Polo del Conocimiento

**Pol. Con. (Edición núm. 50) Vol. 5, No 10**
**Octubre 2020, pp. 1027-1046**
**ISSN: 2550 - 682X**
**DOI: 10.23857/pc.v5i10.2658**

# *Metodologías de desarrollo de software seguro con propiedades agiles*

# *Secure software development methodologies with agile properties*

# *Metodologias seguras de desenvolvimento de software com propriedades ágeis*

Samuel Alfredo López-Rodríguez [I]
samu.lpz@gmail.com
https://orcid.org/0000-0001-8848-537X

Víctor René García-Peña [II]
sercomgar@hotmail.com
https://orcid.org/0000-0002-3088-3559

**Correspondencia:** samu.lpz@gmail.com

Ciencias Sociales y Políticas
Artículo de investigación

*__Recibido:__ 01 de septiembre de 2020 *__Aceptado:__ 23 de septiembre 2020 * __Publicado:__ 28 de octubre de 2020

I. Master Universitario en Ingeniería de Software y Sistemas Informáticos, Ingeniero en Sistemas, Actividades de Docencia en la Metodología Aprendizaje Basado en Proyectos ABP, Universidad Laica Eloy Alfaro de Manabí, Manta, Ecuador.

II. Diploma Superior en Diseño Curricular por Competencias, Magister en Redes de Comunicaciones, Ingeniero en Sistemas e Informática, Licenciado en Sistemas Computacionales, Técnico Ejecutivo Analista de Sistemas, Tecnólogo en Computación e Informática, Universidad Laica Eloy Alfaro de Manabí, Manta, Ecuador.

**Resumen**

Actualmente los avances tecnológicos están acelerando la construcción vertiginosa de software, estos se encargan de procesar grandes volúmenes de datos sensibles ya sea de personas o de organizaciones, aquello hace que la seguridad del software ya no sea considerada como un requisito no funcional, es ahora un elemento de suma importancia. Los ciberdelincuentes se han convertidos en expertos en perpetrar ataques por medio de la explotación de vulnerabilidades en el software, se hace necesario analizar las nuevas tendencias en cuanto a su construcción mediante la implementación de metodologías de desarrollo seguro. La seguridad es un tema que inquieta a investigadores y desarrolladores, que buscan un producto con el menor número de vulnerabilidades posible, pero a su vez también buscan un método de desarrollo ágil, por lo tanto, buscan implementar estas metodologías, pero no es fácil identificar las diferencias entre cada una de ellas, puesto que son variadas y complejas. Se observa entonces la importancia de investigar las metodologías de desarrollo para obtener un producto software más seguro, en base a esto se planteó el objetivo de analizar las metodologías de desarrollo seguro, mediante la recolección y tratamiento de la literatura correspondiente, un análisis comparativo de sus características y su exposición en el desarrollo. Como conclusión principal se corrobora la existencia de diferentes metodologías que se enfocan en la seguridad, pero de acuerdo a la bibliografía consultada solo las metodologías de desarrollo seguro Microsoft SDL y BSIMM poseen propiedades agiles.

**Palabras Clave:** Tecnología de la Información y comunicación; seguridad informática; Metodologías; software; desarrollo.

**Abstract**

Currently technological advances are accelerating the rapid construction of software, these are responsible for processing large volumes of sensitive data either from individuals or organizations, that makes software security is no longer considered as a non-functional requirement, it is now an element of paramount importance. Cybercriminals have become experts in perpetrating attacks by exploiting vulnerabilities in software, it is necessary to analyze new trends in terms of its construction through the implementation of secure development methodologies. Security is an issue that worries researchers and developers, who seek a product with as few vulnerabilities as possible, but in turn also seek an agile development method, therefore, they seek to implement these methodologies, but it is not easy to identify the

differences between each of them, since they are varied and complex. It is then observed the importance of investigating the development methodologies to obtain a more secure software product, based on this, the objective of analyzing the secure development methodologies was raised, through the collection and treatment of the corresponding literature, a comparative analysis of their characteristics and their exposure in the development. As main conclusion it is corroborated the existence of different methodologies that focus on security, but according to the consulted bibliography only the secure development methodologies Microsoft SDL and BSIMM have agile properties.

**Keywords:** Information and communication technology; information security; methodologies; software; development.

## Resumo

Atualmente os avanços tecnológicos estão acelerando a rápida construção de software, estes são responsáveis pelo processamento de grandes volumes de dados sensíveis de indivíduos ou organizações, o que faz com que a segurança de software não seja mais considerada como um requisito não funcional, é agora um elemento de suma importância. Os cibercriminosos se tornaram especialistas em perpetrar ataques explorando vulnerabilidades em software, é necessário analisar novas tendências em termos de sua construção através da implementação de metodologias seguras de desenvolvimento. A segurança é uma questão que preocupa pesquisadores e desenvolvedores, que buscam um produto com o mínimo de vulnerabilidades possível, mas por sua vez também buscam um método de desenvolvimento ágil, portanto, procuram implementar estas metodologias, mas não é fácil identificar as diferenças entre cada uma delas, uma vez que são variadas e complexas. Observa-se então a importância de investigar as metodologias de desenvolvimento para obter um produto de software mais seguro, com base nisso o objetivo de analisar as metodologias de desenvolvimento seguro foi levantado, através da coleta e tratamento da literatura correspondente, uma análise comparativa de suas características e sua exposição no desenvolvimento. Como principal conclusão, é corroborada a existência de diferentes metodologias com foco na segurança, mas de acordo com a bibliografia consultada, apenas as metodologias de desenvolvimento seguro Microsoft SDL e BSIMM possuem propriedades ágeis.

**Palavras-chave:** Tecnologia da informação e comunicação; segurança informática; metodologías; software; desenvolvimento.

## Introducción

Las metodologías de desarrollo se originan como una respuesta a la crisis de los años 70 que desencadeno un conjunto de problemas relacionados al proceso de construcción de un producto software eficaz, de calidad y a tiempo, existen numerosas metodologías dotadas con variadas características que las lleva a afrontar el desarrollo de software de diferentes formas, buscando obtener un producto de calidad, al pasar los años algunas de ellas han quedado obsoletas y no responden a los nuevos desafíos, la realidad actual exige la construcción guiada por métodos adaptables a los cambios basados en la seguridad y agilidad.

Los cambios de requisitos de un producto software cuando se solicita su construcción y además cuando está en plena etapa de desarrollo, conlleva la utilización de metodologías modernas que dejen atrás las tradicionales, debido a que sus conceptos son muy rígidos, no permiten la adaptabilidad y no toman en cuenta la seguridad, en consecuencia, estas no permiten satisfacer las necesidades del cliente. Para afrontar esta problemática se han establecido metodologías que usan técnicas agiles, que disminuyen el tiempo de fabricación del producto software, fallos técnicos, vulnerabilidades, rigidez al cambio de requisitos y a la evolución.

Un producto software puede ser vulnerable, ya sea por propios fallos de construcción o por ataques provocados, para conseguir la disminución de vulnerabilidades y que el producto sea considerado seguro, se deben aplicar medidas como la integración de conceptos de seguridad en todas sus etapas de elaboración. Al utilizar una metodología de construcción que permita integrar la seguridad desde el ciclo de vida del software, se busca conseguir un producto robusto de confianza, que realice solo las funciones para lo que fue creado, minimizando comportamientos inesperados de manera que se asegure la integridad, confiabilidad y confidencialidad.

El propósito del presente artículo es analizar las metodologías seguras, que aplican conceptos de seguridad en el ciclo de vida del software, y que cuenten con propiedades para construir un software de forma ágil, para ello se llevó a cabo una comparativa obteniendo como resultado que existen numerosas metodologías que se enfocan específicamente en la seguridad, pero de acuerdo a la bibliografía consultada solo las metodologías Microsoft SDL y BSIMM aplican la seguridad y además tienen propiedades agiles.

## Materiales y métodos

El tipo de investigación llevada a cabo fue de tipo hermenéutica, descriptiva y heurística, con el propósito de seleccionar y analizar varias fuentes documentales científicas, la información concerniente se sometió a un proceso de exploración, revisión bibliográfica, resumen y descripción (Guirao et al., 2008). El trabajo expondrá una comparativa de las metodologías de desarrollo de software seguro para lo cual se han realizado las siguientes etapas:

Etapa 1. Investigación de estudios previos, relacionados al tema obtenidos de fuentes bibliográficas.

Etapa 2. Selección y tratamiento de la información sobre las diferentes metodologías encontradas.

Etapa 3. Análisis comparativo de la información seleccionada y comprobación de ella.

Etapa 4. Obtención de la discusión y establecimiento de la conclusión sobre las metodologías de desarrollo de software seguro.

**Microsoft Trustworthy Computing SDL**

Microsoft ha establecido formalmente su desarrollo de software de seguridad mejorando el proceso SDL, durante sus "empujes de seguridad" de 2002, modificando sus procesos tradicionales de desarrollo de software integrando tareas y puestos de control destinados expresamente a mejorar la seguridad del software producidos por esos procesos, persiguiendo como propósito disminuir los defectos de diseño y el nivel de daño de cualquier falla de sus productos (Pradeep, 2014).

Microsoft ha declarado que su software desarrollado bajo el proceso SDL Inicialmente demostrado una reducción del 50 por ciento en los boletines de sus principales productos en comparación con versiones de los mismos productos desarrollados antes de SDL; Las estimaciones más recientes de Microsoft demandan hasta un 87 por ciento en los boletines de seguridad (Goertzel et al., 2007).

Modelo del proceso SDL optimizado de Microsoft

El modelo de Microsoft fue creado considerando variables como el tamaño de la organización, sus recursos, la parte directiva además de tomar en cuenta que el desarrollo seguro es un proceso que puede resultar costoso y complicado además de causar preocupaciones de cómo aplicarlo, las fases del modelo son (Microsoft Corporation, 2010):

- Formación, directivas y capacidades organizativas
- Requisitos y diseño
- Implementación

- Comprobación
- Lanzamiento y respuesta

Actividades de seguridad de SDL simplificadas.

- Las actividades propuestas implementadas como parte del proceso de desarrollo del software aportan beneficios en materia de seguridad, además es viable añadir actividades opcionales que quedan a juicio del asesor o equipo de seguridad.

**Ilustración 1:** Ciclo de vida de desarrollo de software de Microsoft: simplificado


**Fuente:** (Microsoft Corporation, 2010)

**Oracle Software Security Assurance Process**

Oracle Corporation afirma haber adoptado un proceso de desarrollo seguro en el que todos los desarrolladores están obligados a seguir normas de codificación segura y utilizar bibliotecas estándar de funciones de seguridad (autenticación, criptografía), y realizar pruebas de seguridad extensivas que incluyen pruebas de penetración, exploración automatizada de vulnerabilidades, validaciones contra listas de verificación de seguridad y seguridad de terceros (gobierno e industria). Sus productos se envían con guías de configuración seguras y las prácticas de gestión de la vulnerabilidad incluyen parches críticos correcciones a la base de código principal, las cuales se utilizan para informar las revisiones (Oracle, 2020). Oracle Software Security Assurance (OSSA) es una metodología de desarrollo seguro que incluye la seguridad en todas las fases del ciclo de vida del desarrollo de software, abarca el diseño y arquitectura, construcción, pruebas y mantenimiento de todos sus productos (Oracle Corporation, 2014).

Oracle ha formado a sus profesionales en la utilización de las normas de codificación seguras de conocimiento general en las áreas de principios, diseños y vulnerabilidades frecuentes para aportar una guía en cuestiones como la privacidad y validación de datos, administración de los usuarios. Los estándares deben estar aplicados en el diseño y construcción de sus productos, las normas de seguridad se han desarrollado por años e incorporan las mejores prácticas, así como

las lecciones aprendidas de las pruebas de vulnerabilidades, además la metodología permite la evaluación, pruebas y análisis de todos sus productos a lo largo de su vida útil (Oracle Corporation, 2014).

La pruebas y análisis incluyen tanto de sus componentes funcionales como aquellos no funcionales con la finalidad de verificar las características y la calidad del producto, se desarrollan diversos tipos de pruebas a las diferentes características de producto. Las pruebas funcionales son ejecutadas típicamente por equipos de control de calidad, los cuales verifican las características de seguridad que previamente se habían planificado (Oracle Corporation, 2014).

Oracle utiliza el software Critical Patch Update para actualizar sus productos mediante parches críticos de seguridad que son lanzados incluso fuera del plazo, como alternativa en caso de vulnerabilidades particularmente criticas también ofrecen instrucciones y procedimientos para el manejo de ellas, esto proporciona ventajas a los clientes como (Oracle Corporation, 2014):

- Seguridad máxima. Las vulnerabilidades son remediadas por orden de severidad, las vulnerabilidades más críticas son corregidas por actualizaciones de parches.
- Menores costos de administración. Mediante un plan de revisión de seguridad fijo se revisa el funcionamiento del producto y de los parches.
- Administración simplificada de parches: las actualizaciones de parches son acumulativas.

**CLASP**

Desarrollado por el experto en seguridad de software John Viega, arquitecto jefe de seguridad y vicepresidente de McAfee, Inc., Comprehensive, Lightweight Application Security Process (CLASP), está metodología fue diseñada para insertar seguridad en cada fase del ciclo de vida. CLASP ha sido puesto en libertad bajo licencia de código abierto. Su principal característica es un conjunto de 30 actividades que se centran en la seguridad, estas pueden integrarse en el proceso de desarrollo de cualquier proyecto. CLASP brinda un enfoque prescriptivo y proporciona documentación continua de actividades que las organizaciones deben realizar para mejorar la seguridad. Algunas de las 30 actividades clave incluyen lo siguiente (OWASP, 2016):

- Monitorizar las métricas de seguridad
- Identificar las funciones y los requisitos del usuario
- Investiga y evalúa las soluciones de seguridad

- Realizar análisis de seguridad del diseño del sistema
- Identificar e implementar pruebas de seguridad.

**Seven Touchpoints for Software Security**

En su libro Building Security, Gary McGraw describe 7 puntos e identifica a su juicio las prácticas de seguridad más destacadas que serán aplicadas a los artefactos en cada fase de desarrollo, estos puntos de contacto basados en la efectividad y experiencia se ordenan de la siguiente manera: (McGraw G. R., 2006).

- Revisión del código (los errores)
- Análisis de riesgo arquitectónico (defectos encontrados)
- Pruebas de penetración
- Pruebas de seguridad basados en el riesgo
- Casos de abuso
- Requisitos de seguridad
- Operaciones de seguridad

Conjuntamente con los siete puntos de contacto, McGraw menciona otro, este "Bonus" es el octavo y consiste en un análisis externo, este punto será realizado de forma independiente y consistirá en la realización de pruebas, evaluaciones y revisiones del software (Goertzel et al., 2007).

**TSP-Secure**

El Centro de Coordinación del instituto de ingeniería de software (SEI) y el Equipo de respuesta de emergencia informática (CERT) de la Universidad Carnegie Mellon CMU (CERT / CC) desarrolló el equipo proceso de software para desarrollo de software seguro (TSP-Secure). Los objetivos de TSP-Secure son reducir o eliminar las vulnerabilidades resultado de errores de diseño e implementación de las aplicaciones, y proporcionar la capacidad de predecir posibles vulnerabilidades en el software entregado (Davis, 2006). Construido en el TSP de SEI, la filosofía central de TSP-Secure incorpora dos valores fundamentales (Goertzel et al., 2007):

1. Los ingenieros y gerentes deben establecer y mantener un ambiente de trabajo en equipo. Los procesos operativos de TSP ayudan a crear ingeniería y fomentar un entorno orientado al equipo.

2. TSP está diseñado para guiar a los ingenieros a través del proceso de ingeniería, se reduzca la probabilidad de que inadvertidamente salte pasos, organizar pasos en un orden improductivo, o pasar tiempo innecesario averiguando el siguiente movimiento.

Es una metodología que se creó con el objetivo de crear un producto en de muy alta calidad en poco tiempo, se enfoca en el trabajo en equipo, en el rendimiento y en aprovechar eficientemente los recursos. TSP cuenta con un cúmulo de procesos bien definidos que muestran lo que se debe realizar en cada etapa y como esta se conecta con otra.

**Building Security In Maturity Model (BSIMM)**

Es una metodología que nació como una iniciativa del estudio de la seguridad en el desarrollo del software, su construcción se basó en datos adquiridos en 67 propuestas de seguridad de diversas empresas. BSIMM es un apoyo para medir la seguridad puesto que por medio de esta es posible comparar y contrastar la propia seguridad con la que tienen otras empresas, estas mediciones son útiles para planificar, estructurar y ejecutar la evolución de la seguridad del software. El propósito de BSIMM es cuantificar las actividades con otras iniciativas de seguridad para ello utiliza un marco de referencia con una terminología general para exponer los elementos más visibles de las iniciativas de seguridad, esto permite comparar iniciativas (McGraw, Migues, & West, 2013).

**Tabla 1:** Modelo de Referencia para la Seguridad del Software (MRSS)

| **Modelo de referencia para la seguridad del software (MRSS)** | | | |
|---|---|---|---|
| Gobernanza | Inteligencia | Puntos de contacto con el SSDL | Despliegue |
| Estrategia y Métricas | Modelos de Ataque | Análisis de la Arquitectura | Pruebas de Penetración |
| Cumplimiento y Política | Características de Seguridad y Diseño | Revisión de Código | Entorno del Software |
| Capacitación | Normas y Requisitos | Pruebas de Seguridad | Gestión de Configuración y Gestión de Vulnerabilidades |

**Fuente:** (McGraw, Migues, & West, 2013)

**Open SAMM**

Modelo de madurez para el aseguramiento de Software (SAMM) que sirve como marco de trabajo para exponer e implementar estrategias de seguridad para el software, fue definido para ser flexible y que pueda ser aplicado en toda organización, los recursos que brinda SAMM tienen como objetivo lo siguiente (Chandra, 2013).

- Evaluar las prácticas de desarrollo seguro en una organización

- Construir un programa de seguridad con etapas bien definidas
- Mostrar mejoras concretas en el programa seguridad de aplicaciones
- Definir y medir las actividades relacionadas a la seguridad en toda organización

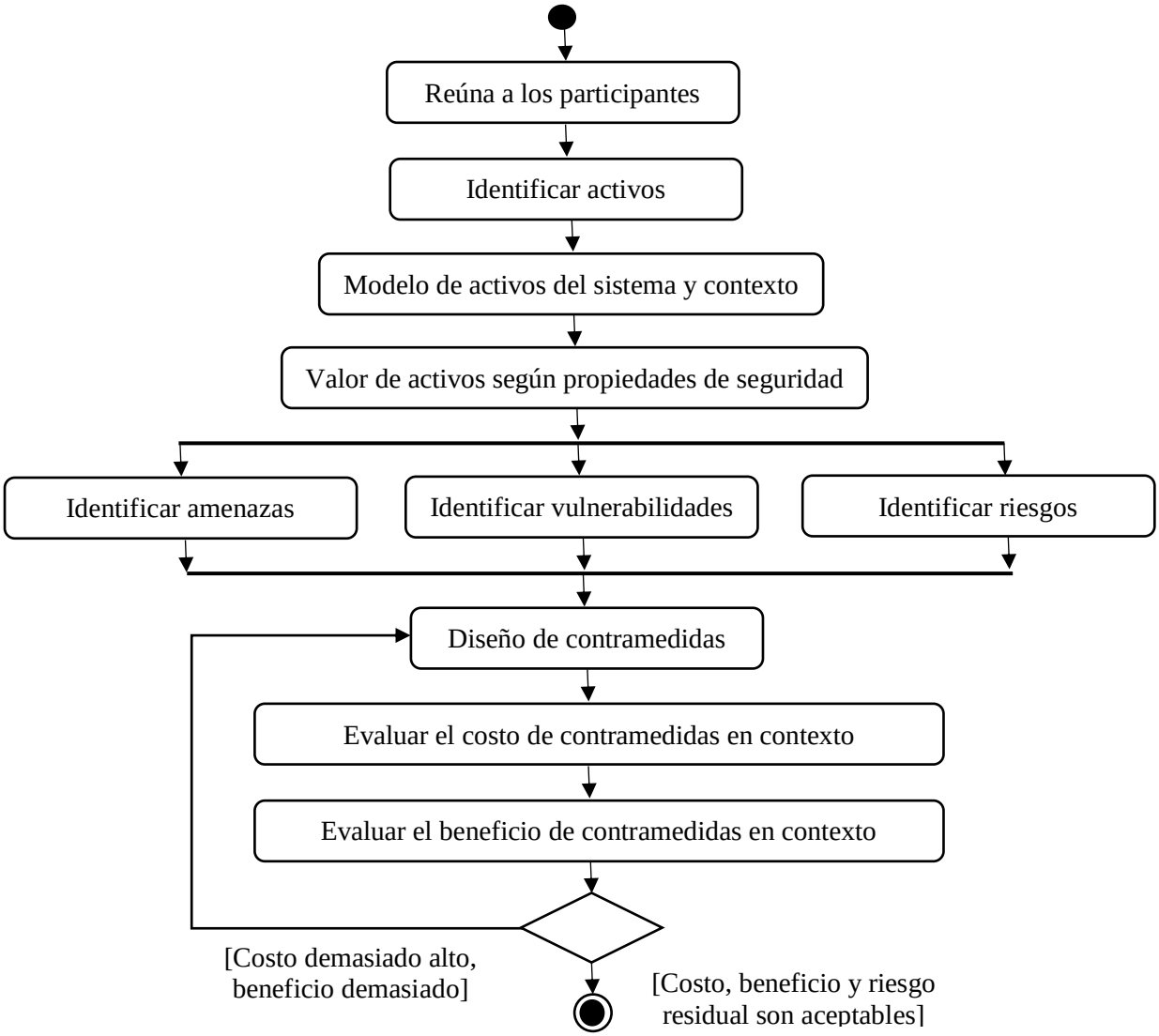
**Ilustración 2:** SAMM / Software Assurance


**Fuente:** (Chandra, 2013)

**Appropriate and Effective Guidance in Information Security (AEGIS)**

Es una metodología socio técnica de ingeniería de software creada para dotar de seguridad a sistemas fundamentados en modelos activos, determinación de requisitos de seguridad, análisis de riesgo y entorno de uso. El propósito es dar a los desarrolladores herramientas intuitivas y simples para desarrollar un software seguro, considerando las necesidades del usuario y fomentando la seguridad (Fléchais, 2005).

**Ilustración 3:** Diagrama de actividad AEGIS




**Fuente:** (Fléchais, 2005)

**Rational Unified Process-Secure (RUPSec)**

Es un modelo extendido del RUP (Proceso racional unificado) al que se le añade extensiones de seguridad, con el propósito de adicionar e integrar artefactos, actividades y roles al RUP para la captura, modelado y documentación de los requisitos y amenazas de seguridad del software, garantizando que este proceso sea implementado en todas las etapas de desarrollo subsiguiente (Ayatollahzadeh et al., 2005)

La metodología RUPSec propone que las actividades y artefactos se basen en los casos de uso del RUP, añadiendo casos de mal uso de uso para después desarrollar medidas que contrarrestaran las amenazas que fueron identificadas previamente con los casos de mal uso, las extensiones de seguridad al RUP se agregan a las siguientes actividades (Goertzel et al., 2007).

- Mantener las reglas comerciales.

- Determinar los actores del negocio y sus casos de uso, documentado los aspectos de seguridad y sus casos de uso.
- Definir las entidades y empleados, precise el nivel de acceso de cada uno.
- Determinar los requisitos de automatización, requisitos de seguridad empresarial y políticas de seguridad.
- Detallar los requisitos del software y definir los requisitos de seguridad.

**Secure Software Development Model (SSDM)**

Modelo desarrollado por el investigador Simon Adesina quie buscaba dar respuesta a los problemas de seguridad surgidos en los procesos de desarrollo de las empresas de software, el SSDM integra ingeniería de seguridad usando un modelo unificado que integra un conjunto de técnicas para producir un software seguro (Sodiya et al., 2006). SSDM delimita un flujo de trabajo seguro que consta de cuatro fases:

- Capacitación en seguridad. Educar al personal en conceptos de seguridad, conciencia, conocimiento de ataques, atacantes, intenciones e interés de los atacantes y el conocimiento de prácticas seguras.
- Modelado de amenazas. Indique a los usuarios los atributos del software, identifique a los posibles atacantes dentro del ambiente de operación, así como sus objetivos, técnicas, patrones y comportamientos, con la finalidad de construir un perfil del atacante. posteriormente determine las vulnerabilidades en función del modelo de amenaza.
- Especificaciones de seguridad. Enliste los posibles ataques y defina las probables medidas a tomar cuando se presenten problemas como, errores de desarrollo, implementación, monitoreo de seguridad, adaptación a situaciones de seguridad.
- Revisión de las especificaciones de seguridad. Revisión, pruebas de penetración de los posibles ataques identificados y los que a futuro se podrían presentar.

**Waterfall-Based Software Security Engineering Process Model**

El modelo en cascada fue propuesto por Winston W. Royce en 1970, es secuencial y su progreso es hacia abajo siguiendo una lista de fase una de tras de otra, pero solo cuando se complete fase anterior se podrá seguir a la siguiente (Bassil, 2012). Ha sido modificado por (Zulkernine &

Ahamed, 2016) y sugieren la integración al modelo actividades de seguridad y artefactos, las actividades se exponen a continuación.

**Tabla 2:** Waterfall-Based Software Security Engineering Process Model

| Fases del ciclo de vida | Actividades de ingeniería de seguridad añadidas |
|---|---|
| Ingeniería de sistemas | Analizar las amenazas de seguridad<br>Definir las necesidades de seguridad y limitaciones del software<br>Producir requisitos de seguridad informales |
| Especificación de requisitos | Identificar escenarios de ataque<br>Producir especificaciones de seguridad<br>Integre los requisitos funcionales de seguridad<br>Producir requisitos de seguridad y software combinados |
| Diseño de software | Diseñar basada en escenarios de ataque |
| Implementación | Identificar las fallas<br>Capacidad de controlar vulnerabilidades en caso de ataque, restructuración de código |
| Operación | Monitoreo de comportamientos inesperados, fallas e intrusiones<br>Integre nuevas especificaciones de ataque dentro de los requisitos formales. |

**Fuente:** (Goertzel et al., 2007).

### Viewnext

Es un modelo S-SDLC con adaptación ágil propuesto por (Caro, 2016) que aplica un concepto que se conoce como ciclo de vida de desarrollo de software seguro ágil, sin embargo, no se especifica cómo se debe implementar, la estructura del metamodelo es la siguiente.

- Políticas: Estrategias y orientación, formación, definición de riesgos.
- Metodología SDL: Validación de requisitos, modelado de amenazas, revisión de código, revisión de desarrollo, Testing de seguridad, validación de salidas.
- Supervisión: Evaluación y métricas, estado del proyecto
- Observatorio: Repositorio de vulnerabilidad, observatorio de seguridad, plan de respuesta e incidentes.

## Análisis y discusión de resultados

**Tabla 3:** Comparación de propuestas de seguridad

| | Nombre | Año | Aspectos destacados |
|---|---|---|---|
| SSDL | CLASP | 2006 | 24 mejores prácticas de seguridad formalizadas que cubren todo el SDL. |
| | Microsoft SDL | 2004 | Completa integración SDL en gran medida basado en el análisis de riesgos y basado en un conjunto de actividades para cada fase, integrado en el paquete de desarrollo de Microsoft. Adecuado para desarrolladores de sistemas operativos y grandes empresas de software. |
| | Touchpoints | 2004 | 7 mejores prácticas que se pueden aplicar al SDL existente |
| Inic iati | OpenSAMM | 2009 | Modelo de madurez medible con 3 niveles de madurez y 12 prácticas de seguridad. Se puede usar como punto de referencia |

| | | | |
|---|---|---|---|
| | BSIMM | 2009 | 12 mejores prácticas reales utilizadas por la industria con 3 niveles de madurez. Puede usarse como punto de referencia |
| | SAFECode | 2008 | Descripción de las mejores prácticas de la industria con énfasis en el liderazgo |
| | Securosis | 2009 | SSDL se enfoca específicamente en las aplicaciones web y se enfoca en la automatización mediante el uso de herramientas |

**Fuente:** (Association Information, 2014)

Las propuestas en la tabla comparativa muestran que todas las contribuciones poseen prácticas de seguridad, con diferentes enfoques, pero tienen similitudes en cuanto a la formación del personal, revisión de código, revisión arquitectónica, pruebas de penetración, documentación y políticas de seguridad, que busca un propósito fabricar un producto software más seguro.

**Tabla 4:** Comparación de las propuestas de seguridad más destacadas

**Fuente:** (Gerr, 2010)

| Característica | SAMM | BSIMM | SSDL | CLASP | SDL |
|---|---|---|---|---|---|
| **Recursos** | Desarrolladores<br>Arquitectos, administradores, QA Tester, Auditores de seguridad, dueños del negocio, personal de apoyo, operaciones | Constructores, (desarrolladores, arquitectos, administradores) Tester Operaciones<br>Administradores<br>Ejecutivos<br>Y mandos medios (dueños del negocio, administradores del producto) | Arquitecto, desarrolladores, testers, administradores de programación | Tester oficia, administrador del proyecto, ingeniero de requerimientos, arquitecto de software | Tester y líderes de equipo |
| **Tiempo** | 20 practicas | 109 actividades | 6 fases | 24 actividades | 5 fases y 16 procedimientos |
| **Popularidad** | 12% | 13% | 7% | 13% | 23% |

En esta comparativa según (Gerr, 2010) se exponen las propuestas de seguridad de más popularidad, en la cual SDL termina siendo la que mayor popularidad debido a la documentación existente, la utilización de menos recursos, esto indica que sería la más indicada para empezar un proyecto, pero se debe considerar que no todos los proyectos son iguales y no todas las organizaciones se acoplan a los diferentes métodos, así que pueden existir diferencias pero tienen en común ser las más conocidas, documentadas y que ofrecen la guía para la construcción de un producto seguro y robusto.

**Tabla 5:** Comparación entre SDL y Touchpoints

| Propiedad | SDL | TOUCHPOINTS |
|---|---|---|
| **General** | | |
| Aplicado a | Fase del ciclo de vida del software | Artefactos de desarrollo de software |
| Adopción | Actividades de seguridad autónomas o todas (recomendadas), dieciséis obligatorias para cumplir con SDL | Independiente o todos los puntos de contacto (recomendado) |
| Proceso | Pesado y rígido | Bastante ligero y flexible |
| Equipo de educación | Parte fundamental de SDL | Se presta poca atención al tema |
| **de acuerdo a las fases de desarrollo** | | |
| Inicio del proyecto | El personal involucrado está organizado y los roles asignados, el asesor de seguridad es el más importante. determinar el tipo de errores de seguridad que serán atacados. Abordar los aspectos de logística (por ejemplo, herramientas). Recopila y evalúa métricas | Recolectar y evaluar métricas, mejorar la estrategia de medición |
| Análisis y requisitos | Análisis de riesgos basado en escenarios de uso | casos de anti adecuaciones y abusos, análisis de riesgos arquitectónicos, requisitos de seguridad adicionales pueden identificarse |
| Diseño arquitectónico y detallado | Modelado riguroso y completo de amenazas. En el nivel de diseño detallado: evaluar el impacto del proyecto en la privacidad del usuario y la reducción de la superficie de ataque | El enfoque principal está en el modelo de amenaza: identificación de amenazas y evaluación de riesgos |
| Implementación y prueba | Aparte de las pautas de codificación segura, SDL no tiene actividades de implementación reales. Enfoque en pruebas de seguridad, sombrero mayormente negro | Enfatiza la importancia de las pruebas de seguridad: tres de los siete puntos de contacto se ocupan de las pruebas de seguridad. Prueba de sombrero blanco y sombrero negro. |
| lanzamiento de implementación y soporte | Se centra en el plan de respuesta que define qué hacer cuando se descubre una nueva vulnerabilidad. La comunicación con el cliente es importante | Los puntos de contacto tienen un soporte muy limitado en esta fase, solo contribuyen con el ajuste fino de los controles de acceso y la configuración de la monitorización y el registro |

**Fuente:** Adaptado de (Tiirik, 2013)

De la tabla comparativa se extrae lo siguiente:

- SDL es más rígido, cuenta con un conjunto de procesos y actividades bien definidos mientras que touchpoints es más ligero y adaptable.
- La educación en seguridad del software es un punto fuerte de SDL el equipo de desarrollo debe estar bien entrenado, mientras que en touchpoints el entrenamiento no es algo fundamental, pero se reconoce su importancia.
- SDL describe de cómo organizar el personal que está inmerso en el proyecto, mientras que touchpoints maneja una estrategia de medición contante mediante métricas.

- En la fase de análisis y requisitos SDL enfatiza la importancia de la codificación segura y utiliza escenarios para realizar el modelado de amenazas, mientras que touchpoints se enfatiza en pruebas de seguridad como caja blanca y caja negra.
- La etapa final del proyecto tanto SDL como touchpoints realizan revisiones de seguridad mientras que la primera realiza pruebas de caja negra y una revisión de seguridad final, touchpoints usa caja negra y caja blanca, las dos se destacan por la utilización de herramientas automatizadas de revisión de código.

**Tabla 6:** Análisis comparativo de S-SDLC a nivel de recursos, artefactos, propiedades ágiles y uso

| **Modelo** | **Recursos** | **Artefactos** | **Ágil** | **Uso en la industria** |
|---|---|---|---|---|
| McGraw | Guías de diseño seguras | Casos de abuso para obtener casos de prueba | **No** | **Reportado** |
| Microsoft SDL | Guías de diseño seguras | - | **Si** | **Reportado** |
| CLASP | Diseño de seguridad y guías de implementación, listas de vulnerabilidad y sus posibles mitigaciones | Pruebas basadas en requisitos | **No** | **Reportado** |
| TSP | - | - | **No** | **No reportado** |
| RUPSec | Experto en seguridad | - | **No** | **No reportado** |
| BSIMM | Modelos de seguridad, estándares, y procesos repetibles | Dominios de seguridad | **Si** | **Reportado** |
| OPEN SAMM | Codificación de seguridad y herramientas de medición | - | **No** | **Reportado** |
| AEGIS | - | Casos de abuso para obtener casos de prueba | **No** | **No reportado** |
| SSDM | - | - | **No** | **No reportado** |
| Writing Secure Code | Guías de mejores prácticas y programación segura | Elementos clave | **No** | **No reportado** |
| Waterfall | Mejores prácticas | - | **No** | **Reportado** |
| Viewnext | políticas | Niveles de seguridad | **Si** | **No reportado** |
| SecSDM | ISO (internacional Organización para Estandarización) normas, NIST (Instituto Nacional de Estándares y Tecnología) pautas | - | **No** | **No reportado** |
| S2D-ProM | - | - | **No** | **No reportado** |
| CbyC | Métodos formales | - | **No** | **No reportado** |
| SQUARE | - | - | **No** | **No reportado** |
| Oracle SSA | Mejores prácticas de codificación segura | Políticas | **No** | **Reportado** |

**Fuente:** Adaptado de (Mohino et al., 2019)

Cada metodología expuesta se fundamenta en la seguridad, pero solo dos resaltan en su uso en la industria y que cuentan con propiedades agiles, Microsoft SDL y BSIMM.

## Discusión

La incógnita planteada en esta investigación salió del propósito de conocer las metodologías seguras que tengan propiedades agiles, se analizó la bibliografía y se hizo una comparativa, destacando las propiedades y actividades principales de las metodologías seguras. Las metodologías Microsoft SDL con su proceso simplificado ágil donde se incluyen actividades de seguridad y BSIMM que tiene atributos agiles que se adaptan a las tendencias del mercado, son las metodologías seguras que tienen propiedades agiles, pero (Mohino et al., 2019) desatacan también Viewnext, propuesta que no se especifica como debe implementarse y que todavía no reporta uso en el mercado, razón por la cual no podría ser usada en los actuales momentos, puesto que como menciona (Ríos & Suntaxi, 2008) para seleccionar una metodología es necesario que el equipo tenga un grado de conocimiento, además (Carvajal, 2008) también destaca que es necesario que exista documentación, certificación y formación, por estos motivos esta no sería una opción para comenzar a construir software seguro de forma ágil.

En cuanto a que metodología segura es mejor que otra (Goertzel et al., 2007) definen 9 metodologías que han sido exitosas en múltiples proyectos a nivel mundial y que en ellas se aplica un modelo estándar de ciclo de vida, todas se fundamentan en la incorporación de prácticas de seguridad. (Hudaib et al., 2017) mencionan que elegir un proceso de desarrollo es un verdadero desafío sin conocer la diferencia entre ellas, y como menciona (Brito Abundis, 2013) la mejor metodología es aquella que se adapta al contexto, lo cierto es que el éxito y buenos resultados que pueda tener una metodología, dependerá del producto a fabricar y de algunos criterios como el grado de formación y conocimiento del equipo, documentación disponible y certificación de la metodología.

Por ultimo existen varios estudios y publicaciones en donde se utilizan metodologías seguras y agiles como lo es el trabajo expuesto por (Siponen et al., 2007) que indica cómo se pueden integrar las funciones de seguridad en un método ágil llamado desarrollo basado en características, mientras que (Ge et al., 2006) presenta un proceso ágil para producir aplicaciones web seguras pero indican además que la investigación que lleva a cabo no es el desarrollo de un nuevo método o proceso que se ocupa de cuestiones de seguridad, por el contrario se investigó los métodos de desarrollo y se los integro para abordar el desarrollo de

aplicaciones web seguras, confirmándose la necesidad de crear nuevas metodologías que integren la seguridad y agilidad al proceso de desarrollo.

Las opiniones de los autores y los resultados obtenidos en el análisis de las diferentes metodologías seguras, resaltan que las metodologías de desarrollo seguras más óptimas para construir un producto software seguro de forma ágil son Microsoft SDL y BSIMM, debido a sus actividades de seguridad, popularidad, uso en la industria, documentación, facilidad de implementación, y sus propiedades agílales.

## Conclusiones

En la actualidad existen varias metodologías de desarrollo seguro, muchas de estas comparten procesos semejantes y sus prácticas de seguridad son muy similares, buscan que el producto software sea desarrollado con el menor número de vulnerabilidades, además que las organizaciones y usuarios cuenten con una capa de protección ante posibles ataques.

El estudio de las metodologías demostró la posibilidad de mejorar la forma en cómo se construye software añadiéndole la seguridad y sus características junto con las técnicas agiles al proceso de desarrollo, permitiendo la construcción de productos software de calidad que pueden acoplarse en entornos altamente cambiantes conservando la seguridad.

Las metodologías Microsoft SDL y BSIMM se enfocan en la seguridad y poseen propiedades agiles, siendo las alternativas más óptimas para la construcción de un producto de calidad, siguiendo prácticas de seguridad y agilidad, están bien documentadas, certificadas y son muy utilizadas en la industria.

## Referencias

1. Association Information. (2014). Software Design and Development: Concepts, Methodologies, Tools, and Applications: Concepts, Methodologies, Tools, and Applications. IGI Global.
2. Ayatollahzadeh, M., Jaferian, P., Elahi, G., Baghi, H., & Sadeghian, B. (2005). RUPSec: An Extension on RUP for Developing Secure Systems - Requirements Discipline. PWASET, 208-212.
3. Bassil, Y. (2012). A Simulation Model for the Waterfall Software Development Life Cycle. ArXiv.
4. Brito Abundis, C. J. (2013). Metodologías para desarrollar software. ReCIBE.

5. Caro, A. (2016). Modelos de Desarrollo Seguro del Software. Obtenido de http://web.fdi.ucm.es/posgrado/conferencias/AndresCaroLindo-slides.pdf
6. Carvajal, J. (2008). Metodologías Ágiles: Herramientas y Modelo de desarrollo para aplicaciones Java EE como metodología empresarial. Obtenido de https://upcommons.upc.edu/handle/2099.1/5608?locale-attribute=es
7. Chandra, P. (2013). Software Assurance Maturity Model. San Fransisco: OWASP.
8. Davis, N. (2006). CISA. Obtenido de https://us-cert.cisa.gov/bsi/articles/knowledge/sdlc-process/secure-software-development-life-cycle-processes#tsp
9. Fléchais, I. (2005). Designing Secure and Usable Systems. University College of London, 57-59.
10. Ge, X., Paige, R., Polack, F., Howard, C., & Brooke, P. (2006). Agile development of secure web applications. ICWE '06 Proceedings of the 6th international conference on Web engineering.
11. Gerr, D. (2010). Companies Actually Using Secure Development Life Cycles? Computer Society.
12. Goertzel, K. M., Winograd, T., McKinley, H. L., Oh, L. J., Colon, M., McGibbon, T., . . . Vienneau, R. (2007). Software Security Assurance. ATAC , DACS.
13. Guirao-Goris, J., Olmedo Salas, A., & Ferrer Ferrandis, E. (2008). El artículo de revisión. Revista Iberoamericana de Enfermeria Comunitaria, 15-25.
14. Hudaib, A., AlShraideh, M., Surakhi, O., & Khanafseh, M. (2017). A Survey on Design Methods for Secure Software Development. International journal of Computersand Technology, 1047-7062.
15. Laskowski, J. (2011). Agile IT Security Implementation Methodology. Reino Unido: Packt Publishing Ltd.
16. McGraw, G. R. (2006). Software Security: Building Security In Addison-Wesley Software Security. Pearson Education.
17. McGraw, G., Migues, S., & West, J. (2013). Bsimm. Obtenido de http://www.fundacionsadosky.org.ar/wp-content/uploads/2014/07/BSIMM-V-esp.pdf
18. Microsoft Corporation. (2010). Microsoft Corporation. Obtenido de https://www.microsoft.com/en-us/sdl/default.aspx

19. Mohino, J. d., Higuera, J. B., & Higuera, J. R. (2019). The Application of a New Secure Software Development Life Cycle (S-SDLC) with Agile Methodologies. Electronics, 04-06.
20. Oracle. (2020). Oracle Software Security Assurance. Obtenido de https://www.oracle.com/es/corporate/security-practices/assurance/
21. Oracle Corporation. (2014). Oracle. Obtenido de http://www.oracle.com/us/support/library/software-security-assurance-2293569.pdf
22. OWASP. (2016). OWASP. Obtenido de https://www.owasp.org/index.php/CLASP_Concepts
23. Pradeep, V. (2014). MSPoweruser. Obtenido de https://mspoweruser.com/microsoft-talks-about-sdl-and-how-it-changed-the-security-landscape-in-the-software-industry/
24. Ríos, E., & Suntaxi, W. (2008). Desarrollo de un sistema informático para los procesos de cosecha y post cosecha de la camaronera Pampas de Cayanca. Obtenido de https://bibdigital.epn.edu.ec/bitstream/15000/1072/1/CD-1905.pdf
25. Siponen, M., Baskerville, R., & Kuivalainen, T. (2007). Extending Security in Agile Software Development Methods. Idea Group.
26. Sodiya, A., Onashoga, S., & Ajayí, O. (2006). Towards Building Secure Software Systems . Issues in Informing Science and Information Technology , 635-645.
27. Tiirik, K. (2013). Comparison of SDL and Touchpoints. Obtenido de https://courses.cs.ut.ee/MTAT.03.246/2013_spring/uploads/Main/essay09.pdf
28. Zulkernine, M., & Ahamed, S. I. (2016). Software Security Engineering: Toward Unifying Software Engineering and Security Engineering. Enterprise Information Systems Assurance and System Security: Managerial and Technical, 215-236.