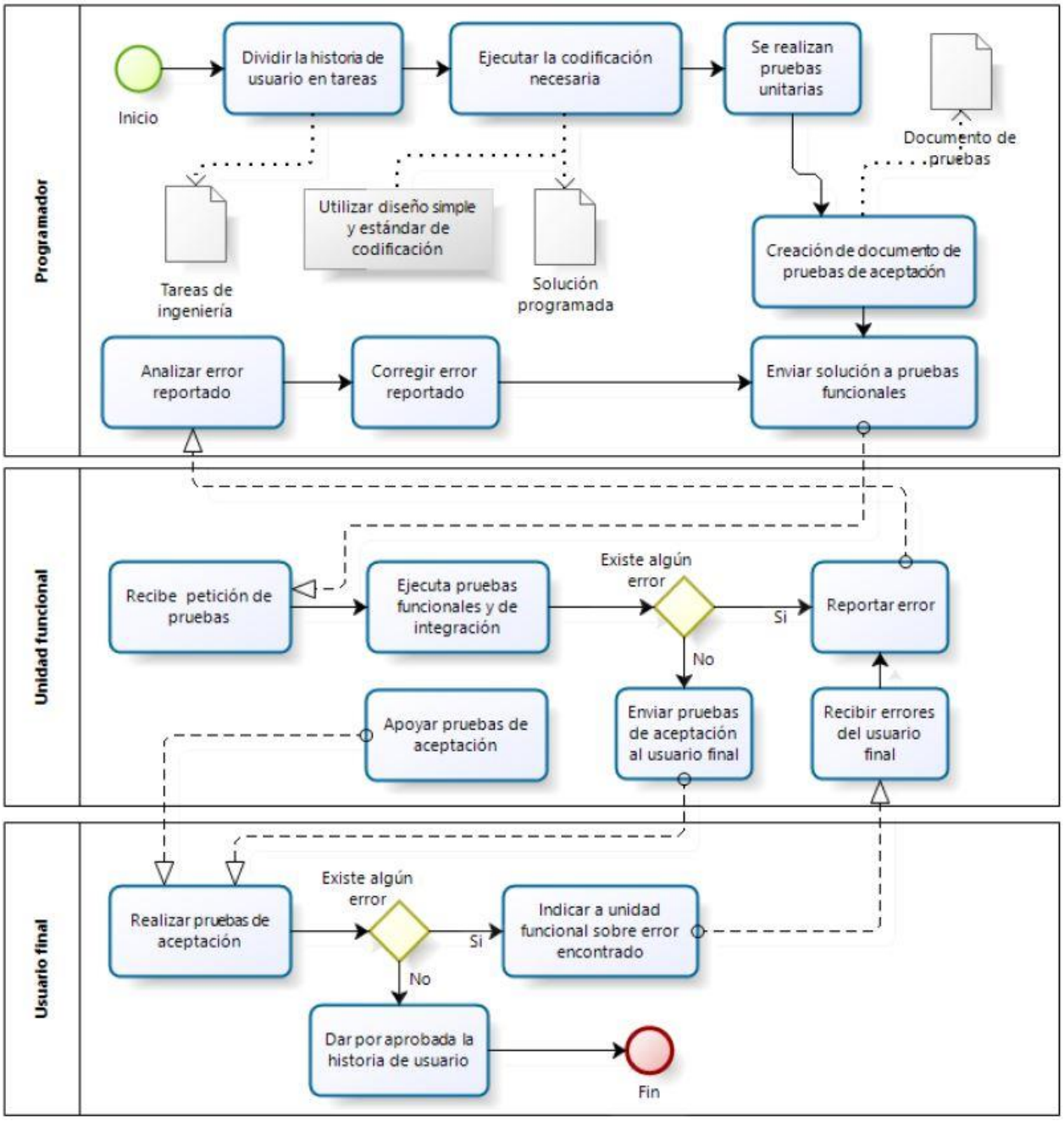


Figura 40
Interacción de los roles en la fase de iteraciones.
Fuente: elaboración propia.

5.4.4.4 Fase de producción

Según lo indicado por la metodología XP, esta fase es alcanzada desde la primera versión que alcance las funcionalidades mínimas que aporten un valor real al negocio y una operativa estable, sin embargo, para el SIMA la estrategia a utilizar será realizar los pases a producción por cada una de las iteraciones (cada dos o tres semanas), un avance muy significativo considerando que actualmente los requerimientos son puestos en ambiente de producción hasta el final de toda la etapa de desarrollo y en muchas ocasiones este proceso puede tardar entre seis y 12 meses.

En la figura 41 se muestra la interacción que existe entre las distintas actividades que componen la fase de producción y los roles involucrados en esta etapa.

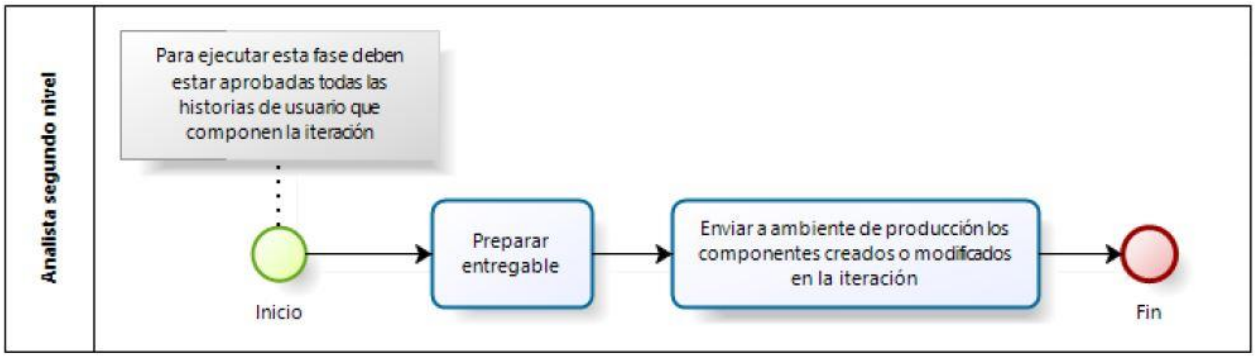


Figura 41
Interacción de los roles en la fase de producción.
Fuente: elaboración propia.

5.4.4.5 Fase de mantenimiento

La fase de mantenimiento es aquella en la cual se da soporte a la primera versión de la iteración, ya sea por eventuales errores, problemas o debido a que el usuario final tiene la necesidad de que el sistema cuente con nuevas funcionalidades.

En esta etapa los desarrolladores y todo el equipo comienzan una nueva entrega y se regresa nuevamente a la fase de planificación.

5.4.4.5 Análisis de resultados

En esta sección se pretende demostrar la forma en que fueron abordados los objetivos específicos del proyecto.

5.4.4.5.1 Objetivo específico 1

En el primer objetivo del proyecto se requería conocer la situación actual del proceso de mantenimiento y desarrollo del SIMA. Para alcanzar dicho objetivo, se realizó una entrevista al Licenciado Johnatan Solano Díaz, líder de producto DSI-MSI ORACLE, para así obtener información relevante de los procesos involucrados en el SIMA; adicionalmente, mediante la observación participante el investigador logró fortalecer el conocimiento adquirido en la entrevista para así tener un panorama aún más claro sobre el área de estudio.

5.4.4.5.2 Objetivo específico 2

El segundo objetivo se refiere al estudio de las metodologías de desarrollo de *software* con el fin de tener un criterio sólido para proponer mejoras a la situación presentada actualmente en el área de mantenimiento y desarrollo del SIMA. Para lograr el objetivo se realizó una revisión de la literatura para estudiar y comprender la aplicación de algunas de las metodologías de desarrollo de *software* más utilizadas en la actualidad.

5.4.4.5.3 Objetivo específico 3

El tercer objetivo está basado en la identificación de las necesidades del área de mantenimiento y desarrollo del SIMA. Para alcanzar el objetivo se realizó una entrevista al líder del producto y se aplicó una encuesta a todos los miembros del grupo para obtener dichas necesidades; asimismo, aprovechando el conocimiento adquirido sobre la situación actual, fue posible determinar las brechas existentes en los procesos de mantenimiento y desarrollo para así poder atacar las debilidades y obtener mejoras significativas.

5.4.4.5.4 Objetivo específico 4

El cuarto objetivo se enfoca en el diseño de una propuesta metodológica para los procesos de mantenimiento y desarrollo del SIMA y, gracias al estudio y comparación de las distintas metodologías, se logra comprender que todas las necesidades que tiene

el área pueden ser abarcadas perfectamente por la metodología de desarrollo eXtreme Programming, ya que esta cuenta con una serie de roles, valores, principios y herramientas que se adaptan de muy buena manera al proceso actual para mejorarlo significativamente y así lograr alcanzar el objetivo indicado. Teniendo en cuenta que dichos valores y principios

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES

- ✓ Acerca de la situación actual del área de mantenimiento y desarrollo del SIMA con respecto a las distintas actividades realizadas, se concluye que existen varias debilidades que requieren ser mejoradas con la utilización de alguna metodología o marco de trabajo, donde se establezca la forma en la cual se deben realizar las tareas de mantenimiento y desarrollo de *software*.

  Es importante destacar que el área actualmente no cuenta con ningún tipo de metodología para la gestión de mantenimientos o desarrollos y la forma de trabajar es cercana a las metodologías tradicionales o pesadas, con una gran cantidad de documentación y burocracia.

- ✓ El realizar un estudio comparativo de las metodologías de desarrollo de *software* permitió encontrar las ventajas y desventajas que poseen las mismas, para identificar con claridad la metodología que más se ajusta a las necesidades del área de mantenimiento y desarrollo SIMA.

- ✓ Por medio de la entrevista y las encuestas aplicadas a todo el grupo de mantenimiento y desarrollo del SIMA, se han logrado identificar las necesidades existentes, que de ser atendidas pueden mejorar de forma significativa aspectos como: la calidad, la comunicación y la eficiencia en la atención de las distintas tareas.

- ✓ Se concluye que la implementación de la metodología XP en el área de mantenimiento y desarrollo del SIMA ayudaría a mejorar aspectos de gran importancia tales como: la comunicación con el usuario final, la calidad del producto creado o modificado, la estandarización de código, el proceso de implementaciones en producción, entre otros.

6.2 RECOMENDACIONES

- ✓ Según la situación actual del área de mantenimiento y desarrollo del SIMA, se recomienda estandarizar y definir la forma en la cual se deben llevar a cabo las distintas actividades realizadas.

- ✓ Se recomienda que todo el grupo de mantenimiento y desarrollo del SIMA se mantenga estudiando y afinando la aplicación de la metodología utilizada.

- ✓ Para este proyecto se han descrito una serie de necesidades requeridas por el área de mantenimiento y desarrollo del SIMA, sin embargo, se recomienda que se realice un análisis constante para identificar nuevas carencias y definir de qué forma pueden ser solventadas con la metodología de desarrollo de *software* adquirida.

- ✓ Se recomienda aplicar la metodología XP para el área de mantenimiento y desarrollo del SIMA, ya que esto permitirá que los desarrolladores puedan trabajar de forma organizada y estandarizada, reduciendo esfuerzo de trabajo y obteniendo la satisfacción del cliente.

- ✓ Mantener la comunicación entre el usuario final y los programadores durante el proceso de desarrollo de *software*, con la finalidad de lograr una retroalimentación concreta y frecuente que permita desplegar resultados funcionales que cumplan con las expectativas del usuario.

- ✓ Realizar las pruebas del sistema de forma continua y oportuna, de manera que ayuden a reducir los posibles problemas que pueden presentarse a lo largo de una implementación de una iteración, con el fin de lograr un producto funcional y de alta calidad.

BIBLIOGRAFÍA

Alaimo, D. (2013). *Proyectos Ágiles con Scrum: Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos*. 1ª ed. Argentina: Ediciones Kleer.

Bahit, E. (2012). *Scrum & eXtreme Programming para programadores*. Argentina: Safe Creative.

Barrantes, R. (2014). *Investigación: un camino al conocimiento un enfoque cuantitativo y cualitativo*. UNED

Barrantes, R. (2002). *Investigación: un camino al conocimiento un enfoque cuantitativo y cualitativo*. San José, C.R.: Editorial Universidad Estatal a Distancia.

Beck et al. (2001). *Manifiesto por el Desarrollo Ágil de Software*. Utah, EEUU. Disponible en: http://agilemanifesto.org.

Bernal, C. (2010). *Metodología de la Investigación tercera edición*. Colombia: Pearson Educación.

Cámara de Tecnologías de Información y Comunicación, Promotora de Comercio Exterior de Costa Rica. (2015). *Mapeo Sectorial de Tecnologías Digitales 2014*. San José, Costa Rica. Disponible en: https://www.camtic.org/.

Chanto, C. (2016). *Propuesta de una metodología para la actualización de software en la empresa Clinverse S.A. durante el primer semestre del 2016*. (Tesina inédita de Bachillerato). Universidad Hispanoamericana, San José, Costa Rica.

CMMI Institute. (2017). *Certificaciones CMMI-DEV v1.3*. Disponible en: https://sas.cmmiinstitute.com/pars/pars.aspx.

Deemer et al. (2012). *Scrum Primer Una introducción básica a la teoría y práctica de Scrum versión 2.0*. Disponible en: http://scrumprimer.org.

Domínguez P. (2017). *Gestiona tu proyecto de desarrollo*. Disponible en: https://openclassrooms.com/courses/gestiona-tu-proyecto-de-desarrollo/en-que-consiste-el-modelo-en-cascada.

Fernández, J. (2013). *Introducción a las Metodologías Ágiles*. España: Universitat Oberta de Catalunya UOC.

Grijalva N. (2012). *Ingeniería del Software I*: Disponible en: http://sofware1nathalygrijalva.blogspot.com/2012/10/modelo-espiral.html

Instituto Nacional de Seguros. (2017). *Organigrama Institucional*. Disponible en: http://portal.ins-cr.com/portal.ins-cr.com/Institucional/Organigrama/Organigrama.

Instituto Tecnológico de Apizaco. (2013). *Unidad IV.- Programación Extrema (XP)*. Ciudad de México. Disponible en: https://modulopoo.wordpress.com/unidad-iv/.

Laínez, J. (2014). *Desarrollo de Software Ágil: Extreme Programming y Scrum. 2ª ed*: Createspace Independent Publishing Platform.

León, N., Gonzales, Y., Hernández, J, & Medina, M. (2013). *Ranking Mundial en certificaciones CMMI-DEV ver.1.3. Revista Iberoamericana de Producción Académica y Gestión Educativa ISSN 2007-8412, 6-7*. Disponible en: http://www.rcs.cic.ipn.mx

Marchena, G. (2013). *Análisis, Mejoramiento y Estandarización de los Procesos del Departamento de Control y Protección de Vida Silvestre del Minae*. (Tesina inédita de Bachillerato). Universidad Fidélitas, San José, Costa Rica.

Martínez, D. (2010). *Ingeniería de Software Tema 7: Mantenimiento del software*. Universidad Tecnológica de la Mixteca, León, México. Disponible en: http:// http://www.utm.mx.

Menzinsky, A.; López, G. & Palacio, J. (2016). *Scrum Manager Guía de Información versión 2.6*. Ed. Safe Creative. Disponible en: http://www.scrummanager.net.

Ocampo, M. (2016). *Propuesta de Modelo de Calidad para la Gestión de Requerimiento de Sistemas de Información del INDER*. (Tesis inédita de Licenciatura). Universidad Hispanoamericana, San José, Costa Rica.

Pineda, M. (2015). *Proponer una herramienta para la administración de cambios a través de un marco de trabajo de gestión de configuración y de los cambios para los procesos de desarrollo en Dinámica Consultores*. (Tesina inédita de Licenciatura). Universidad Fidélitas, San José, Costa Rica.

Place, E. (2009). *Programación Orientada a Objetos para PHP5*. Autoedición.

Pressman, R. (2010). *Ingeniería del software un enfoque práctico*. México D.F.: McGraw-Hill.

Hernández, R. (2014). *Metodología de la Investigación sexta edición*. México D.F.: McGraw-Hill/Interamericana Editores

Software Enginnering Institute. (2010). *CMMI para Desarrollo, Versión 1.3 Mejora de los procesos para el desarrollo de mejores productos y servicios*.

Sommerville, I. (2011). *Ingeniería de Software novena edición*. México D.F.: Pearson Educación.

Universidad Autónoma del Caribe. (2013). *Revisión de metodologías ágiles para el desarrollo de software*. Colombia. Disponible en: http://www.redalyc.org/articulo.oa?id=496250736004.

Universidad Politécnica de Valencia. (2006). *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. España. Disponible en: http://www.cyta.com.ar/ta0502/v5n2a1.htm.

Versionone Agile Made Easier. (2017). *11th Anual State of Agile Report*. Disponible en: https://www.versionone.com/.

Vila J. *Eventos Scrum: El Scrum Diario*: Disponible en: http://managementplaza.es/blog/el-scrum-diario/

GLOSARIO

Término	Significado
Ambiente de producción	Se refiere al ambiente real que utiliza el usuario final, normalmente todo el desarrollo se lleva a cabo en ambientes de desarrollo.
Analista de sistemas	Es un profesional especializado en las tecnologías de información, capacitado para interpretar necesidades empresariales o crear proyectos de Desarrollo de *Software* en diferentes áreas. Es capaz de analizar y diseñar sistemas computacionales que permitan aumentar la productividad y competitividad de las empresas.
Backlog	Es una lista ordenada de todo el trabajo pendiente.
CAMTIC	Cámara de Tecnologías de Información y Comunicación.
Clase	Una clase es un conjunto de objetos que comparten una estructura y comportamiento comunes.
CMMI-DEV	*Capability Maturity Model Integration,* en español Integración de modelos de madurez de capacidades, es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de *software*.
Código o código fuente	Es un conjunto de líneas de texto con los pasos que debe seguir la computadora para ejecutar un programa informático.
Cross-funcional	Es aquella persona que tiene todos los perfiles necesarios para entregar una tarea de *software* terminada al final de cada fase del proyecto.
Desarrollador	Es un especialista en informática que es capaz de concebir y elaborar sistemas informáticos.

Desarrollo	Se refiere a desarrollo de *software* que consiste en la creación de un programa informático que realiza alguna función específica.
Estandarización	Es el proceso de ajustar o adaptar características en un producto, servicio o procedimiento; con el objetivo de que éstos se asemejen a un tipo, modelo o norma en común.
Historia de usuario	Especificación de requisitos (acompañadas de las discusiones con los usuarios y las pruebas de validación).
Implementación	Es el acto de agregar o actualizar cierta funcionalidad del sistema informático.
Incidencia	Situación que se produce en el transcurso de algo y que repercute en su desarrollo. En informática una incidencia corresponde a un mal funcionamiento del programa.
INS	Instituto Nacional de Seguros.
Iteración	Es el acto de repetir un proceso con la intención de alcanzar una meta deseada, objetivo o resultado.
Mantenimiento de *Software*	Es la modificación de un programa informático después de la entrega, para corregir errores, mejorar el rendimiento, u otros atributos.
Método	Un método es un conjunto de instrucciones que realizan una determinada tarea y son similares a las funciones de los lenguajes estructurados.
Metodología de Desarrollo	Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información.
Monopolio	Productor o vendedor, es el único que explota un bien o un servicio, lo que le confiere un gran poder y le brinda una posición de privilegio.

MSI-DSI	Mantenimiento y Desarrollo de Sistemas del Instituto Nacional de Seguros.
ORACLE	Es básicamente una herramienta cliente/servidor para la gestión de Bases de Datos.
Parametrización	La acción de crear un tipo variable para asignarle valores específicos y que el tipo de valor pueda influir en el comportamiento del sistema informático.
PROCOMER	Promotora de Comercio Exterior de Costa Rica.
QA	Significa *Quality Assurance*, o aseguramiento de la calidad. Se trata de un conjunto de actividades de evaluación de las distintas etapas del proceso de desarrollo para garantizar que el producto final sea de calidad.
ROI	Retorno de la inversión, es decir obtener un valor superior al dinero invertido.
Rol	Función que una persona desempeña en un lugar o en una situación.
SCRUM	Es una metodología de desarrollo de *software* en la que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto.
SIMA	Sistema Integral Médico Administrativo.
Sintaxis	En la informática, la sintaxis se entiende como el grupo de normas que marcan las secuencias correctas de los elementos propios de un lenguaje de programación.
Software	Conjunto de programas y rutinas que permiten a la computadora realizar determinadas tareas.

Sprint	Es una simple iteración llevada a cabo por los miembros del equipo de desarrollo de *software*.
Stakeholders	Significa 'interesado' o 'parte interesada', y que se refiere a todas aquellas personas u organizaciones afectadas por las actividades y las decisiones de una empresa.
Usuario final	Persona o personas que van a manipular de manera directa un producto de *software*.
WIP	Por sus siglas *Work in Progress,* indica el trabajo en curso.
XP	*Extreme Programming* o programación extrema, es una metodología de desarrollo de *software* ágil que tiene como principal objetivo aumentar la productividad a la hora de desarrollar un proyecto *software*.

ANEXOS

ANEXO I

Martes 04 de julio del 2017

Ing. Yenory Rojas Hernández, Ph.D.
Directora de carrera Ing. Informática
Universidad Hispanoamericana

Reciba un cordial saludo de nuestra parte.

Esperando se encuentre de la mejor manera, tengo el agrado de dirigirme a usted con la finalidad de hacer de su conocimiento que el señor FREDDY ESQUIVEL FUENTES portador de la cédula de identidad 1-1286-0504, colaborador del Grupo INS y estudiante de la Universidad Hispanoamericana a la cual usted representa, cuenta con todo el apoyo para realizar su proyecto de graduación en nuestra Institución.

Dicho proyecto, consiste en la creación de una propuesta de estandarización para el proceso de Desarrollo de Software del Sistema Integral Médico Administrativo, lo anterior con el fin de mejorar la eficiencia y eficacia del servicio que brindamos, este proyecto del Sr. Esquivel se considera de sumo interés para nuestro departamento debido a los grandes beneficios que nos pueda aportar.

Quedo a su disposición para cualquier información adicional que le pueda brindar.

Saludos.

Lic. Johnatan Solano Díaz
Departamento de Tecnologías de Información
Líder de producto DSI-MSI ORACLE
Tel: (+506) 2284-8500 Ext: 8727
johnasol@ins-cr.com

Carta de aceptación de la empresa.

ANEXO II

Minuta para la definición de la metodología a proponer.

INS	MINUTA DE REUNIÓN	Minuta Versión 01 Página 1 de 2

Número	Fecha	Hora	Sitio de Reunión
M-TI-007	08/02/2018	3:30 p.m. a 05:20 p.m.	Laboratorio, Departamento de Informática en piso 10, Oficinas Centrales

Participantes:

Nombre	Departamento	Puesto
Johnatan Solano Díaz	Tecnología de Información	Líder de producto, ORACLE
Randall Durán Fallas	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Wendy Chaves Valverde	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Jorge Daniel Ulloa Gómez	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Luis Daniel Martínez Campos	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Limberg Artavia Barboza	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Jonathan Hernández Porras	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Marlon Sánchez Ureña	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Luis Diego Martínez Dobla	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales
Freddy Esquivel Fuentes	Tecnología de Información	Desarrollo y Mantenimiento de Sistemas de Información Institucionales

OBJETIVO
1. Definir la metodología ágil a utilizar en el área de mantenimiento y desarrollo del SIMA

INS	MINUTA DE REUNIÓN	Minuta
		Versión 01
		Página 2 de 2

Temas discutidos:

1-Calidad del producto desarrollado.

Se reconoció por parte de todo el grupo que la calidad del producto creado o modificado no es la mejor y por este motivo la cantidad de incidencias reportadas es tan elevada, se comenta sobre la necesidad de un estándar de desarrollo y la corrección de muchos aplicativos que están llenos de "parches", es decir soluciones rápidas para corregir un determinado problema, pero la funcionalidad general no es la más adecuada.

Se espera fortalecer la forma en que se realizan las pruebas funcionales por parte de la Unidad Funcional, esto con el fin de que el software que se pone en producción sea lo más fiable posible.

2-Comunicación con el usuario final.

Todo el grupo está de acuerdo en que la comunicación con el usuario debe mejorar ya que actualmente la misma es muy poca y la forma de comunicarse es por medio de la herramienta Gestic, lo que se pretende es tener una comunicación más directa ya que las veces que esto ha sucedido al atender incidentes o requerimientos la mejoría en agilidad y efectividad ha sido muy amplia.

3-Implementación de requerimientos en ambiente de producción.

Se comenta sobre el problema que ha existido al implementar en producción requerimientos de gran tamaño donde se empiezan a presentar múltiples errores, mismos que en algunas ocasiones impiden la atención de pacientes en todos los centros médicos del país.

4-Trabajo en equipo.

Se comenta sobre el hecho de que los requerimientos de gran tamaño sean asumidos por más de una persona, actualmente se realizan requerimientos individualmente y no existe el apoyo entre desarrolladores, además el conocimiento de todo el requerimiento queda en una sola persona, se desea el apoyo y conocimiento sea más grupal.

5-Proponer una metodología de desarrollo.

Se han revisado los valores, principios, herramientas y roles de las metodologías Scrum y Extreme Programming (XP), para definir cuál de las dos se ajusta de mejor forma a lo esperado por el grupo, por decisión unánime se propone la utilización de XP ya que la misma cuenta con una serie de principios que se adaptan de excelente forma a lo que se requiere; Además al proponer estandarización de código simple y la refactorización (Corrección de código incorrecto) se espera poder mejorar la gran cantidad de aplicativos "mal desarrollados" (Con malas prácticas de programación y código complejo) y así disminuir la gran cantidad de incidentes reportados diariamente.

Al final de la reunión se llega al acuerdo de que la metodología propuesta será Extreme Programming, esto por cuanto cumple de mejor manera los objetivos del área de Mantenimiento y Desarrollo del SIMA.

Lic. Johnatan Solano Díaz
Departamento de Tecnologías de Información
Líder de producto DSI-MSI ORACLE
Tel (+506) 2284-8500 Ext. 8727
johnasol@ins-cr.com

ANEXO III

Entrevista realizada al líder de producto DSI-MSI-ORACLE.

Universidad Hispanoamericana
Ingeniería Informática
Para optar por el grado de Bachillerato

UNIVERSIDAD HISPANOAMERICANA
Seriedad y Prestigio

La presente entrevista tiene como objetivo evidenciar la situación actual de la compañía en relación al uso de Metodologías de Desarrollo de Software.

Funcionario: Nombre Apellido1 Apellido2
Fecha: dd/mm/yyyy
Entrevistador: Freddy Esquivel Fuentes

Aspecto consultado	Respuesta suministrada
1. ¿Según su conocimiento, se cuenta con una metodología de desarrollo de software para llevar a cabo las actividades de mantenimiento y desarrollo requeridas por el SIMA? (Si la respuesta es NO pase a la pregunta número 3).	
2. ¿Cuál es la metodología de desarrollo de software que se utiliza actualmente para llevar a cabo las actividades de mantenimiento y desarrollo del SIMA? (En caso de recibir una respuesta pase a la pregunta número 4).	
3. ¿De qué modo se realizan las tareas de mantenimiento y desarrollo del SIMA al no contar con una metodología de desarrollo definida?	
4. ¿Según su criterio la calidad del mantenimiento desarrollo de software implementado en el SIMA es la esperada?	
5. ¿Considera usted que el uso de una metodología de desarrollo de software puede mejorar la calidad del software creado o modificado?	
6. ¿Cree usted que metodologías de desarrollo ágiles tales como: Scrum, Kanban, Extreme Programming, entre otras, son más eficientes que las metodologías tradicionales?	
7. ¿Considera usted que el método utilizado para llevar a cabo las actividades de mantenimiento y desarrollo del SIMA cubre todas las necesidades de la compañía?	

Universidad Hispanoamericana
Ingeniería Informática
Para optar por el grado de Bachillerato

La presente entrevista tiene como objetivo evidenciar la situación actual de la compañía en relación al uso de Metodologías de Desarrollo de Software.

Funcionario: Nombre Apellido1 Apellido2
Fecha: dd/mm/yyyy
Entrevistador: Freddy Esquivel Fuentes

Aspecto consultado	Respuesta suministrada
8. ¿Estaría dispuesto a impulsar la implementación de una metodología de desarrollo de software para optimizar las tareas de mantenimiento y desarrollo ejecutadas en el SIMA?	
9. ¿Estaría de acuerdo en que se realice un estudio de las distintas metodologías de desarrollo de software para identificar cual puede proporcionar una mejora significativa en el proceso actual de mantenimiento y desarrollo del SIMA?	
10. ¿Cuál cree que sea el nivel de aceptación por parte del grupo de analistas y programadores en caso de implementar una nueva metodología de desarrollo de software?	

ANEXO IV

Encuesta realizada al grupo de mantenimiento y desarrollo del SIMA.

Universidad Hispanoamericana
Ingeniería Informática
Para optar por el grado de Bachillerato

La presente encuesta tiene como objetivo obtener información relevante por parte del grupo de trabajo del SIMA en relación al conocimiento de las metodologías de desarrollo de software existentes y en general del proceso actual de mantenimiento y desarrollo.

Fecha: dd/mm/yyyy
Entrevistador: Freddy Esquivel Fuentes

Aspecto consultado	Opciones
1. ¿Cuál es el puesto que desempeña actualmente en la empresa?	a Administrador de proyectos b Analista de sistemas c Desarrollador/programador
2. ¿Cuánto tiempo tiene de trabajar con el Sistema Integral Médico Administrativo?	a Menos de 1 año b Entre 1 y 2 años c Entre 2 y 5 años d Entre 5 y 10 años e Más de 10 años
3. ¿Ha escuchado hablar sobre metodologías de desarrollo de software ágiles o tradicionales?	a Si b No
4. ¿Cuáles de las siguientes metodologías de desarrollo de software le resultan conocidas?	a SCRUM b Extreme Programming (XP) c Kanban d Modelo de la cascada e Modelo espira de Boehm f Modelo de desarrollo incremental g Ninguna
5. ¿Según su conocimiento cuál de las siguientes metodologías de desarrollo de software cree que se adaptaría de buena forma al modo de trabajar en el SIMA?	a SCRUM b Extreme Programming (XP) c Kanban d Modelo de la cascada e Modelo espira de Boehm f Modelo de desarrollo incremental g Ninguna

Universidad Hispanoamericana
Ingeniería Informática
Para optar por el grado de Bachillerato

La presente encuesta tiene como objetivo obtener información relevante por parte del grupo de trabajo del SIMA en relación al conocimiento de las metodologías de desarrollo de software existentes y en general del proceso actual de mantenimiento y desarrollo.

Fecha: dd/mm/yyyy
Entrevistador: Freddy Esquivel Fuentes

Aspecto consultado	Opciones	
6. ¿Con cuáles de las siguientes premisas puede identificar el proceso actual de mantenimiento y desarrollo de software del SIMA?	a	Algunos de los mantenimientos o desarrollos se realizan de manera informal para la solución de problemas urgentes
	b	Los requerimientos de gran tamaño son desarrollados en su totalidad y hasta el final del proceso son implementados en ambiente de producción
	c	Existe un proceso de mantenimiento y desarrollo de software pero no se encuentra formalmente documentado
	d	Los desarrollos son rígidos y basados estrictamente en la especificación del requerimiento
	e	Los requerimientos de gran tamaño son desarrollados por etapas y se implementan incrementalmente en secciones o partes completamente funcionales
	f	Se utiliza una metodología de desarrollo de software totalmente definida, documentada y consolidada
	g	Se realizan reuniones diarias para aclaración de dudas o validación de funcionalidades
	h	Existe una comunicación constante con el usuario para verificar si el desarrollo se está realizando acorde a lo requerido
	i	Existe un departamento de QA donde se realicen pruebas exhaustivas para evitar pases a producción con posibles errores
7. ¿Según su criterio cual sería el nivel de comunicación que existe entre el usuario final y los programadores o analistas desde el inicio de un desarrollo y hasta la culminación del mismo?	a	Muy bajo
	b	Bajo
	c	Medio
	d	Alto
	e	Muy alto

Universidad Hispanoamericana
Ingeniería Informática
Para optar por el grado de Bachillerato

La presente encuesta tiene como objetivo obtener información relevante por parte del grupo de trabajo del SIMA en relación al conocimiento de las metodologías de desarrollo de software existentes y en general del proceso actual de mantenimiento y desarrollo.

Fecha: dd/mm/yyyy
Entrevistador: Freddy Esquivel Fuentes

Aspecto consultado	Opciones
8. Existen actividades de desarrollo en equipo tales como: apoyo entre los miembros del grupo para la solución efectiva de errores o la programación en parejas para mejorar la calidad y disciplina ¿En qué medida cree usted que se promueven este tipo de actividades en el equipo de trabajo del SIMA?	a Muy bajo b Bajo c Medio d Alto e Muy alto
9. ¿Cómo considera que es el apoyo de la unidad funcional con los programadores o analistas cuando requieren conocer en detalle alguna funcionalidad del SIMA?	a Bajo b Medio c Alto
10. ¿Está de acuerdo a que el uso de una metodología de desarrollo de software en el SIMA puede tener un impacto positivo en la calidad y eficiencia del producto desarrollado?	a Si b No
11. ¿Estaría de acuerdo en adaptarse al uso de una nueva metodología de desarrollo de software sabiendo que esto mejoraría las actividades realizadas en el SIMA?	a Si b No