MARCO DE BUENAS PRACTICAS BAJO LA METODOLOGÍA SCRUM, PARA DESARROLLAR CÓDIGO SEGURO - (SCRUMSEC)

Autor:
LUIS CARLOS FIGUEREDO LEYTON

UNIVERSIDAD EAN
FACULTAD DE INGENIERÍA
MAESTRÍA EN GERENCIA DE SISTEMAS DE INFORMACIÓN Y PROYECTOS TECNOLÓGICOS
Bogotá, Colombia
2020

MARCO DE BUENAS PRACTICAS BAJO LA METODOLOGÍA SCRUM, PARA DESARROLLAR CÓDIGO SEGURO - (SCRUMSEC)

Autor:
LUIS CARLOS FIGUEREDO LEYTON

Trabajo de grado presentado como requisito para optar al título de:
MAGISTER EN GERENCIA DE SISTEMAS DE INFORMACIÓN Y PROYECTOS TECNOLÓGICOS

Director (a):
Ing. ALIX ERICA ROJAS HERNÁNDEZ

Modalidad:
MONOGRAFÍA

UNIVERSIDAD EAN
FACULTAD DE INGENIERÍA
MAESTRÍA EN GERENCIA DE SISTEMAS DE INFORMACIÓN Y PROYECTOS TECNOLÓGICOS
Bogotá, Colombia
2020

Nota de aceptación

_____

_____

_____

_____

_____

_____

Firma del jurado

_____

Firma del jurado

_____

Firma del director del trabajo de grado

Bogotá D.C. Día - mes – año

Resumen

Con la creciente demanda del uso de aplicaciones, y procesos de transformación digital que están viviendo las organizaciones, desarrollar código seguro, es una actividad que hoy debe recibir especial atención, ya que, al masificar la utilización de herramientas, plataformas, y medios transaccionales, es importante que las empresas e instituciones garanticen la seguridad y privacidad de los datos e información de su entorno y de sus usuarios.

Estos cambios han llevado a adoptar el uso de metodologías ágiles, que permiten dar eficiencia, flexibilidad y agilísimo a las entregas de los proyectos de desarrollo de software, generando calidad y buenos resultados entre el cliente y proveedores de servicio. Una de las que ha sido adoptada alrededor del mundo es Scrum, la cual está, en cerca del 80% de las organizaciones (Consultancy.eu, 2020). Desarrollar software ágil, es importante, pero la seguridad debe ser un elemento que trasciende desde la estrategia y la orientación del negocio en las organizaciones.

El presente proyecto, busca con la ayuda de las principales metodologías de desarrollo de código seguro, generar un marco que permita a través de Scrum como metodología ágil, la apropiación de los elementos de seguridad que no se tienen en cuenta, en el desarrollo de código y aplicaciones, con el fin de mitigar fallos, vulnerabilidades y riesgos en el uso de estas. Con los siguientes temas se busca dar cumplimiento: identificación y análisis de las vulnerabilidades, fallos y riesgos en aplicaciones; análisis de las principales metodologías de desarrollo de software seguro; definición y resultado de criterios de evaluación; definición de las fases del marco de buenas prácticas de desarrollo de código seguro; validación del marco mediante juicio de expertos; conclusiones y recomendaciones.

Palabras clave: Marco de desarrollo de código seguro, buenas prácticas, vulnerabilidades, fallos y riesgos en aplicaciones, metodología ágil Scrum.

Abstract

With the increasing demand for the use of applications and digital transformation processes that organizations are experiencing, developing secure code is an activity that today must receive special attention, since, by massively using tools, platforms, and transactional media, it is important for companies and institutions to ensure the security and privacy of data and information in their environment and their users.

These changes have led to the adoption of agile methodologies, which allow to give efficiency, flexibility, and agility to the deliveries of software development projects, generating quality and good results between the customer and service providers. One of the methodologies that has been adopted around the world is Scrum, which is, in about 80% of the organizations (Consultancy.eu, 2020). Developing agile software is important, but security must be an element that transcends strategy and business orientation in organizations.

This project, with the help of the main methodologies of secure code development, seeks to generate a framework that allows, through Scrum as an agile methodology, the appropriation of security elements that are not taken into account, in the development of code and applications, in order to mitigate failures, vulnerabilities and risks in the use of these. The following issues are addressed: identification and analysis of vulnerabilities, failures and risks in applications; analysis of the main methodologies of secure software development; definition and result of evaluation criteria; definition of the phases of the framework of good practices of secure code development; validation of the framework through expert judgment; conclusions and recommendations.

Keywords: Secure code development framework, good practices, vulnerabilities, failures and risks in applications, agile Scrum methodology.

Tabla de contenido

Pág.

Lista de figuras

Pág.

Lista de tablas

Pág.

1. Introducción

En la actualidad, personas y organizaciones utilizan herramientas y aplicaciones que generan productividad y dinamismo a sus actividades; estas ayudan a alcanzar los retos y metas que se proponen; además de facilitar tareas sencillas que incluso pueden tomar mucho tiempo. Hoy en día, existen bastantes aplicaciones, y desarrollos que sirven para tal fin, y son diseñados y construidos desde el uso de metodologías ágiles como Scrum, pero que nos hacen cuestionar, si son realmente seguras, e incluso cómo estas manejan los datos e información.

La seguridad es un aspecto fundamental que debe ser incluido en los procesos de desarrollo de código, en los equipos de trabajo que implementa Scrum, y en los proyectos en los que las organizaciones buscan mejorar su capacidad digital y cobertura de servicios. Debido a lo anterior el agilísimo se ha posicionado, como un movimiento de entregas rápidas de calidad y con un alto nivel de complejidad, en el que es posible mediante la transparencia, inspección y adaptación, la adopción de estos procesos en las organizaciones e instituciones.

Es aquí en donde diseñar un marco de buenas prácticas, para desarrollo de código seguro bajo la metodología Scrum, incorpora la importancia de la implementación de elementos de seguridad, a la hora de desarrollar código desde una metodología ágil como lo es Scrum, partiendo del hecho fundamental, que ninguna aplicación o herramienta se construye 100% segura, y que los datos e información, pueden ser susceptibles de ser extraídos, vulnerados o robados con fines económicos o terroristas, por grupos o personas al margen de la ley.

Este trabajo está estructurado en quince capítulos. El **primer capítulo**, compuesto por la introducción, la formulación del problema, la pregunta general y las preguntas específicas; el **segundo capítulo** contiene el objetivo general y los objetivos específicos; **el tercer capítulo** la hipótesis del proyecto; y el **cuarto capítulo** la justificación.

En el **quinto capítulo** se presenta el marco de referencia, en el que se da un acercamiento a los antecedentes, la seguridad de la información; se habla sobre el desarrollo seguro, y las metodologías ágiles en el desarrollo de software, aquí se profundiza sobre *agile* y Scrum. Después se presenta el **sexto capitulo** en el que se incorpora la metodología de investigación.

Con el **séptimo capítulo** se da cumplimiento al primer objetivo. Este da un acercamiento a las vulnerabilidades, fallos y riesgos comunes en las aplicaciones que afectan la seguridad en las organizaciones y en las instituciones. Así mismo, para cumplir con el segundo objetivo se crea el **octavo capitulo**, en el que se analizan las principales metodologías de desarrollo de software seguro.

Para dar cumplimiento con el tercer objetivo al contener un alcance más amplio, se crea el noveno, decimo y undécimo capitulo**.** El **noveno capítulo** compara las principales metodologías de desarrollo de software seguro, con el fin de identificar los elementos de seguridad que deben ser incluidos en el marco de buenas prácticas.

Con el fin de dar un contexto más amplio sobre los procesos de Scrum, se incluye el **capítulo decimo**, que habla sobre el manifiesto ágil, la visión general de la metodología, el equipo, los eventos, los artefactos y procesos de Scrum; para posteriormente en el **capítulo once**, diseñar el marco de buenas prácticas para desarrollar código seguro bajo la metodología Scrum, y se definen las fases y actividades que presentan relevancia para ser agregadas al mismo. Ya en el **capítulo doce** se da cumplimiento al cuarto objetivo; este valida la pertinencia del marco, a través de la metodología de juicio de expertos.

Adicional en el **capítulo trece**, se cierra este trabajo de maestría con las conclusiones generales; y en el **capítulo catorce** con las recomendaciones, para fortalecer lo investigado y diseñado en el marco de buenas prácticas para desarrollo seguro. Por último, en el **capítulo quince**, están incluidas las referencias que aportaron al establecimiento de conocimiento e investigación, y los **anexos** que ayudan a las actividades de seguridad desde la práctica.

1.1. Formulación del problema

En la actualidad el desarrollo de software es una práctica que ha tenido que evolucionar de manera acelerada desde el inicio de la computación. Aproximadamente, desde las últimas dos décadas ha habido un auge en el desarrollo de código y de aplicaciones, debido a las múltiples soluciones que existen para satisfacer necesidades y servicios, que dan simpleza e interoperabilidad y han llevado a grandes y pequeñas organizaciones a integrarlas en sus procesos de transformación hacia los clientes y usuarios finales.

Esta tendencia ha hecho que las empresas evolucionen y adopten nuevas tecnologías, tanto así, como para proveer el desarrollo y asegurar la entrega efectiva de sus aplicaciones. De igual manera el crecimiento acelerado del uso de dispositivos móviles ha permitido que la industria del desarrollo de software impacte de manera gradual las actividades económicas de los países, incorporando productividad y distintas soluciones para diferentes tipos de problemas (Bastos, 2019). Los cambios que se dan en este ámbito requieren agilidad y el uso de metodologías que permitan adaptar de manera rápida las condiciones de los nuevos proyectos de desarrollo, en los que la eficiencia y la flexibilidad generan menores costes de producción, y una alta calidad (Garcia, 2019).

En el desarrollo de código, la entrega es fundamental, y es aquí donde las metodologías ágiles han jugado un papel importante desde hace ya tres décadas en la historia del software (Roche, 2018); generando conocimiento, aprovechando el valor del cliente y acompañando a este en lograr buenos resultados en una sociedad cambiante y dada a innovar. Tanto así, que se estima que al año 2019 a nivel mundial, el uso de estas en la gestión de proyectos en las organizaciones fue de alrededor del 71%, como lo revela un estudio del *Project Management Institute (PMI).* Y a nivel de Latinoamérica en Colombia la adopción de la cultura ágil está cerca al 49% en las empresas (Cano, 2019).

Como lo muestra la versión 14 del informe anual sobre el estado de la concepción ágil - *14th Anual State Of Agile Report - 2020*, se encuestaron más de 40 mil profesionales, ejecutivos y consultores, los cuales participaron de este estudio en el que demostró que Scrum, es la metodología ágil más utilizada con al menos una participación del 75% entre los encuestados, incluso aumento está en un 5% con respecto al año 2019 respectivamente (Digital.ai, 2020).

Scrum se enfoca principalmente en el trabajo en equipo, la colaboración entre el cliente y el proveedor y el desarrollo iterativo, todo esto aplicando un conjunto de buenas prácticas, las cuales tienen gran capacidad para adaptarse al cambio siendo flexible. Scrum se caracteriza por sus iteraciones, en las que mediante el desarrollo de pequeños productos dan valor al proyecto en un corto plazo; cada ciclo (*Sprint*) tiene un tiempo de duración de dos a cuatro semanas, que garantiza el incremento y las entregas del producto en sí. Este marco de trabajo aborda situaciones complejas, en las que de manera adaptativa da el mayor valor posible a los resultados generados dentro de los equipos de trabajo y en las organizaciones.

Desde un punto de vista práctico, los beneficios que destacan las metodologías ágiles han ayudado en muchos aspectos a la cultura del desarrollo de software, incluyendo a Scrum, claro está; en el que las entregas rápidas, optimizadas y de calidad han marcado un punto importante en nuestra era actual. Sin embargo, la deuda técnica que resulta de la construcción de productos de software de la manera en que se hacen con estas metodologías, ha generado una alta tendencia en los últimos años a los ataques cibernéticos, lo cual genera una dinámica creciente en el número de incidentes de este tipo no solo en nuestro país, sino a nivel mundial. Según el informe Tendencias Cibercrimen Colombia 2019 - 2020[1], durante el año 2019 se registraron aproximadamente 30.140 casos de ciberdelitos, los cuales fueron dispuestos a mano de la Policía Nacional, y se estima que alrededor del 57% de estos tenían relación con la infracción de la Ley 1273 de 2009[2] la cual modifica el código penal colombiano, y crea un bien jurídico denominado "de la protección de la información y de los datos" y preservación integral de los sistemas que utilizan tecnologías de la información y las comunicaciones. No es un secreto, que una de las principales motivaciones frente a estas prácticas es la económica y posteriormente lo que se denomina la monetización generada por estos ciberataques. De manera general el delito más denunciado en nuestro país es el hurto por medios informáticos con un total de 31.058 casos, luego la violación de datos personales con 8.037 casos, y el tercer delito más denunciado es el acceso abusivo a sistema informático con 7.994 casos. Otros delitos estuvieron asociados a amenazas de secuestro de datos

[1] Nacional, Policia. (29 de octubre de 2019). Tendencias Cibercrimen Colombia 2019-2020. Obtenido de https://caivirtual.policia.gov.co/sites/default/files/tendencias_cibercrimen_colombia_2019_-_2020_0.pdf

[2] Distrital, S. J. (5 de enero de 2009). Ley 1273 de 2009 Nivel Nacional. Obtenido de https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=34492

(ransomware), ataques de denegación de servicio *(DDOS),* y el uso de programas maliciosos (Nacional, Policia, 2019).

Lo anterior demuestra el valor que se debe dar al uso de los datos e información, en cualquier entorno; y la necesidad de desarrollar código seguro para aplicaciones, que brinden confiabilidad y calidad a los usuarios sin comprometer su privacidad e información, y que estas se desarrollen bajo metodologías ágiles como Scrum.

Es importante considerar las consecuencias de los problemas de seguridad que aparecen en las entregas de desarrollo de software, que se dan con una mayor velocidad y con el uso de las metodologías ágiles, y más aún, es importante que dentro de la metodología más utilizada en el mundo para desarrollar software, en este caso Scrum, se incluyan todos los elementos de seguridad necesarios, para dar una experiencia de usuario, no solo de calidad, ni de funcionalidad, sino también dar una experiencia segura en los datos e información que genera con el uso de las aplicaciones.

1.2. Pregunta general

¿Cuáles son los elementos necesarios de un marco de buenas prácticas, para desarrollar código seguro bajo la metodología Scrum?

1.3. Preguntas especificas

¿Cuáles son los riesgos, amenazas y vulnerabilidades más comunes a los que se ve expuesto un usuario con el uso de aplicaciones?

¿Cuáles son las principales metodologías, o buenas prácticas de desarrollo de código seguro, que existen?

¿Es posible crear un marco de buenas prácticas bajo metodología Scrum, para desarrollar código seguro?

¿Por qué es importante desarrollar código seguro, e incluir esta práctica dentro de una metodología como Scrum?

2. Objetivos

2.1. Objetivo general

Diseñar un marco de buenas prácticas, para el desarrollo de código seguro bajo la metodología Scrum.

2.2. Objetivos específicos

- Categorizar las vulnerabilidades, fallos y riesgos de seguridad más comunes, para prevenir brechas y ataques en el desarrollo de código.
- Determinar los dominios, prácticas, herramientas, y roles a partir de las principales metodologías de desarrollo de código seguro.
- Establecer las buenas prácticas, elementos, artefactos, y herramientas de seguridad que han de ser conjugados dentro de la metodología Scrum.
- Validar y socializar el marco de buenas prácticas bajo la metodología Scrum.

3. Hipótesis

A partir de la investigación y análisis sobre las principales metodologías de desarrollo de código seguro, y mediante la profundización del uso y los procesos de la metodología ágil Scrum, se espera mitigar los incidentes de seguridad de la información en aplicaciones terminadas.

3.1. Hipótesis específicas

- Las funciones de negocio deberán estar alineadas al plan estratégico de la organización y en concordancia con los principios de seguridad de la información para construir aplicaciones seguras.
- La integración de Scrum y de elementos de seguridad de las principales metodologías de desarrollo de código seguro, permitirá mitigar los errores de programación que generan vulnerabilidades en las aplicaciones.
- El equipo de Scrum será capacitado y contará con formación en seguridad. Esto permitirá identificar y disminuir la generación de brechas de seguridad que serán remediadas oportunamente.
- La implementación de un plan de respuesta a incidentes en la fase de despliegue permitirá tener una base de como contener ataques informáticos en aplicaciones terminadas.

4. Justificación

En un mundo globalizado donde la información y la tecnología van de la mano, y donde se ha masificado el uso de aplicaciones, el valor de los datos e información se vuelve crucial, no solo para las organizaciones, sino también para las personas. De lo anterior se generan nuevas necesidades; y es por eso, por lo que cada día se crea una nueva aplicación, dando movilidad y acceso a plataformas, consumos y productos que generan una relación directa entre organizaciones y consumidores.

Tal y como lo revela un reciente estudio de *Social Media & Branding,* las categorías de las aplicaciones más usadas en el mundo corresponden en un 89% a medios sociales y mensajería instantánea, a un 65% a aplicaciones de entretenimiento, un 47% son aplicaciones de juegos, las de compras ocupan un 66%, y aplicaciones de música un 52%, (Xie, 2020). Para dar más claridad sobre el estudio, este fue realizado a marzo del año 2020 y el porcentaje corresponde al consumo de internet en personas en edades de 16 a 64 años respectivamente[3]. Un factor común entre todas ellas está asociado a que administran datos e información de usuarios, y tienen acceso a información privada y confidencial, a los contenidos y preferencias que se generan minuto a minuto al interactuar con ellas, y en muchas ocasiones a información de productos financieros.

Algo que deben garantizar las organizaciones que proveen estas aplicaciones es la seguridad en los datos, información, y políticas de privacidad, tanto en el uso de estas, como en la generación de buenas prácticas para el desarrollo de código, basado en normas y disposiciones que guíen el que hacer de las mismas. Dado lo anterior, el enfoque que tienen estas aplicaciones desde su concepción y desarrollo va orientado al uso de metodologías ágiles dentro de sus equipos de trabajo, esto con el fin de acelerar los procesos y proyectos de desarrollo de código, para poder responder de manera rápida y ágil, a las necesidades y mercados emergentes de posibles usuarios que se dan día a día.

Un hecho interesante a resaltar, es: ¿cuáles fueron los motivos por los cuales las organizaciones, de un modo u otro decidieron adoptar metodologías ágiles al interior de sus equipos, y en sus procesos de desarrollo de software?; según lo investigado, estos

[3] Xie, Y. M. (4 de marzo de 2020). Situación Global Mobile 2020. Obtenido de https://yiminshum.com/mobile-movil-app-2020/

están asociados a: acelerar las entregas de los productos, mejorar la capacidad de adaptabilidad y respuesta al cambio de requerimientos, por supuesto también está el incremento a la productividad, la calidad y la predictibilidad en las entregas del mismo (TI, 2015).

Si bien es cierto, en la actualidad existen varias metodologías ágiles y buenas prácticas adoptadas por la industria como: *Scrum*, *Kanban*, *Crystal Clear*, *Extreme Programming XP*, y *Agile Inception*; estas tienen un objetivo común y es la entrega del software funcional entre dos semanas y dos meses, o en un tiempo menor si este se llega a estipular; estas se enfocan principalmente en satisfacer al cliente con entregas continuas de productos terminados. Sin embargo, los resultados de un estudio realizado en el año 2017 demuestran, que las etapas más importantes en el desarrollo de software para las organizaciones son las de análisis y definición de objetivos, excluyendo por completo los requisitos de seguridad; lo que deja en evidencia que la seguridad de la información no se contempla en los proyectos de desarrollo como un requerimiento más dado por el cliente, esto a pesar de las vulnerabilidades e incidentes de seguridad que afectan la integridad, disponibilidad y confidencialidad de la información, como los ataques de inyección de código, perdida de autenticación, fuga de información, problemas criptográficos, y exposición de datos sensibles; adicional a lo anterior, este estudio demuestra que los clientes consideran más importante la funcionalidad sobre la seguridad, lo que genera una deuda técnica en el uso de las metodologías ágiles entorno a la seguridad de las aplicaciones (AGUILERA, 2017), es por eso, que con el marco de buenas prácticas bajo la metodología Scrum, se incluirán los elementos de seguridad que permitan a las organizaciones gestionar de manera rápida y eficiente los riesgos, vulnerabilidades y errores que se den durante todas las fases de desarrollo en los ciclos *(Sprint)* que propone Scrum.

5. Marco de referencia

Es innegable que en la actualidad se está viviendo una revolución tecnológica, y distintos sectores están siendo objeto de digitalización de sus procesos, lo que aumenta el desarrollo de aplicaciones, herramientas, interacción y uso directo de los usuarios y las organizaciones entorno a estos nuevos ambientes. Tal y como lo comenta *Sicrom Team* (2018), desde hace años los objetivos principales de los criminales cibernéticos eran las personas, pero en la actualidad esto cambio, y la mayoría de los ataques e incidentes de seguridad son dirigidos principalmente a empresas y a las aplicaciones que se exponen como uso común a los usuarios.

Como lo reporta el Instituto Nacional de Ciberseguridad (INCIBE) de España, lo anterior genera pérdidas millonarias causadas por incidentes de seguridad en las empresas y las *pymes*; entre los cuales se presentan (Sicrom, 2018):

- Ciber espionaje – 73.368,53 US
- Intrusión en la red – 72305,62 US
- Denegación de servicio – (*DDOS*) – 56331,61 US
- Suplantación de identidad – 51018,26 US
- Vulnerabilidades en el software – 44.634,98 US
- Infección por programa maligno – 32.945,37 US
- Fuga de información – 35.103,95 US

Es alarmante como año tras año estas cifras que generan pérdidas millonarias a las organizaciones e instituciones aumentan y generan cierto grado de sofisticación en los métodos que utilizan para realizar los ataques cibernéticos, lo que indica una evolución de estas malas prácticas también. Si bien es cierto, existen diversos tipos de ataques, vulnerabilidades, riesgos y fallos los cuales han estado presentes desde hace varios años, la gestión de la seguridad de la información también ha evolucionado y ha estado de la mano de los avances tecnológicos, incrementando entre otras sus controles, y el uso de metodologías que permiten desarrollar código seguro, para las aplicaciones y herramientas que están presentes en los distintos sectores de la sociedad.

5.1. Antecedentes

Hoy con la transformación digital, en la que se usan aplicaciones de todo tipo, en cualquier lugar y momento; para solicitar transporte, pedir servicios, para que traigan comida y medicamentos, entre otros. Incluso resulta más cómodo, y casi inaceptable como usuarios salir a buscar algunos productos que son de uso frecuente. Pero esto como todo en su curso normal trae consecuencias, en las que en muchas ocasiones no se reconoce por muchos factores si una aplicación es segura o no, pero ¿qué sucedería si al usar una de estas aplicaciones se perdiera información?, o ¿qué sucedería si la información comprometida es, entre otras, de carácter personal o financiera? Bueno, pues es algo que busca el presente documento, y es en lo que las empresas deben enfocarse ahora y en los próximos años, implementar modelos de seguridad en los desarrollos de software.

A partir de lo anterior, el software es aquella parte lógica en la que se convierten estas aplicaciones, y es utilizada por los usuarios de distintas maneras, y en distintos dispositivos. Con la creciente dependencia del uso de herramientas, dispositivos y aplicaciones para que se hagan trabajos críticos significa que se está dando un valor a este que no reside únicamente en la capacidad de mejorar o quizás mantener la eficiencia. Sino que tiene un valor en el que es derivado de su capacidad de poder continuar operando en forma que sea fiable e incluso en momentos en los que cuentan con eventos que lo pueden llegar a amenazar (Matei, 2015). Esto genera una interdependencia en lo que respecta al software, ya que, incluso asumiendo un nivel constante de fallos, no se piensa en si este es fiable o no, y se asume en que la capacidad de confianza de este seguirá bajo cualquier circunstancia.

Las organizaciones tienen control sobre la información que se genera, ya que en las políticas de privacidad y en los acuerdos de confidencialidad se da acceso a la misma, pero esto sucede por usar la aplicación y el software que se desarrolló con el fin de solucionar una necesidad que incluso no se sabía que existía, y la información es inmersa en un mundo en el que viaja a través de sus servidores y redes inseguras como internet; y en el que las transacciones financieras y otro tipo de información viaja de manera recurrente y global, exponiendo la confidencialidad y haciendo que la capacidad con la que se desarrolló y se implemento sea fácilmente vulnerada.

Desarrollar software seguro debe considerarse importante, ya que como se mencionó este es usado por grandes masas, y así tenga funciones específicas o distintas para las que fue desarrollado, seguirá siendo el punto de inflexión de los criminales en esta era cibernética y de transformación digital. Por los distintos tipos de software, en muchas entidades pasan procesos que son críticos, y esto hace que para quienes lo quieran vulnerar tenga un objetivo bastante amplio, quizás no solo sea económico, sino también reputación, además de penal, competitivo o incluso hasta con fines terroristas.

Sobre esta problemática de desarrollo de código seguro, se ha venido trabajando de manera progresiva, sin embargo, las actividades de seguridad en el desarrollo de software y codificación de aplicaciones son pocas, y transcienden a tal punto, que es necesario implementar políticas y metodologías para esta práctica. Tal y como lo comenta Parra (2011), la perdida de información en los aplicativos, se da por falta de organización en el desarrollo y mantenimiento de aplicaciones fiables, lo que ocasiona pérdidas económicas y perdidas de información, por lo que se hace necesario diseñar un marco de buenas prácticas que asegure el éxito en la implementación de actividades de seguridad (PARRA, 2011). Además, por el volumen de datos e información, que circulan a través de los aplicativos y herramientas, a diario se ejecutan millones de líneas de código que son hechas por desarrolladores que no son conscientes de los riesgos de seguridad, a los que pueden estar expuestas aplicaciones, es por eso la importancia de adoptar políticas que cumplan con las exigencias de seguridad pertinentes (Alejandro, 2017).

En un informe realizado en los Estados Unidos en el año 2005, el cual se tituló "*Cyber Security: A Crisis of Prioritization*", contemplan el problema de la seguridad del software como una disciplina que aún no ha sido desarrollada, y que en su ciclo de vida no ha sido procesada en su totalidad, más aun es comparado como la enfermedad del cáncer, en la que así se trate con un determinado tratamiento esta llegara a infectar a todo el sistema y lo dañara llevando consigo la información siendo una amenaza para las demás aplicaciones (Benioef & Lazowska, 2005).

Lo que nos indica esta problemática, debe tomarse en serio, ya que las vulnerabilidades que son explotadas pueden llegar a remediarse e incluso conocerse durante el diseño y construcción del código fuente; y lo más preocupante aun es que es cierto, en que, si es explotada de manera adecuada una de estas vulnerabilidades, puede llegar a copiarse e infectar otros sistemas en la red más grande del mundo, y por consiguiente la información

puede llegar a ser pertenencia de manos criminales, y vendidas en sitios oscuros de redes de computadoras como la *Deep Web* o la *Dark Web.*

5.2. Seguridad de la información

Desde el inicio la seguridad ha sido ligada de varias maneras a la humanidad; y está, alrededor de los años ochenta ha girado en torno a los cambios que han masificado el desarrollo progresivo y creciente de las tecnologías de la información y la comunicación (FromLinux, 2016).

La seguridad de la información está compuesta por un conjunto de técnicas y medidas que permiten el control de los datos e información que se manejan en una organización o institución; esta salvaguarda los datos que se encuentran en un sistema informático, en todos los contextos posibles con el fin de responder a tres cualidades: critica, valiosa y sensible. Hoy en día, hablar de seguridad de la información es un aspecto esencial, ella enmarca y resguarda de forma integral la información de un sujeto, en el contexto de personas, empresas, instituciones, organismos, sociedad y gobiernos. También es considerada como la disciplina que trata sobre los riesgos, amenazas y posibles escenarios en los que las buenas prácticas y esquemas normativos exigen a las organizaciones, niveles de aseguramiento óptimos en las tecnologías para elevar los niveles de confianza en la creación, almacenamiento, transmisión, recuperación, y disposición final de la información. (Etica, 2013). Por ende, la esencia de esta es mantener protegidos los datos que son importantes para los sujetos que debe defender.

La seguridad de la información cuenta con tres pilares, en los que cimienta sus principios.

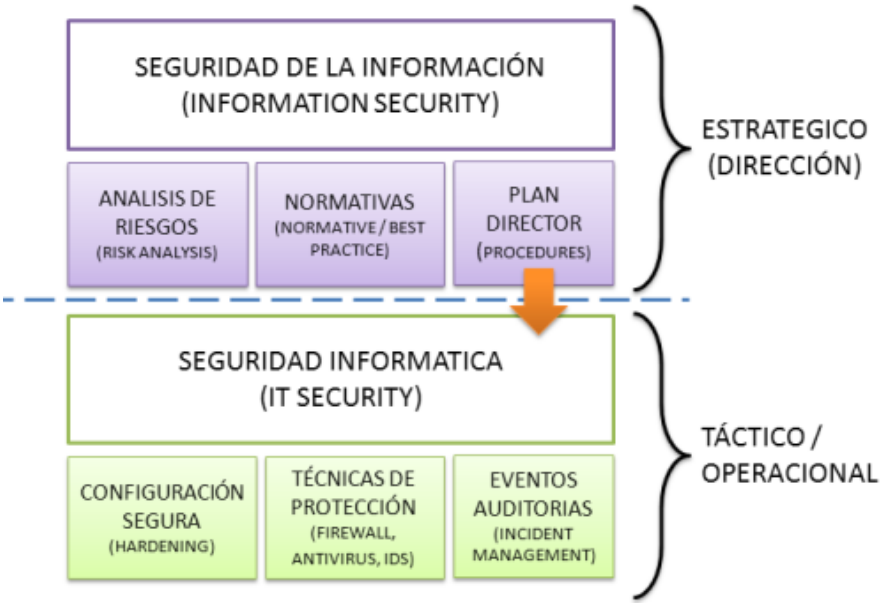
- **Confidencialidad.** La información debe ser salvaguardada y estar a disposición de una entidad o individuos autorizados únicamente.
- **Integridad.** Se debe velar por que la información, está completa a través de los métodos de consulta y su procesamiento; y esta no debe ser alterada por sujetos ajenos a la misma.
- **Disponibilidad.** Se garantiza el acceso oportuno, y la utilización de esta en los sistemas autorizados cuando esta sea requerida.

Los campos de la seguridad de la información se encuentran vinculados estrechamente con la seguridad informática y la ciberseguridad. Como lo comenta Catherine A. Theohary

(2015), para algunos la ciberseguridad es proteger de cualquier manera la infraestructura de información de un ataque cuyo alcance puede ser físico o electrónico respectivamente.

Tal y como lo muestra la Figura 1**,** una de las diferencias de la seguridad de la información frente a la seguridad informática, es el sentido estratégico de la misma, en cuanto a su alcance de gestión de análisis de riesgos, normatividad y procedimientos de acción. Por otro lado, está la seguridad informática que a nivel táctico tiene un enfoque desde el que hacer de la seguridad, basando su operación en la configuración, técnicas de protección en herramientas y eventos de seguridad.

Figura 1. Seguridad de la información – Estratégica.



Fuente. Tomado de (Etica, 2013).

La importancia de la seguridad de la información radica en la participación de todos los recursos con los que cuentan las organizaciones, ya que de acuerdo con los controles y análisis de riesgos que estén implementados, se puede lograr una gestión adecuada de la misma desde el gobierno en estas.

La seguridad de la información es fundamental como proceso mismo, en el que es integrada a la construcción de software, y esta requiere una ardua planificación bajo los fundamentos indicados en el presente capítulo, y espera generar consenso, aportando al cambio gradual de las prácticas de desarrollo de software seguro.

5.3. Desarrollo seguro

La seguridad debe estar implícita desde el surgimiento de la necesidad de un proyecto de desarrollo de software, además es un error postergarla a las demás etapas del ciclo de vida de desarrollo. No solo basta con tener implementados controles de seguridad informática como software antivirus, o incluso herramientas perimetrales como firewalls o controles que no satisfacen la premisa del desarrollo seguro (Becerra & Sanjuan, 2014). En un artículo publicado por la ACIS – Asociación Colombiana de Ingenieros de Sistemas, en el año 2019, se indica como en algunos países se orienta el desarrollo de software como un compromiso hacia la industria, y como se considera este como un producto clave en la seguridad nacional del estado. Uno de estos países es Estados Unidos, y este considera la alta dependencia de las tecnologías de la información y uso del software como eje central para la vida cotidiana de las personas y organizaciones de su nación; aquí se considera inaceptable las fallas del software como consecuencia del riesgo, que puede llegar a afectar la infraestructura nacional, la economía, la vida de las personas, la pérdida de confianza pública y la identidad de sus ciudadanos. Si bien es cierto que países como Colombia, no generan un volumen tan importante de desarrollo de software como lo abarca Estados Unidos, es importante reconocer la utilidad de los planes que se generan a nivel global sobre esta industria que cada vez toma más fuerza (Sguerra, 2019).

Los conceptos asociados a código seguro, están relacionados a controlar la seguridad en el desarrollo; y es claro que la calidad es un factor que debe incluir la seguridad y es un elemento de medición importante que las organizaciones y las naciones deben implementar, ya que en los productos terminados puede ser más costoso remediar un incidente que evaluarlo al inicio (Código, 2019).

Como lo comenta Sguerra (2019), un desarrollo de software que genere confianza tiene atributos orientados a garantizar fiabilidad, rendimiento, garantía, y supervivencia; esto enmarcado dentro de un amplio contexto de compromisos y adversidades basadas en niveles aceptables de riesgo. Así mismo, el estudio de este autor determina que una formación profesional y una intención continua por parte de las organizaciones, es una buena opción para satisfacer estas necesidades, que a su vez deben ser requeridas desde las fases de diseño, con métricas de calidad definidas, y evaluadas con metodologías de desarrollo seguro (Sguerra, 2019).

A continuación, se abordan algunos marcos de seguridad, y estándares internacionales que son de gran importancia para el desarrollo del presente proyecto.

- **ISO/IEC 27000.** Son estándares internacionales dados por ISO, y están enfocados en la gestión de la seguridad de la información, estos son utilizables por cualquier organización
- **ISO/IEC 15408:2009 CCF.** Esta más orientado a la realización y análisis de pruebas, así como a la evaluación de seguridad dentro de un marco definido de ISO
- **ISO/IEC 21827:2008 SSE-CMM.** Define un modelo que mide la madurez orientada a mejorar y evaluar toda la capacidad que debe ser implementada en la seguridad de la organización
- **IEEE P107074-2005.** Su enfoque va asociado a la gestión del software y a los ciclos de vida, priorizando los proyectos desde el enfoque de seguridad, así como la construcción de controles y protocolos

Así mismo, algunas metodologías con enfoque seguro permiten identificar los procesos clave y los pasos necesarios para la gestión de vulnerabilidades que ayudan a mitigar los riesgos, en los procesos de desarrollo. Para esto se recomienda altamente el uso de MAGERIT, pero aun así se describen otros modelos los cuales se especifican a grandes rasgos aquí.

- **CRAMM.** Es un método adoptado por el Reino Unido, va orientado al análisis y gestión de los riesgos de seguridad de la información
- **MAGERIT.** Es el método más usado por la administración española, es muy reconocido a nivel mundial y va orientado a al análisis y gestión de los riesgos de seguridad de la información
- **OCTAVE.** Es un método desarrollado por Estados Unidos, y busca realizar análisis de riesgos, es altamente aceptado a nivel internacional
- **OWASP.** Es un proyecto que tiene implícita una gestión de riesgos, e identifica el top 10 de estos, con el fin de crear conciencia de la seguridad en las aplicaciones

Es fundamental, tener una postura de seguridad de desarrollo, en la que, desde el ciclo de vida, y con metodologías como Scrum, se incluyan además estas normatividades dentro de las organizaciones, ya que permitirán identificar como están nuestros procesos al

interior de esta, y si estos cuentan con las regulaciones necesarias de los productos que se desarrollan y usan los clientes y usuarios de cada institución o empresa.

5.4. Metodologías ágiles en el desarrollo de software

Los ciclos de vida dan gran valor a lo que se desea desarrollar desde el punto de vista de calidad, pero realmente son las metodologías ágiles quienes pueden aportar al desarrollo de software seguro, por eso en el presente documento, se abordan dos de las metodologías con más auge en estos días, las cuales van orientadas a ser mucho más efectivas y dinámicas generando confiabilidad y disponibilidad a los productos de software desarrollado.

5.4.1. Agile

Actualmente, quienes hacen actividades de ingeniería y están en esta profesión prefieren en muchas ocasiones elegir alguna metodología tradicional frente a una ágil. Como introducción a estas, se orientan a proyectos en los que en muchas ocasiones se cuentan con tiempos muy cortos, así como costos reducidos; e incluso los requerimientos se basan en tecnologías que son emergentes y están asociadas a estas nuevas formas de trabajo.

Como lo comenta, (Duarte & A., 2008) en su documento evolución de las metodologías de desarrollo en la ingeniería, el termino ágil, se da por primera vez en los Estados Unidos, con el fin de aplicarse al desarrollo de software, sobre los ciclos de vida en calidad, principios y valores que permiten a los equipos desarrollar software de manera mucho más rápida, respondiendo a todos los cambios que puedan llegar a darse como consecuencia de los entregables periódicos que se dan a los usuarios.

Dado lo anterior se creó el llamado manifiesto ágil, en el que luego de varias sesiones en las que se trataron los temas actuales de las metodologías existentes, se llegó a la determinación de que se estaban considerando muy pesadas, y rígidas frente a los cambios que se dieron en ese momento en el año 2001; además que consideraban una dependencia normativa y de planificación bastante amplias, lo que retrasaba de gran manera los procesos de las fases de los ciclos de vida actuales para ese momento.

Valores y Principios. Así mismo, el manifiesto ágil se fundamentó en:

- Las personas son lo más importante dentro de los equipos de trabajo, y son el principal insumo con el que se debe contar en un proyecto, además son fundamentales para que este cuente con éxito. Incluso, se debe crear primero los equipos de trabajo, y luego con este ya conformado construir el entorno de cómo serán las interacciones, las herramientas y funciones a desarrollar sobre estos.
- Se debe entregar software que funcione, en conseguir que lo que se está haciendo sea útil para el cliente, se basa también en documentar lo fundamental, y en realizar esta tarea después de finalizado los desarrollos propuestos.
- Más que fijarse en pactar o hacer cumplir un contrato, se debe colaborar con los clientes y usuarios, y establecer interacciones constantes para que la marcha del proyecto sea exitosa.
- Responder a los cambios de manera ágil y rápida, no se trata de seguir al pie de la letra todo como se pactó, sino más bien estar atentos a lo nuevo, a lo que genere valor para adaptarlo al proyecto.

De los valores nombrados anteriormente nacen los principios los cuales son 12, y determinan los procesos orientados al logro de la metodología ágil.

1. Satisfacer continuamente al cliente, mediante entregas continuas y tempranas, con el fin de aportar valor al software y a las necesidades de los requerimientos

2. Ser abiertos a los cambios, para dar ventajas competitivas a los clientes

3. Reducir considerablemente los tiempos entre entregas de software

4. Trabajar de la mano con los responsables de los procesos para alinear la estrategia del negocio, con el desarrollo

5. Motivar al equipo de trabajo, darles confianza y contar con ellos para alcanzar el éxito

6. Dialogar abiertamente con los miembros del equipo, sobre lo que está pasando actualmente, con el fin de conseguir información oportuna y efectiva

7. Si el software que se entrega es funcional, dará progreso al proyecto

8. Se consideran desarrollos sostenibles los proyectos orientados con la metodología ágil

9. La calidad es un factor determinante para promover técnicas adecuadas de desarrollo

10. Ser simple es lo mejor y da agilidad

11. Ser organizados surgen de las mejores ideas y de la organización

12. Ser reflexivo, y crítico con uno mismo ayuda a ser mucho más efectivo en los equipos de trabajo

Basado en el documento evolución de las metodologías de desarrollo en la ingeniería, se realiza un comparativo entre metodologías tradicionales vs ágiles, en la Tabla 1.

Tabla 1. Comparación metodologías Tradicionales vs Ágiles

Metodologías Tradicionales	Metodologías Ágiles
Se caracteriza por sus normas y estándares, dados por los entornos de desarrollo	Son basadas en métodos de aprendizaje prácticas y ágiles para la producción del código
Estigmatiza de gran manera cambios repentinos	Siempre están preparados para los cambios durante los proyectos
El trabajo siempre será impuesto por reglas a nivel externo	El equipo de trabajo se orienta bajo sus propias reglas
La ejecución de sus procesos se hace de manera más controlada	Los procesos son menos controlados, pero con más principios
Siempre existen contratos previamente establecidos	Los contratos cuentan con flexibilidad, por la respuesta al cambio que se da en esta modalidad
No existen reuniones esporádicas entre el cliente y los equipos de trabajo	Los clientes son parte activa de los equipos de desarrollo
Se dan grupos grandes de desarrollo, y distribuidos en tareas distintas	Son grupos pequeños de menos de diez integrantes
Hay muchos más artefactos	No hay tantos artefactos
Hay muchos más roles	No hay tantos roles
El software es basado en modelos de arquitecturas	No se enfatiza en la arquitectura del software

Fuente. Tomado y adaptado de (Duarte & A., 2008).

5.4.2. Scrum

Es una metodología que es usada fundamentalmente para trabajar en equipo, en el que se da repeticiones o iteraciones que son llamadas ciclos *(Sprints)*. El objetivo principal de esta es saber controlar, y planificar proyectos que contienen un gran volumen de cambios que pueden ser resueltos o reprogramables con una incertidumbre bastante grande. Esta se centra realmente en ajustar y saber responder a las expectativas de los clientes, con el fin de afinar mediante el paso y la entrega de productos terminados, las necesidades identificadas al principio del proyecto, en el análisis de requerimientos.

Basados en las teorías de control de procesos empíricos, Scrum lleva a cabo una serie de ciclos breves para los pequeños proyectos llamados ciclos *(Sprints)*, cada uno de estos tiene un tiempo determinado y limitado, y para estos, se tiene que en cada finalización se haya entregado una funcionalidad mayor a la del anterior ciclo *(Sprint)*. Para el cumplimiento de estos ciclos *(Sprints)*, se crean equipos de trabajo conformados, en los que, mediante roles, artefactos, eventos, y reglas se relacionan entre sí, para conformar lo que se llama el marco de trabajo. Los equipos de trabajo están compuestos por dueños de producto, equipos de desarrollo y responsables de asegurar que todo sea aplicado y entendido para los grupos.

En los ciclos *(Sprints)* se tienen una serie de actividades determinadas, las cuales se detallan a continuación:

- Se planifican reuniones en los que los ciclos *(Sprints)* son presentados a los equipos de trabajo, y es aquí donde se inician estos proyectos. Así mismo se marcan los objetivos, se organiza el equipo para todas las actividades grupales
- Se hacen reuniones de seguimiento, aproximadamente de 15 a 20 minutos, estas pueden llegar a ser diarias según la disponibilidad de todo el equipo, o en su defecto se hacen según se requieran con una planeación dada. Se plantean preguntas asociadas a conocer el estado de los proyectos, la participación de los miembros de los equipos, el desarrollo y consecución de los objetivos planteados, y más importante aún los avances relacionados e impedimentos de cada etapa en específico
- Al final se evalúa el incremento o avance general de los proyectos

- Por último, se hace una reunión de ingeniería inversa del ciclo *(Sprint)* en específico, que busca evaluar todo el trabajo realizado, y elaborar un plan de mejora para el nuevo que va a empezar.

Los ciclos *(Sprints)* como se nombró anteriormente son considerados proyectos cortos, que se utilizan para responder a necesidades específicas y como fin lograr algo. Cada ciclo *(Sprint)* tiene un plan de trabajo definido, y tiene cinco etapas en las que desarrolla su trabajo.

- **Reunión Planificación.** Tiene una duración determinada de acuerdo con el alcance del proyecto, y en este se define las funcionalidades, y los objetivos del desarrollo que se alcanzaran con este ciclo *(Sprint)*, así mismo se responde interrogantes que están relacionados a ¿que será entregado? y ¿cómo se realizara el trabajo?
- **Scrum Diario.** Son reuniones de 15 minutos como mínimo en el que los equipos de trabajo intercambian ideas, actividades y establecen nuevos planes para ser ejecutados en las próximas 24 horas o más si llega a decidirse así. Estos Scrum diarios tienen muchas ventajas ya que permiten mejorar de manera significativa las comunicaciones que se deben dar entre los participantes y además ayuda a acelerar y promover el conocimiento entre los equipos de desarrolladores
- **Trabajo de Desarrollo.** Existen una serie de actividades que se deben asegurar durante cada ciclo *(Sprint)*, y es no realizar cambios significativos que hagan retrasar el proyecto, no cambiar de rumbo ni disminuir los objetivos, ni mucho menos la calidad; y si es necesario dar un nuevo alcance a los objetivos iniciales o al proyecto negociado con el cliente usuario. Una de las características más importantes de los ciclos *(Sprints)*, es que ayudan a alcanzar metas en un tiempo determinado
- **Revisiones.** Estas se llevan a cabo al final de cada ciclo *(Sprint)*, y sirven para realizar inspecciones a los productos entregados, así mismo se valida lo que se denomina la lista de producto *(Product Backlog)*, esta se hace de manera informal y se destina a la presentación de retroalimentación de lo concluido por el equipo de trabajo

- **Retrospectiva.** En este punto se crean planes de mejoras orientados a revisar, identificar y crear planes que ayuden a mejorar procesos, potencializar el equipo y ser más eficientes en el trabajo desarrollado

La metodología ágil Scrum es aplicable en principio a cualquier tipo de proyecto, pero su creación y aplicabilidad está orientada a proyectos en los que se involucre desarrollo de software en lo que se denomina entornos complejos. Ya que garantiza según la naturaleza de este, su estructura, organización, y los entregables que son propios de cada una de las fases de las que se componen los ciclos *(Sprints)*.

6. Metodología

La metodología formulada busca cumplir con el objetivo general y los objetivos específicos que se plantearon en el trabajo de grado; así como aplicar el método científico en el que se realiza la observación, el reconocimiento del problema, se genera una hipótesis y predicción, se hace una experimentación o validación, y se analizan y comunican los resultados de los hallazgos.

El enfoque utilizado en esta investigación para la creación del marco de buenas prácticas bajo la metodología Scrum, para desarrollo de código seguro, es cualitativo tomando como referencia que esta se enfoca en el diseño y construcción de un marco de buenas prácticas que busca fortalecer el desarrollo de software, comprendiendo las actividades técnicas, recursos, y significado de generar aplicaciones seguras, que ayuden a mitigar los riesgos, fallos y vulnerabilidades a los que están expuestos usuarios, organizaciones e instituciones; debido a que el tema de estudio ha tomado importancia por el alto número de delitos informáticos e incidentes de seguridad que se han presentado en los últimos años (Hernández & Baptista, 2014).

El tipo de investigación se realiza como descriptiva, ya que se plantea una solución al problema, y se identifican las situaciones objeto de estudio. Tal y como lo indica Hernández & Baptista (2014), estos estudios describen situaciones, contextos y sucesos, que buscan especificar las características del fenómeno que se someten a un análisis. La población identificada son organizaciones e instituciones que, dentro de sus procesos y servicios, desarrollen software.

Para la validez del estudio se realiza mediante el juicio de expertos, que consiste en solicitar a un grupo de personas el juicio sobre un instrumento, material o marco en específico para su opinión en concreto. La técnica de recolección de datos se realiza mediante un cuestionario o planilla, que evaluara los objetivos de la investigación desde el juicio de expertos. La Figura 2 muestra la metodología de investigación por juicio de expertos con más profundidad.

Figura 2. Metodología de investigación – Juicio de expertos.

Fuente. Tomado y adaptado de (Pérez & Martínez, 2008).

7. Vulnerabilidades, fallos y riesgos en aplicaciones

En el año 2011, Marc Andreessen, creador del primer navegador gráfico *(Mosaic)* (Menn, 2000), escribió un artículo en el *Wall Street Journal* que incluía la famosa frase *Why Software Is Eating The World*, ("Porque el software se está comiendo el mundo") (Andreessen, 2011). Ocho años después, esa declaración suena más verdadera que nunca. No es exagerado decir que el software se está comiendo el mundo de la ciberseguridad también.

En la actualidad nos movemos y trabajamos en una era globalizada, con más tecnología que hace algunos años, pero lo que representa hoy en acceso a datos e información, también nos hace vulnerables desde el punto de vista informático y de privacidad. El uso de programas y herramientas ha superado nuestras expectativas y ha llegado a un nivel de sofisticación en el que día a día, se generan nuevas y mejores.

7.1. Vulnerabilidades y fallos en aplicaciones

Según el informe del año 2019 de Veracode, (organización encargada de realizar análisis estático y mejorar la seguridad de las aplicaciones), *State of Software Security Vol. 10*, el 83% de las 85.000 aplicaciones que fueron probadas por esta herramienta tenían al menos un fallo a nivel de seguridad, y el numero promedio de días usados para la remediación de estas era de 171 aproximadamente, lo cual indica que resolver los hallazgos no ocurre de una manera rápida; así mismo el 20% de esas aplicaciones tuvieron un fallo grave en seguridad (Veracode, 2019).

Otra cifra que resulta alarmante en el informe, indica que en promedio de 2 a 3 aplicaciones fallan las pruebas iniciales de seguridad, esto basándose en las reconocidas metodologías de desarrollo de código seguro y en el análisis estático realizado al código fuente. En cuanto a cumplimiento, el 68% de las aplicaciones no pasaron *OWASP*, y el 67% SANS, (OWASP, Una guía para integrar seguridad en el desarrollo de software, 2009) (SANS, 2011). Así mismo está comprobado, que cuanto más tiempo permanezca una

vulnerabilidad o defecto en las aplicaciones, la acumulación de estas generara un tiempo mayor en la remediación y una degradación en la prestación del servicio a largo plazo.

Si bien es cierto, que existen un sin número de aplicaciones móviles, web, basadas en red y para distintos dispositivos, la seguridad es un factor primordial que debemos evaluar como usuarios, y como dueños de nuestros datos e información.

Según lo investigado y publicado en su informe, los principales tipos de fallos más comunes en aplicaciones a nivel general son:

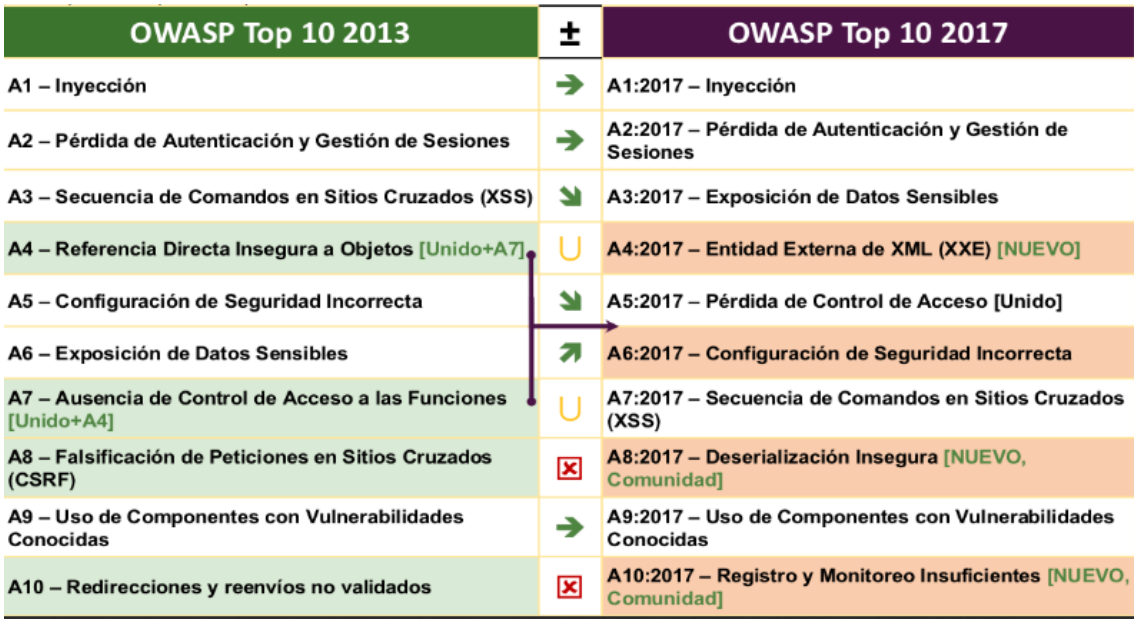
- Fuga de información **(64%)**
- Problemas criptográficos **(62%)**
- Inyección de CRLF **(61%)**
- Código de calidad **(56%)**
- Insuficiente validación de entrada **(48%)**
- Secuencias de comandos entre sitios **(47%)**
- Recorrido del directorio **(46%)**
- Gestión de credenciales **(45%)**

En enero de 2020 Imperva (empresa de servicios y software de seguridad cibernética), público un informe que muestra en el año 2019, el aumento de las vulnerabilidades en aplicaciones web, en un 17,6% (20.362) en comparación con 2018 (17.308), y un 44,5% en comparación con 2017 (14.086). Así mismo el CVSS *(Common Vulnerability Scoring System)* clasifico estas vulnerabilidades, con el 8% como gravedad baja, el 61% como media, mientras que entre el 18 y el 13% fueron de gravedad alta y critica respectivamente. (Dima Bekerman, 2020). Según este informe de Imperva casi la mitad de las vulnerabilidades (47%), tienen una explotación *(exploit)* pública, el cual desafortunadamente está disponible para cualquier atacante informático. Además, el 40,2% de estas vulnerabilidades no tienen una solución disponible, una actualización de software o un parche de seguridad.

7.2. Riesgos en seguridad en aplicaciones

La comunidad en línea que produce artículos, documentos, metodologías y herramientas en el campo de la seguridad en aplicaciones web *OWASP*, ha publicado los 10 principales riesgos y vulnerabilidades de seguridad desde el año 2003, esta es actualizada cada tres o cuatro años, y su última versión fue en el año 2017. Tal y como lo muestra la Figura 3, estos son los riesgos más críticos en aplicaciones web. (OWASP, OWASP Top Ten, 2017).

Figura 3. OWASP Top 10 – 2013 vs 2017 Los diez riesgos más críticos en Aplicaciones Web.



OWASP Top 10 2013	±	OWASP Top 10 2017
A1 – Inyección	→	A1:2017 – Inyección
A2 – Pérdida de Autenticación y Gestión de Sesiones	→	A2:2017 – Pérdida de Autenticación y Gestión de Sesiones
A3 – Secuencia de Comandos en Sitios Cruzados (XSS)	↘	A3:2017 – Exposición de Datos Sensibles
A4 – Referencia Directa Insegura a Objetos [Unido+A7]	U	A4:2017 – Entidad Externa de XML (XXE) [NUEVO]
A5 – Configuración de Seguridad Incorrecta	↘	A5:2017 – Pérdida de Control de Acceso [Unido]
A6 – Exposición de Datos Sensibles	↗	A6:2017 – Configuración de Seguridad Incorrecta
A7 – Ausencia de Control de Acceso a las Funciones [Unido+A4]	U	A7:2017 – Secuencia de Comandos en Sitios Cruzados (XSS)
A8 – Falsificación de Peticiones en Sitios Cruzados (CSRF)	☒	A8:2017 – Deserialización Insegura [NUEVO, Comunidad]
A9 – Uso de Componentes con Vulnerabilidades Conocidas	→	A9:2017 – Uso de Componentes con Vulnerabilidades Conocidas
A10 – Redirecciones y reenvíos no validados	☒	A10:2017 – Registro y Monitoreo Insuficientes [NUEVO, Comunidad]

Fuente. Tomado de (OWASP, OWASP Top Ten, 2017).

A continuación, se describe cada uno de los 10 riesgos y las posibles implicaciones de las víctimas y los atacantes.

- **A1:2017 - Inyección.** Las fallas de inyección de código pueden darse en SQL, NoSQL, OS o LDAP, estas normalmente ocurren cuando se envían datos no confiables a un intérprete, como parte de un comando o consulta. Los datos maliciosos de los atacantes engañan al interprete para ejecutar comandos involuntarios conseguir acceder a información restringida, modificar valores en bases de datos, instalar programas maliciosos, subir privilegios y atacar páginas web, entre otros.

Puede causar divulgación, pérdida o corrupción de información, perdida de auditabilidad, o denegación de acceso y servicio.

- **A2:2017 - Pérdida de Autenticación.** Las fallas de funciones de autenticación y gestión de sesiones de usuarios, al ser implementadas de manera incorrecta, comprometen a las contraseñas, *token* de sesiones, y permiten explotar otras fallas con el fin de asumir la identidad de otros usuarios. Esto se puede dar de manera temporal o permanente. Un atacante al obtener acceso a una cuenta de administrador puede llegar a comprometer el sistema.

  Puede causar de acuerdo con el nivel de compromiso robo de identidad, lavado de dinero, divulgación de información sensible, perdida intelectual de datos, compromiso total al sistema involucrado.

- **A3:2017 - Exposición de Datos Sensibles.** Uno de los principales riesgos está asociado a que muchas aplicaciones web y *APIs* no generan la protección suficiente de información sensible como a: datos personales, contraseñas, correos electrónicos, datos financieros, médicos entre otros. Estos datos e información son usados por los atacantes con el fin de robarla y cometer distintos tipos de fraude.

  Puede causar de acuerdo con el nivel de compromiso robo de identidad, divulgación de información sensible, fraudes es distintas entidades, perdida intelectual de datos, entre otros.

- **A4:2017 - Entidades Externas XML (XXE).** La inyección de código ocurre en las entidades, estos son lugares que almacenan cualquier tipo de dato, o en programación se conoce como variable. Estas entidades tienen acceso a contenidos locales o remotos, e incluso a *URIs* (identificador de recursos uniforme) cuando son procesadas. Estas vulnerabilidades explotan estos accesos de tal manera que un atacante puede acceder a ficheros del sistema o incluso hacer peticiones determinadas a la máquina, con el fin de escanear puertos en redes LAN, o ataques de denegación de servicio entre otros.

Puede causar divulgación, perdida o corrupción de información, ataques de denegación de servicio, enumeración de puertos abiertos en un escaneo, exfiltración de credenciales, entre otros.

- **A5:2017 - Pérdida de Control de Acceso.** Las restricciones que son implementadas en las aplicaciones, a nivel de usuarios autenticados se ha identificado que son limitadas, lo que puede permitir a un atacante tomar control de otros usuarios, suplantar información, obtener cuentas de administrador, sabotear los registros, modificar el sistema al que tiene acceso, y cambiar derechos y permisos.

  Puede causar acceso indebido a servidores, servicios, información confidencial y sensible, enumerar roles de usuario, enviar ataques de fuerza bruta, referenciar objetos y variables susceptibles a manipulación, y fraude, entre otros.

- **A6:2017 - Configuración de Seguridad Incorrecta.** Los problemas de configuraciones por defecto pueden comprometer un sistema. La configuración de seguridad es un problema común que se establece de hacerlo de manera manual y ocurren por omisión o falta de políticas de aseguramiento. Algunos ejemplos son cabeceras HTTP erróneas, mensajes de error con información sensible, falta de actualizaciones, e implementación de plantillas de aseguramiento.

  Puede causar acceso no autorizado a datos, funciones y errores del sistema, el impacto depende de la aplicación y el nivel de compromiso del ataque.

- **A7:2017 - Cross-Site Scripting (XSS).** Ocurren cuando aplicaciones toman datos no confiables y estos son enviados al navegador web sin validación y codificación apropiada. Para esto un atacante inyecta una secuencia de comandos *(script)* usualmente de *Javascript*, en la salida de una aplicación web, permitiendo ejecutar comandos en el navegador de la víctima, con el fin de secuestrar la sesión del usuario, o redireccionarlo a sitios maliciosos.

  Puede causar robo de credenciales, secuestro de sesiones de usuario de determinada aplicación, instalación de software malicioso, redirección a sitios no confiables, entre otros.

- **A8:2017 - Deserialización Insegura.** La serialización es la codificación de un objeto en secuencias de bytes, con el fin de almacenarlo o transmitirlo. Esta falla de seguridad permite al atacante realizar repeticiones, inyecciones o elevar permisos de ejecución en la aplicación comprometida. En algunos casos este fallo de seguridad permite la ejecución remota de código en los servidores.

  Puede causar perdida de datos, funciones y errores del sistema, el impacto depende de la aplicación y el nivel de compromiso del ataque.

- **A9:2017 - Uso de Componentes con Vulnerabilidades Conocidas.** Varios componentes como bibliotecas, módulos y marcos de referencia *(frameworks)* son el foco de los atacantes, esto debido a que se ejecutan con los mismos privilegios de las aplicaciones. Si uno de estos componentes contiene alguna vulnerabilidad conocida el atacante puede tomar control sobre el servidor y sus aplicaciones.

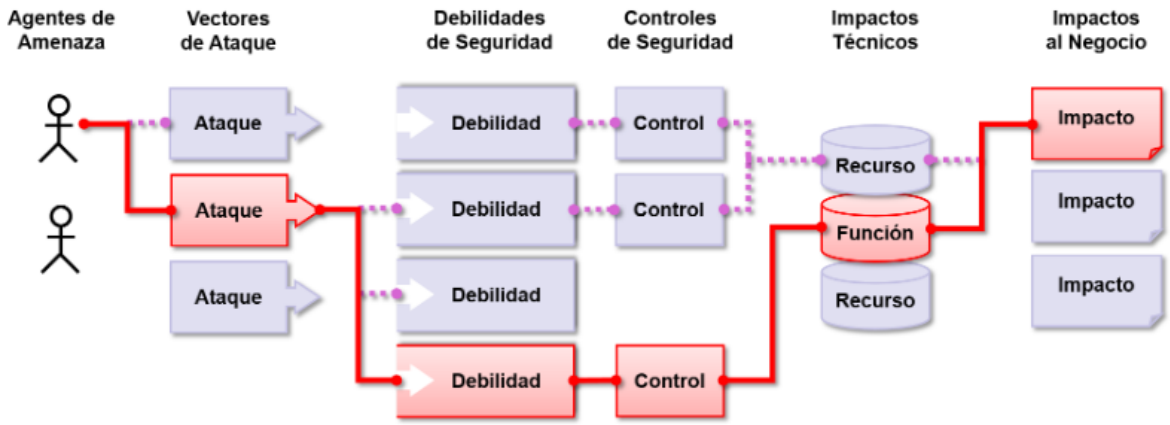
  Puede causar robo de identidad, divulgación de información sensible, perdida intelectual de datos, compromiso total al sistema involucrado, entre otros.

- **A10:2017 - Registro y Monitoreo Insuficientes.** Un inexistente monitoreo o registro de eventos en las aplicaciones, y acciones que ejecutan los usuarios es común, y es catalogado como un fallo severo. Un atacante puede mantenerse un tiempo determinado sin ser detectado, y tomar control de sistemas para manipularlos, extraer o destruir datos e información. Los estudios demuestran que la detección de brechas de seguridad y compromiso es mayor a 200 días.

  Puede causar sondeo de vulnerabilidades y aumentar las probabilidades de éxito de los atacantes, así como pérdida de datos, funciones y errores del sistema, ex filtración de información, entre otros.

Desde un punto de vista teórico, siempre existe un riesgo inherente en el uso de las aplicaciones, el cual un atacante puede materializar, usando distintas rutas, métodos y medios, para perjudicar una organización, o incluso comprometer la información de los usuarios de esta. Cada uno de estos caminos tal y como lo muestra la Figura 4, puede llegar a ser o no lo suficientemente grave, pero estos lo pueden saber aprovechar para obtener el beneficio que desean (OWASP, OWASP Top Ten, 2017).

Figura 4. OWASP - Riesgos en seguridad de aplicaciones.



Fuente. Tomado de (OWASP, OWASP Top Ten, 2017).

Otras debilidades comunes encontradas en el software se encuentran en el *CWE (Common Weakness Enumeration),* la cual lista cuales son las más peligrosas, comunes e impactantes. Estas son consideradas debilidades porque son fáciles de encontrar, explotar y permiten a los atacantes comprometer un sistema, robar datos e información e impedir que una aplicación funcione correctamente (CWE, 2020), algunas para tener en cuenta son:

Tabla 2. CWE - Las 25 debilidades más peligrosas del software 2020.

Rango	ID	Nombre
[1]	CWE-79	*Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*
[2]	CWE-787	*Out-of-bounds Write*
[3]	CWE-20	*Improper Input Validation*
[4]	CWE-125	*Out-of-bounds Read*
[5]	CWE-119	*Improper Restriction of Operations within the Bounds of a Memory Buffer*

[6]	CWE-89	*Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')*
[7]	CWE-200	*Exposure of Sensitive Information to an Unauthorized Actor*
[8]	CWE-416	*Use After Free*
[9]	CWE-352	*Cross-Site Request Forgery (CSRF)*
[10]	CWE-78	*Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*
[11]	CWE-190	*Integer Overflow or Wraparound*
[12]	CWE-22	*Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')*
[13]	CWE-476	*NULL Pointer Dereference*
[14]	CWE-287	*Improper Authentication*
[15]	CWE-434	*Unrestricted Upload of File with Dangerous Type*
[16]	CWE-732	*Incorrect Permission Assignment for Critical Resource*
[17]	CWE-94	*Improper Control of Generation of Code ('Code Injection')*
[18]	CWE-522	*Insufficiently Protected Credentials*
[19]	CWE-611	*Improper Restriction of XML External Entity Reference*
[20]	CWE-798	*Use of Hard-coded Credentials*
[21]	CWE-502	*Deserialization of Untrusted Data*
[22]	CWE-269	*Improper Privilege Management*
[23]	CWE-400	*Uncontrolled Resource Consumption*
[24]	CWE-306	*Missing Authentication for Critical Function*

[25]	CWE-862	*Missing Authorization*

Fuente. Tomado de Common Weakness Enumeration (CWE, 2020).

Es importante reconocer las vulnerabilidades, defectos y amenazas que pueden comprometer los activos de información, datos en la organización y aplicaciones expuestas en internet. Tanto así, como preocuparse en quien puede explotarlas, y como se va a beneficiar de nuestra información.

Actualmente existen fuentes externas en seguridad, como *OWASP*[4], *CVE*[5], y el *CWE*[6] entre otras, que se ocupan de identificar y categorizar este tipo de amenazas y vulnerabilidades que se registran en el mundo. Estas se pueden utilizar como fuente de información de primera línea, para tener un acercamiento de cómo es su modo de operación, nivel de ataque y compromiso, y como se puede llegar a mitigar; esto con el fin de estar preparados y que el software que se desarrolle sea seguro, incluso antes de empezar su etapa de diseño y codificación. Si bien es conocido, las amenazas son usualmente externas a los activos, pero estas no provienen necesariamente de fuera de las organizaciones, de hecho, la mayoría de los incidentes de seguridad se originan dentro del perímetro de estas. (Excelencia, 2019).

8. Análisis de las principales metodologías de desarrollo de software seguro

Con el fin de identificar los elementos necesarios para diseñar un marco de buenas prácticas para desarrollo de código seguro, se ha decidió investigar tres de las principales metodologías, las cuales son reconocidas por su aplicabilidad y funciones de negocio particulares relacionadas con el desarrollo de software seguro. Estas metodologías son: *SAMM - Software Assurance Maturity Model, BSIMM - Building Security In Maturity Model y Microsoft Security Development Lifecycle*, así mismo, los factores que se determinaron

[4] OWASP. (30 de noviembre de 2017). OWASP Top Ten. Obtenido de https://owasp.org/www-project-top-ten/
[5] CVE. (18 de septiembre de 2020). News & Events. Obtenido de https://cve.mitre.org/news/archives/2020/news.html
[6] CWE. (20 de agosto de 2020). 2020 CWE Top 25 Most Dangerous Software Weaknesses. Obtenido de https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html

para investigar sobre estas fueron: rapidez en su implementación y despliegue, generación de estándares de cumplimento, aplicación en organizaciones del sector real, reducción de vulnerabilidades en aplicaciones comerciales conocidas, y contribución a los modelos de TI.

Uno de los tres principales modelos es *SAMM - Software Assurance Maturity Model*, el cual fue escogido por ser un referente internacional en la evaluación de seguridad en la construcción de aplicaciones, es reconocido por ser desarrollado en el 2009 por *OWASP - Open Web Application Security Project.* Este incluye el mapeo de estándares existentes como ISO, PCI, COBIT e ISM3; es rápido, fácil de desplegar y ayuda a mejorar el entorno de seguridad y el estado de la construcción de software. Se caracteriza por ser personalizable y adaptable en las organizaciones, según las necesidades y los niveles de seguridad que se deseen alcanzar. Contiene alrededor de 100 organizaciones las cuales han compartido su experiencia y han contribuido de manera abierta a la gestión de conocimiento y de evolución del modelo, entre los que se encuentran expertos en seguridad, desarrolladores, arquitectos, gerentes de TI, entre otros (Cornell, 2017).

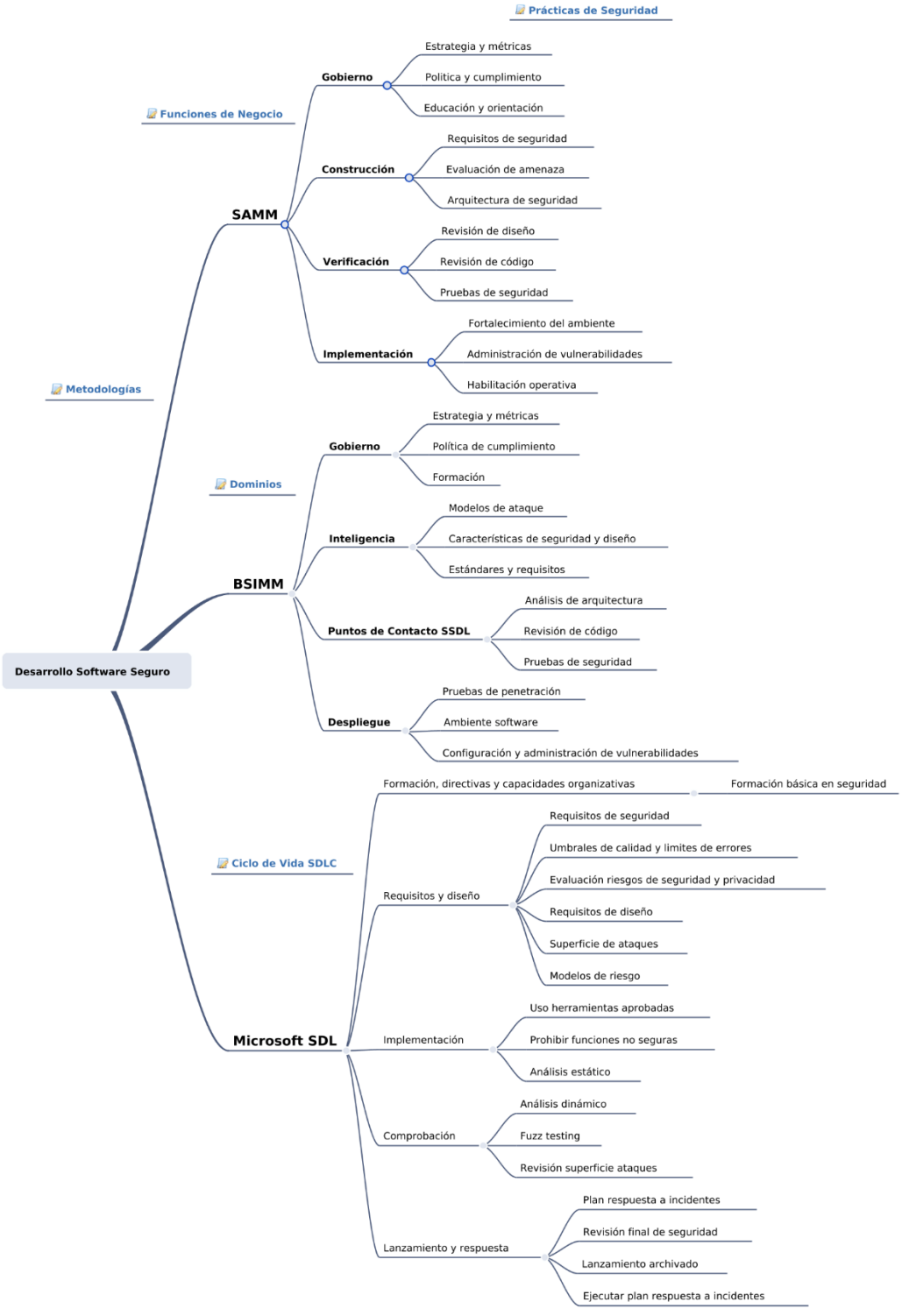
Por su parte *BSIMM - Building Security In Maturity Model*, es una de las principales metodologías de desarrollo de código seguro orientada a medir la seguridad del software, la cual es considerada como un modelo descriptivo y una de las primeras herramientas de seguridad que basa su uso en datos reales de las organizaciones, para la décima versión, tiene un total de 122 empresas que han aplicado su modelo a nivel mundial, y estas permiten tener un comparativo de gran nivel, de lo que las organizaciones deben hacer (News, 2016). En términos generales *BSIMM* destaca en su modelo, como hacen y planean la seguridad en sus organizaciones, a tal punto que uno de los gigantes de tecnología como Huawei, opto por adaptarlo desde el año 2013, dentro de sus áreas y modelos de negocio. Luego de esto Huawei creo lo que se llama el *SDP Track (Security Development Platfrom)*, la cual permite monitorear las distintas actividades de negocio, en los procesos de desarrollo a través de seguridad y gestión de seguridad, con un mayor nivel de comprensión y seguimiento sobre sus nuevos productos (Sougth, 2018).

La importancia de escoger el *Microsoft Security Development Lifecycle*, radica en la alta reducción de las vulnerabilidades que ha impactado de manera positiva la seguridad a sus productos desde el año 2009. Algunas cifras importantes asociadas a este modelo son: la reducción del 91% de las vulnerabilidades identificadas en productos de SQL Server, las

cuales son remediadas en tan solo unos meses luego de la liberación de los productos al mercado; así mismo, en cuanto a la familia Windows de sistemas operativos, en tan solo un año del lanzamiento se redujo el 45% de las vulnerabilidades entre una versión y otra. En su navegador *Internet Explorer* no fue la excepción, también en tan solo un año, redujo el 35% de las vulnerabilidades, y además el 63% de estas fueron categorizadas con una severidad alta, lo cual demuestra la validez de su aplicación (Sullivan, 2009). Este modelo tiene una aplicabilidad universal orientada a los sistemas operativos, lenguajes de programación, distintos escenarios de desarrollo, y una alta tasa de adopción de metodologías ágiles, destacando entre estas a Scrum dentro del primer lugar.

A continuación, en la Figura 5**,** se muestra un mapa mental, con el fin de dar un acercamiento al entorno general de las metodologías de desarrollo de software seguro que serán analizadas en el presente capitulo, en este se detallan los dominios, las funciones de negocio, y las prácticas de seguridad que abarca cada una.

Figura 5. Mapa mental metodologías seleccionadas.



Fuente. Elaboración propia.

8.1. SAMM - Software Assurance Maturity Model

El *SAMM*, o modelo de madurez de aseguramiento de software fue diseñado, desarrollado y escrito en el año 2009, por Pravir Chandra un consultor de seguridad independiente. La escritura y creación del primer borrador (v1.0), se dio gracias a los fondos destinados de la compañía *Fortify Software*, este proyecto se ha convertido en parte del reconocido proyecto abierto de seguridad en aplicaciones web (OWASP, Una guía para integrar seguridad en el desarrollo de software, 2009). Fue diseñado originalmente como un marco de trabajo abierto, cuyo objetivo es ayudar a las organizaciones a implementar una estrategia de seguridad de software adecuada, de acuerdo con las necesidades específicas que pueden enfrentar hoy en día las organizaciones.

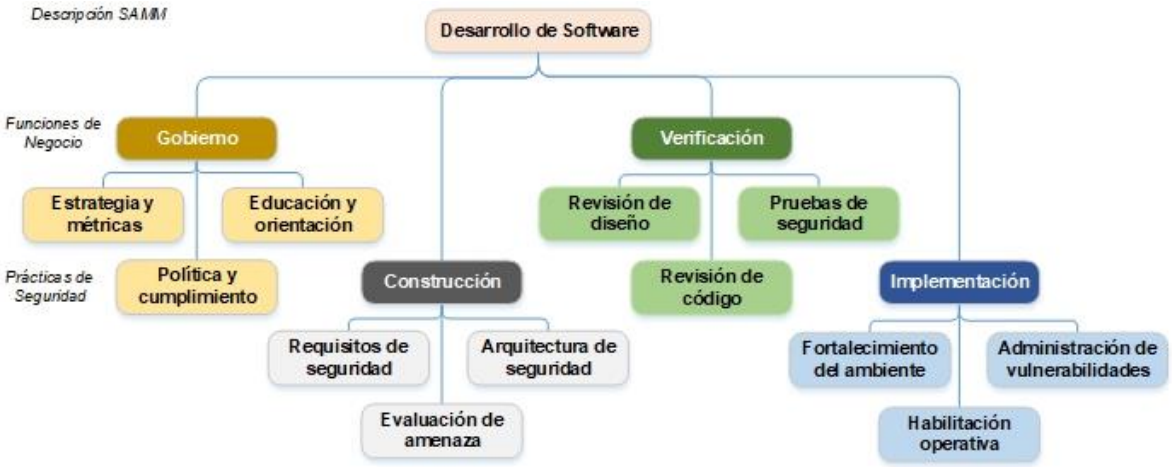
SAMM funciona de manera flexible, y puede llegar a ser implementado en organizaciones pequeñas, medianas o grandes que tengan en su operación cualquier metodología o estilo de desarrollo. Su aplicabilidad está dada en la organización, por líneas de negocio o en proyectos determinados. Esta metodología fue construida bajo los siguientes principios:

- Cambios en las organizaciones a través del tiempo. El establecimiento de un programa de seguridad para software debe generar entregas en ciclos pequeños como en las metodologías ágiles actuales, y considerar metas alcanzables a largo plazo.
- El funcionamiento en cada organización es particular y diferente. Este marco fue construido para ser flexible y adaptarse a las organizaciones basándose en su tolerancia al riesgo y su manera de trabajo actual.
- Las actividades de seguridad son específicas, tanto así, como sus lineamientos y relaciones. Todos los pasos construidos bajo esta metodología deben tener la característica de ser sencillos, definidos, medibles y sostenibles a través del tiempo en la organización.

Tal y como lo muestra la Figura 6, *SAMM* ha definido cuatro niveles de negocio, asociados a: Gobierno, Construcción, Verificación e Implementación. Cada categoría se encuentra relacionada a tareas específicas de la práctica de desarrollo de software. Para cada función de negocio, este define tres prácticas de seguridad, las cuales constituyen actividades de aseguramiento para las funciones relacionadas. Así mismo, para cada práctica de seguridad *SAMM*, ha definido tres niveles de madurez, pero como objetivos. Estos

objetivos caracterizados por las prácticas de seguridad definen actividades específicas mayores y exigentes de acuerdo con las métricas de éxito definidas en el nivel anterior.

Figura 6. SAMM - Esquema descripción.



Fuente. Tomado de Software Assurance Maturity Model (OWASP, Una guía para integrar seguridad en el desarrollo de software, 2009).

Al nivel más alto dentro de *SAMM*, las cuatro funciones de negocio tienen un enfoque determinado dentro de los procesos y actividades, en la implementación de la estrategia como metodología de desarrollo seguro en la organización.

- **Gobierno.** Asociado al gobierno de TI, enfocado en procesos y actividades de gestión al interior de la organización, incluye aquí la preocupación del desarrollo de software de manera global, y su alineación al negocio y sus actividades.

- **Construcción.** Refiere a los procesos y actividades definidas a través de la gestión de productos, requisitos de seguridad y especificaciones de alto nivel de arquitectura.

- **Verificación.** Se enfoca en los procesos y actividades típicamente producidos, en el desarrollo de software, tanto, así como el aseguramiento de calidad y las pruebas de revisión y evaluación.

- **Implementación.** Abarca los procesos y actividades en la liberación de productos que han sido creados en la organización, como el envío de paquetes finales a los usuarios e instalación en ambientes de ejecución.

En la Tabla 3, se describe cada práctica de seguridad asociada a la función de negocio establecida por *SAMM*.

Tabla 3. SAMM – Función de negocio, práctica de seguridad y descripción.

Función de Negocio	Práctica de Seguridad	Descripción Práctica de Seguridad
Gobierno	Estrategia y métricas (SM)	Involucra la dirección estratégica del programa de aseguramiento de software, tanto así, como sus actividades, procesos y métricas, con el fin de acercar estas a la postura de seguridad de la organización.
	Política y cumplimiento (PC)	Establece una estructura de control y auditoria asociada a seguridad, para dar cumplimiento a regulaciones, estándares y normas.
	Educación y orientación (EG)	Orienta a incrementar el conocimiento en seguridad, en el personal de desarrollo de software, con herramientas de entrenamiento.
Construcción	Evaluación de amenazas (TA)	Identifica y determina posibles ataques contra el software de una organización, con el fin de caracterizar los riesgos y su gestión.
	Requisitos de seguridad (SR)	Promueve la necesidad de incluir la seguridad durante los procesos de desarrollo de software, para especificar está dentro de las funcionalidades.
	Arquitectura de seguridad (SA)	Fortalece desde el proceso de diseño, la promoción de actividades de seguridad con marcos de trabajo definidos.

Verificación	Revisión de diseño (DR)	Inspecciona artefactos creados a partir del proceso de diseño, para asegurar la incepción de mecanismos de seguridad en la organización.
	Revisión de código (CR)	Evalúa el código fuente de la organización, para descubrir vulnerabilidades y definir las actividades de mitigación y seguridad en la programación.
	Pruebas de seguridad (ST)	Prueba el software en ambientes de ejecución propicios para descubrir vulnerabilidades y establecer estándares mínimos en pasos a producción.
Implementación	Administración de vulnerabilidades (VM)	Establece procesos para administrar reportes internos o externos de vulnerabilidades limitando la exposición de los riesgos y mejorando el programa de aseguramiento.
	Fortalecimiento de ambientes (EH)	Implementa controles para la administración y operación de los ambientes, con el fin de mejorar y reforzar la postura de seguridad.
	Habilitación operativa (OE)	Identifica y captura información relevante de seguridad para la operación y puesta en marcha del software en la organización.

Fuente. Tomado y adaptado de Software Assurance Maturity Model (OWASP, Una guía para integrar seguridad en el desarrollo de software, 2009)

Cada una de las prácticas de seguridad tiene tres niveles de madurez definidos y un nivel inicial cero (0) implícito. Para cada nivel representa:

- 0 - Punto inicial implícito, las actividades en la práctica no se han realizado.
- 1 - Muestra entendimiento inicial y provisión de la práctica de seguridad.
- 2 - Muestra incremento en la eficiencia y/o efectividad de la práctica de seguridad.

- 3 - Muestra un dominio amplio de la práctica de seguridad.

Tal y como lo indica (OWASP, Una guía para integrar seguridad en el desarrollo de software, 2009), al usar los niveles de madurez en cada práctica de seguridad, se genera por cada una:

- **Objetivos.** Están enfocados en alcanzar el nivel de madurez asociado, conforme se incrementa de nivel los objetivos representan metas más sofisticadas.

- **Actividades.** Requisitos indispensables para obtener en cada nivel, estas representan funciones de seguridad principales.

- **Resultados.** Demuestran la capacidad de ejecución al obtener un nivel dado.

- **Métricas de Éxito.** Especifican ejemplos de mediciones que son usadas para verificar el desempeño en un nivel dado.

- **Costos:** Describen cualitativamente los costos que incurren en una organización al obtener un nivel dado.

- **Personal.** Indican los costos de operación en términos de recursos humanos para operar en los distintos niveles.

Para obtener un panorama general acerca de las prácticas de seguridad definidas en una organización, se deben realizar revisiones con el fin de entender las actividades adoptadas por la misma. Dado lo anterior, *SAMM* puede ser implementado a futuro, como un plan de mejora continua o simplemente para evaluar el desempeño de una organización. Para esta evaluación se han definido unas hojas de trabajo, las cuales se califican en base a respuestas dadas. Como complemento se realiza un trabajo de auditoria para verificar las actividades prescritas tanto para cada actividad, su funcionamiento y métricas de éxito.

Para calificar las hojas de trabajo se pueden crear tarjetas de calificación, con el fin de mediante un conjunto de 12 calificaciones facilitar el entendimiento del programa de aseguramiento dentro de la organización. Adicional *SAMM* recomienda realizarlo en periodos de tiempo definidos, esto con el fin de conocer la evolución de esta, de ser necesario.

8.2. BSIMM – Building Security In Maturity Model

El *BSIMM* - o modelo de madurez de desarrollo seguro fue construido en el año 2008, con las prácticas de seguridad del entonces modelo *SSF – Software Security Framework*, en la actualidad se encuentra en la versión (v.10). (BSIMM, 2019). Este es un modelo basado en datos que ha ido evolucionando, el cual incluye descripciones de actividades de 122 empresas que se encuentran en diversos mercados. Actualmente para esta última versión, cuenta con 119 actividades asociadas a prácticas de seguridad, las cuales evolucionan dentro del modelo, de acuerdo con patrones de comportamiento, los cuales son rigurosamente estudiados a través de los años.

Para determinar las actividades a desarrollar por el equipo de trabajo, se considera importante iniciar con la asignación de funciones de responsables que guiaran la implementación, lo cual es parte fundamental para que el modelo funcione.

- **Líder Ejecutivo**. Permite a estos recursos, brindar apoyo mientras maduran los distintos grupos que dan soporte al desarrollo de aplicaciones. El objetivo es poner a cargo directamente de la seguridad a un ejecutivo senior que puede abordar temas de gestión, responsabilidad y empoderamiento al interior de la organización, con el fin de generar un propósito y que el modelo evolucione positivamente.
- **Grupo de Seguridad de Software.** Es el segundo rol más importante, y según los estudios realizados, se identificó que cada una de las 122 iniciativas contiene este grupo, dedicado a la seguridad del software. Muy a menudo en las organizaciones que van a adoptar la metodología *BSIMM*, se debe iniciar con los desarrolladores y enseñarles a estos temas de seguridad. Así mismo se deben crear planes de capacitación, para mejorar estas habilidades y dar el conocimiento necesario como herramienta en la organización.
- **Satélite.** Hace referencia a profesionales en el área de desarrollo, evaluación y arquitectura, los cuales tienen un interés asociado a la seguridad del software. Para esto el modelo hace referencia de manera colectiva a estos como satélite, quienes aprenden de nuevas tecnologías y amplían la comprensión de esta práctica.
- **Los demás miembros.** Son quienes pueden involucrarse en partes del modelo y contribuyen a garantizar que los productos desarrollados sean seguros, tales roles pueden ser, evaluadores, equipos de diseño, administradores, y proveedores.

BSIMM facilita la gestión y evolución del modelo, para que cualquier persona a cargo de la seguridad del software en las organizaciones tenga éxito. Para la alineación de la metodología, al interior de estas, no se requiere incorporar distintas perspectivas sobre la gestión de riesgos, más, sin embargo, se puede colaborar y aprovechar las fortalezas que este modelo proporciona.

Independiente de la cultura y estructura de las organizaciones, estas se esfuerzan por el crecimiento de seguridad durante sus proyectos y trayectorias, tanto así que generalmente avanzan en los siguientes tres estados, los cuales ubican a la organización en un estado de madurez, según su necesidad propia:

- **Emergente.** Son organizaciones que están en cero (0), e inicia un proceso de formalización emergente usando una estrategia holística del modelo. Es aquí donde se traza una hoja de ruta en la que de 12 a 24 meses y mediante recursos limitados, se trabaja sobre los cimientos del programa.
- **Madurado.** Son organizaciones con un enfoque de seguridad de software ya existente o emergente, que tienen la capacidad de madurar agregando actividades, profundidad y amplitud a los proyectos en curso.
- **Optimizado.** Son organizaciones que están afinando y evolucionando en sus capacidades de seguridad existentes, lo cual se ve reflejado muy a menudo en uso de enfoques basados en riesgos, con visiones claras de las expectativas y métricas que se tienen.

Dado lo anterior, las organizaciones pueden autoevaluarse y determinar que luego de realizar un número determinado de actividades su calificación de nivel de madurez puede estar en emergente, maduras u optimizadas respectivamente, sin embargo, las versiones anteriores de *BSIMM* han identificado que se puede alcanzar un nivel de madurez en etapas específicas, sin tener en cuenta el total de las actividades propuestas por el modelo.

BSIMM está organizado como un conjunto de 119 actividades, en un marco de seguridad de software, el cual incluye 12 prácticas que se organizan en cuatro dominios, tal y como lo muestra la Tabla 4, a continuación, se describen las practicas por cada dominio.

- **Gobierno.** Prácticas que ayudan a organizar, administrar y medir las iniciativas de software, incluyendo la gobernanza.

- **Inteligencia.** Prácticas que dan lugar, al conocimiento corporativo utilizado en el desarrollo de software.
- **Puntos de contacto *SSDL*.** Prácticas de análisis y aseguramiento de software, con la ayuda de artefactos y procesos. Todas las metodologías de seguridad incluyen estas prácticas.
- **Despliegue.** Prácticas de configuración y mantenimiento del software que tienen un impacto directo en la seguridad.

Tabla 4. BSIMM – Marco de seguridad, prácticas y dominios.

PRÁCTICAS			
Gobierno	**Inteligencia**	**Puntos de Contacto SSDL**	**Despliegue**
1. Estrategia y métricas (SM) 2. Políticas de cumplimiento (CP) 3. Formación (T)	4. Modelos de ataques (AM) 5. Características de seguridad y diseño (SFD) 6. Normas y requisitos (SR)	7. Análisis de arquitectura (AA) 8. Revisión de código (CR) 9. Pruebas de seguridad (ST)	10. Pruebas de penetración (PT) 11. Entorno del software (SE) 12. Administración y vulnerabilidades (CMVM)

Fuente. Tomado de Building Security In Maturity Model (BSIMM, 2019).

En la Tabla 5, se describe cada práctica asociada a cada dominio establecida por *BISMM*.

Tabla 5. BSIMM – Dominio, práctica de seguridad y descripción.

Dominio	**Práctica de Seguridad**	**Descripción Práctica de Seguridad**
Gobierno	Estrategia y métricas (SM)	Se concentra en la planificación, asignación de roles, responsabilidades e identificación de metas de seguridad de software.
	Política y cumplimiento (CP)	Abarca la identificación de controles para el cumplimiento regulatorio, como PCI DSS y HIPAA. Adicional, maneja contractualmente los

		niveles de acuerdo de servicio, con el fin de controlar el riesgo de software comercial.
	Formación (T)	Desempeña un papel fundamental, ya que usualmente los desarrolladores y los arquitectos de software no cuentan con conocimiento de seguridad.
Inteligencia	Modelos de ataques (AM)	Captura información relacionada a los posibles atacantes, tanto, así como amenazas, casos de abuso, clasificación de datos entre otros.
	Características de seguridad y diseño (SFD)	Mediante controles de seguridad, estudia patrones que cumplen con normas definidas.
	Normas y requisitos (SR)	Permite orientar a la organización, en construcción de software, adquisición de herramientas comerciales y crear comités de normas.
Puntos de contactos SSDL	Análisis de arquitectura (AA)	Este contiene la captura de arquitectura de software, mediante la diagramación del desarrollo de aplicaciones, incluyendo listas de chequeo de riesgos y amenazas.
	Revisión de código (CR)	Mediante el uso de herramientas de comprobación de código, se hacen análisis y seguimiento a los resultados, para conocer posibles fallas de seguridad.
	Pruebas de seguridad (ST)	Se centran básicamente en encontrar fallas en el desarrollo que puedan llegar a ser vulnerables.
Despliegue	Pruebas de penetración (PT)	Su enfoque principal es el hallazgo de vulnerabilidades en la configuración final del software, defectos y mitigación.

	Entorno del software (SE)	Se ocupa de la actualización y aplicación de parches de los sistemas como plataforma, además de la documentación, monitoreo y gestión de cambios.
	Administración y vulnerabilidades (CMVM)	Se ocupa de la actualización y aplicación de parches de control de versiones, tanto, así como el seguimiento a defectos y correcciones, además del manejo de incidentes.

Fuente. Tomado y adaptado de Building Security In Maturity Model (BSIMM, 2019).

Cada uno de los cuatro dominios del *BSIMM*, tiene asociadas unas metas particulares, las cuales dan sentido a la aplicabilidad del modelo en la organización.

- **Gobierno.** Responsabilidad, transparencia, verificación y balance.
- **Inteligencia.** Diligencia, auditoria, estandarización.
- **Puntos de contacto *SSDL*.** Control de calidad.
- **Despliegue.** Gestión de cambios y control de calidad.

Un aspecto importante para la implementación de cualquier modelo o metodología es el diseño de una hoja de ruta, la cual dará compromiso y estabilidad a la estrategia de seguridad de software, además esta ayudará a conocer las capacidades actuales y futuras con las que cuenta la organización. Algo que resalta *BSIMM* en su estudio, es que se puede comparar cualquier organización que implemente este modelo con los datos de las empresas que ya hacen parte del programa, tanto así que, si su puntuación es menor al 80%, es hora de replantear la estrategia de postura de seguridad en el desarrollo de software.

En la Tabla 6, se muestra cuáles son las actividades más comunes por práctica que han sido implementadas en las 122 empresas que han adoptado el modelo. Lo anterior no concluye que estás sean implícitamente necesarias para cualquier organización.

Tabla 6. BSIMM - Actividades más comunes por práctica

Código	Actividades
[SM1.4]	Identificar las ubicaciones de las puertas, reunir los artefactos necesarios.
[CP1.2]	Identificar las obligaciones de PII. (Información de identificación personal).
[T1.1]	Llevar a cabo un entrenamiento de concienciación.
[AM1.2]	Crear un esquema de clasificación de datos e inventario.
[SFD1.1]	Construir y publicar elementos de seguridad.
[SR1.3]	Traducir las restricciones de cumplimiento a los requisitos.
[AA1.1]	Realizar una revisión de las características de seguridad.
[CR1.4]	Utilizar herramientas automatizadas junto con la revisión manual.
[ST1.1]	Hay que asegurar que el control de calidad apoye las pruebas de condición de valor límite/bordes.
[PT1.1]	Usar probadores de penetración externos para encontrar problemas.
[SE1.2]	Hay que asegurar que los fundamentos de seguridad del host y de la red estén en su lugar.
[CMVM1.1]	Crear o interactuar con la respuesta a incidentes.

Fuente. Tomado de Building Security In Maturity Model (BSIMM, 2019).

Para determinar en donde se encuentra una organización en relación con otras empresas, frente al modelo de seguridad de software. Lo que se debe hacer es, tener un registro con las actividades que ya se tienen asociadas a seguridad, en la organización, y crear una tarjeta de puntuación¡Error! No se encuentra el origen de la referencia.. Lo siguiente es, crear un cuadro de mando y llamarlo con el nombre de la empresa, y compararlo con los cuatro dominios, cada práctica y actividades del *BSIMM*, ubicando con el número 1 donde aplique en la columna que lleva el nombre de la empresa. Así mismo la columna con

nombre *BSIMM10* muestra el número de observaciones para actividad en las que las 122 empresas tienen esta práctica dentro de su modelo de seguridad.

Luego de determinar la posición con respecto a las actividades, es indispensable diseñar un plan para mejorar las prácticas con actividades incluidas en el *BSIMM*. Es importante adoptar las actividades que den sentido a la organización. De igual forma las de nivel 1 son a menudo sencillas y de aplicación universal, las de nivel 2 son más difíciles de ejecutar y requieren más coordinación, y las de nivel 3 son más difíciles de ejecutar y no siempre son aplicables. Como tal, los niveles podrían ilustrar una progresión natural a través de las actividades asociadas con cada práctica, pero no es necesario realizar todas las actividades en un nivel determinado antes de pasar a las actividades en un nivel más alto.

8.3. Microsoft SDL - Security Development Lifecycle

Microsoft SDL o ciclo de vida de desarrollo seguro, es una iniciativa de Microsoft la cual fue lanzada en el año 2002, con su primera versión llamada “Computación Confiable”; esta busca ayudar a ser más seguros productos de software con el fin de contar con más disponibilidad, confiabilidad e integridad (Microsoft, 2002). *Microsoft SDL*, fue el resultado del trabajo de grupos de desarrollo de software, que orientaron un modelo fácil de aplicar incorporando código seguro. Para el año 2004, los productos desarrollados por Microsoft contaban ya con la implementación de esta metodología, como una estrategia constante en la evolución de la organización y de la seguridad misma. Uno de los principales objetivos de esta metodología es minimizar el número e impacto de las amenazas y vulnerabilidades en el software y en el uso de aplicaciones.

El ciclo de vida de desarrollo seguro *(SDL),* ha sido una iniciativa que ha desempeñado un papel muy importante en la integración de seguridad y privacidad en la cultura de Microsoft, siendo un referente para otras áreas de estudio en seguridad. La aplicación de esta metodología se probó en diferentes fases del ciclo de vida del desarrollo de software, y esta no se limita a grandes empresas, por lo que tiene un amplio espectro de orientación al uso en las organizaciones.

El modelo se encuentra estructurado en base a cinco áreas de capacidades, que corresponden a las fases del ciclo de vida del desarrollo de software.

- Formación, directivas y capacidades organizativas

- Requisitos y diseño
- Implementación
- Comprobación
- Lanzamiento y respuesta

Dado lo anterior el modelo define cuatro niveles de madurez para las distintas áreas de capacidades, estos son: básico, estandarizado, avanzado y dinámico, tal y como lo muestra la Figura 7.

Figura 7. Microsoft SDL - Modelo de optimización.

Fuente. Tomado de (Microsoft, 2002).

El *SDL*, inicia con el nivel de madurez básico, el cual tiene pocos procesos, formación en seguridad, herramientas y poca implementación. Desde este punto es donde se llega al nivel dinámico, en el que las organizaciones tendrán un nivel de madurez suficiente con procesos y procedimientos eficaces, profesionales con formación en seguridad y altamente calificados, herramientas de uso especializadas y un alto grado de responsabilidad en el mantenimiento y actualización de este. Uno de los principales enfoques del presente modelo, es proporcionar de manera práctica como pasar de un nivel más bajo a uno más alto y evitar el manejo de listados de actividades como suele hacerse con otras metodologías.

La aplicabilidad depende en gran parte de lo que esperan las organizaciones, frente a los tipos de planes en los que será sometido el modelo, es por ello por lo que Microsoft recomienda proyectos bajo las siguientes características, en lo posible:

- Aplicaciones de entornos empresariales
- Aplicaciones que procesen información de tipo *PII* (información de identificación personal), u otro tipo de información confidencial
- Aplicaciones que tienen comunicaciones a través de internet o redes distintas principalmente

Algunos de los principales roles de seguridad que sirven para consultar y proporcionar a la estructura organizacional, la identificación, clasificación y mitigación de los principales problemas que se pueden presentar en un proyecto de desarrollo de software seguro incluye:

- **Roles de revisión y asesoramiento**. Permiten supervisar la seguridad y la privacidad de los proyectos. Entre estos están los asesores de seguridad, auditores y expertos.
- **Líderes de equipo.** Se asignan a los expertos en materia del equipo, sus funciones principales son negociar, hacer seguimiento, validar requisitos mínimos de seguridad y privacidad. Entre estos están líder de seguridad, líder de privacidad y combinación de roles.

Como lo muestra la Figura 8, el proceso de *SDL* de Microsoft tiene un conjunto de actividades que son obligatorias, y están representadas en el orden en el que deben seguirse o llevarse a cabo, y están agrupadas por las fases de un ciclo de vida de desarrollo tradicional. A través de los años y la experiencia Microsoft recomienda seguir las actividades en su totalidad, ya que da más beneficios como una implementación sistémica en modo para este propósito (*ad hoc).*

Figura 8. Microsoft SDL - Ciclo de vida de desarrollo de software.

Fuente. Tomado de (Microsoft, 2002).

Para la implementación de un proyecto de desarrollo de software seguro, se deben realizar dieciséis (16) actividades que son obligatorias para lograr en conformidad el *SDL* de Microsoft, estas se mencionan a continuación.

Tabla 7. Microsoft SDL – Fases y actividades del modelo.

Fase	Actividad	Descripción
Requisitos anteriores: Formación en seguridad	1.Requisitos de formación	Los miembros del equipo de desarrollo de software deben ser capacitados y recibir formación en conceptos básicos de seguridad y privacidad. La formación como mínimo debe contener: diseño seguro, modelos de riesgo, codificación segura, pruebas de seguridad, privacidad, mitigación de amenazas entre otras.
Primera: Requisitos	2.Requisitos de seguridad	La seguridad es un requisito que debe considerarse desde el inicio y debe proveer fiabilidad a quien usa el software. Al principio del proyecto se deben analizar los requisitos de seguridad y privacidad, así mismo las especificaciones técnicas a ser implementadas para mitigar las posibles vulnerabilidades.
	3.Umbrales de calidad y límites de errores	El establecimiento de umbrales de calidad y límites de errores, definen los criterios asociados

		a los problemas de seguridad que se identificaran para ser corregidos y proteger el código en cada fase del desarrollo.
	4.Evaluación de los riesgos de seguridad y privacidad	Son procesos obligatorios que ayudan a la identificación de los aspectos funcionales del software, requieren una revisión al detalle. Las evaluaciones incluyen modelos de riesgos, revisiones de diseño, pruebas de penetración, análisis adicionales, impactos sobre la propiedad privada.
Segunda: Diseño	5.Requisitos de diseño	La mitigación de problemas de seguridad y privacidad es menos costosa si evalúa en las etapas iniciales del ciclo de vida del proyecto. Las especificaciones de diseño prescriben como se implementa de manera segura las funcionalidades dadas en características o funciones determinadas.
	6.Reducción de la superficie de ataques	Es específicamente una forma de reducir los riesgos dando a los atacantes menos oportunidad de aprovechar los posibles puntos débiles o las vulnerabilidades. Aquí se dan los privilegios mínimos posibles mediante un modelo de seguridad en capas.
	7.Modelos de riesgos	Permite considerar, documentar y establecer de forma estructurada las implicaciones de seguridad, dentro del marco de los diseños y su entorno operativo previsto.
Tercera: Implementación	8.Usar herramientas aprobadas	Los equipos de desarrollo en esta fase implementan una lista, con herramientas de seguridad aprobadas, tanto, así como compiladores, y actualizaciones que den protección y una detección temprana de vulnerabilidades.

	9.Prohibir funciones no seguras	Se deben analizar el llamado de las funciones y el uso de las API en entornos de amenazas. Estas que se evalúen como inseguras deben ser prohibidas dentro del código y reemplazadas por aquellas que si lo sean.
	10.Análisis estático	Los equipos que participan en el desarrollo de software deben hacer análisis estático sobre el código fuente. El cual permite revisar el código de seguridad, contribuyendo a observar las directivas de desarrollo seguro.
Cuarta: Comprobación	11.Análisis dinámico de los programas	La comprobación de software en tiempos de ejecución, aseguran las funcionalidades con el diseño inicial. Aquí se especifican las herramientas de supervisión y el comportamiento de las aplicaciones para detectar problemas de seguridad críticos.
	12.Pruebas de exploración de vulnerabilidades mediante datos aleatorios	Son formas especializadas de análisis dinámicos que introducen deliberadamente datos aleatorios o de manera incorrecta. Son ejecutadas por asesores de seguridad en tiempos determinados.
	13.Revisión de modelos de riesgo y la superficie de ataques	Es importante la revisión de los modelos de riesgos, durante las fases de requisitos y diseño dentro de los proyectos de desarrollo de software.
Quinta: Lanzamiento	14.Plan de respuesta a incidentes	Se especifica que, en cada lanzamiento de software, se debe contemplar incluir un plan, y un equipo de respuestas a incidentes.
	15.Revisión final de seguridad	Son inspecciones deliberadas de todas las actividades de seguridad realizadas hasta el momento.

	16.Lanzamiento o archivado	El asesor de seguridad que fue asignado certifica que el equipo y personal del proyecto cumplió con todos los requisitos de seguridad. Además, se archiva los datos e información pertinentes del mantenimiento de este.

Fuente. Tomado y adaptado de (Microsoft, 2002).

Las organizaciones que han implementado el uso del ciclo de vida *SDL* de Microsoft, deben comprobar que se han seguido todos los pasos descritos. Así mismo se debe garantizar la centralización de los datos referentes a los desarrollos de código seguro, con el fin de contribuir a la oportuna toma de decisiones en escenarios como auditorias y revisiones de cumplimiento por los distintos entes a lugar.

La comprobación de la seguridad en las distintas aplicaciones desarrolladas con esta metodología implica:

- Usar software diseñado específicamente para realizar seguimiento, y que centralice todos los artefactos del *SDL*.
- Únicamente personal autorizado tendrá acceso al software anteriormente descrito.
- Se debe realizar una segmentación de roles al interior de los procesos y del uso de la metodología.
- Se deben capturar con exactitud los requisitos de seguridad y privacidad suministrados al inicio del proyecto, con el fin de ser exhaustivos en las revisiones de seguridad.

El *SDL* de Microsoft, permite mejorar la seguridad y privacidad del software, y está demostrado que este ha sido aplicado a un número importante de programas y proyectos alrededor del mundo. Así mismo, la combinación de procesos de formación con el uso de herramientas aporta beneficios y una capacidad técnica al desarrollo de código seguro en las organizaciones.

9. Comparativo de las metodologías de desarrollo de código seguro

Luego de investigar y profundizar más sobre las tres principales metodologías de desarrollo de código seguro, tratadas en el presente documento, en esta sección se realiza una definición de aspectos, con el fin compararlos y conocer cuáles de ellos y que elementos se ajustan al enfoque del marco de buenas prácticas que será el resultado de la presente investigación. Para esto se definieron los siguientes:

- **Consideraciones Iniciales.** Este primer aspecto tiene el conocimiento propio de las metodologías, que han sido analizadas en la sección anterior. Busca dar un vistazo de cada una a nivel general, para saber cuál es el alcance, dominios, actividades, complejidad, facilidad de implementación, uso, y el tipo de organización en la que puede aplicar. Los elementos que constituyen este aspecto son: dominios, actividades, facilidad de implementación, facilidad de uso, tipo de organización.

- **Alineación Estratégica del Negocio.** Este segundo aspecto es fundamental, busca medir la alineación de la estrategia y necesidades del negocio, con la propuesta de desarrollo de código seguro en las aplicaciones y herramientas en cualquier organización. Este incluye conocer si estas cuentan con un plan de formación que apoyara la propuesta y el compromiso de los funcionarios, e indagar sobre el cumplimiento normativo, extralegal y regulatorio al implementar estándares de seguridad internos. Los elementos que constituyen este aspecto son: estrategia y métricas, cumplimiento normativo, formación en seguridad.

- **Requisitos de Seguridad.** El tercer aspecto de validación para comparar las metodologías busca identificar la orientación técnica desde el punto de vista de arquitectura de seguridad, esta debe contener requisitos de seguridad en el diseño e implementación de las aplicaciones, así como también velar por la identificación de los riesgos, amenazas y la evaluación de la privacidad de los datos e información que almacenara la misma. Los elementos que constituyen este aspecto son:

arquitectura de seguridad, requisitos de seguridad y diseño, evaluación de riesgos y privacidad.

- **Verificación de Contexto:** El cuarto aspecto, busca hacer una evaluación de la seguridad dentro del contexto del desarrollo de código, con el fin de encontrar posibles vulnerabilidades, errores, y potenciales vectores de compromiso. Es importante que la metodología contemple esta fase preliminar de auditoria, en el desarrollo y codificación de las aplicaciones, de tal forma que de un vistazo inicial de cómo está la seguridad y el uso funcional de esta. Los elementos que constituyen este aspecto son: uso de artefactos y/o herramientas de seguridad, pruebas de seguridad estáticas, pruebas de seguridad dinámicas, detección de funciones seguras.

- **Despliegue y Mejora Continua:** El quinto y último aspecto, busca fortalecer el ecosistema de seguridad en el desarrollo de código, a través de la fase de despliegue y la mejora continua, de los procesos establecidos con la metodología. Esta debe contemplar las pruebas de vulnerabilidad de las aplicaciones, en su última fase de desarrollo, el mantenimiento de estas vulnerabilidades, y tener una expectativa clara sobre el manejo de posibles eventos de incidentes de seguridad en los distintos ambientes y contextos. Los elementos que constituyen este aspecto son: pruebas de vulnerabilidad, administración de vulnerabilidades, plan de respuesta a incidentes.

La Tabla 8, contiene los criterios de evaluación y calificación que serán aplicados a las tres metodologías de desarrollo de código seguro seleccionadas. Cada aspecto será evaluado y categorizado de acuerdo con tres dimensiones: bajo, medio y alto. Es de aclarar que el criterio bajo, está asociado a la ausencia total del aspecto, el medio cuenta con algunos elementos del aspecto, el alto claramente cuenta con todos los elementos del aspecto, esto para cada metodología a comparar.

Tabla 8. Comparativo - Criterios de calificación.

ASPECTOS	**CRITERIOS DE CALIFICACIÓN – (Puntuación)**		
	BAJO (1-2)	**MEDIO (3)**	**ALTO (4-5)**

Consideraciones Iniciales	No cuenta con elementos de consideraciones iniciales	Cuenta con los elementos de consideraciones iniciales necesarias	Las consideraciones iniciales, dar un acercamiento claro a la metodología
Alineación Estratégica del Negocio	No contiene elementos de alineación estratégica del negocio	Contiene algunos elementos de alineación estrategia del negocio	La alineación estratégica del negocio es clara y generara valor a los procesos de la organización
Requisitos de Seguridad	No posee elementos de requisitos de seguridad	Posee algunos elementos de requisitos de seguridad	Los requisitos de seguridad están claramente definidos
Verificación de Contexto	No cuenta con elementos de verificación de contexto	La verificación del contexto no está definida en su totalidad	La verificación de contexto tiene definidos sus elementos y estos claros
Despliegue y Mejora Continua	No contiene elementos de despliegue y mejora continua	Los elementos de despliegue y mejora continua no cubren los requisitos de seguridad	El despliegue y la mejora continua cubren los requisitos de seguridad y el mantenimiento del proceso

Fuente. Elaboración propia.

Tal y como lo muestra la tabla anterior los criterios de calificación están clasificados en bajos, medios y altos. Para medir de manera más precisa cada aspecto y sus elementos, se dará una puntuación de 1 a 5, siendo 1 menor y 5 la puntuación máxima a obtener. Para el criterio bajo se dará una puntuación de 1 a 2, para el medio la puntuación será de 3, y para el criterio alto una puntuación máxima de 4 a 5 respectivamente, tal y como lo muestra la Tabla 9. Así mismo luego de evaluar cada metodología y cada aspecto con sus elementos, se calcula un promedio entre estas puntuaciones, las cuales categorizaran al aspecto en un criterio de calificación según lo explicado en el presente párrafo.

Tabla 9. Comparativo - Criterios de puntuación.

	CRITERIOS		
	BAJO	**MEDIO**	**ALTO**
PUNTUACIÓN	1-2	3	4-5

Fuente. Elaboración propia.

Luego de especificar los criterios de calificación y puntuación, en la Tabla 10 se han comparado las metodologías, con los aspectos y elementos definidos en este capítulo.

Tabla 10. Comparativo – Metodologías.

			Metodologías		
No.	**Aspecto**	**Elemento**	**SAMM**	**BSIMM**	**Microsoft - SDL**
1	Consideraciones Iniciales	Dominios	5	5	5
		Actividades	5	5	5
		Facilidad de Implementación	3	2	3
		Facilidad de Uso	3	2	3
		Tipo de Organización	5	5	5
Calificación Aspecto 1			**4,2**	**3,8**	**4,2**
2	Alineación Estratégica del Negocio	Estrategia y Métricas	5	5	1
		Cumplimiento Normativo	5	5	1
		Formación en Seguridad	5	5	5
Calificación Aspecto 2			**5,0**	**5,0**	**2,3**
3	Requisitos de Seguridad	Arquitectura de Seguridad	5	5	1
		Requisitos de Seguridad y Diseño	4	4	4
		Evaluación de Riesgos y Privacidad	3	3	5
Calificación Aspecto 3			**4,0**	**4,0**	**3,3**
4	Verificación de Contexto	Artefactos y/o Herramientas de Seguridad	4	4	4
		Pruebas de Seguridad Estáticas	4	4	5
		Pruebas de Seguridad Dinámicas	4	4	5
		Detección de Funciones Seguras	2	2	4
Calificación Aspecto 4			**3,5**	**3,5**	**4,5**
5		Pruebas de Vulnerabilidad	5	5	4

	Despliegue y Mejora Continua	Administración de Vulnerabilidades	5	5	4
		Plan de Respuesta a Incidentes	3	4	5
Calificación Aspecto 5			**4,3**	**4,7**	**4,3**

Fuente. Elaboración propia.

Como resultado, en el primer aspecto **consideraciones iniciales**, se puede observar un equilibro entre las tres metodologías, lo cual indica que cuentan con dominios, funciones de negocio o fases por cada una, lo que da orden, secuencia y un cumplimiento ALTO, a las tres. Por otro lado, tienen definidas actividades que son específicas a cada una, desde la visión de desarrollo de código seguro, orientado a proveer los pasos necesarios para el acompañamiento e implementación en las organizaciones, lo que da un cumplimiento ALTO.

En cuanto a facilidad de uso e implementación, las tres cuentan con documentación importante, más sin embargo al tener un número de actividades alto, el proceso de inmersión en una organización puede tomar alrededor de 12 meses aproximadamente, por lo cual se asigna un cumplimiento MEDIO.

El tipo de organización al que van dirigidas son pequeñas, medianas o grandes, lo cual abarca un espectro amplio en la industria del software, o empresas que tengan dentro de sus procesos, desarrollo de código indistintamente de su tamaño, por lo que se da un cumplimiento ALTO respectivamente.

Algo a resaltar sobre este aspecto, es que permitió conocer e identificar, la generalidad de cada metodología que fue evaluada, para diseñar el marco de buenas prácticas de desarrollo de código seguro y su posible impacto en la organización.

En el segundo aspecto **alineación estratégica del negocio**, claramente la metodología *SAMM* y *BSIMM* tienen un impacto ALTO desde el gobierno como eje transversal a las tecnologías de información, lo cual es importante, ya que desde la estrategia buscara que el código sea seguro, y sea validado en todas las distintas fases de desarrollo. En cuanto a *Microsoft SDL*, este resalta algunas capacidades organizativas desde el punto de vista de asociación de roles, herramientas, formación entre otros, pero no desde el gobierno, por lo que su cumplimiento es BAJO.

En cuanto a cumplimiento normativo *SAMM* y *BSIMM*, tienen un enfoque en comprender requisitos legales y regulatorios de entes externos para asegurar los objetivos del negocio en la organización, por lo cual su cumplimiento es ALTO; más sin embargo *Microsoft SDL*, tiene un enfoque más hacia la calidad y el límite de errores esperado, por lo cual debería trabajar en conocer las regulaciones internacionales y propias de un país como Colombia, para incluirlo dentro del *SDL*, su cumplimiento es BAJO.

Para resaltar en este aspecto, las tres metodologías comparadas tienen un plan de formación en seguridad, para los distintos miembros y roles que cada una incluye, esto con el fin de mejorar en los equipos de los proyectos, la identificación y mitigación de los posibles riesgos en seguridad que son aplicables a la organización en las fases de desarrollo e implementación de los productos de software, por lo que su cumplimiento es ALTO.

En el tercer aspecto **requisitos de seguridad**, *SAMM* y *BSIMM* conciben la arquitectura de seguridad, como una práctica que a partir de recomendaciones y marcos de desarrollo de software, permiten la inserción y la evolución de patrones de diseño funcionales seguros en las aplicaciones que se construyen, dando un cumplimiento ALTO, sobre el *SDL* de Microsoft, el cual concibe este elemento dentro de los programas de formación y los conocimientos del personal que proporcionara la organización, para este da un cumplimiento BAJO.

La evaluación de los requisitos de seguridad y diseño son incluidos claramente en las tres metodologías, especificando a través de actividades los objetivos funcionales del software que será desarrollado, basados en prácticas de seguridad, ejerciendo auditorias de calidad, seguimiento y sofisticación. Para este elemento su cumplimiento es ALTO.

Es indiscutible que la evaluación de riesgos y privacidad va de la mano con los requisitos de seguridad y diseño, ya que al tener una participación e involucrando la seguridad desde que se concibe el proyecto, el entorno se centrara en la identificación y comprensión de los riesgos y privacidad que lo rodean. Será eficaz la toma de decisiones, y la remediación de los posibles hallazgos y amenazas, para lo cual *SAMM* y *BSIMM*, obtuvieron puntuación MEDIO, lo que demuestra que estas dos metodologías dentro de sus actividades hacen evaluación de amenazas y vulnerabilidades, pero no como lo hace el *SDL* de Microsoft, quien define específicamente dentro del proyecto de desarrollo de software la evaluación

de riesgos y privacidad, lo cual demuestra el carácter comercial de la metodología en sus productos. Para este elemento su cumplimiento es ALTO.

Para resaltar en este aspecto, es importante contar con un esquema de arquitectura de seguridad, ya que este permitirá a la organización adaptar de manera natural los procesos y validaciones de seguridad en el desarrollo de las aplicaciones.

Ya en el cuarto aspecto **verificación de contexto**, en cuanto al uso de artefactos y herramientas, las tres metodologías lograron obtener una puntuación ALTA, debido a que estas en su modelo incluyen listas de verificación y uso de herramientas, asociadas a la revisión de requisitos de seguridad, y a automatizar las distintas pruebas de revisión de código fuente. Adicional, un punto a destacar de *SAMM* y *BSIMM*, es que tienen procesos de fortalecimiento del ambiente y análisis de la arquitectura, en donde se valida el entorno de operación de las aplicaciones y si este es asegurado, tanto así, a nivel de herramientas, como de sistemas operativos y compiladores.

Por otro lado, en cuanto a la ejecución de pruebas estáticas y dinámicas, los tres se destacan. Estas tienen una puntuación ALTA, debido a que buscan analizar el código, con el fin de encontrar directivas inseguras. Las pruebas tienen una cobertura importante de la seguridad, sin dejar atrás los aspectos funcionales y productivos. A resaltar sobre Microsoft *SDL* en este elemento, es su capacidad de disponer las pruebas estáticas y dinámicas de manera separada, lo que permite detectar problemas asociados a privilegios, usuarios, y otras directivas en las que se pone un gran número de datos e información que da veracidad a los distintos análisis.

En cuanto a la detección de funciones seguras, de las tres herramientas sobresale el *SDL* de Microsoft, con una puntuación ALTA, esto ya que este busca determinar con una lista, el uso de funciones prohibidas en los distintos compiladores, que debe ser tenido en cuenta por los equipos de desarrollo, para ser reemplazadas por funciones más seguras. *SAMM* y *BSIMM*, asumen este elemento dentro de las pruebas de revisión de código, y las características de seguridad y diseño.

Por último, en el quinto aspecto **despliegue y mejora continua**, dos elementos que van de la mano son las pruebas y administración de las vulnerabilidades, en estos, las tres metodologías sobresalieron con una puntuación ALTA, lo cual es importante ya que da un

entendimiento mayor y un apetito del riesgo a la organización limitado, frente a las posibles vulnerabilidades y errores en la codificación de las aplicaciones. Además, el plan de respuesta a incidentes prepara a la organización y su proceso, frente a posibles problemas de seguridad, por lo que la metodología *SAMM* obtuvo una puntuación MEDIO, mientras que *BSIMM* y *Microsoft SDL*, tienen una puntuación ALTA. La importancia del plan de respuesta a incidentes radica en implementar planes de seguridad, asociados a posibles vectores de ataque y la respuesta frente a estas.

En la Tabla 11, se muestra un resumen general de los puntajes obtenidos y el criterio alcanzado, de acuerdo con los aspectos definidos y los elementos de comparación.

Tabla 11. Comparativo – Resumen metodologías.

		Metodologías		
No.	**Aspecto**	**SAMM**	**BSIMM**	**Microsoft - SDL**
1	Consideraciones Iniciales	ALTO	MEDIO	ALTO
2	Alineación Estratégica del Negocio	ALTO	ALTO	BAJO
3	Requisitos de Seguridad	ALTO	ALTO	MEDIO
4	Verificación de Contexto	MEDIO	MEDIO	ALTO
5	Despliegue y Mejora Continua	ALTO	ALTO	ALTO

Fuente. Elaboración propia.

Del presente comparativo, se puede concluir que las tres metodologías son amplias en su aplicación, tienen definidos sus dominios, funciones de negocio y fases por cada una, lo que genera actividades de desarrollo y objetivos alineados con métricas para evaluar la gestión e implementación de cualquiera de estos modelos. La estrategia y el gobierno de TI, es un eje fundamental que ayuda a la adopción de estas metodologías, además de dar cumplimiento normativo y formación en temáticas de seguridad a quienes ejercen roles de desarrollo, o de alguna manera aportan a los proyectos de software.

Los requisitos de seguridad y la verificación de contexto permiten profundizar en la identificación de riesgos, tanto así, como en el uso de artefactos y/o herramientas de seguridad, encaminadas a automatizar la detección de funciones inseguras y las pruebas estáticas y dinámicas en el código.

Los aspectos finales orientados al despliegue y la mejora continua son las herramientas de los modelos que permiten optimizar los procesos al interior de la organización, y administrar de manera eficiente las vulnerabilidades y posibles hallazgos que se dan en las aplicaciones, luego de su salida a producción, esto se considera como una fase de estabilización.

10. Desarrollo ágil de software con Scrum

La gestión de proyectos ha evolucionado, y las organizaciones deben responder rápidamente ante los cambios que se dan en el mundo. El método ágil, permite adaptar las nuevas formas de trabajo a las distintas condiciones de los proyectos y su entorno, dando respuestas rápidas, flexibles y adaptables (Fajardo, 2020). Este permite crear equipos de trabajo idóneos, que analizan y mejoran los productos de desarrollo, aumentando la capacidad de fabricación haciendo estos más competitivos.

10.1. El manifiesto ágil y Scrum

En el año 2001 se creó el manifiesto ágil, con el fin de mejorar la gestión de los proyectos de desarrollo de software, el cual tiene cuatro aspectos importantes, los cuales se nombran a continuación:

- Las personas y las interacciones son el foco, más que los procesos y herramientas.
- La documentación pasa a un segundo plano, se debe dar más importancia al software funcional.
- La colaboración con el cliente es un hito importante, más que el negocio.
- Los procesos deben responder al cambio más que seguir un plan en específico.

Con el fin de encontrar una alternativa a las limitaciones que daba el modelo de desarrollo en cascada, un grupo de programadores de software se enfocaron en los beneficios del desarrollo ágil, siendo este más eficiente y comprometido con la satisfacción del cliente (Gonçalves, 2020). Algunos de estos beneficios son:

- Compromiso y satisfacción a los inversores

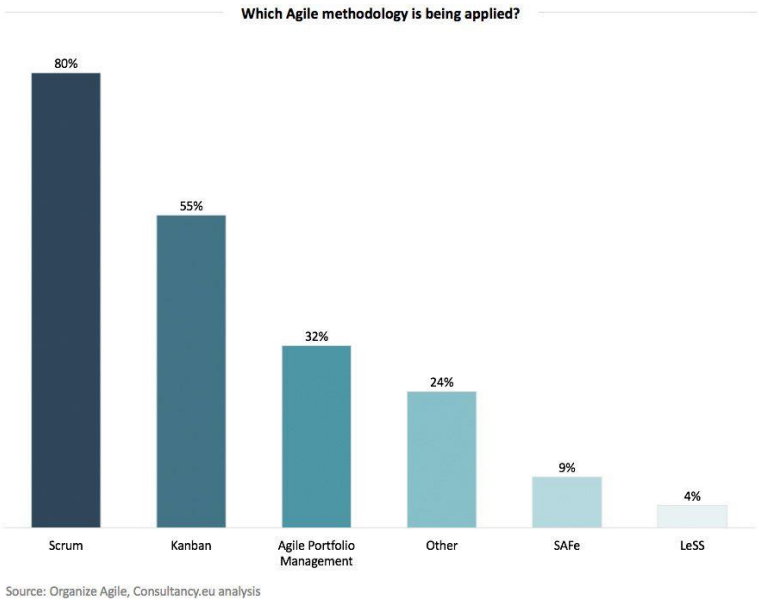
- Transparencia
- Entrega temprana y predecible
- Horarios y costes predecibles
- Priorización flexible
- Cambios permitidos
- El foco son los usuarios
- Mejor calidad
- Objetivos de equipo

Adicional, en el año 2009, se realizó la publicación de la investigación *The Business Value of Agile Software Methods* (Rico, 2009), en este se comparó el método ágil con los métodos tradicionales en la gestión de proyectos de software. Durante esta investigación se analizaron 23 procesos ágiles, comparándolos con 7.500 proyectos tradicionales, por lo cual se encontró 20 beneficios en los proyectos ágiles, entre lo que se destaca:

- El 83% mostro un menor tiempo de lanzamiento del producto al mercado.
- El 41% fueron mejores en cuanto a valor empresarial y alineación del negocio.
- El 50% tenía mayor calidad y menores costes.
- El 83% fue más productivo.

Hoy en día, existen bastantes metodologías bajo el marco ágil; al igual que Scrum comparten características y prácticas similares. Es un hecho, que la mitad de las organizaciones del mundo, han aplicado metodologías ágiles por más de tres años. Dentro de las más usadas, según el *Organize Agile*, una consultora europea encargada de realizar estudios a nivel mundial, Scrum es la metodología más usada, con cerca del 80% de aplicación según los encuestados, así mismo le sigue *Kanban* con el 55% y el 32% usan un portafolio de administración ágil, tal y como lo muestra la Figura 9**.** (Consultancy.eu, 2020)**.**

Figura 9. Scrum **-** ¿Qué metodología ágil está siendo más aplicada?



Fuente. Tomado de (Consultancy.eu, 2020).

Scrum con el paso del tiempo, se ha consolidado como una de las metodologías ágiles más usadas para desarrollar software y gestionar proyectos, porque facilita el hallazgo de soluciones en el menor tiempo posible, y se destaca por la manera en que sus fases iterativas encuentran distintas soluciones en cualquiera de los proyectos en el que participa. Este se enfoca en el trabajo colaborativo, en reunir a todos los involucrados, y en satisfacer las necesidades del cliente, priorizando las entregas de desarrollo de código. Además, esta creado para entornos complejos y de gran incertidumbre en el que la innovación, flexibilidad y competitividad es necesaria (Mariño, 2019).

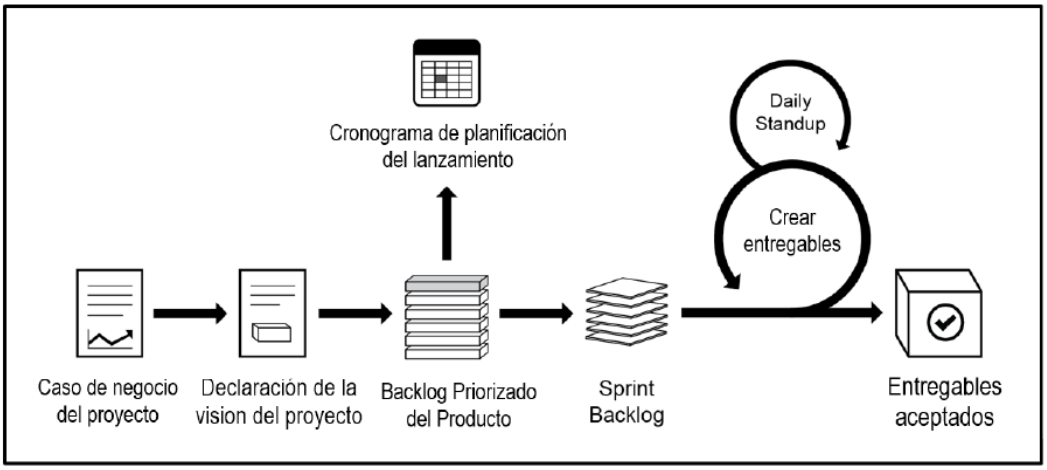
10.2. Visión general de Scrum

Scrum fue desarrollado y soportado por Ken Schwaber y Jeff Sutherland, presentado en la conferencia sobre programación, lenguajes y aplicaciones orientadas a objetos *(OOPSLA)* en el año de 1995 en Austin, Texas. Desde entonces, Scrum ha aumentado su popularidad, y hoy es considerado como el método de proyectos más usado en organizaciones a nivel mundial. (SCRUMstudy, 2017). Este consiste en los equipos Scrum, sus roles, eventos, artefactos y reglas asociadas. Cada componente dentro de este marco de trabajo tiene un propósito específico y es esencial para el éxito de la metodología. Se basa en seis

principios: control de proceso empírico, auto organización, colaboración, priorización basada en valor, gestión del tiempo, y desarrollo iterativo.

Una de las grandes fortalezas de este marco de trabajo, radica en el uso de equipos interfuncionales, auto organizados y empoderados que dividen su trabajo en ciclos *(Sprints)* de trabajos cortos y concentrados, tal y como lo muestra la Figura 10.

Figura 10. Scrum **–** Flujo para un ciclo (Sprint).



Fuente. Tomado de (SCRUMstudy, 2017).

Este trabaja bajo la teoría del control de procesos empírico, el cual asegura que el conocimiento proviene de la experiencia y de tomar decisiones basadas en lo que se conoce. Está compuesto por tres pilares soportan el control de procesos empírico, transparencia, inspección y adaptación.

- **Transparencia.** Todo aspecto significativo debe ser visible por aquellos responsables del resultado, y se debe compartir un uso de lenguaje común para referirse a un proceso.

- **Inspección.** Continuamente se debe inspeccionar los artefactos de Scrum, y los progresos hacia los objetivos con el fin de detectar variaciones no deseadas.

- **Adaptación.** Se debe realizar ajustes en el evento en que se determine que se han desviado de los límites aceptables del proyecto, para lo cual Scrum prescribe cuatro

eventos formales dentro de cada ciclo *(Sprint)*: planificación del ciclo *(Sprint)*, Scrum diario, revisión del ciclo *(Sprint),* retrospectiva del ciclo *(Sprint)*.

El éxito de Scrum depende del compromiso de cada persona, así como la aplicación e interpretación que cada uno le pueda dar a los valores que tiene este marco de trabajo, tales como el compromiso, el coraje, foco, apertura y respeto. Algo importante que se debe aplicar aquí es que todos se deben enfocar en el trabajo del ciclo *(Sprint)* y conseguir juntos llegar a la meta propuesta.

10.3. El equipo Scrum

El equipo de Scrum, compuesto por el dueño de producto *(Product Owner),* el equipo de desarrollo *(Development Team)* y un facilitador *(Scrum Master)*. Estos equipos son autoorganizados y eligen la forma de trabajar, además no son dirigidos por personas externas. Los equipos multifuncionales trabajan sin depender de otras personas que no son parte del equipo. Estos entregan productos iterativa e incrementalmente, con el fin de obtener retroalimentación oportuna de estos, con esto se asegura que habrá una versión potencialmente útil y funcional del producto esperado por el cliente.

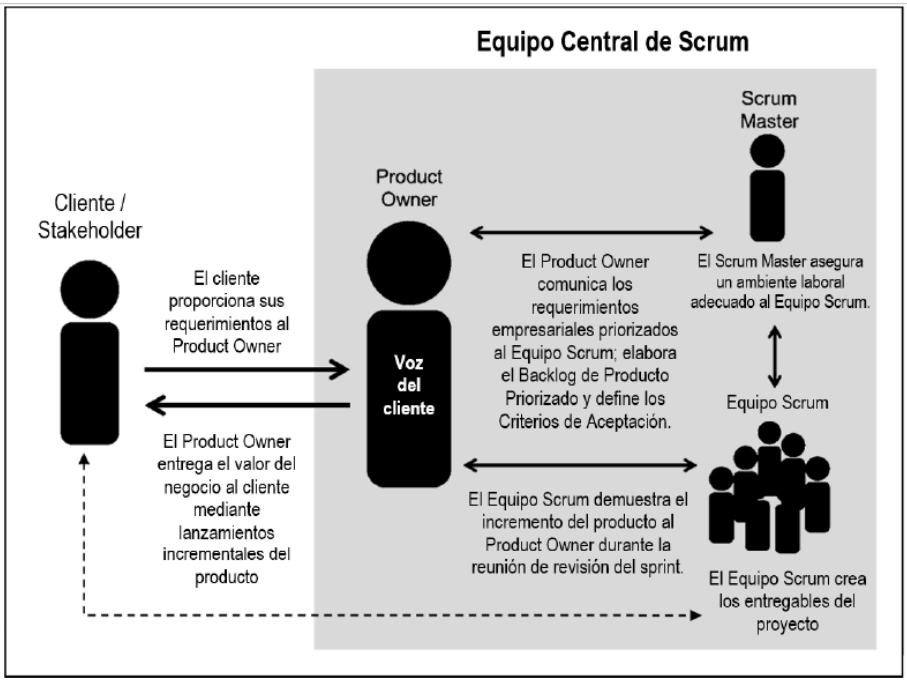
- **Dueño de producto *(Product Owner).*** Es una única persona, que debe tener el respaldo de toda la organización, para la toma de decisiones. Este está a cargo de responder por la lista de producto *(Product Backlog)*, y maximizar el valor del producto resultante, del trabajo del equipo de desarrollo.

- **Equipo de desarrollo *(Development Team).*** Son los profesionales que se encargan de trabajar en las entregas de incremento de productos “terminados”, que tienen la característica de poder colocarlos en producción. Estos productos “terminados” son obligatorios en las revisiones de los ciclos *(Sprints)*.

- **Facilitador *(Scrum Master)*.** Promueve y apoya a los miembros del equipo a entender la teoría, practicas, reglas y valores de Scrum. Es un líder, que ayuda a las personas externas a entender las interacciones para maximizar los esfuerzos del equipo.

En la Figura 11, se muestra la descripción de los roles de Scrum, y como estos interactúan entre sí en cada ciclo *(Sprint).*

Figura 11. Scrum **–** Roles descripción general



Fuente. Tomado de (SCRUMstudy, 2017).

10.4. Eventos de Scrum

Con el fin de dar regularidad y maximizar el tiempo disminuyendo las reuniones, Scrum cuenta con una serie de eventos predefinidos. Estos eventos se llaman bloques de tiempo *(time-boxes),* y tienen una duración fija, de tal manera que todos tienen una duración máxima, lo que implica que este tiempo no podrá acortarse o alargarse. Es de aclarar que algunos de los eventos pueden terminar, siempre y cuando estos hayan alcanzado los objetivos propuestos.

Scrum recomienda tener presente los siguientes eventos, y si llegase a faltar alguno, se puede ver reducida la transparencia y la oportunidad de la adaptación e inspección del marco de trabajo. Estos eventos son:

- **El ciclo *(Sprint).*** Es el evento principal de Scrum, tiene definido un bloque de tiempo de un mes o menos, en el cual se crean incrementos de productos “terminados”. Cada nuevo ciclo *(Sprint)* inicia luego de la finalización del ciclo *(Sprint)* anterior, y se debe considerar como un proyecto con un objetivo definido,

y estos se usan con el fin de lograr algo, o conseguir algo en específico. Al durar tan solo un mes, se controla el riesgo y se limitan los objetivos planteados asociados a los costos. Es de aclarar que el alcance y los objetivos planteados no se pueden cambiar, ya que pueden poner en riesgo a los objetivos del ciclo *(Sprint Goal)*.

Los ciclos *(Sprints)* contienen: Planificación *(Sprint Planning)*, Scrum Diario *(Daily Scrum)*, Revisión *(Sprint Review)* y Retrospectiva *(Sprint Retrospective).*

- **Planificación del Sprint *(Sprint Planning).*** Aquí se planifica el trabajo a realizar, todo el equipo completo de Scrum participa. Esta tiene una duración de ocho horas para un mes, para eventos más cortos se puede reducir el tiempo. Uno de los principales deberes del facilitador *(Scrum Master)* es que todos los asistentes comprendan el propósito y los tiempos determinados para el fin que se busca. Usualmente responde a dos preguntas. ¿Qué se entregará al final del ciclo *(Sprint)*?, ¿Qué será necesario hacer para conseguir el objetivo planteado?

- **Objetivo del ciclo *(Sprint Goal).*** Este se crea durante la planificación, se concibe como una meta para el ciclo *(Sprint)*, se apoya principalmente en la lista de implementación del producto y es coherente con el objetivo trazado. Guía al equipo de desarrollo del porqué de la construcción del producto en incremento. Si el trabajo resultante llega a ser distinto, el equipo de desarrollo acuerda con el dueño de producto el alcance de la lista de pendientes del ciclo *(Sprint Backlog).*

- **Scrum Diario *(Daily Scrum).*** Son reuniones que se realizan a diario en el mismo lugar y a la misma hora, durante el ciclo *(Sprint)*, con una duración de 15 minutos. En esta reunión el equipo de desarrollo planea el trabajo a realizar durante las próximas 24 horas, con el fin de que el equipo inspeccione el trabajo avanzado desde el último Scrum diario. Además, sirve para evaluar el progreso hacia el objetivo del ciclo *(Sprint)*, y tener presente la lista de pendientes. Usualmente esta reunión es guiada por el equipo de desarrollo, aunque es planeada por el facilitador *(Scrum Master)*.

- **Revisión de Sprint *(Sprint Review).*** Estas suceden al finalizar el ciclo *(Sprint)*, con el fin de inspeccionar el incremento, y adaptar si fuera necesario la lista de producto. En esta reunión se evalúa lo alcanzado según los objetivos planteados,

y se colabora con el equipo en determinar las cosas que quedan pendientes por hacer. Esta es más reunión informal, que tiene una duración de cuatro horas para un ciclo *(Sprint)* de un mes, para eventos más cortos se puede reducir el tiempo. Entre otros temas se genera una revisión de la lista de producto, que definirá los elementos posibles de la lista de producto del siguiente ciclo *(Sprint)*, además de revisar la línea de tiempo, costes, presupuesto, capacidades, próximas entregas o funcionalidades.

- **Retrospectiva *(Sprint Retrospective).*** Es considerada como una oportunidad en la que el equipo Scrum se inspecciona así mismo, y se crean planes de mejora para el siguiente. Esta reunión se da luego de la revisión, y antes de la planificación del ciclo *(Sprint)*, tiene una duración de tres horas para uno de un mes, para eventos más cortos se puede reducir el tiempo. En esta la responsabilidad cae sobre el facilitador *(Scrum Master),* quien dirige la sesión y alienta al equipo a mejorar y ser más efectivos en los procesos de desarrollo y en la práctica.

10.5. Artefactos de Scrum

Los artefactos proporcionan mayor claridad al marco de Scrum, y son útiles para dar transparencia, inspección y adaptación. Estos son necesarios ya que dan sentido y acompañamiento a las actividades de este.

- **Lista de producto *(Product Backlog).*** Es una lista que contiene los elementos necesarios del producto, es dinámica y está considerada como la fuente en la que se establecen los requisitos y que puede llegar a ser susceptible a cambios en la medida en que el producto evoluciona de acuerdo con las necesidades del cliente. Esta tiene las funcionalidades, características, requisitos, mejoras y correcciones sobre las entregas del producto. Al pasar el tiempo, es normal que esta lista se extienda, ya que con la retroalimentación oportuna surgen nuevas necesidades a implementar. El responsable de esta es el dueño de producto *(Product Owner).* Estas tienen una etapa de refinamiento *(Refinement),* para dar orden, añadir detalles y posibles estimaciones.

- **Lista de pendientes *(Sprint Backlog)*.** Pertenece únicamente al equipo de desarrollo. Es un plan que tiene el nivel de detalle suficiente, que guarda el conjunto de elementos de la lista de producto, más un plan de entrega de incremento para conseguir los objetivos trazados. También es una predicción hecha por el equipo de desarrollo sobre las posibles funcionalidades que harán parte del próximo incremento, y del trabajo que debe hacerse para ello. Esta lista de pendientes acompaña al ciclo *(Sprint)* durante el periodo de tiempo de su ejecución, además ayuda al Scrum diario a visualizar el plan con los detalles necesarios, cambios y progresos de avances. En algunos casos el equipo de desarrollo adiciona a esta lista, nuevo trabajo que debe hacerse.

- **Incremento.** Es el resultado de todos los elementos que hacen parte de la lista de producto, que tienen un estado "completado" en un ciclo *(Sprint)*, y el valor de los incrementos de todos los ciclos *(Sprints)* anteriores. Al final como resultado de un ciclo *(Sprint)* el incremento debe estar "terminado" y este debe contar con lo necesario para ser utilizable.

10.6. Procesos de Scrum

Como lo muestra la Tabla 12**,** Scrum cuenta con 19 actividades específicas, enmarcadas en cinco fases, las cuales aplican a todos los proyectos. Estas fases describen los procesos fundamentales de Scrum.

Tabla 12. Scrum – Procesos fundamentales.

Fase	Procesos fundamentales de Scrum
Inicio	1. Crear la visión del proyecto 2. Identificar al facilitador (*Scrum Master) e* interesado(s) *(Stakeholder(s))* 3. Formar equipos Scrum 4. Definir épica (s) 5. Crear la lista priorizada del producto 6. Realizar la planificación de lanzamiento

Planificación y estimación	7. Crear historias de usuario 8. Estimar historias de usuario 9. Comprometer historias de usuario 10. Identificar tareas 11. Estimar tareas 12. Crear la lista de pendientes *(Sprint Backlog)*
Implementación	13. Crear entregables 14. Realizar Scrum diario 15. Refinar la lista priorizada del producto
Revisión y retrospectiva	16. Demostrar y validar el ciclo *(Sprint)* 17. Retrospectiva del ciclo *(Sprint)*
Lanzamiento	18. Enviar entregables 19. Retrospectiva del proyecto

Fuente. Tomado de (SCRUMstudy, 2017).

A continuación, se hace una breve descripción de cada proceso fundamental por sus fases:

Inicio.

- **Crear la visión del proyecto.** Se identifica el dueño de producto *(Product Owner)*, se revisa el caso de negocio del proyecto, para analizar su enfoque.

- **Identificar al facilitador *(Scrum Master)* e interesado(s) *(Stakeholder(s))*.** Mediante un proceso de selección específico se identifica al facilitador *(Scrum Master)* y a los interesados *(Stakeholders)*.

- **Formar equipos Scrum.** El dueño de producto *(Product Owner)*, identifica a los miembros del equipo Scrum, lo puede hacer colaborativamente con el facilitador *(Scrum Master)*.

- **Definir épica (s).** Se hace la declaración de visión del proyecto, se llevan a cabo reuniones de usuarios de grupos para revisar las épicas adecuadas.

- **Crear la lista priorizada del producto.** Se crean y se refinan las épicas, estas son usadas para crear una lista priorizada del producto. Aquí se dan los criterios de producto “terminado”.

- **Realizar la planificación de lanzamiento.** Se determina el tiempo del ciclo *(Sprint)* y se revisan las historias de usuario en la lista priorizada del producto, para hacer un cronograma de implementación por fases, el cual se puede compartir con los interesados *(Stakeholders)*.

Planificación y estimación.

- **Crear historias de usuario.** Las historias de usuario las escribe el facilitador *(Product Owner)*, y se diseñan para asegurar el cumplimiento de los requisitos del producto propuesto por el cliente, además estas son incorporadas a la lista priorizada del producto.

- **Estimar historias de usuario.** El dueño de producto *(Product Owner)* explica las historias de usuario, al facilitador *(Scrum Master)* y el equipo Scrum para que estimen los esfuerzos necesarios para desarrollar la funcionalidad descrita allí.

- **Comprometer historias de usuario.** El equipo Scrum, se compromete a entregar al dueño de producto *(Product Owner)* las historias de usuarios revisadas y aprobadas para un ciclo *(Sprint)*.

- **Identificar tareas.** En base a las historias de usuario comprometidas se crean tareas específicas en una lista de tareas.

- **Estimar tareas.** El equipo Scrum estima los esfuerzos para cumplir con las tareas identificadas en la lista de tareas.

- **Crear la lista de pendientes *(Sprint Backlog)*.** El equipo Scrum elabora una lista de pendientes *(Sprint Backlog)*, este contiene las tareas que deben ser completadas en un ciclo *(Sprint)*.

Implementación.

- **Crear entregables.** El equipo Scrum trabaja en las tareas de la lista de pendientes *(Sprint Backlog)* para crear entregables. Usualmente se usa un tablero *(Scrumboard)*, para hacer seguimiento a las actividades del ciclo *(Sprint)*.

- **Realizar Scrum diario.** Es una reunión que se lleva a cabo diariamente, en donde los miembros del equipo Scrum se actualizan para conocer progresos e impedimentos que se pueden llegar a presentar.

- **Refinamiento de la lista priorizada del producto.** Se puede considerar hacer una reunión de seguimiento, para realizar cambios o actualizaciones a lista priorizada del producto e incorporarlos a este si fuera necesario.

Revisión y retrospectiva.

- **Demostrar y validar el ciclo *(Sprint)*.** En una reunión de revisión se muestran los entregables del ciclo *(Sprint)* al dueño de producto *(Product Owner)* y a los interesados *(Stakeholders)*, con el propósito de asegurar que se aprueben y acepten.

- **Retrospectiva del ciclo *(Sprint)*.** Es considerada como una reunión de aprendizaje, para analizar las lecciones aprendidas durante el ciclo *(Sprint)* que va a finalizar. Es importante documentar estas lecciones para aplicarlas a próximos ciclos *(Sprints)*.

Lanzamiento.

- **Enviar entregables.** Los entregables aceptados se entregan o en su efecto se envían a los interesados *(Stakehorlders)* definidos. En el documento acuerdo de entregables funcionales *(Working Deliverables Agreement)*, allí se documentan las conclusiones del ciclo *(Sprint)*.

- **Retrospectiva del proyecto.** Los interesados *(Stakeholders)* se reúnen junto con el equipo Scrum para hacer una retrospectiva del proyecto e identificar y documentar las lecciones aprendidas.

Scrum es un marco de trabajo que tiene gran aplicabilidad, este se puede adaptar a cualquier tipo de entorno productivo en el que la demanda de los clientes evolucione rápidamente, es muy práctico y se centra más en las personas que en los procesos. Es muy fácil de implementar y muy popular alrededor del mundo ya que permite conseguir resultados muy rápidos.

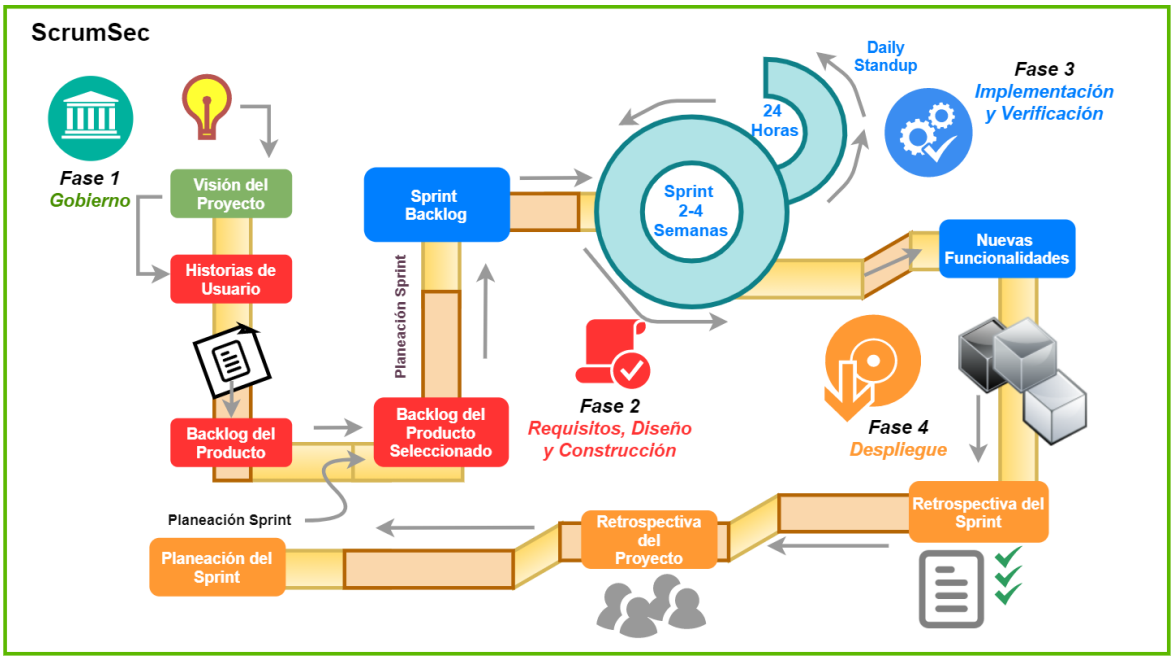
Es importante que cualquier organización que adopte este marco, este en la capacidad de dar un paso hacia adelante y conozca cuáles son los fundamentos del agilísimo, ya que Scrum invita a trabajar y dar avances importantes desde el primer momento, y que esto afecte en lo menos posible en tiempos, costes y a los equipos de trabajo al interior de esta.

11. Marco de buenas prácticas para desarrollar código seguro

Construir software de calidad, ágil y rápido, es tan importante como construir software seguro. Es un hecho que la tecnología avanza, pero consigo trae vulnerabilidades y amenazas que exponen a las organizaciones y a sus usuarios. Es por esto por lo que el marco de buenas prácticas para desarrollar código seguro implícitamente busca dar confiabilidad, protección y seguridad, al entorno de desarrollo de software desde Scrum como metodología ágil.

ScrumSec *(Scrum Secure),* es un marco de buenas prácticas que busca incorporar elementos de seguridad, de las metodologías de desarrollo de código seguro como: *SAMM*, *BSIMM* y *Microsoft SDL*, a los procesos de Scrum como metodología ágil.

Figura 12. ScrumSec – Diagrama marco buenas prácticas.



Fuente. Elaboración propia.

Este consiste en cuatro fases, y a su vez, en 30 actividades que mejoran la práctica de seguridad en el desarrollo de software. Las fases son:

- Gobierno.
- Requisitos, diseño y construcción.
- Implementación y verificación.
- Despliegue.

Dentro de estas fases se encuentran los 19 procesos de Scrum, los cuales contienen un identificador asociado al proceso de Scrum seguido del número, tal y como lo muestra la Tabla 13.

Tabla 13. Marco – procesos Scrum + Secure.

Scrum		Secure
Inicio	IN-1. Crear la visión del proyecto	Gobierno
	IN-2. Identificar al facilitador *(Scrum Master)* e interesado(s) *(Stakeholder(s))*	
	IN-3. Formar Equipos Scrum	
	IN-4. Definir la épica(s)	Requisitos, Diseño y Construcción
	IN-5. Crear la lista priorizada del producto	
	IN-6. Realizar la planificación de lanzamiento	
Planificación y Estimación	PE-7. Crear historias de usuario	
	PE-8. Estimar historias de usuario	
	PE-9. Comprometer historias de usuario	
	PE-10. Identificar tareas	
	PE-11. Estimar tareas	
	PE-12. Crear la lista de pendientes *(Sprint Backlog)*	Implementación y Verificación
Implementación	IM-13. Crear entregables	
	IM-14. Realizar Scrum diario	
	IM-15. Refinar la lista priorizada del producto	
Revisión y Retrospectiva	RY-16. Demostrar y validar el ciclo *(Sprint)*	Despliegue
	RY-17. Retrospectiva del ciclo *(Sprint)*	
Lanzamiento	LA-18. Enviar entregables	
	LA-19. Retrospectiva del proyecto	

Fuente. Elaboración propia.

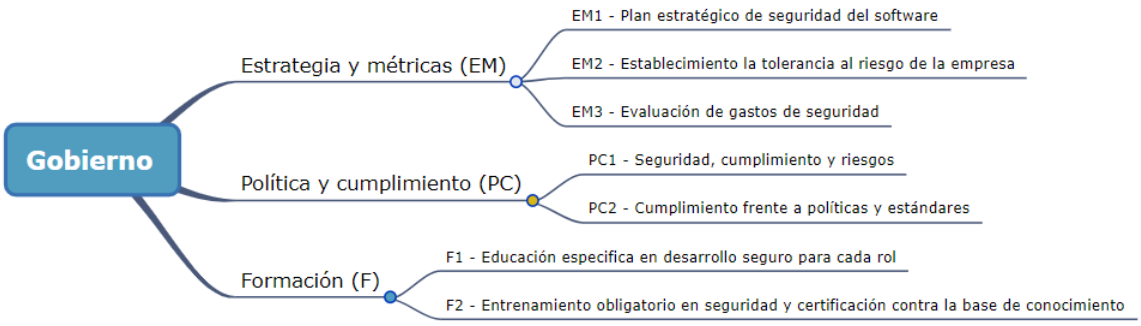
11.1. Primera fase. Gobierno

Incorporando al gobierno de TI, el marco de buenas prácticas para desarrollar código seguro, tanto sus fases y actividades se asegura la adopción de este al interior de la organización. El gobierno ayuda a implementar y alinear, los procesos y planes de las tecnologías de la información, así como el desarrollo de aplicaciones con enfoque seguro.

En esta fase de inicio con Scrum, el interesado *(Stakeholder)* con el dueño de producto *(Product Owner)* definen los objetivos estratégicos del negocio aplicados al producto de software que será puesto en producción, constituyen el equipo de trabajo, y se identifican aspectos de seguridad como el plan estratégico, la tolerancia al riesgo, y se hace una evaluación de los gastos.

En la Figura 13**,** se muestran las actividades que mejoran la práctica de seguridad desde el gobierno.

Figura 13. Gobierno - Actividades que mejoran la práctica de seguridad.



Fuente. Elaboración propia.

IN-1. Crear la visión del proyecto. Hace parte de la fase de inicio de Scrum. Los clientes e interesados del proyecto toman decisiones e identifican el valor que genera este una vez haya finalizado, aquí se evalúa la seguridad como un eje transversal del mismo. Los elementos de seguridad involucrados son:

- **EM1 - Plan estratégico de seguridad del software.** Estima el plan estratégico de riesgo del negocio, asegura la existencia de un programa de aseguramiento de software.

- **EM2 - Establecimiento la tolerancia al riesgo de la empresa.** Establece, clasifica y mide las aplicaciones basadas en riesgo y su tolerancia. Tiene un entendimiento organizacional importante, puede construirse en base a proyectos como eje central del negocio. (Ver Anexo A. Análisis de Riesgo con OWASP).

- **EM3 - Evaluación de gastos de seguridad.** Estima las perdidas en base a costes y problemas de seguridad, y considera los gastos por perdida potencial por proyecto.

- **PC1 - Seguridad, cumplimiento y riesgos.** Identifica, crea y mantiene los lineamientos de cumplimiento, establece prácticas de auditoria de riesgos en aplicaciones.

- **PC2 - Cumplimiento frente a políticas y estándares.** Garantiza el cumplimiento frente a políticas y estándares en conformidad a los proyectos de la organización. (Ver Anexo B. Normas y Cumplimiento).

La Tabla 14, muestra las entradas, herramientas y salidas con enfoque seguro de crear la visión del proyecto, el texto rojo indica los nuevos elementos.

Tabla 14. Marco – Crear la visión del proyecto. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Caso de negocio del proyecto EM1 - Plan estratégico de seguridad del software Identificación del dueño del producto *(Product Owner)* Proyecto de prueba Prueba de concepto	Reunión de visión del proyecto Sesiones de clasificación de datos y aplicaciones basadas en riesgo Sesiones de diseño de aplicación conjunta Reunión de objetivos y clasificación de seguridad	Dueño del producto *(Product Owner)* identificado Declaración de la visión del proyecto Plan de aseguramiento organizacional Acta constitutiva del proyecto Lista de riesgos críticos de software

EM2 - Establecimiento de la tolerancia al riesgo de la empresa PC1- Seguridad, cumplimiento y riesgos PC2- Cumplimiento frente a políticas y estándares Visión de la empresa Misión de la empresa Estudio del mercado EM3 - Evaluación de gastos de seguridad	Análisis e investigación de políticas, cumplimiento en base a seguridad Análisis FODA Análisis de brechas y seguridad	Practica de auditoria de proyecto Puntos de control de cumplimiento Presupuesto del proyecto

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IN-2. Identificar al facilitar *(Scrum Master)* e interesado(s) *(Stakeholder(s))*. Hace parte de la fase de inicio de Scrum. En conjunto el dueño del producto *(Product Owner)* y el área de recursos humanos de la organización, definen las competencias que tendrá el facilitador *(Scrum Master)*, el cual debe poseer conocimientos del negocio, entender el marco Scrum, la estrategia de seguridad de la organización, aportar a la gestión y garantizar el éxito del proyecto. Los elementos de seguridad involucrados son:

- **F1 - Educación especifica en desarrollo seguro para cada rol.** Ofrece capacitación al personal que interviene en el desarrollo del proyecto, en temas de seguridad específicos para cada rol.

- **F2 - Entrenamiento obligatorio en seguridad y certificación contra la base de conocimiento.** Establece lineamientos de entrenamiento de seguridad, en un plan de carrera, con soporte a seguridad, exámenes y certificaciones por rol.

La Tabla 15, muestra las entradas, herramientas y salidas con enfoque seguro de Identificar al facilitador *(Scrum Master)* e interesado(s) *(Stakeholder(s))*, el texto rojo indica los nuevos elementos.

Tabla 15. Marco – Identificar al Scrum Master y Stakeholder(s). Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Dueño del producto *(Product Owner)* Declaración de la visión del proyecto Requerimientos de las personas Disponibilidad y compromiso de las personas Matriz de recursos organizacionales Matriz de las destrezas requeridas F1 - Educación especifica en desarrollo seguro para cada rol F2 - Entrenamiento obligatorio en seguridad y certificación contra la base de conocimiento	Criterios de selección Asesoramiento de expertos en recursos humanos Capacitación y costos de capacitación Contratación de mentor en temas de seguridad Plataformas de capacitación con certificaciones y cursos de seguridad en desarrollo Costos de recursos	Facilitador *(Scrum Master)* identificado Interesado(s) *(Stakeholder(s))* identificados Personal capacitado y certificado en temas de seguridad acorde a su rol

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IN-3. Formar Equipos Scrum. Hace parte de la fase de inicio de Scrum. El dueño de producto *(Product Owner)* y el facilitador *(Scrum Master)*, identifican las competencias

necesarias de los integrantes del equipo de trabajo, con el fin de encontrar los roles que garanticen el avance del proyecto, los cuales deben poseer conocimientos del negocio, entender el marco Scrum, la estrategia de seguridad de la organización, aportar a la gestión y garantizar el éxito del proyecto. Los elementos de seguridad involucrados son:

- **F1 - Educación especifica en desarrollo seguro para cada rol.** Ofrece capacitación al personal que interviene en el desarrollo del proyecto, en temas de seguridad específicos para cada rol.

- **F2 - Entrenamiento obligatorio en seguridad y certificación contra la base de conocimiento.** Establece lineamientos de entrenamiento de seguridad, en un plan de carrera, con soporte a seguridad, exámenes y certificaciones por rol.

La Tabla 16**,** muestra las entradas, herramientas y salidas con enfoque seguro de formar equipos Scrum, el texto rojo indica los nuevos elementos.

Tabla 16. Marco – Formar equipos Scrum. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Dueño de producto *(Product Owner)* Facilitador *(Scrum Master)* Declaración de la visión del proyecto Requisitos de las personas Disponibilidad y compromiso de las personas Matriz de recurso organizacional Matriz de las destrezas requeridas	Selección del equipo Scrum Asesoramiento de expertos en recursos humanos Capacitación y costos de capacitación Contratación de mentor en temas de seguridad Plataformas de capacitación con certificaciones y cursos de seguridad en desarrollo Costos del personal	Equipo Scrum identificado Substitutos Plan de colaboración Plan de formación del equipo Personal capacitado y certificado en temas de seguridad acorde a su rol

Requerimientos de recursos F1 - Educación especifica en desarrollo seguro para cada rol F2 - Entrenamiento obligatorio en seguridad y certificación contra la base de conocimiento	Capacitación y costos de capacitación Costos de recursos	

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

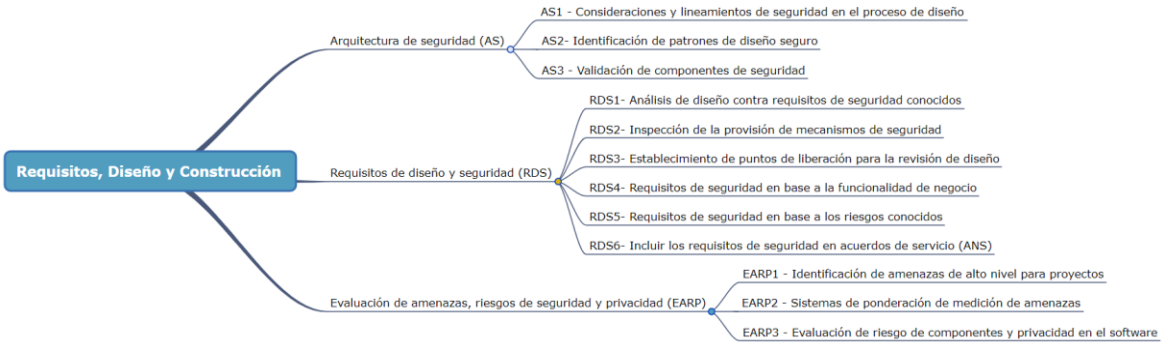
11.2. Segunda fase. Requisitos, diseño y construcción.

Las actividades de seguridad se deben incluir como requisitos dentro del diseño, y se deben analizar desde el inicio del proyecto, tanto, así como en las especificaciones técnicas, como funcionales. Es esencial que los problemas asociados a seguridad se evalúen desde esta fase, ya que los costos de remediación incluso en la construcción de software pueden superar las expectativas de la organización, e impactar negativamente sus características o funciones determinadas.

En esta fase de planificación y estimación con Scrum, el facilitador *(Scrum Master)* y el equipo en conjunto muestran al dueño de producto *(Product Owner)*, un estimado en tiempo y esfuerzo para completar lo definido por este último en la lista de pendientes *(Sprint Backlog)* en las historias de usuario.

En la Figura 14**,** se muestran las actividades que mejoran la práctica de seguridad desde los requisitos, diseño y construcción.

Figura 14. Requisitos, diseño y construcción - Actividades que mejoran la práctica de seguridad.



Fuente. Elaboración propia.

IN-4. Definir épica(s). Hace parte de la fase de inicio de Scrum. Las épicas son las definiciones generales del producto que se desea construir durante el proyecto. Estas son presentadas en la visión y en los requerimientos del cliente. Durante la definición de las épicas es importante incluir los lineamientos y consideraciones en seguridad que el desarrollo de código debe incluir, tanto, así como los patrones de diseño seguro. Los elementos de seguridad involucrados son:

- **AS1 - Consideraciones y lineamientos de seguridad en el proceso de diseño.** Mantiene y aplica las recomendaciones y principios de seguridad, mediante una lista con el marco de trabajo de diseño de software. Estas recomendaciones hacen parte de los equipos y estos son conscientes de los principios de diseño y desarrollo seguro. (Ver Anexo C. Recomendaciones Generales de Seguridad).

- **AS2 - Identificación de patrones de diseño seguro.** Se promueve la seguridad en la infraestructura y los servicios de desarrollo, desde la arquitectura basada en patrones previstos y conocidos. (Ver Anexo D. Patrones de diseño seguro).

La Tabla 17, muestra las entradas, herramientas y salidas con enfoque seguro de la definición de la épica(s), el texto rojo indica los nuevos elementos.

Tabla 17. Marco – Definir épica(s). Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo Principal de Scrum Declaración de la visión del proyecto Interesado(s) *(Stakeholder(s))* AS1 - Consideraciones y lineamientos de seguridad en el proceso de diseño Solicitudes de cambios aprobadas Solicitudes de cambios no aprobadas Riesgos del portafolio y del programa AS2- Identificación de patrones de diseño seguro Leyes y regulaciones Contratos aplicables Información de proyectos previos	Reuniones del grupo de usuarios Talleres de historias de usuario Reuniones del grupo de enfoque en seguridad Entrevistas al usuario o cliente Documentación de patrones de aseguramiento Cuestionarios Técnicas de identificación de riesgos	Épica(s) Prototipos Cambios aprobados Protocolo de desarrollo en las aplicaciones Riesgos identificados Comparativo de funciones de usuario de diseño seguro

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IN-5. Crear la lista priorizada del producto. Hace parte de la fase de inicio de Scrum. Es un documento que contiene las funcionalidades que necesita la aplicación del cliente y el negocio. Algunos beneficios están asociados a, optimizar los distintos recursos, aumentar la productividad, hacer las validaciones preliminares de los componentes de seguridad, y

analizar los requisitos del diseño para reducir la materialización de los riesgos. Los elementos de seguridad involucrados son:

- **AS3 - Validación de componentes de seguridad.** Valida la utilización de componentes de seguridad, en base a arquitecturas y plataformas de referencia. Verificando patrones de trabajo de desarrollo, bajo esquemas controlados.

- **RDS1- Análisis de diseño contra requisitos de seguridad conocidos.** Identifica y analiza el diseño contra requisitos de seguridad, además de la superficie de ataques en el software.

La Tabla 18, muestra las entradas, herramientas y salidas con enfoque seguro de la creación de la lista priorizada del producto, el texto rojo indica los nuevos elementos.

Tabla 18. Marco – Crear el Backlog Priorizado del Producto. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Épica(s) AS3 - Validación de componentes de seguridad Prototipos Interesado(s) *(Stakeholder(s))* Declaración de la visión del proyecto Requerimientos del negocio RDS1- Análisis de diseño contra requisitos de seguridad conocidos	Métodos de priorización de historias del usuario Talleres de historias del usuario Planificación de valor Técnicas de evaluación del riesgo Documentación del proyecto de los perímetros de ataque Estimación del valor del proyecto Métodos de estimación de historias de usuario	Lista priorizada del producto Criterios de terminado Lista de validación con componentes y vectores de ataque de seguridad identificados Proceso ligero para conducir revisiones de seguridad a nivel del proyecto

Solicitudes de cambios aprobadas Riesgos identificados Contratos Aplicables		

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IN-6. Realizar la planificación de lanzamiento. Hace parte de la fase de inicio de Scrum. Son los puntos de entrega de los incrementos de producto que se hacen al cliente para su paso a producción, lo cual constituye en la generación de un cronograma, y en la información por parte del cliente, si cada vez que se genere un producto este será lanzado inmediatamente a producción, o por el contrario se lanzara en fechas específicas. En la planificación antes del lanzamiento se inspecciona la provisión de los mecanismos de seguridad suministrados para la aplicación, tanto, así como la revisión de diseño y código. Los elementos de seguridad involucrados son:

- **RDS2- Inspección de la provisión de mecanismos de seguridad.** Mediante la provisión de mecanismos de seguridad, implementa servicios de revisión de diseño para los equipos de proyecto.

- **RDS3- Establecimiento de puntos de liberación para la revisión de diseño.** Genera una vista de los puntos débiles en el diseño y la revisión, de acuerdo con los lineamientos dados en la visión del proyecto y arquitectura de seguridad, antes del lanzamiento del producto terminado.

La Tabla 19**,** muestra las entradas, herramientas y salidas con enfoque seguro de realizar la planificación de lanzamiento, el texto rojo indica los nuevos elementos.

Tabla 19. Marco – Lanzar la planificación de lanzamiento. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Interesado(s) *(Stakeholder(s))*	Sesiones de planificación del lanzamiento	Cronograma de planificación del lanzamiento

Declaración de la visión del proyecto Lista priorizada del producto Criterios de terminado RDS2- Inspección de la provisión de mecanismos de seguridad Requerimientos del negocio RDS3- Establecimiento de puntos de liberación para la revisión de diseño Calendario de días festivos	Métodos de priorización del lanzamiento Sesión de análisis de inspección con los miembros del equipo, en base a arquitectura de seguridad	Cronograma de revisión y puntos de liberación y seguridad Duración del ciclo *(Sprint)* Clientes meta para el lanzamiento Lista priorizada del producto refinado

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

PE-7. Crear historias de usuario. Hace parte de la fase de planificación y estimación de Scrum. Da una visión al dueño del producto *(Product Owner)* de las funcionalidades y eventos que tendrá el producto final terminado de software, incluyendo los requerimientos mínimos de aceptación de cada historia de usuario, las tareas a ser ejecutadas, el análisis de los *QA*, los requisitos de seguridad en base a la funcionalidad de negocio, y en base a riesgos conocidos. Los elementos de seguridad involucrados son:

- **RDS4- Requisitos de seguridad en base a la funcionalidad de negocio.** Deduce los requisitos de seguridad con conocimiento previo de acuerdo con el negocio, además evalúa los lineamientos y cumplimiento respecto a las necesidades del cliente.

- **RDS5- Requisitos de seguridad en base a los riesgos conocidos.** Se especifican los requisitos de seguridad en base a riesgos conocidos, y en la retroalimentación de otras actividades de seguridad. (Ver Anexo E. Riesgos Conocidos).

La Tabla 20**,** muestra las entradas, herramientas y salidas con enfoque seguro de crear las historias de usuario, el texto rojo indica los nuevos elementos.

Tabla 20. Marco – Crear historias de usuario. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Lista priorizada del producto Criterios de terminado RDS4- Requisitos de seguridad en base a la funcionalidad de negocio RDS5- Requisitos de seguridad en base a los riesgos conocidos Prototipos Interesado(s) *(Stakeholder(s))* Épica(s) Requerimientos del negocio Leyes y regulaciones Contratos aplicables	Experiencia en la redacción de historias de usuario Talleres de historias del usuario Reuniones del grupo de usuarios Reuniones del grupo de enfoque Sesiones de especificación de requisitos de seguridad y lineamientos de cumplimiento Matriz de control de acceso para proyectos Entrevistas al cliente o usuario Cuestionarios	Historias de usuarios Criterio de aceptación de historias del usuario Lista priorizada del producto actualizada Escenarios de ataques en contra de la lógica del negocio Toma de decisiones entre características de seguridad y diseño Prototipos actualizados o refinados

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

PE-8. Estimar historias de usuario. Hace parte de la fase de planificación y estimación de Scrum. Luego de la definición de las historias de usuario que serán parte del ciclo *(Sprint)*, se identifican los esfuerzos de ejecución para la implementación de estas. El equipo de desarrollo hace un análisis de las tareas, riesgos, esfuerzo y requisitos de seguridad para dedicarse a este trabajo en el ciclo *(Sprint)*, también se da un alcance a los

acuerdos de nivel de servicio de los requisitos. Los elementos de seguridad involucrados son:

- **RDS3- Establecimiento de puntos de liberación para la revisión de diseño.** Genera una vista de los puntos débiles en el diseño y la revisión, de acuerdo con los lineamientos dados en la visión del proyecto y arquitectura de seguridad, antes del lanzamiento del producto terminado.

- **RDS6- Incluir los requisitos de seguridad en acuerdos de servicio (ANS).** Incorpora los requisitos de seguridad a acuerdos con proveedores, y con las mesas de gestión de servicios de TI al interior de la organización, para centralizar los esfuerzos de seguridad en el proyecto, en el evento que se presenten incidentes de seguridad que deban ser remediados en tiempos específicos.

La Tabla 21**,** muestra las entradas, herramientas y salidas con enfoque seguro de estimar las historias de usuario, el texto rojo indica los nuevos elementos.

Tabla 21. Marco – Estimar historias de usuario. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum RDS3- Establecimiento de puntos de liberación para la revisión de diseño RDS6- Incluir los requisitos de seguridad en acuerdos de servicio (ANS) Historias de usuarios	Reuniones de planificación del ciclo *(Sprint)* Reuniones de revisión de la lista priorizada del producto Sesiones de especificación de requisitos de seguridad y lineamientos de cumplimiento Métodos de estimación Sesión de identificación requisitos de seguridad específicos y riesgos	Historias del usuario estimadas Lista priorizada del producto actualizada Escenarios de ataques en contra de la lógica del negocio Criterios de aceptación del usuario actualizado Acuerdo de nivel de servicios (ANS), de incidentes de seguridad

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

PE-9. Comprometer historias de usuario. Hace parte de la fase de planificación y estimación de Scrum. La ejecución de las tareas que darán cumplimiento a las historias de usuario de un ciclo *(Sprint)* es fundamental, el compromiso es del equipo de trabajo, además del cumplimiento de los tiempos con las mesas de gestión de servicios de TI al interior de la organización. Los elementos de seguridad involucrados son:

- **RDS6- Incluir los requisitos de seguridad en acuerdos de servicio (ANS).** Incorpora los requisitos de seguridad a acuerdos con proveedores, y con las mesas de gestión de servicios de TI al interior de la organización, para centralizar los esfuerzos de seguridad en el proyecto, en el evento que se presenten incidentes de seguridad que deban ser remediados en tiempos específicos.

La Tabla 22**,** muestra las entradas, herramientas y salidas con enfoque seguro de comprometer las historias de usuario, el texto rojo indica los nuevos elementos.

Tabla 22. Marco – Comprometer historias de usuario. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum RDS6- Incluir los requisitos de seguridad en acuerdos de servicio (ANS) Historias del usuario estimadas Duración del ciclo *(Sprint)* Velocidad del ciclo *(Sprint)* anterior	Reuniones de planificación del sprint Sesión de identificación requisitos de seguridad específicos y riesgos Técnicas de comunicación	Historias de usuario comprometidas Acuerdo de nivel de servicios (ANS), de incidentes de seguridad

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

PE-10. Identificar tareas. Hace parte de la fase de planificación y estimación de Scrum. Se identifican las tareas comunes que contienen las historias de usuario que serán ejecutadas durante el ciclo *(Sprint)*. Para ello se hace un análisis de esfuerzo, riesgos e

identificación de amenazas de alto nivel, y se hace una ponderación de medición a las amenazas identificadas. Los elementos de seguridad involucrados son:

- **EARP1 - Identificación de amenazas de alto nivel para proyectos.** Desarrolla y mantiene modelos de amenazas para cada aplicación. Genera un perfil del atacante desde la arquitectura de software.

- **EARP2 - Sistemas de ponderación de medición de amenazas.** Desarrolla y mantiene modelos de casos de abuso, y un sistema de ponderación para la medición de amenazas.

La Tabla 23**,** muestra las entradas, herramientas y salidas con enfoque seguro de identificar las tareas, el texto rojo indica los nuevos elementos.

Tabla 23. Marco – Identificar tareas. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum EARP1 - Identificación de amenazas de alto nivel para proyectos EARP2 - Sistemas de ponderación de medición de amenazas Historias de usuario comprometidas	Reuniones de planificación del ciclo *(Sprint)* Uso de modelos de casos de abuso por proyecto Reunión de validación de tipos de atacantes Descomposición Determinación de dependencia	Lista de tareas Marco de toma de decisiones para los equipos de proyecto Listado de amenazas y ponderación de estas Historias de usuario comprometidas actualizadas Dependencias

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

PE-11. Estimar tareas. Hace parte de la fase de planificación y estimación de Scrum. La estimación de tareas consiste en la identificación de esfuerzos y tiempo por parte del equipo de desarrollo, que se requiere para la ejecución de las tareas, de acuerdo con las historias de usuario definidas. Es una de las actividades más importantes, ya que incluye

identificar y evaluar el riesgo de los componentes y la privacidad en el software. Los elementos de seguridad involucrados son:

- **EARP3 - Evaluación de riesgo de componentes y privacidad en el software.** Identifican aspectos del software a nivel funcional que deben ser revisados de manera exhaustiva, y se mide el nivel de impacto de privacidad en base a pautas específicas.

La Tabla 24**,** muestra las entradas, herramientas y salidas con enfoque seguro de estimar las tareas, el texto rojo indica los nuevos elementos.

Tabla 24. Marco – Estimar tareas. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum EARP3 - Evaluación de riesgo de componentes y privacidad en el software Lista de tareas Criterios de aceptación de historia de usuario Dependencias Riesgos identificados	Reuniones de planificación del ciclo *(Sprint)* Sesión de revisión de identificación de riesgos por componentes críticos Niveles de impacto de privacidad Criterios de estimación Métodos de estimación	*Effort Estimated Task List* Lista de tareas actualizada

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

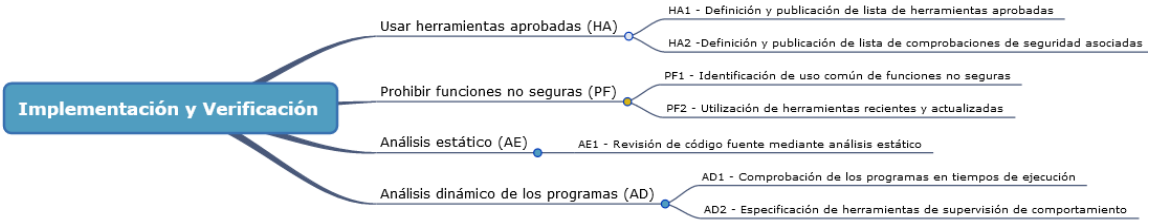
11.3. Tercera fase. Implementación y verificación.

El enfoque de seguridad en la implementación y verificación es fundamental, ya que, en esta fase, varios de los elementos del presente marco se encuentran en la codificación, en el desarrollo de software y en las prácticas que propone la metodología Scrum. Aquí se hacen pruebas de seguridad, orientadas al trabajo típicamente de aseguramiento de calidad, revisiones y evaluación.

En esta fase de implementación con Scrum, el equipo conoce las tareas que se deben ejecutar para cumplir con los elementos de las historias de usuario definidas dentro de la lista de pendientes *(Sprint Backlog)*.

En la Figura 15**,** se muestran las actividades que mejoran la práctica de seguridad desde la implementación y verificación.

Figura 15. Implementación y verificación - Actividades que mejoran la práctica de seguridad.



Fuente. Elaboración propia.

PE-12. Crear la lista de pendientes *(Sprint Backlog)*. Hace parte de la fase de planificación y estimación de Scrum. El equipo se compromete a implementar las historias de usuario durante el ciclo *(Sprint)*, obtenidas de la lista de producto *(Product Backlog)*. El tiempo recomendado para esta actividad es de máximo 4 semanas. Dentro de la lista de pendientes *(Sprint Backlog)*, se define la lista de herramientas aprobadas, y como se van a realizar las revisiones de seguridad asociadas al proyecto en curso, así como la identificación y uso de funciones no seguras. Estas actividades son importantes realizarlas antes de su salida a producción, o que queden marcadas como productos terminados dentro de los entregables al cliente. Los elementos de seguridad involucrados son:

- **HA1 - Definición y publicación de lista de herramientas aprobadas.** El equipo de desarrollo define y publica mediante una lista, todas las herramientas que han de ser aprobadas y asociadas a seguridad. Se deben usar versiones recientes y actualizadas que permitan dar protección y funciones adicionales.

- **HA2 -Definición y publicación de lista de comprobaciones de seguridad asociadas.** El equipo de desarrollo define y publica mediante una lista, como opciones de seguridad las advertencias vinculadas al desarrollo de código y

funciones potencialmente no seguras, que deberán ser remediadas antes del producto terminado.

- **PF1 - Identificación de uso común de funciones no seguras.** Analiza e identifica las funciones que serán trabajadas durante el proyecto de desarrollo de código, con el fin de prohibir las que se consideran inseguras.

La Tabla 25, muestra las entradas, herramientas y salidas con enfoque seguro de crear la lista de pendientes *(Sprint Backlog)*, el texto rojo indica los nuevos elementos.

Tabla 25. Marco – Crear la lista de pendientes *(Sprint Backlog)*. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum HA1 - Definición y publicación de lista de herramientas aprobadas HA2 -Definición y publicación de lista de comprobaciones de seguridad asociadas PF1 - Identificación de uso común de funciones no seguras *Effort Estimated Task List* Duración del ciclo *(Sprint)* Dependencias Calendario del equipo	Reuniones de planificación del ciclo *(Sprint)* Reunión de validación de herramientas del proyecto Herramientas de seguimiento del ciclo *(Sprint)* Parámetros de seguimiento del ciclo *(Sprint)*	Lista de pendientes *(Sprint Backlog)* *Sprint Burndown Chart* Lista de herramientas aprobadas Lista de seguridad de advertencias y funciones no seguras

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IM-13. Crear entregables. Hace parte de la fase de implementación de Scrum. Da la posibilidad dentro del marco, de entregar productos funcionales al cliente, desde la

perspectiva de priorizar el valor, mitigar los riesgos, evaluar el uso de herramientas, revisar el código de manera estática y comprobar las aplicaciones en ambientes de laboratorio con aspectos de seguridad, claro está como prerrequisito antes de la entrega al cliente. Los elementos de seguridad involucrados son:

- **PF2 - Utilización de herramientas recientes y actualizadas.** La infraestructura de las aplicaciones debe contar con herramientas que sean recientes y actualizadas, con el fin de proveer a los entornos de desarrollo, seguridad en sus últimas versiones.

- **AE1 - Revisión de código fuente mediante análisis estático.** Se programan revisiones de código fuentes estáticas con frecuencia óptimas, por el equipo de proyecto.

- **AD1 - Comprobación de los programas en tiempos de ejecución.** Se debe proporcionar mediante el uso de herramientas, la verificación de las aplicaciones y su comportamiento, para detectar problemas de seguridad críticos.

La Tabla 26**,** muestra las entradas, herramientas y salidas con enfoque seguro de crear entregables, el texto rojo indica los nuevos elementos.

Tabla 26. Marco – Crear entregables. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Lista de pendientes *(Sprint Backlog)* Tablero *(Scrumboard)* Lista de impedimentos *(Impediment Log)* PF2 - Utilización de herramientas recientes y actualizadas	Experiencia del equipo Software Otras herramientas de desarrollo Matriz de estado de herramientas Herramientas de análisis estático y dinámico de código	Entregables del sprint* Tablero *(Scrumboard)* actualizado* Lista de impedimentos *(Impediment Log)* actualizado* Solicitudes de cambios no aprobadas Listado de herramientas por estado de proyecto

AE1 - Revisión de código fuente mediante análisis estático AD1 - Comprobación de los programas en tiempos de ejecución Cronograma de planificación del lanzamiento Dependencias		Riesgos identificados Riesgos mitigados Dependencias actualizadas

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IM-14. Realizar Scrum diario. Hace parte de la fase de implementación de Scrum. Se planean reuniones de seguimiento diario, para informar los avances a nivel general del Sprint, así como las dificultades e impedimentos, y encontrar posibles soluciones al proyecto.

La Tabla 27**,** muestra las entradas, herramientas y salidas de realizar el Scrum diario.

Tabla 27. Marco – Realizar Scrum diario. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo Scrum Facilitador *(Scrum Master)* Gráfico de trabajo pendiente del ciclo *(Sprint Burndown Chart)* Lista de impedimentos *(Impediment Log)* Dueño de producto *(Product Owner)*	Scrum diario Tres preguntas diarias *War Room* Videoconferencia	Gráfico de trabajo pendiente del ciclo *(Sprint Burndown Chart)* actualizado Lista de impedimentos *(Impediment Log)* actualizado Equipo Scrum motivado Scrumboard actualizado Solicitud de cambios no aprobados

Experiencia del día anterior de trabajo Tablero *(Scrumboard)* Dependencias		Riesgos identificados Riesgos mitigados Dependencias actualizadas

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

IM-15. Refinar la lista priorizada de producto. Hace parte de la fase de implementación de Scrum. El refinamiento de la lista priorizada del producto es un documento que lista e incluye funcionalidades adicionales que desean alcanzar en el proyecto. Aquí se especifican posibles herramientas para hacer monitoreo de ciberseguridad e identificar problemas a reparar e incluir en la lista de producto priorizada. Los elementos de seguridad involucrados son:

- **AD2 - Especificación de herramientas de supervisión de comportamiento.** Mediante la identificación de herramientas se realiza el monitoreo de ciberseguridad de las aplicaciones con el fin de detectar posibles problemas de seguridad. (Ver Anexo F. Herramientas de Monitoreo (Cyber Risk Rating)).

La Tabla 28**,** muestra las entradas, herramientas y salidas con enfoque seguro de crear entregables, el texto rojo indica los nuevos elementos.

Tabla 28. Marco – Refinar la lista priorizada de producto. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum AD2 - Especificación de herramientas de supervisión de comportamiento Lista priorizada de producto Entregables rechazados	Reunión de revisión de la lista priorizada de producto Reunión de validación y especificación de necesidades Técnicas de comunicación	Listado de herramientas de ciberseguridad de supervisión de comportamiento Lista priorizada de producto actualizada Cronograma de planificación del lanzamiento actualizado

Solicitudes de cambios aprobados Solicitud de cambios rechazados Riesgos identificados Registro(s) de la retrospectiva del ciclo *(Sprint)* Dependencias Cronograma de planificación del lanzamiento	Otras técnicas de refinamiento de la lista priorizada de producto	

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

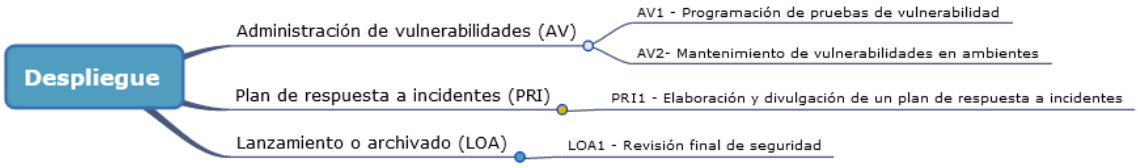
11.4. Cuarta fase. Despliegue.

El despliegue es la cuarta fase del marco, aquí se relacionan las actividades de la liberación del código, y de las aplicaciones que se desarrollaron a través de todo el proyecto. Desde el enfoque de seguridad se busca administrar de manera óptima las posibles vulnerabilidades que surgen, y se aborda la implementación de un plan de respuesta a incidentes que debe ser socializado al interior de la organización, para posteriormente hacer una revisión de seguridad final y archivar la documentación correspondiente sobre lo aprendido durante el proceso.

En esta fase de retrospectiva y lanzamiento con Scrum, el facilitador *(Scrum Master)* junto con el equipo analizan en conjunto los resultados operativos del ciclo *(Sprint)*, con el fin de identificar posibles oportunidades de mejora para el ciclo *(Sprint)* que inicia. Finalmente, el facilitador *(Scrum Master)* entrega al dueño de producto *(Product Owner)* y al cliente el producto de incremento terminado con base en seguridad.

En la Figura 16**,** se muestran las actividades que mejoran la práctica de seguridad desde el despliegue.

Figura 16. Despliegue - Actividades que mejoran la práctica de seguridad.



Fuente. Elaboración propia.

RY-16. Demostrar y validar el ciclo *(Sprint)*. Hace parte de la fase de revisión y retrospectiva de Scrum. Consiste en la exposición y validación por parte del dueño de producto *(Product Owner)*, clientes y usuarios del producto de incremento generado, que cumple con las expectativas del ciclo *(Sprint)*, de acuerdo con las historias de usuario, y a los elementos de seguridad que fueron incorporados en momentos específicos. De igual manera, dentro de los criterios de aceptación se encuentra la realización periódica de pruebas de vulnerabilidad, y el mantenimiento de estas, según se programe. Los elementos de seguridad involucrados son:

- **AV1 - Programación de pruebas de vulnerabilidad.** Se adopta al interior de la organización un plan de pruebas de vulnerabilidad, que dará visibilidad al estado del desarrollo de la aplicación con el paso del tiempo. Lo que pretende es hacer seguimiento a los productos terminados.

- **AV2 - Mantenimiento de vulnerabilidades en ambientes.** Se identifican las vulnerabilidades, y se les da un tratamiento de acuerdo con los planes de remediación dispuestos en la organización.

La Tabla 29**,** muestra las entradas, herramientas y salidas con enfoque seguro de demostrar y validar el ciclo *(Sprint)*, el texto rojo indica los nuevos elementos.

Tabla 29. Marco – Demostrar y validar el ciclo *(Sprint)*. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Entregables del ciclo *(Sprint)*	Reuniones de revisión del ciclo *(Sprint)* Análisis del valor ganado	Entregables aceptados Entregables rechazados

Lista de pendientes *(Sprint Backlog)* Criterios de terminado Criterio de aceptación de las historias del usuario AV1 - Programación de pruebas de vulnerabilidad AV2 - Mantenimiento de vulnerabilidades en ambientes Interesado(s) *(Stakeholder(s))* Cronograma de planificación del lanzamiento Riesgos identificados Dependencias	Formato de vulnerabilidades y mantenimiento Experiencia del Scrum (*Guidance Body)*	Riesgos actualizados Resultados del análisis del valor ganado Cronograma de planificación del lanzamiento actualizado Dependencias actualizadas

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

RY-17. Retrospectiva del ciclo *(Sprint)*. Hace parte de la fase de revisión y retrospectiva de Scrum. Esta sesión se realiza luego de finalizado el ciclo *(Sprint)*, y se identifican aspectos importantes que se dieron durante el mismo. Durante el lanzamiento y en la retrospectiva se incluye y divulga el plan de respuesta a incidentes de seguridad, que espera abordar el presente y futuros ciclos *(Sprints)*, en los productos terminados. Los elementos de seguridad involucrados son:

- **PRI1 - Elaboración y divulgación de un plan de respuesta a incidentes.** El plan de respuesta a incidentes debe incluir los planes de servicio de seguridad, la adopción del proceso y el análisis de la causa raíz para los incidentes.

La Tabla 30**,** muestra las entradas, herramientas y salidas con enfoque seguro de la retrospectiva del ciclo *(Sprint)*, el texto rojo indica los nuevos elementos.

Tabla 30. Marco – Retrospectiva del Sprint. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Facilitador *(Scrum Master)* Equipo Scrum Salidas de demostrar y validar el ciclo *(Sprint)* PRI1 - Elaboración y divulgación de un plan de respuesta a incidentes Dueño de producto *(Product Owner)*	Reunión de retrospectiva del ciclo *(Sprint)* ECVP *Speed Boat* Parámetros y técnicas de medición	*Agreed Actionable Improvements* *Assigned Action Items* y fechas límite Elementos no funcionales propuestos para la lista priorizada de productos Registros de la retrospectiva del ciclo *(Retrospect Sprint Log(s))* Lecciones aprendidas del equipo de Scrum Plan de respuesta a incidentes

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

LA-18. Enviar entregables. Hace parte de la fase de lanzamiento de Scrum. Los entregables aceptados se envían a los clientes como productos terminados, y documentación adicional para el proceso.

La Tabla 31**,** muestra las entradas, herramientas y salidas de enviar entregables.

Tabla 31. Marco – Enviar entregables. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Dueño de producto *(Product Owner)* Interesado(s) *(Stakeholder(s))*	Métodos de desplazamiento organizacional Plan de comunicación	Acuerdo de entregables funcionales Entregables funcionales

Entregables aceptados Cronograma de planificación del lanzamiento Facilitador *(Scrum Master)* Equipo Scrum Criterios de aceptación de las historias del usuario Plan de pilotaje		Lanzamientos del producto

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

LA-19. Retrospectiva del proyecto. Hace parte de la fase de lanzamiento de Scrum. Es una reunión que evalúa a nivel general el proyecto, y se identifican e interiorizan los aspectos más relevantes, aciertos y desaciertos de este. Se hace una revisión final con el fin de inspeccionar todas las actividades de seguridad realizadas sobre el proyecto de software antes de su lanzamiento.

- **LOA1 - Revisión final de seguridad.** Consiste en la revisión final del proyecto a nivel de seguridad, de los resultados y el rendimiento de calidad, límites de errores, vulnerabilidades, mitigación e impacto y capacidad de respuesta a incidentes.

La Tabla 32**,** muestra las entradas, herramientas y salidas con enfoque seguro de la retrospectiva del proyecto, el texto rojo indica los nuevos elementos.

Tabla 32. Marco – Retrospectiva del proyecto. Entradas, herramientas y salidas.

Entradas	Herramientas	Salidas
Equipo principal de Scrum Facilitador *(Chief Scrum Master)*	Reunión de la retrospectiva del proyecto Otras herramientas para la retrospectiva del proyecto	*Agreed Actionable Improvements* *Assigned Action Items* y fechas límite

Dueño de producto *(Chief Product Owner)* Interesados *(Stakeholder(s))* LOA1 - Revisión final de seguridad		Estado de revisión final de seguridad Elementos no funcionales propuestos para la lista priorizada del producto Recomendaciones del Scrum *Guidance Body* actualizadas

Fuente. Tomado y adaptado de (SCRUMstudy, 2017).

ScrumSec, aborda los problemas de seguridad en el desarrollo de código, desde una perspectiva, dinámica y constante al interior de los equipos de trabajo y desarrollo. Es importante que se conserve el agilísimo, pero que no se deje de lado la seguridad ni la privacidad de los datos. El éxito de la aplicación, además de ser funcional es que corresponda a unos niveles de seguridad aceptados y mitigados durante todo el ciclo de construcción del desarrollo.

Tal y como lo muestra la Tabla 33, con las cuatro fases se busca una alineación estratégica transversal, que ayude a disminuir los costos de remediación de incidentes de seguridad en un 25%. Es importante que los procesos de desarrollo sean eficientes y los productos terminados contemplen las actividades propuestas para cada fase.

Tabla 33. Marco – Fases ScrumSec, actividades.

Fases	ScrumSec	Actividades
Gobierno	Estrategia y métricas (EM)	7
	Política y cumplimiento (PC)	
	Formación (F)	
Requisitos, Diseño y Construcción	Arquitectura de Seguridad (AS)	12
	Requisitos de diseño y seguridad (RDS)	
	Evaluación de amenazas, riesgos de seguridad y privacidad (EARP)	
Implementación y Verificación	Usar herramientas aprobadas (HA)	7
	Prohibir funciones no seguras (PF)	
	Análisis estático (AE)	

	Análisis dinámico de los programas + Pruebas de Vulnerabilidad (AD)	
Despliegue	Administración de vulnerabilidades (AV)	4
	Plan de respuesta a incidentes (PRI)	
	Lanzamiento o archivado (LOA)	

Fuente. Elaboración propia.

12. Validación del marco de buenas prácticas para desarrollar código seguro

La validación del marco de buenas prácticas bajo la metodología Scrum, para desarrollar código seguro; se realizó mediante un método llamado juicio de expertos, que consiste en solicitar a un grupo de personas el juicio sobre un instrumento, material o marco en específico para su opinión en concreto. También se define esta, como una opinión informada de personas con trayectoria en el tema, que se reconocen por otros como expertos cualificados en este, y que pueden dar información, evidencia, juicios y valoraciones sobre un tema en específico (Pérez & Martínez, 2008).

Algunas de las ventajas de esta estrategia de evaluación son: la calidad de respuesta teórica de las personas, ofrece un nivel de profundización importante, facilita la puesta en acción, permite la ausencia de recursos técnicos y humanos para su ejecución, se pueden utilizar distintas estrategias para recolectar información de gran utilidad para determinar contenidos temáticos complejos, además de obtener información específica sobre los temas sometidos al estudio planteado (Almenara & Llorente, 2013).

12.1. Objetivo del juicio de expertos

Validar la pertinencia del marco de buenas prácticas bajo la metodología Scrum, para desarrollar código seguro.

12.2. Selección de los jueces

La selección del grupo de expertos, parte de las necesidades propias identificadas en el proyecto objeto de estudio en el presente documento; algunos criterios son: ser profesional

de las TIC, ejecutar acciones relacionadas con las tecnologías de información con formación en gerencia de TI, programación de software, seguridad de la información e informática, poseer un nivel alto de conocimiento y experiencia en la metodología Scrum o haber participado ampliamente en los procesos de calidad del software. El número de expertos puede llegar a variar, algunos autores recomiendan, entre 7 y 30 (García & Fernández, 2008), y por su parte otros no definen un numero en concreto, pero la recomendación es que sea menor a 50 en función de los objetivos que se persiguen.

La identificación y el perfilamiento de las personas que forman parte del juicio de expertos fue sujeta a los siguientes criterios generales:

a) Formación académica y entrenamiento

b) Amplio conocimiento y trayectoria en el tema

c) Motivación y disponibilidad para participar

d) Imparcialidad y capacidad de argumentación sobre la temática específica propuesta

Adicional se consideró importante la participación de los siguientes perfiles profesionales con determinadas características, desde la perspectiva del modelo de buenas prácticas de desarrollo de código seguro:

- **Gerente de proyecto TI.** Perfil.

  Profesional en Ingeniería de Sistemas, Telecomunicaciones o Electrónica. Con formación en gerencia de tecnologías de la información y comunicación TIC. Con especialización y/o maestría en proyectos o gerencia de sistemas de información, telecomunicaciones o administración de tecnología y procesos de TI.

  - **Conocimientos y Habilidades.** Conocimiento o certificación en ITIL, COBIT5, PMP, procesos de desarrollo de software, administración de procesos de tecnología, planificación estratégica, recursos y personal a cargo de equipos de trabajo.

- **Competencias.** Liderazgo y comunicación**,** toma de decisiones, habilidades de negociación y administrativas, competencias técnicas para la asimilación de nuevas tecnologías.

- **Experiencia y Antigüedad en el Área.** De 7 a 10 años.

- **Senior *Scrum Master*.** Perfil.

  Profesional en Ingeniería de Sistemas, Telecomunicaciones o Electrónica. Con experiencia y certificación de *Scrum Master*, desarrollo de software y metodologías ágiles. Con especialización y/o maestría en desarrollo de software o afines.

  - **Conocimientos y Habilidades.** Desarrollo de software, uso de metodologías ágiles y sólida experiencia en este rol. Utilización de herramientas de gestión de proyectos, gestión de la producción de soluciones de software garantizando la alineación estratégica.

  - **Competencias.** Liderazgo, comunicación y servicio**,** toma de decisiones, resolución de problemas, mejoramiento continuo, competencias técnicas para el manejo de herramientas y equipos interdisciplinares.

  - **Experiencia y Antigüedad en el Área.** De 5 a 7 años.

- **Especialista en Seguridad de Información.** Perfil.

  Profesional en Ingeniería de Sistemas, Telecomunicaciones o Electrónica. Con formación en seguridad digital, seguridad informática, investigación forense, gobierno y gestión de procesos de TI. Con especialización y/o maestría en seguridad de la información o afines.

  - **Conocimientos y Habilidades.** Seguridad informática, identificación, análisis y gestión de riesgos, arquitectura de seguridad, transformación digital, acompañamiento e implementación de proyectos de seguridad, ISO27001, Ley 1582 de 2012 de Colombia, y Ley europea de protección de datos GDPR. Monitoreo de seguridad, para identificar amenazas y eventos

de comportamientos sospechosos, e implementación de controles para la gestión de la información y ciberseguridad.

- **Competencias.** Liderazgo y comunicación**,** toma de decisiones, visión global de la estrategia de seguridad, conocimiento técnico de la ejecución de proyectos.

- **Experiencia y Antigüedad en el Área.** De 5 a 7 años.

- **Desarrollador de Software.** Perfil.

  Profesional en Ingeniería de Sistemas, de Software o Electrónica. Con formación en estructuras de datos, lenguajes de programación, bases de datos y analítica. Con especialización y/o maestría en desarrollo de software o afines.

  - **Conocimientos y Habilidades.** Modelos y metodologías de desarrollo de software, calidad para la industria del software. Dominio alto de lenguajes de programación (estructuras, orientada a objetos, bases de datos), herramientas de control, metodologías ágiles en proyectos, análisis de código fuente, conocimiento específico en seguridad de entornos web.

  - **Competencias.** Liderazgo y comunicación**,** toma de decisiones, capacidad de análisis y resolución de problemas, trabajo en equipo, creativo e innovador, con gran capacidad de aprendizaje.

  - **Experiencia y Antigüedad en el Área.** De 7 a 10 años.

- **QA Tester.** Perfil.

  Profesional en Ingeniería de Sistemas, de Software o Electrónica. Con formación en metodologías y modelos de calidad. Con especialización y/o maestría en desarrollo de software o afines.

  - **Conocimientos y Habilidades.** Modelos y metodologías de desarrollo de software, calidad para la industria del software. Certificación en *Testing – ISTQB*, conocimientos en Scrum, lenguajes de programación, manejo en

ambientes, consultas a bases de datos, herramientas para la ejecución y seguimiento de pruebas, y automatización de procesos.

- **Competencias.** Liderazgo y comunicación**,** toma de decisiones, capacidad de análisis y resolución de problemas, trabajo en equipo, responsabilidad, compromiso, pensamiento analítico y crítico, capacidad de abstracción, pragmático en la construcción de técnicas y seguimiento de instrucciones.

- **Experiencia y Antigüedad en el Área.** De 5 a 7 años.

12.3. Acción y evaluación del marco

La estrategia de juicio de expertos a poner en acción se realizó bajo la propuesta del **Método de Consenso**, (Almenara & Llorente, 2013). En donde de manera grupal y en conjunto, los expertos que fueron seleccionados llegan a conseguir un acuerdo. Para esto se realizó una sesión virtual, a través de una herramienta colaborativa, con el grupo de expertos seleccionado, en un horario definido, en el que en un periodo determinado de tiempo se expone la propuesta de valor, objetivos, antecedentes, y el marco, para conocer su juicio sobre el tema en específico; al final se realiza una evaluación sobre la propuesta.

Para la evaluación del marco, por parte del grupo de expertos se adaptó la planilla de Pérez & Martínez (2008). La cual fue dispuesta en un formulario de Google esto por su practicidad y concordancia con la sesión virtual que se ha realizado. Alli se almacenara la informacion para realizar las conclusiones tal y como lo contempla el Capitulo 13. (Ver Anexo G. Planilla juicio de expertos).

12.4. Cálculo de la concordancia entre los jueces

El cálculo de concordancia entre los jueces se realizó mediante el estadístico de Kendall (W), este es utilizado para conocer el grado de asociación entre los conjuntos de rangos (k), particularmente cuando los jueces expertos asignan calificaciones a rangos específicos (Pérez & Martínez, 2008).

La prueba estadística del coeficiente de concordancia de Kendall (W), da el valor de concordancia entre los expertos. Como la muestra la Tabla 34, el valor de Kendall (W) oscila entre 0 y 1 respectivamente; por consiguiente 1 muestra una concordancia total y 0

desacuerdo total; la tendencia a el valor 1 es lo deseado indicando la significación alcanzada (EcuredCu, 2012).

Tabla 34. Validación **-** Interpretación del valor de Kendall (W).

W Kendall	Interpretación
0,1≤W<0,3	Consenso muy débil
0,3≤W<0,5	Consenso débil
0,5≤W<0,7	Consenso moderado
0,7≤W<0,9	Consenso fuerte
W≥0,9	Consenso extraordinariamente fuerte

Fuente. Tomado de (Martínez, 2017).

La fórmula para el cálculo está dada por (Moguel, 2013).

$$W = \frac{S}{{}^{1}/_{12}\ K^2\,(N^3 - N) - K\ \sum Li}$$

En dónde.

- W = Coeficiente de concordancia de Kendall
- S = Suma de los cuadrados de las diferencias observadas con respecto a un promedio
- N = Tamaño de la muestra en función del número de tripletes, etc.
- K = Numero de variables incluidas
- Li = Sumatoria de las ligas o empates entre los rangos

La hipótesis planteada para la exposición del caso a los jueces en base a.

- **Alterna (Ha).** Existe asociación entre variables.

- **Nula (Ho).** No existe correlación o asociación entre variables.

Los ítems para calificar de acuerdo con el Anexo G. (Ver Anexo G. Planilla juicio de expertos), se presentan en la Tabla 35**,** allí se exponen cuatro categorias en torno a las cuales se genera la reflexion y la discusión, tales como: suficiencia, claridad, coherencia y relevancia; en base a estas categorias se espera obtener una calificacion de acuerdo a indicadores especificos, denotando una puntuacion minima de 1 (No cumple con el criterio) hasta 4 (Alto nivel) como maxima.

Tabla 35. Validación **-** Clasificación de los ítems a calificar.

Categoría	Calificación	Indicador
Suficiencia. La información expuesta pertenece a una misma dimensión y basta para obtener la medición de esta.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta no es suficiente para medir la dimensión La información expuesta mide algún aspecto de la dimensión, pero no corresponde con la dimensión total Se debe incrementar alguna información para poder evaluar la dimensión completamente La información expuesta es suficiente
Claridad. La información expuesta se comprende fácilmente, es decir, su sintáctica y semántica son adecuadas.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta no es clara La información expuesta requiere bastantes modificaciones o una modificación muy grande en el uso de las palabras de acuerdo con su significado o por la ordenación de estas Se requiere una modificación muy específica de algunos de los términos de la información expuesta La información expuesta es clara, tiene semántica y sintaxis adecuada
Coherencia. La información expuesta tiene relación lógica	1. No cumple con el criterio 2. Bajo nivel	La información expuesta no tiene relación lógica con la dimensión

con la dimensión o indicador que está midiendo.	3. Moderado nivel 4. Alto nivel	La información expuesta tiene una relación tangencial con la dimensión La información expuesta tiene una relación moderada con la dimensión que está midiendo La información expuesta se encuentra completamente relacionada con la dimensión que está midiendo
Relevancia. La información expuesta es esencial o importante, es decir debe ser incluida.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta puede ser eliminada sin que se vea afectada la medición de la dimensión La información expuesta tiene alguna relevancia, pero otra distinta puede estar incluyendo lo que mide éste La información expuesta es relativamente importante La información expuesta es muy relevante y debe ser incluida

Fuente. Tomado y adaptado de (Pérez & Martínez, 2008).

12.5. Resultados concordancia de Kendall (W)

La muestra de recolección de datos fue obtenida de siete expertos, los cuales calificaron con los siguientes resultados, que fueron analizados en el software estadístico IBM SPSS.

Para la carga de datos se definieron las siguientes variables en el entorno de IBM SPSS.

Figura 17. Validación – Declaración de variables de datos en carga.

	Nombre	Tipo	Anchura	Decimales	Etiqueta	Valores	Perdidos	Columnas	Alineación	Medida
1	Experto	Cadena	16	0		Ninguno	Ninguno	10	Izquierda	Nominal
2	Suficiencia	Numérico	16	0		{1, No cump...	Ninguno	10	Derecha	Ordinal
3	Claridad	Numérico	16	0		{1, No cump...	Ninguno	10	Derecha	Ordinal
4	Coherencia	Numérico	16	0		{1, No cump...	Ninguno	10	Derecha	Ordinal
5	Relevancia	Numérico	16	0		{1, No cump...	Ninguno	10	Derecha	Ordinal

Fuente. Elaboración propia en IBM SPSS.

Como lo muestra la Figura 18¡Error! No se encuentra el origen de la referencia.**,** estos fueron los resultados obtenidos que fueron analizados con la concordancia de Kendall (W) por los jueces expertos.

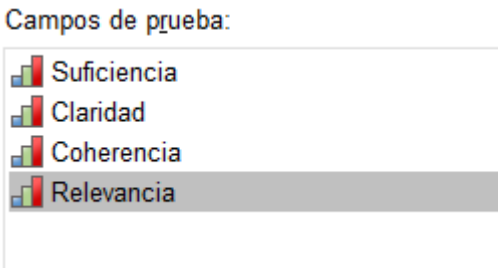
Figura 18. Validación – Datos obtenidos en juicio de expertos.

	Experto	Suficiencia	Claridad	Coherencia	Relevancia
1	Experto_1	Alto Nivel	Moderado Nivel	Alto Nivel	Moderado Nivel
2	Experto_2	Moderado Nivel	Moderado Nivel	Moderado Nivel	Moderado Nivel
3	Experto_3	Alto Nivel	Moderado Nivel	Alto Nivel	Moderado Nivel
4	Experto_4	Alto Nivel	Moderado Nivel	Alto Nivel	Moderado Nivel
5	Experto_5	Alto Nivel	Moderado Nivel	Alto Nivel	Moderado Nivel
6	Experto_6	Alto Nivel	Moderado Nivel	Alto Nivel	Alto Nivel
7	Experto_7	Alto Nivel	Moderado Nivel	Alto Nivel	Moderado Nivel

Fuente. Elaboración propia en IBM SPSS.

Una vez cargados los datos al software estadístico IBM SPSS, se analizan por la opción ->Analizar,>Pruebas no paramétricas, >Muestras relacionadas. En la opción campos, se seleccionaron los campos de prueba, los cuales tienen una configuración de medida ordinal.

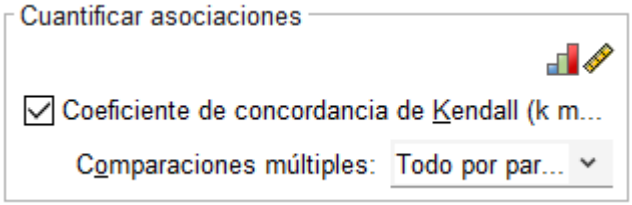
Figura 19. Validación – Selección campos de prueba.



Fuente. Elaboración propia en IBM SPSS.

En la opción >Configuración, >Personalizar pruebas, selecciona >Coeficiente de concordancia de Kendall (W), y >Ejecutar.

Figura 20. Validación – Selección coeficiente de concordancia de Kendall (W).



Fuente. Elaboración propia en IBM SPSS.

Los resultados de las pruebas no paramétricas concluyen lo siguiente:

Como lo muestra la Figura 21, se estableció la decisión de hipótesis nula, con el fin de verificar la veracidad de esta; lo que quiere decir, es rechazada, y además no hay una relación entre las variables: Suficiencia, Claridad, Coherencia y Relevancia, lo que generó una significación del 0,50 respectivamente.

Figura 21. Validación – Resumen de contrastes e hipótesis.

Resumen de contrastes de hipótesis

	Hipótesis nula	Prueba	Sig.[a,b]	Decisión
1	Las distribuciones de Suficiencia, Claridad, Coherencia y Relevancia son iguales.	Coeficiente de concordancia de Kendall para muestras relacionadas	,001	Rechace la hipótesis nula.

a. El nivel de significación es de ,050.

b. Se muestra la significancia asintótica.

Fuente. Elaboración propia en IBM SPSS.

El valor estadístico de Kendall (W), tal y como lo muestra la Figura 22, arroja un nivel de concordancia entre los expertos de **0,764**, y según la interpretación de la Tabla 34, existe un **consenso fuerte** entre los jueces con un grado de libertad amplio (3), respectivamente. Lo que indica una aceptación superior del 76% al marco de buenas prácticas bajo la metodología Scrum para desarrollar código seguro.

Figura 22. Validación – Resumen de coeficiente de concordancia de Kendall para muestras relacionadas.

Suficiencia, Claridad, Coherencia, Relevancia

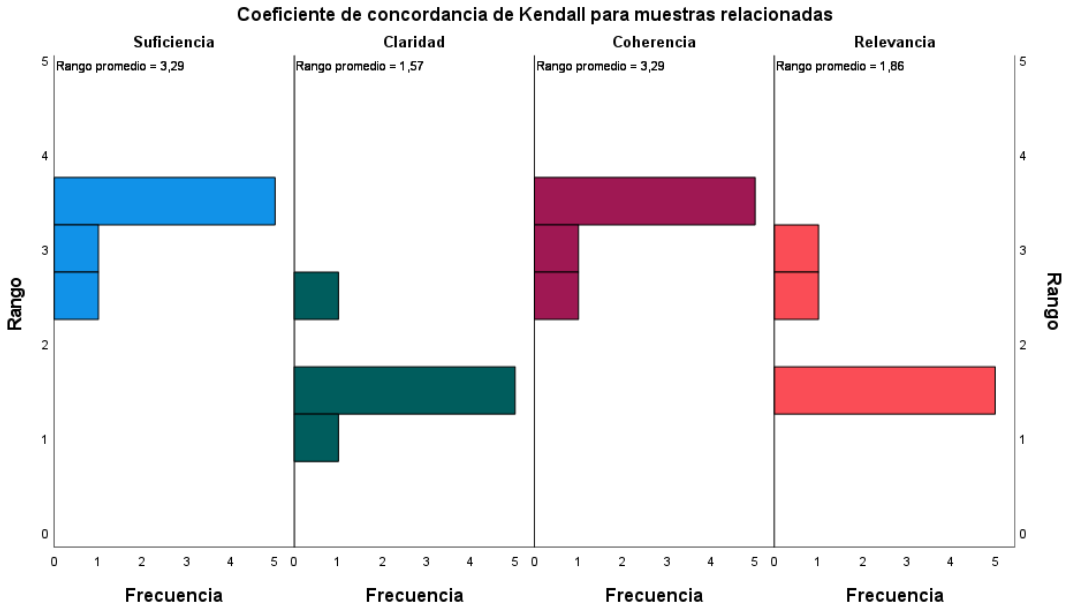
Resumen de coeficiente de concordancia de Kendall de muestras relacionadas

N total	7
W de Kendall	,764
Estadístico de prueba	16,043
Grado de libertad	3
Sig. asintótica (prueba bilateral)	,001

Fuente. Elaboración propia en IBM SPSS.

Algunos puntos relevantes en la validación del presente marco, y con la ayuda de la evaluación de los jueces expertos, demostraron cifras importantes en los coeficientes de concordancia para las categorías de Suficiencia y Coherencia, obteniendo rangos promedios con una valoración de 3,29, por encima de Relevancia 1,86 y Claridad 1,57. Lo anterior indica que el marco propuesto, cuenta con los criterios en suficiencia para su puesta en marcha, y es coherente con las necesidades propias identificadas en la formulación del problema, así mismo tiene un nivel de relevancia importante como aspecto social, y debe reforzar la claridad de lo que desea implementar en las organizaciones.

Figura 23. Validación – Figura coeficiente de concordancia de Kendall para muestras relacionadas.



Fuente. Elaboración propia en IBM SPSS.

En la comprobación por parejas que se generó en el software IBM SPSS, como hechos a resaltar se encuentran la Claridad-Relevancia y Suficiencia-Coherencia con un nivel de correlación ajustado cerca al 100%; y las parejas Claridad-Suficiencia, Claridad-Coherencia, obtuvieron un nivel de correlación ajustado superior al 70%, lo que indica que la propuesta se encamino a ser clara, relevante, y contar con suficiencia y coherencia en los que se propone mitigar desde el punto de vista de seguridad en las aplicaciones.

Figura 24. Validación – Comprobación por parejas.

Comparaciones por parejas

Sample 1-Sample 2	Estadístico de prueba	Desv. Error	Desv. Estadístico de prueba	Sig.	Sig. ajustada[a]
Claridad-Relevancia	-,286	,690	-,414	,679	1,000
Claridad-Suficiencia	1,714	,690	2,484	,013	,078
Claridad-Coherencia	-1,714	,690	-2,484	,013	,078
Relevancia-Suficiencia	1,429	,690	2,070	,038	,231
Relevancia-Coherencia	1,429	,690	2,070	,038	,231
Suficiencia-Coherencia	,000	,690	,000	1,000	1,000

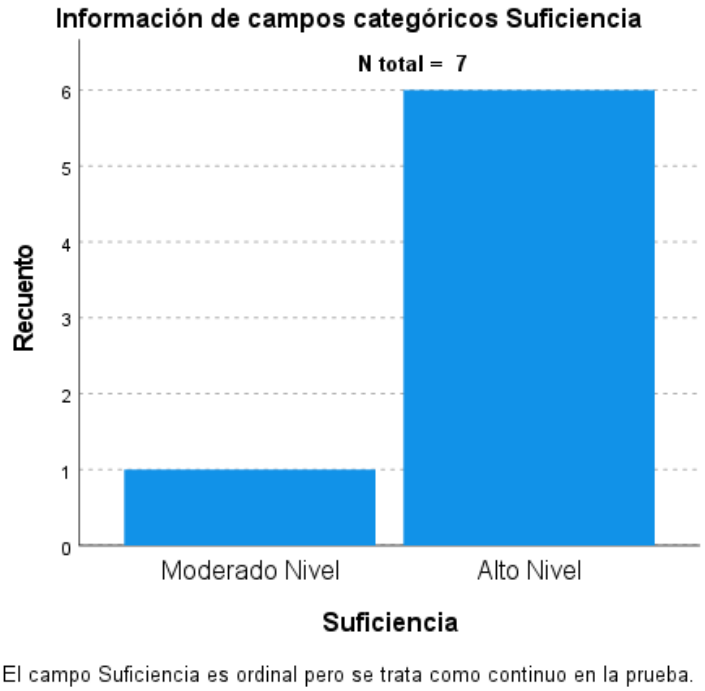
Cada fila prueba la hipótesis nula que las distribuciones de la Muestra 1 y la Muestra 2 son iguales.
Se visualizan las significaciones asintóticas (pruebas bilaterales). El nivel de significación es de ,050.

a. Los valores de significación se han ajustado mediante la corrección Bonferroni para varias pruebas.

Fuente. Elaboración propia en IBM SPSS.

A continuación, en la Figura 25, se muestran los resultados de la categoría suficiencia, con una aceptación del 85,71% del total de la muestra, lo que indica que la información que fue expuesta a los jurados fue suficiente y está en un alto nivel; en contraposición el 14,29% considera que se debe incrementar información específica relacionada con cifras, costos, y estadísticas en las que el marco de buenas prácticas genere un nivel de madurez esperado en las organizaciones, esto con el fin de poder evaluar la dimensión completamente y estimar el impacto de la inclusión de esta.

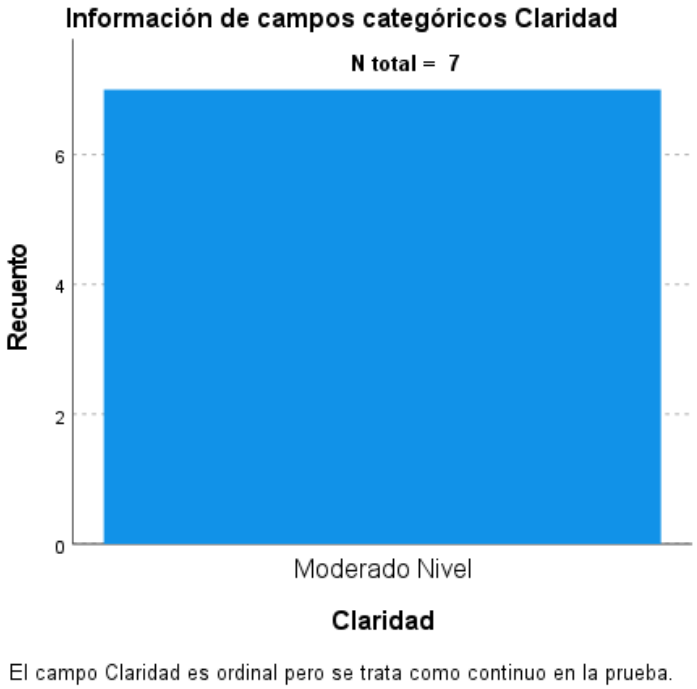
Figura 25. Validación – Información de campos por Suficiencia.



Fuente. Elaboración propia en IBM SPSS.

En la Figura 26**Figura 25,** se muestran los resultados de la categoría claridad, con una aceptación del 100% del total de la muestra, lo que indica que la información fue investigada, presentada a alto nivel, y clara; y además contaba con la semántica y sintaxis adecuada. Lo que sobrepone al estudio realizado y respalda los resultados obtenidos en la concordancia de los jueces expertos, es muy importante que el marco de buenas prácticas sea un instrumento de apoyo en las organizaciones, y este ayude a disminuir los errores, y las brechas de seguridad que se generan al entregar los productos terminados de software.

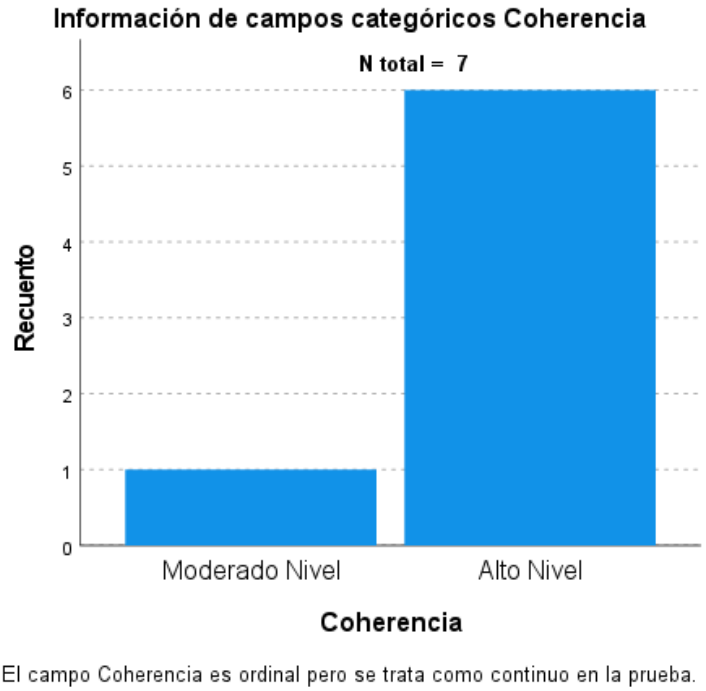
Figura 26. Validación – **I**nformación de campos por Claridad.



Fuente. Elaboración propia en IBM SPSS.

Por otra parte en la Figura 27**Figura 26Figura 25,** se muestran los resultados de la categoría coherencia, con una aceptación del 85,71% del total de la muestra, lo que indica que la información que fue expuesta a los jurados se encuentra completamente relacionada con la dimensión que se está midiendo; en contraposición el 14,29% considera que la información expuesta tiene una relación moderada con la dimensión que se está midiendo, lo cual indica que el proyecto debe incrementar información específica relacionada con cifras, costos, y estadísticas en las que el marco de buenas prácticas genere un nivel de madurez esperado en las organizaciones, esto con el fin de poder evaluar la dimensión completamente y estimar el impacto de la inclusión de esta.

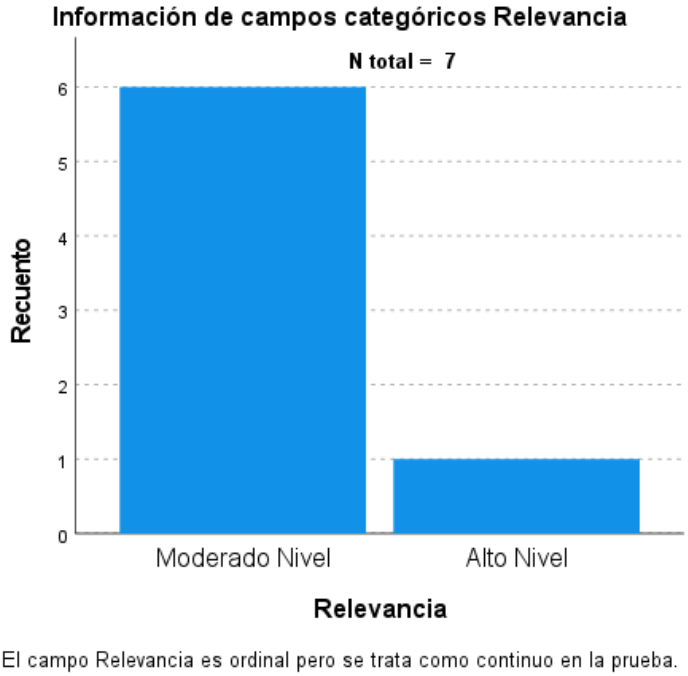
Figura 27. Validación – información de campos por Coherencia.



Fuente. Elaboración propia en IBM SPSS.

En la Figura 28**Figura 27Figura 26Figura 25,** se muestran los resultados de la categoría relevancia, con una aceptación del 14,29% del total de la muestra relacionada con la dimensión que se está midiendo a alto nivel, lo que indica que la información expuesta es relevante y debe ser incluida; por otro lado el 85,71% con un moderado nivel, muestra relatividad al momento de exponer la información que fue de su interés, esto asociado al incluir la seguridad dentro de Scrum como marco de buenas prácticas para desarrollo de código.

Figura 28. Validación – información de campos por Relevancia.



Fuente. Elaboración propia en IBM SPSS

Como resultado del presente estudio, se identificaron en las características expuestas a los jueces expertos, la pertinencia y la apropiación del marco de buenas prácticas; estos llegaron a un nivel de consenso fuerte en la concordancia, con una aproximación del 76%, entre los que se destacaron calificaciones de moderado nivel en un 53,57%, y un alto nivel del 46,43% respectivamente. A resaltar en la evaluación, no se obtuvieron calificaciones de bajo nivel, ni de no cumplimiento de los criterios. Para más información sobre el cálculo estadístico, se recomienda ver el Anexo H, al final del documento. (Ver Anexo H. Resultados coeficiente de Kendall (W)).

13. Conclusiones

Después de investigar sobre las vulnerabilidades, fallos y riesgos en seguridad, se logró identificar que las aplicaciones que son desarrolladas y analizadas con herramientas de código fuente estático, tienen al menos un fallo a nivel de seguridad grave, lo que es preocupante, porque estas aplicaciones terminadas están siendo expuestas a los usuarios finales, con este tipo de errores que pueden causar principalmente ataques de inyección de código, perdida de autenticación, fuga de información, problemas criptográficos, y exposición de datos sensibles.

La seguridad es un factor que debe ser evaluado desde la misma concepción de la idea, e incluida durante toda su fase de desarrollo, hasta el despliegue; ya que siempre existen riesgos inherentes al uso de estas, que los atacantes pueden llegar a materializar, usando distintas técnicas y métodos, con el fin de comprometer los datos e información, no solo de los usuarios, sino también de las organizaciones. Es importante conocer los riesgos, a los que podemos estar expuestos, y reconocer las fuentes de información que categorizan estos fallos y riesgos a nivel mundial.

Las metodologías de desarrollo seguro analizadas tienen un enfoque de gobierno que ayudan a las prácticas de seguridad, con actividades determinadas que pueden ser integradas en cualquier tipo de organización. Es necesario que estas metodologías sean involucradas dentro de prácticas ágiles, y en entornos reales de equipos de desarrollo, con el fin de ser probadas, no solo por los profesionales en seguridad, sino por todos aquellos que intervienen en esta práctica.

La importancia de escoger alguna de las que fueron objeto de análisis en este documento, puede ser representada por el enfoque mismo de la organización que desee adoptarla; sin embargo, *SAMM* se destaca por ser creada por la reconocida metodología *OWASP*, y por ser un modelo rápido, y fácil de desplegar, además por la documentación que la respalda en línea. *BSIMM*, es aplicada en ambientes empresariales reales, y cuenta ya con su décima versión, y es reconocida por 122 organizaciones que ayudan a medir el nivel de madurez al interior de estas. *Microsoft SDL*, ha impactado de manera positiva sus productos desde el año 2009, con una reducción promedio del 57% de sus vulnerabilidades en un periodo de un año, entre el lanzamiento de estos y los pasos a producción

comerciales respectivamente. Implementar alguna de estas es cuestión de decisión, y de impactar al negocio transversalmente, ya que incluso aquí los costos de remediación de alguno de estos ataques se pueden llegar a disminuir considerablemente.

Desde el momento en que se empezó a analizar la información referente a las metodologías de desarrollo seguro, fue importante identificar cual era la metodología ágil, más utilizada en las organizaciones, y que cumpliera con lo estipulado en el manifiesto ágil en su practicidad, efectividad, tiempos de desarrollo y entrega; para esto se identificó a Scrum como un marco de trabajo, que permite con sus procesos, cumplir con los objetivos a nivel de negocio, desde el punto de vista de desarrollo de software, aun así, este marco de trabajo es usado con distintos enfoques para la dirección de proyectos, pero en cuanto al desarrollo de aplicaciones es un referente importante en la actualidad.

La integración del marco de buenas prácticas de seguridad, con Scrum, surge como una necesidad que debe ser resuelta desde las organizaciones y en las prácticas de desarrollo. Para esto se creó *ScrumSec*, el cual tomo los 19 procesos de Scrum, y creo cuatro fases y 30 actividades que mejoran la práctica de seguridad en el desarrollo de software; estas cuatro fases dividen teóricamente los procesos de Scrum, no modificando sus artefactos, ni actividades, sino inyectando las mejores técnicas de seguridad recomendadas por las mejores metodologías para hacer software más seguro, y confiable. Cada uno de los procesos de *ScrumSec*, muestra de manera detallada las entradas, herramientas y salidas que se deben tener presentes al implementar en las organizaciones el marco de buenas prácticas. Lo que se consideró como una tarea compleja, ardua y de trabajo a conciencia que involucro análisis a profundidad de las tres metodologías propuestas en este estudio.

Validar y socializar el marco de buenas prácticas mediante la metodología de juicio de expertos, es una forma de medir el instrumento que se ha creado, ya que permite conocer la opinión en concreto de quienes son expertos en áreas del conocimiento en específico, tener calidad en las respuestas, y recolectar la información que permitirá someter el estudio planteado a un análisis estadístico, mediante la pertinencia de este. Los expertos que fueron seleccionados para este fin contaban con conocimiento y experiencia certificada en tecnologías de la información, gerencia de TI, programación de software, metodologías ágiles, y seguridad de la información.

Con el método de consenso, se realizó la exposición de la propuesta del marco de buenas prácticas, a través de herramientas colaborativas e internet; para la evaluación del marco se adaptó la planilla recomendada por Pérez & Martínez (2008), la cual sirvio como instrumento de verificacion. Por ultimo se calculo la concordancia entre los participantes y se demostro mediante el estadistico de Kendall (W), que habia lo que se denomina un consenso fuerte entre los jueces al validar lo propuesto en el marco ($0,7 \leq W < 0,9$), con un porcentaje de aceptacion del 76%, lo que indica que la propuesta contaba con suficiencia 3,29% para su puesta en marcha, claridad 1,57% en lo que se queria demostrar, coherencia 3,29% en relacion a las necesidades propias identificadas en la formulacion del problema, y relevancia 1,86% en el impacto positivo en las organizaciones que lo implementen, en la disminucion de brechas de seguridad de las aplicaciones desarrolladas.

Disminuir las brechas de seguridad en el proceso de creación y construccion de las aplicaciones, es en lo que las organizaciones deben trabajar, este marco de buenas practicas bajo la metodologia Scrum, para desarrollar codigo seguro, espera hacer parte de estos procesos que buscan la mejora contunia en la gestion de la seguridad, de las nuevas aplicaciones y sistemas de informacion que salen a produccion.

14. Recomendaciones

Con relación a los resultados alcanzados, y luego de realizar un análisis en conjunto con el grupo de expertos, es recomendable para el marco de buenas prácticas, establecer un número de días promedio de remediación de las vulnerabilidades con *ScrumSec*, que ayude a disminuir la cifra estipulada en el entorno de seguridad, la cual es de 171; ya que permitirá conocer el impacto de la inclusión del marco en las organizaciones, y dará una mayor aproximación en cuanto al nivel de madurez del mismo, así mismo esta disminución no deberá modificar los tiempos en los que se manejan los ciclos *(Sprints)* que se tienen estipulados actualmente bajo la metodología Scrum.

En cuanto al uso de herramientas, para el seguimiento de las vulnerabilidades que se pueden dar durante la construcción y puesta en marcha del código fuente y aplicaciones, es importante que las organizaciones, mediante el uso de artefactos, incorpore estas dentro de los ciclos *(Sprints)*, con el fin de que, al interior de los equipos de trabajo, estas sean visibles, y puedan dar trazabilidad a las mismas. Es de aclarar que las organizaciones pueden crear sus propios tableros de control, asociados a las necesidades del negocio, también así, estos pueden tener un enfoque de mitigación de vulnerabilidades de acuerdo con la criticidad y severidad que se pueda derivar del análisis de riesgo hecho con *OWASP* (Ver Anexo A. Análisis de Riesgo con OWASP).

Como se comentó en el juicio de expertos, el marco de *ScrumSec*, tiene una orientación exclusiva para proyectos de desarrollo de software, por lo anterior no se contempla abrir el enfoque de esta a otro tipo de actividades o proyectos derivadas de los procesos de Scrum, ya que las buenas prácticas que fueron incluidas tienen una definición propia para el análisis, descubrimiento y mitigación de errores de codificación, vulnerabilidades y hallazgos en las aplicaciones propias que se deriven del uso de este marco.

Dentro de las mejoras que pueden dar valor agregado al marco de *ScrumSec*, es importante hacer un análisis de costos derivados por la utilización de este. No solo para los eventos que se dan dentro del marco, sino para contemplar que existen costos asociados a la remediación de incidentes de seguridad. En este estudio de costos se debe incluir los asociados a servicios profesionales, y a la formación del equipo de incidentes de respuestas bajo un modelo de gestión en la organización.

Otra mejora que fue planteada en el juicio de expertos es la utilización de estándares de aseguramiento, para los entornos y contenedores de desarrollo y aplicaciones. Estas prácticas ofrecen recomendaciones que son bien definidas, y basadas en consensos muchas veces por los fabricantes, que ayudan a las organizaciones a evaluar y mejorar los entornos de seguridad. En la charla técnica con los expertos, surgió un ente internacional que promueve estas buenas prácticas a nivel de aseguramiento de las plataformas que se llama *CIS Security Benchmarks* (referencias de seguridad del CIS). Sería importante relacionar en otra fase del proyecto de *ScrumSec*, cual serian estos estándares que impactarían de manera positiva la seguridad en los entornos de desarrollo y en las aplicaciones propiamente.

Un tema que surgió con el hallazgo de la remediación de vulnerabilidades es la evolución de los entornos corporativos, y como se utilizan las pruebas de penetración para encontrar fallos en las aplicaciones. Hoy en día existe una técnica que es muy usada, llamada *Bug Bounty*, en la cual plataformas de seguridad en internet, buscan de manera informal casa recompensas que permiten auditar la seguridad de las aplicaciones en la red. Claro está, estos profesionales se conectan a través de plataformas reconocidas para ofrecer sus servicios de *hackers* éticos, con el fin de encontrar *bugs* en los distintos escenarios que son puestos a prueba. Esta es una práctica muy común, que puede ser agregada en una fase posterior de *ScrumSec*, ya que la tercerización de pruebas funcionales de seguridad permitiría detectar con una fiabilidad mayor los errores de programación, vulnerabilidades y fallos en las aplicaciones, y tendrían un nivel de aceptación importante durante los ciclos *(Sprints)* que se trabajen en las organizaciones.

Una recomendación adicional que se generó en el juicio de expertos fue considerar establecer unos niveles de madurez que reflejen el estado de la seguridad y la implementación de *ScrumSec* en las organizaciones. Esto debido a que, así este apalancado desde la estrategia este marco, este tomara tiempo adaptarlo al interior de los equipos de desarrollo, ya que se propone capacitación continua y certificación en temáticas de seguridad. En una línea de tiempo optima, un equipo de *ScrumSec*, estaría trabajando con el marco, y en las iteraciones con los ciclos *(Sprints)* en un tiempo aproximado de 12 meses, con este nuevo enfoque. Sin embargo, se debe hacer un caso de estudio en ambientes productivos y reales, para conocer las cifras y estadísticas sobre el

comportamiento de este, por consiguiente, en una nueva fase de *ScrumSec*, esto debe ser tenido en cuenta.

15. Referencias

AGUILERA, J. A. (2017). *IMPLICACIONES DE SEGURIDAD EN METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE*. Obtenido de https://repository.libertadores.edu.co/bitstream/handle/11371/1163/ramirezjonathan2017.pdf?sequence=2&isAllowed=y

Alejandro, P. C. (2017). *Buenas Prácticas para el Desarrollo de Código Seguro*. Obtenido de http://polux.unipiloto.edu.co:8080/00001881.pdf

Almenara, J. C., & Llorente, C. (Enero de 2013). *La aplicación del juicio de experto como técnica de evaluación de las tecnologías de la información y comunicación (TIC)*. Obtenido de https://www.researchgate.net/publication/260750592_La_aplicacion_del_juicio_de_experto_como_tecnica_de_evaluacion_de_las_tecnologias_de_la_informacion_y_comunicacion_TIC

Álvarez García, J. I. (30 de Noviembre de 2016). *Catálogo de patrones de ciclo de vida en desarrollo de software aplicables en la industria colombiana*. Obtenido de https://repository.ean.edu.co/bitstream/handle/10882/8947/AlvarezJorge2016.pdf

Andalucia, J. d. (2017). *Ciclo de vida clásico o en cascada*. Obtenido de http://agrega.juntadeandalucia.es/repositorio/20022017/6b/es-an_2017022012_9122843/51_ciclo_de_vida_clsico_o_en_cascada.html

Andreessen, M. (20 de Agosto de 2011). *Why Software Is Eating The World* . Obtenido de https://www.wsj.com/articles/SB10001424053111903480904576512250915629460

Architects, I. (2017). *The Seven Phases of the System-Development Life Cycle*. Obtenido de https://www.innovativearchitects.com/KnowledgeCenter/basic-IT-systems/system-development-life-cycle.aspx

AUSCERT. (2020). *Security Bulletins*. Obtenido de https://www.auscert.org.au/resources/security-bulletins/

Bastos, P. (9 de Septiembre de 2019). *Una mirada a la industria del software en América Latina*. Obtenido de https://markamagazine.com/una-mirada-a-la-industria-del-software-en-america-latina/

Becerra, P., & Sanjuan, M. (2014). *Revisión de estado del arte del ciclo de vida de desarrollo de software seguro con la metodología SCRUM*. Obtenido de http://publicaciones.unisimonbolivar.edu.co/rdigital/ojs/index.php/identic/article/view/1525

Benioef, M. R., & Lazowska, E. D. (2005). *Overview of Cyber Security: A Crisis of Prioritization*. Obtenido de https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1439495

BSIMM. (18 de Septiembre de 2019). *BSIMM Framework*. Obtenido de https://www.bsimm.com/download.html

Cano, C. C. (21 de Agosto de 2019). *Metodologías ágiles, una oportunidad para las empresas colombianas*. Obtenido de https://www.capitalhumano.com.co/actualidad/metodologias-agiles-una-oportunidad-para-las-empresas-colombianas-12530/

CISA, C. &. (2020). *2020 Alerts*. Obtenido de https://us-cert.cisa.gov/ncas/alerts/2020

Código, A. d. (01 de Agosto de 2019). *Auditoría de Código*. Obtenido de https://auditoriadecodigo.com/desarrollo-de-software-seguro-ciclo-de-vida-de-desarrollo-de-software-seguro-s-sdlc/

COLOMBIA, C. D. (17 de Octubre de 2012). *LEY ESTATUTARIA 1581 DE 2012*. Obtenido de https://www.defensoria.gov.co/public/Normograma%202013_html/Normas/Ley_1581_2012.pdf

colombiafintech. (8 de Septiembre de 2020). *Superfinanciera fortalece la protección de la información de los consumidores financieros* . Obtenido de https://www.colombiafintech.co/novedades/superfinanciera-fortalece-la-proteccion-de-la-informacion-de-los-consumidores-financieros-ante-riesgos-de-ciberseguridad-y-la-realizacion-de-operaciones-en-pasarelas-de-pago#:~:text=DEL%20GREMIO-,Superfinanciera%20

Consultancy.eu. (7 de Mayo de 2020). *Half of companies applying Agile methodologies & practices*. Obtenido de https://www.consultancy.eu/news/4153/half-of-companies-applying-agile-methodologies-practices

Cornell, D. (Mayo de 2017). *OWASP San Antonio: Open Software Assurance Maturity Model (OpenSAMM)*. Obtenido de https://www.slideshare.net/denimgroup/open-samm10-owaspsanantonio20100519

CVE. (18 de Septiembre de 2020). *News & Events* . Obtenido de https://cve.mitre.org/news/archives/2020/news.html

CWE. (20 de Agosto de 2020). *2020 CWE Top 25 Most Dangerous Software Weaknesses*. Obtenido de https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html

Digital.ai. (2020). *stateofagile.com*. Obtenido de https://stateofagile.com/#ufh-i-615706098-14th-annual-state-of-agile-report/7027494

Dima Bekerman, S. Y. (23 de Enero de 2020). *The State of Vulnerabilities in 2019*. Obtenido de https://www.imperva.com/blog/the-state-of-vulnerabilities-in-2019/

Distrital, S. J. (5 de Enero de 2009). *Ley 1273 de 2009 Nivel Nacional*. Obtenido de https://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=34492

Duarte, O., & A., &. R. (24 de Mayo de 2008). *Las Metodologías de desarrollo Ágil como una oportunidad para la Ingeniería del software educativo.* Obtenido de http://www.bdigital.unal.edu.co/15430/1/10037-18216-1-PB.pdf

EcuredCu. (2012). *Coeficiente de Kendall*. Obtenido de https://www.ecured.cu/Coeficiente_de_Kendall

Erdman, B. (18 de Octubre de 2019). *Cumplimiento PCI DSS y PA DSS*. Obtenido de https://www.helpsystems.com/es/recursos/articulo/cumplimiento-pci-y-pa-dss-que-es-requisitos-latinoamerica-espana#:~:text=Los%20est%C3%A1ndares%20PCI%2DDSS%20y,forma%20correcta%2C%20segura%2C%20y%20que

Etica, S. y. (16 de Septiembre de 2013). *¿SEGURIDAD INFORMÁTICA O SEGURIDAD DE LA INFORMACIÓN?* Obtenido de https://seguridadetica.wordpress.com/2013/09/16/seguridad-informatica-o-seguridad-de-la-informacion/

Excelencia, E. E. (13 de Noviembre de 2019). *Listado de amenazas y vulnerabilidades en ISO 27001*. Obtenido de https://www.escuelaeuropeaexcelencia.com/2019/11/listado-de-amenazas-y-vulnerabilidades-en-iso-27001/

Fajardo, D. G. (31 de Marzo de 2020). *Las metodologías ágiles más usadas*. Obtenido de https://blog.cobiscorp.com/metodologias-agiles-mas-usadas

Figueredo, L. C. (10 de Octubre de 2020). *RELATED TEST - PDF (Suficiencia Claridad Coherencia Relevancia) KENDALL(COMPARE=PAIRWISE)*. Obtenido de IBM SPSS

FromLinux. (2016). *Seguridad de la Información: Historia, Terminología y Campo de acción*. Obtenido de https://blog.desdelinux.net/seguridad-informacion-historia-terminologia-campo/

Garcia, L. A. (9 de Agosto de 2019). *¿Qué son las metodologías ágiles?* Obtenido de https://www.luisan.net/blog/transformacion-digital/que-son-las-metodologias-agiles

García, L., & Fernández, S. J. (Marzo de 2008). *Procedimiento de aplicación del trabajo*. Obtenido de https://www.redalyc.org/pdf/3291/329127758006.pdf

Gonçalves, L. (19 de Agosto de 2020). *Qué es la metodología Ágil, todo lo que necesitas saber*. Obtenido de https://adaptmethodology.com/es/que-es-la-metodologia-agil/

Hernández, C., & Baptista, P. (2014). *Metodología de la investigación.* México: McGraw Hill.

Lastra Colobón, I. C. (2018). *Análisis de los beneficios de implementar controles financieros bajo la Ley Sarbanes Oxley (Sox) en empresas colombianas del Sector Real*. Obtenido de https://repository.ugc.edu.co/handle/11396/4581

Mariño, Z. C. (6 de Noviembre de 2019). *SCRUM: la metodología ágil más usada* . Obtenido de https://zoraidaceballosdemariño.info/scrum/scrum-la-metodologia-agil-mas-usada/

Martínez, M. S. (2017). *CONCLUSIONS FROM A DELPHI STUDY*. Obtenido de http://revintsociologia.revistas.csic.es/index.php/revintsociologia/article/view/678/841

Matei, A. (7 de Enero de 2015). *Guía para el desarrollo de*. Obtenido de http://oa.upm.es/34770/1/PFC_ADRIANA_MATEI.pdf

Menn, J. (13 de Marzo de 2000). *Marc Andreessen: el padre de Netscape*. Obtenido de https://www.lanacion.com.ar/tecnologia/marc-andreessen-el-padre-de-netscape-cambia-de-rumbo-nid187857/

Microsoft. (31 de Enero de 2002). *Microsoft SDL*. Obtenido de https://www.microsoft.com/en-us/securityengineering/sdl

Moguel, A. R. (22 de Julio de 2013). *COEFICIENTE DE CONCORDANCIA DE KENDALL*. Obtenido de https://sites.google.com/site/tecnicasdeinvestigaciond38/estadisticas-no-parametricas/3-7-coeficiente-de-concordancia-de-kendall

Nacional, Policia. (29 de Octubre de 2019). *Tendencias Cibercrimen Colombia 2019-2020*. Obtenido de https://caivirtual.policia.gov.co/sites/default/files/tendencias_cibercrimen_colombia_2019_-_2020_0.pdf

News, I. (2016). *Instrumentan herramienta de seguridad del software en empresas argentinas*. Obtenido de http://www.infosecurityvip.com/newsletter/toolbox_set14.html

NVD, N. V. (2020). *Last 20 Scored Vulnerability IDs & Summaries*. Obtenido de https://nvd.nist.gov/

OWASP. (25 de Marzo de 2009). *Una guía para integrar seguridad en el desarrollo de software*. Obtenido de https://opensamm.org/downloads/SAMM-1.0-es_MX.pdf

OWASP. (1 de Diciembre de 2013). *ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP*. Obtenido de https://owasp.org/www-pdf-archive/Analisis_de_riesgo_usando_la_metodologia_OWASP.pdf

OWASP. (2017). *OWASP Top 10 2017Riesgos en Seguridad de Aplicaciones*. Obtenido de https://wiki.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

OWASP. (30 de Noviembre de 2017). *OWASP Top Ten*. Obtenido de https://owasp.org/www-project-top-ten/

PARRA, F. J. (2011). *MODELO DE ESTÁNDARES DE DESARROLLO SEGURO EN APLICATIVOS WEB (EDSAW)*. Obtenido de https://repository.unilibre.edu.co/bitstream/handle/10901/8837/MODELO%20DE%20EST%C3%81NDARES%20DE%20DESARROLLO%20SEGURO%20EN%20APLICATIVOS%20WEB%20EDSAW.pdf?sequence=1&isAllowed=y

Pérez, J. E., & Martínez, A. C. (Enero de 2008). *Validez de contenido y juicio de expertos: Una aproximación a su utilización*. Obtenido de https://www.researchgate.net/publication/302438451_Validez_de_contenido_y_juicio_de_expertos_Una_aproximacion_a_su_utilizacion

Rico, D. F. (1 de Octubre de 2009). *THE BUSINESS VALUE OF AGILE SOFTWARE METHODS*. Obtenido de http://www.davidfrico.com/agile-book-reviews.pdf

Roche, J. (12 de Diciembre de 2018). *Historia del movimiento Agile*. Obtenido de https://www2.deloitte.com/es/es/pages/technology/articles/historia-movimiento-agile.html

Rogerio, G. G., & Egas, L. M. (12 de Diciembre de 2013). *EVOLUCIÓN DE LAS METODOLOGÍAS DE DESARROLLO DE LA INGENIERÍA DE SOFTWARE EN EL PROCESO LAINGENIERÍA DE SISTEMAS SOFTWARE*. Obtenido de https://incyt.upse.edu.ec/ciencia/revistas/index.php/rctu/article/view/29/28

SANS. (27 de Junio de 2011). Obtenido de What Errors Are Included in the Top 25 Software Errors?: https://www.sans.org/top25-software-errors/

SCRUMstudy. (2017). *Scrum Framework. I. SCRUMstudy™. II. SBOK™ Guide*. Obtenido de https://www.scrumstudy.com/sbokguide/download-free-buy-sbok

Segovia, A. J. (18 de Abril de 2011). *Una introducción simple a los aspectos básicos ISO27001*. Obtenido de https://advisera.com/27001academy/es/que-es-iso-27001/

Sguerra, M. D. (2019). *ACIS*. Obtenido de http://acistente.acis.org.co/typo43/fileadmin/Revista_119/Uno.pdf

Sicrom. (2018). *Los tipos de ataques informáticos más comunes*. Obtenido de https://sicrom.com/blog/tipos-ataques-informaticos/

SL, T. P. (2017). *Software - Ciclo de Vida de Desarrollo - Modelo V*. Obtenido de https://www.tutorialspoint.com/es/software_engineering/software_development_life_cycle.htm

Sougth, P. (2018). *When Huawei met BSIMM*. Obtenido de https://programmersought.com/article/55881981569/

Sullivan, B. (12 de Noviembre de 2009). *The Microsoft® Security Development Lifecycle (SDL)*. Obtenido de https://slideplayer.com/slide/13903228/

TI, L. (13 de Mayo de 2015). *¿Qué métodos, técnicas y herramientas Ágiles se utilizan más y Por qué?* Obtenido de https://www.laboratorioti.com/2015/05/13/que-metodos-tecnicas-herramientas-agiles-utilizan-mas-por-que/

Veracode. (22 de Octubre de 2019). *State of Software Security*. Obtenido de https://www.veracode.com/sites/default/files/pdf/resources/sossreports/state-of-software-security-volume-10-veracode-report.pdf

Xie, Y. M. (4 de Marzo de 2020). *Situación Global Mobile 2020*. Obtenido de https://yiminshum.com/mobile-movil-app-2020/

A. Anexo. Análisis de Riesgo con OWASP

Un atacante puede utilizar cualquier ruta para vulnerar una aplicación, hacer daño a una organización o negocio. *OWASP* tiene una metodología de clasificación de riesgos, que cuenta con los siguientes aspectos (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013):

- Existen estándares internacionales y metodologías que se deben adaptar al negocio
- Un hallazgo de una vulnerabilidad critica para un negocio no lo es necesariamente para otro

Riesgo = Probabilidad * Impacto

El primer paso es: identificar un riesgo de seguridad a ser tratado:

Identifica agentes de amenaza.

Tabla 36. Anexo A – Agentes de amenaza.

Habilidades técnicas	Prueba de penetración (1) Redes y Programación (3) Usuario avanzado en computación (4) Habilidades técnicas medias (6) No cuenta con habilidades técnicas (9)
Motivación	Baja o sin motivación (1) Algo de interés (4) Bastante interesado (9)
Oportunidad	Acceso total (0) Acceso especial (4) Algunos accesos (7) Sin acceso (9)

Tamaño	Desarrolladores (2) Administradores de sistemas (2) Usuarios internos (4) Socios de negocio (5) Usuarios autenticados (6) Anónimos (9)

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Identifica vulnerabilidades que pueden ser explotadas.

Tabla 37. Anexo A – Vulnerabilidades.

Facilidad de descubrimiento	Dificultad alta (1) Dificultad media (3) Sencilla (7) Herramientas automatizadas disponibles (9)
Facilidad de explotación	Complejo (1) Dificultad media (3) Sencilla (5) Herramientas automatizadas disponibles (9)
Conciencia o conocimiento	Desconocido (1) Medianamente conocido (4) Común (6) De conocimiento público (9)
Detectores de intrusión	Detección activa en la aplicación (1) Autenticado y monitoreado (3) Autenticado sin monitoreo (8) No autenticado (9)

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Estima el impacto sobre el negocio de materialización de una amenaza, puede ser técnico o en el negocio.

Impacto técnico.

Tabla 38. Anexo A – Impacto técnico.

Perdida de confidencialidad	Mínima (data no critica) (2) Mínima (data critica) (6) Considerable (data no critica) (6) Considerable (data critica) (7) Corrupción de datos total (9)
Perdida de integridad	Mínima (data no critica) (1) Mínima (data critica) (3) Considerable (data no critica) (5) Considerable (data critica) (7) Corrupción de datos total (9)
Perdida de disponibilidad	Mínima (servicios no críticos) (1) Mínima (servicios críticos) (5) Considerable (servicios no críticos) (5) Considerable (servicios críticos) (7) Pérdida total de los servicios (9)
Perdida de auditabilidad	Totalmente auditable (1) Posiblemente auditable (7) No auditable (9)

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Impacto en el negocio.

Tabla 39. Anexo A – Impacto en el negocio.

Daño económico	Menor que el costo de la solución total (1) Efecto menor en el costo anual (3) Efecto significante en el costo anual (7) Efecto devastador (bancarrota) (9)
Daño de imagen	Daño mínimo (1) Pérdida de grandes cuentas (4) Pérdida de credibilidad a gran escala (5) Daño total de imagen (9)
No cumplimiento	Mínimo (2) Medio (5) Alto (7)
Violación de la privacidad	Una persona (3) Cientos de personas (5) Miles de personas (7) Millones de personas (9)

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Determina la severidad del riesgo. Se estima:

La **probabilidad** de ocurrencia de la amenaza e **impacto** generado sobre el negocio.

- **BAJA:** No ocasionaría mayores inconvenientes sobre la organización. Esta no se solucionaría inmediatamente.
- **MEDIA:** Ocasionaría un impacto leve sobre la organización. Esta debe solucionarse en un tiempo prudente.

- **ALTA:** Ocasionaría un impacto negativo sobre la organización. Esta debe solucionarse inmediatamente.

Tabla 40. Anexo A – Niveles de probabilidad e impacto.

Niveles de probabilidad e impacto	
0 a < 3	BAJA
3 a <6	MEDIA
6 a 9	ALTA

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Ejemplo: Probabilidad

Tabla 41. Anexo A – Ejemplo: Agentes de amenaza.

Agentes de Amenaza			
Habilidades técnicas	Motivación	Oportunidad	Tamaño
5	2	7	1

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Tabla 42. Anexo A – Ejemplo: Factores de vulnerabilidad.

Factores de Vulnerabilidad			
Facilidad de descubrimiento	Facilidad de explotación	Conciencia o conocimiento	Detectores de intrusión
3	6	9	2

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Promedio de probabilidad: 4,375 (Media)

Ejemplo: Impacto.

Tabla 43. Anexo A – Ejemplo: Impacto técnico.

Impacto Técnico			
Perdida de confidencialidad	Perdida de integridad	Perdida de disponibilidad	Perdida de auditabilidad
9	7	5	8

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Promedio de impacto técnico: 7,25 (Alto)

Tabla 44. Anexo A – Ejemplo: Impacto del negocio.

Impacto del Negocio			
Daño económico	Daño de imagen	No cumplimiento	Violación de la privacidad
1	2	1	5

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Promedio de impacto del negocio: 2,25 (Bajo)

Tabla 45. Anexo A – Gravedad del riesgo.

Gravedad del Riesgo				
Impacto	ALTO	Medio	Alto	Critico
	MEDIO	Bajo	Medio	Alto
	BAJO	Nota	Bajo	Medio
		BAJO	MEDIO	ALTO
		Probabilidad		

Fuente. Tomado de (OWASP, ANÁLISIS DE RIESGOS APLICANDO LA METODOLOGÍA OWASP, 2013).

Para el ejemplo la gravedad del riesgo es Bajo.

B. Anexo. Normas y Cumplimiento

A continuación, se listan algunas normas y estándares que dan cumplimiento de seguridad de la información e informática, a nivel internacional y en Colombia.

- **ISO/IEC 27001.** Norma internacional que describe la gestión de seguridad de la información en las organizaciones. La revisión de esta norma se realizó por última vez en el año 2013, puede ser implementada en cualquier tipo de organización, su cumplimiento es importante ya que confirma el eje central de la misma desde la protección de la confidencialidad, integridad y disponibilidad de la información y recurso de las organizaciones. (Segovia, 2011).

- **PCI DSS.** *(Payment Card Industry Data Security Standard).* Este fue desarrollado por el comité PCI SSC *(Payment Card Industry Security Standards Council),* y está enfocado en la seguridad de los sistemas, redes y otros equipos que participan en el procesamiento de las transacciones realizadas con tarjetas de pago, crédito y débito respectivamente. Tiene como finalidad asegurar que todos los pagos que se realicen sean de forma segura; se centra en la colección, procesamiento y transferencia de datos de las tarjetas. (Erdman, 2019).

- **SOX.** *(Sarbanes-Oxley Act).* Define los lineamientos sobre las medidas que se deben implementar a nivel de control interno para asegurar la fiabilidad de la información financiera. Esta puede ser implementada en Colombia para diseñar controles internos. (Lastra Colobón, 2018).

- **CE 007 de 2018 - SFC.** Imparte los requerimientos mínimos de seguridad y calidad para la realización de operaciones a través de pasarelas de pago. Establece instrucciones para la gestión de riesgos de ciberseguridad en entidades vigiladas por la Superintendencia Financiera de Colombia. (colombiafintech, 2020).

- **Ley 1581 – Protección de datos personales.** Tiene por objeto, dar derecho a las personas a conocer, actualizar y rectificar la información que ha sido suministrada

en bases de datos y el tratamiento que se hace de la misma por las distintas entidades. (COLOMBIA, 2012).

C. Anexo. Recomendaciones Generales de Seguridad

Las recomendaciones generales de seguridad para los proyectos permiten dar un vistazo inicial de aspectos a implementar, y tener en cuenta en el análisis de requerimientos de las aplicaciones.

- Los proyectos de desarrollo de software y aplicaciones deben garantizar la aplicación de buenas prácticas de desarrollo seguro, incluidas en el presente marco, ejecutar revisiones de seguridad sobre el código fuente y sobre la infraestructura, con los procesos definidos en la organización.

- Luego de la entrega del producto terminado en cada ciclo *(Sprint)*, se deberán aplicar pruebas de vulnerabilidad y *hacking* ético, pruebas estáticas y dinámicas al código fuente sobre nuevas funcionalidades que sean agregadas.

- Se deberá disponer de ambientes por separado e implementar controles de seguridad, para desarrollo, laboratorio y producción.

- Se debe garantizar la implementación de registros de auditoria en los contenedores de la aplicación y durante el desarrollo de código. Así mismo, en las aplicaciones las acciones de los usuarios deberán ser almacenadas por el tiempo que la organización disponga.

- El desarrollo de las aplicaciones, de acuerdo con su funcionalidad, debe contar con esquemas de autenticación que confirmen la identidad de los usuarios.

- La implementación de técnicas de cifrado fuerte para herramientas, comunicaciones, almacenamiento y exposición de servicios es necesaria.

- Se deben implementar controles de validación de datos de entrada y aseguramiento de calidad en el desarrollo de código.

D. Anexo. Patrones de diseño seguro

Algunos patrones de diseño seguro pueden ser consultados en la guía general de *OWASP*, aquí se tratan los riesgos, y se especifican los vectores de ataque, debilidades de seguridad, e impacto. Así mismo, muestra información asociada a las vulnerabilidades de las aplicaciones, como se previene, ejemplos de escenarios de ataque, y referencias externas a documentación especifica de seguridad.

OWASP. (2017). *OWASP Top 10* 2017 Riesgos en Seguridad de Aplicaciones. Obtenido de https://wiki.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

E. Anexo. Riesgos Conocidos

Los riesgos conocidos y las últimas actualizaciones sobre vulnerabilidades se pueden obtener de las siguientes fuentes de información, es importante consultarlas para cada proyecto de desarrollo de código:

CWE. (20 de agosto de 2020). *2020 CWE Top 25 Most Dangerous Software Weaknesses*. Obtenido de https://cwe.mitre.org/top25/archive/2020/2020_cwe_top25.html

CVE. (18 de septiembre de 2020). *News & Events*. Obtenido de https://cve.mitre.org/news/archives/2020/news.html

OWASP. (30 de noviembre de 2017). *OWASP Top Ten*. Obtenido de https://owasp.org/www-project-top-ten/

CISA, C. &. (2020). *2020 Alerts*. Obtenido de https://us-cert.cisa.gov/ncas/alerts/2020

AUSCERT. (2020). *Security Bulletins*. Obtenido de https://www.auscert.org.au/resources/security-bulletins/

NVD, N. V. (2020). *Last 20 Scored Vulnerability IDs & Summaries*. Obtenido de https://nvd.nist.gov/

F. Anexo. Herramientas de Monitoreo (Cyber Risk Rating).

Las herramientas de monitoreo constante, como las clasificaciones de seguridad (*Cybersecurity Rantings)*, permiten obtener una medida objetiva y dinámica de datos de las organizaciones asociadas a la postura de seguridad. Estas herramientas permiten conocer y mitigar los riesgos y vulnerabilidades, de los sitios y aplicaciones que están expuestos en internet. Algunas que existen y pueden ayudar a la seguridad, actualmente son:

- UpGuard – www.upguard.com
- Security Scorecard – www.securityscorecard.com
- BitSight – www.bitsight.com
- Riskrecon – www.riskrecon.com
- Fico – Cyber Risk Score – www.fico.com

G. Anexo. Planilla juicio de expertos

El contenido del formulario muestra la siguiente información.,

URL Formulario Google. https://forms.gle/1Cczi8bSLC9BZeWJ8

Titulo. Validación - Marco de Buenas Prácticas SCRUMSEC

Descripción. La evaluación de los instrumentos es de gran relevancia para lograr que sean válidos y que los resultados obtenidos a partir de éstos sean utilizados eficientemente; aportando tanto al área investigativa de la psicología como a sus aplicaciones. Agradecemos su valiosa colaboración (Pérez & Martínez, 2008).

Manejo de datos personales. La información aquí recolectada es con fines académicos, su información personal, académica y profesional no será tratada con ningún fin distinto al indicado.

Información requerida.

- Nombres y Apellidos.
- Formación Académica.
- Área de Experiencia Profesional.
- Cargo Actual – Institución.

Calificación de los Ítems. De acuerdo con la información expuesta en la presentación califique cada uno de los ítems según corresponda.

Tabla 46. Anexo F – Clasificación de los ítems.

Categoría	Calificación	Indicador
Suficiencia. La información expuesta pertenece a una misma dimensión y basta para obtener la medición de esta.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta no es suficiente para medir la dimensión La información expuesta mide algún aspecto de la dimensión, pero no corresponde con la dimensión total Se debe incrementar alguna información para poder evaluar la dimensión completamente La información expuesta es suficiente
Claridad. La información expuesta se comprende fácilmente, es decir, su sintáctica y semántica son adecuadas.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta no es clara La información expuesta requiere bastantes modificaciones o una modificación muy grande en el uso de las palabras de acuerdo con su significado o por la ordenación de estas Se requiere una modificación muy específica de algunos de los términos de la información expuesta

		La información expuesta es clara, tiene semántica y sintaxis adecuada
Coherencia. La información expuesta tiene relación lógica con la dimensión o indicador que está midiendo.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta no tiene relación lógica con la dimensión La información expuesta tiene una relación tangencial con la dimensión La información expuesta tiene una relación moderada con la dimensión que está midiendo La información expuesta se encuentra completamente relacionada con la dimensión que está midiendo
Relevancia. La información expuesta es esencial o importante, es decir debe ser incluida.	1. No cumple con el criterio 2. Bajo nivel 3. Moderado nivel 4. Alto nivel	La información expuesta puede ser eliminada sin que se vea afectada la medición de la dimensión La información expuesta tiene alguna relevancia, pero otra distinta puede estar incluyendo lo que mide éste La información expuesta es relativamente importante La información expuesta es muy relevante y debe ser incluida

Fuente. Tomado y adaptado de (Pérez & Martínez, 2008).

Redirección a la URL del formulario. Con el fin dar acceso de una manera fácil, sencilla y amigable a los expertos al instrumento de valoración se creó un código QR, que se encuentra junto con la URL del formulario al final de la presentación, allí ellos evaluaran el contenido y la pertinencia de lo expuesto.

H. Anexo. Resultados coeficiente de Kendall (W)

Figueredo, L. C. (10 de octubre de 2020). RELATED TEST - PDF (Suficiencia Claridad Coherencia Relevancia) KENDALL(COMPARE=PAIRWISE). Obtenido de IBM SPSS.