ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE SISTEMAS

PROYECTO DE DESARROLLO

PROPUESTA DE MODELO DE SCRUM SEGURO APLICADO A UN CASO DE ESTUDIO

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL GRADO DE MAGISTER EN SOFTWARE MENCIÓN SEGURIDAD

WILSON GEOVANNY PANJÓN QUINDE

wilson.panjon@epn.edu.ec

Director: PhD. JENNY GABRIELA TORRES OLMEDO

jenny.torres@epn.edu.ec

Codirector: PhD. PAMELA CATHERINE FLORES NARANJO

pamela.flores@epn.edu.ec

2019

APROBACIÓN DEL DIRECTOR

Como director del trabajo de titulación PROPUESTA DE MODELO DE SCRUM SEGURO APLICADO A UN CASO DE ESTUDIO desarrollado por Wilson Geovanny Panjón Quinde, estudiante de Maestría en Software mención seguridad, habiendo supervisado la realización de este trabajo y realizado las correcciones correspondientes, doy por aprobada la redacción final del documento escrito para que prosiga con los trámites correspondientes a la sustentación de la Defensa oral.

PhD. JENNY TORRES

DIRECTOR

DECLARACIÓN DE AUTORÍA

Yo, Wilson Geovanny Panjón Quinde declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

La Escuela Politécnica Nacional puede hacer uso de los derechos correspondientes a este trabajo, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Wilson Geovanny Panjón Quinde

DEDICATORIA

A mi amada esposa Vanessa, y a mis pequeñas Camila y Emilia por compartir este viaje conmigo y ser esa fuente de amor y coraje para culminar esta etapa de mi vida.

A Javier y Carmen esos padres que realmente son un regalo precioso de Dios y que me guiaron y me siguen guiando en este proceso llamado vida.

Wilson.

AGRADECIMIENTO

A Dios por darme esa seguridad y confianza de que nunca estuve, estoy o estaré solo.

A mi directora de tesis PhD. Jenny Torres y mi co-directora PhD. Pamela Flores por el apoyo brindado y sus palabras de aliento durante la realización de este proyecto.

A Ericka, mi jefa y amiga por el apoyo y comprensión que me brindo durante mis estudios de maestría.

A Paul, mi amigo y consejero que me dio la oportunidad de trabajar en otra ciudad sin conocerme, que me apoyo con ideas y sugerencias para completar este documento.

A mis compañeros de trabajo que, con cada alegría, tristeza y frustración me ayudaron a identificar mis dificultades e influyeron a que estudie para seguir mejorando.

Wilson.

ÍNDICE DE CONTENIDO

LISTA DE FIGURAS ……..…….....i
LISTA DE TABLAS …….……..…..……...……..…...ii
RESUMEN …..…….…….……..iii
ABSTRACT …….…….……...…….....iv

1. INTRODUCCIÓN.....1
1.1. PREGUNTA DE INVESTIGACIÓN..... 2
1.2. OBJETIVO GENERAL..... 2
1.3. OBJETIVOS ESPECÍFICOS..... 2
2. MARCO TEÓRICO.....3
2.1. METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE..... 3
2.1.1. KANBAN..... 4
2.1.2. EXTREME PROGRAMMING..... 5
2.1.3. SCRUM..... 5
2.1.4. SCRUMBAN..... 7
2.1.5. SCRUM XP HYBRID..... 8
2.2. METODOLOGÍAS DE DESARROLLO DE SOFTWARE SEGURO..... 9
2.2.1. MICROSOFT SD3+C PRINCIPLES. 11
2.2.2. MICROSOFT SECURITY DEVELOPMENT LIFECYCLE SDL..... 11
2.2.3. NIST 800-64..... 12
2.2.4. COMPREHENSIVE LIGHTWEIGHT APPLICATION SECURITY PROCESS ... 13
2.2.5. BUILDING SECURITY IN MATURITY MODEL 13
2.2.6. AGILE DEVELOPMENT USING MICROSOFT SECURITY DEVELOPMENT LIFECYCLE..... 14
2.2.7. SSE-CMM SYSTEM SECURITY ENGINEERING – CAPABILITY MATURITY MODEL..... 15
2.2.8. VAHTI 16
2.2.9. OWASP SECURE SOFTWARE DEVELOPMENT LIFECYCLE..... 16
2.2.10. SAMM SOFTWARE ASSURANCE MATURITY MODEL 16

3. REVISIÓN SISTEMÁTICA DE LITERATURA 17
3.1. FASE DE BÚSQUEDA 17
3.1.1. CADENA DE BÚSQUEDA 17
3.1.2. CRITERIOS DE INCLUSIÓN Y EXCLUSIÓN 17
3.1.3. FORMAS DE EXTRACCIÓN 18
3.1.4. CRITERIOS DE SELECCIÓN 18
3.2. FASE DE EJECUCIÓN 18
3.3. FASE DE EXTRACCIÓN 19
3.4. RESULTADOS DE LA REVISIÓN DE LA LITERATURA 31
4. METODOLOGÍA 32
4.1. INVESTIGACIÓN – ACCIÓN 32
4.2. CASO DE ESTUDIO: EMPRESA PRIVADA 32
4.2.1. CONTEXTO 32
4.3. DISEÑO DE LA INVESTIGACIÓN 33
5. IMPLEMENTACIÓN DEL DISEÑO DE INVESTIGACIÓN 36
5.1. DIAGNÓSTICO 36
5.2. PLAN DE ACCIÓN 36
5.3. TOMA DE ACCIÓN 36
5.4. EVALUACIÓN 44
5.5. APRENDIZAJE 46
6. DISCUSIÓN 66
7. CONCLUSIONES Y RECOMENDACIONES 68
7.1. CONCLUSIONES 68
7.2. RECOMENDACIONES 68
REFERENCIAS BIBLIOGRÁFICAS 69

LISTA DE FIGURAS

Figura 1.- Flujo de trabajo de Kanban 4
Figura 2.- Modelo de eXtreme Programming 5
Figura 3.- Modelo de Scrum 7
Figura 4.- Modelo de proceso de ScrumBan 8
Figura 5.- Modelo Scrum XP Hybrid 8
Figura 6.- Reporte de uso de Metodologías Ágiles 9
Figura 7.- Conceptos de seguridad 10
Figura 8.- Modelo SAMM 16
Figura 9.- Modelo de Scrum Seguro 24
Figura 10.- Modelo de Scrum Seguro 25
Figura 11.- Modelo de Scrum Seguro 26
Figura 12.- Modelo de Scrum Seguro 28
Figura 13.- Modelo de eXteme Programming seguro 29
Figura 14.- Modelo de Desarrollo de Aplicaciones Web seguras 30
Figura 15.- Organigrama de la Compañía 33
Figura 16.- Diseño de la investigación 34
Figura 17.- Ciclo de mejora continua de Deming 37
Figura 18.- Modelo propuesto de Scrum seguro 37
Figura 19.- Síntesis modelo Scrum seguro 43
Figura 20.- Respuestas de la pregunta 1 62
Figura 21.- Respuestas de la pregunta 2 63
Figura 22.- Respuestas de la pregunta 3 63
Figura 23.- Respuestas de la pregunta 4 64
Figura 24.- Respuestas de la pregunta 5 64
Figura 25.- Respuestas de la pregunta 6 65
Figura 26.- Respuestas de la pregunta 7 65
Figura 27.- Respuestas de la pregunta 8 65

LISTA DE TABLAS

Tabla 1.- Cadena de búsqueda 17
Tabla 2.- Criterios de inclusión y exclusión 17
Tabla 3.- Formas de extracción 18
Tabla 4.- Cadenas de búsqueda resultados totales 18
Tabla 5.- Cadena de búsqueda resultados relevantes 19
Tabla 6.- Extracción de resultados 19
Tabla 7.- Modelos y Metodologías de Seguridad 21
Tabla 8.- Análisis de los modelos en la literatura 30
Tabla 9.- Matriz de requerimientos de la compañía 34
Tabla 10.- Lista de verificación de la etapa de planificación 44
Tabla 11.- Lista de verificación de las etapas de identificación e implementación 45
Tabla 12.- Lista de verificación en la etapa de verificación 46
Tabla 13.- Lista de verificación de etapa de ajuste 46
Tabla 14.- Tareas del Sprint 0 48
Tabla 15.- Lista de verificación de la etapa de planificación Sprint 0 49
Tabla 16.- Tareas del Sprint 1 51
Tabla 17.- Lista de verificación etapa identificación e implementación Sprint 1 52
Tabla 18.- Lista de verificación en la etapa de verificación Sprint 1 53
Tabla 19.- Lista de verificación de etapa de ajuste Sprint 1 53
Tabla 20.- Tareas del Sprint 2 56
Tabla 21.- Lista de verificación de la etapa de planificación Sprint 2 57
Tabla 22.- Lista de verificación etapa identificación e implementación Sprint 2 58
Tabla 23.- Lista de verificación en la etapa de verificación Sprint 2 59
Tabla 24.- Lista de verificación de etapa de ajuste Sprint 2 60

RESUMEN

Esta tesis de Maestría crea un modelo de desarrollo de software seguro con metodologías ágiles. Utilizando Scrum como metodología base para agregar componentes de seguridad propuestos por estándares de seguridad y mejores prácticas de compañías de software. Considerando el hecho de que la detección temprana de requerimientos, fallas y errores de seguridad en el ciclo de desarrollo produce un producto de software seguro. El modelo resultante agrupa los eventos de Scrum en etapas y las define como: planificación, identificación, implementación, verificación y ajuste para listar las tareas que deben ocurrir en cada etapa. Además, el modelo propone un nuevo rol de seguridad encargado de coordinar y asegurar que las tareas de seguridad ocurran durante el proyecto. El modelo propone una lista de verificación de tareas de seguridad que deben ocurrir durante los eventos de Scrum para verificar si el proceso se está cumpliendo y detectar oportunidades de aprendizaje para el equipo. Este modelo puede contribuir a equipos de desarrollo de software a incluir seguridad en sus tareas en el ciclo de desarrollo.

Palabras clave: Scrum Seguro, Ciclo de Desarrollo de Software Seguro, Desarrollo de Software Seguro con Metodologías Ágiles.

ABSTRACT

This Master Thesis creates a secure software development model with agile methodologies. Scrum is used as base methodology to adding security components taking advantage of current security standards and best practices in software companies. There is a fact that secure product is due to early detection of security requirements, flaws and bugs in software development cycle. The proposed model bundles the Scrum events in stages such as: planification, identification, verification and acting to list task involved in each stage. Model proposes a new Scrum role to coordinate and verify security tasks are executed in each event defined in model. Besides, model creates checklist for the security tasks executed in the Scrum events to verify if they were is executed in the specific sprint and detect learning opportunities for the Scrum team. This model could contribute to software development teams to include security in development cycle.

Keywords: Secure Scrum. Secure Software Development Lifecycle, Secure Software Development in Agile.

1. INTRODUCCIÓN

El ciclo de desarrollo de software utiliza marcos de trabajo, para la gestión de desarrollo de sistemas de software desde la captura de requerimientos hasta el mantenimiento y posterior retiro de estos. Existen diversos tipos de metodologías como cascada, iterativo e incremental, prototipos, ágiles, etc. cada uno con ventajas o desventajas al momento de ser utilizados, aunque en los últimos años la preferencia por las metodologías ágiles se ha incrementado sin dejar de lado las metodologías tradicionales [1].

El desarrollo de software seguro es un proceso modificado del ciclo de desarrollo, en donde se utilizan buenas prácticas y consideraciones de seguridad como NIST 800-64 [2], Microsoft SDL [3] y OWASP S-SDLC [4]. Estas buenas prácticas y consideraciones de seguridad se ajustan a entornos lineales y secuenciales en donde cada etapa del proceso de desarrollo tiene su tiempo y se pueden ir definiendo los requerimientos al pasar a la siguiente etapa [5]. Sin embargo, existen metodologías ágiles de desarrollo de software como Scrum, en donde en una o dos iteraciones de 15 o 30 días el producto es entregado al cliente y las especificaciones funcionales y no funcionales como seguridad y confiabilidad pueden o no ser priorizadas y definidas por el equipo de Scrum [6]. Además, que cada actualización de código entregado anula el aseguramiento de calidad previo [7].

Los incidentes de seguridad en computadores y redes continúan aumentando a medida que aumenta la accesibilidad a la red y la conectividad de dispositivos, estos incidentes tienen altos costos económicos debido a vulnerabilidades de seguridad en el software que permiten el robo de información personal, gubernamental, militar o empresarial además de afectar la reputación de la compañía [8]. Estos incidentes podrían haberse reducido y hasta eliminado si hubieran sido detectados los defectos en etapas tempranas del ciclo de desarrollo de software.

El objetivo de esta investigación es definir que eventos, artefactos o actores de Scrum se deben modificar o agregar para el manejo de seguridad en el diseño e implementación del ciclo de desarrollo [6, 9], utilizando buenas prácticas de seguridad, las cuales se identificarán en la literatura actual. Adicionalmente, se revisarán diferentes propuestas en las cuales se presenta la modificación del proceso de Scrum, principalmente con dos enfoques: agregar más artefactos y roles al proceso o definir los ítems del product backlog que presentan amenazas de seguridad o son requerimientos de seguridad [9].

En el mercado nacional e internacional Scrum se ha convertido en la metodología de desarrollo preferida por los beneficios que entrega al cliente, empresa de software y

personal humano [6]. Agregar un componente de seguridad contribuirá entonces a la comunidad de software a reducir las vulnerabilidades, incrementar los beneficios económicos a las compañías además de mejorar el conocimiento del equipo de desarrollo en temas de seguridad.

El presente documento está estructurado de la siguiente forma. En el capítulo 1 se define los objetivos de la investigación. El capítulo 2 recompila información sobre las metodologías de desarrollo ágiles y de seguridad. El capítulo 3 documenta los trabajos hasta ahora realizados con respecto a Scrum seguro. El capítulo 4 define la metodología a utilizar, el caso de estudio y el modelo propuesto. El capítulo 5 implementa el modelo propuesto en la compañía, el capítulo 6 las discusiones del modelo propuesto y el capítulo 7 las conclusiones del estudio.

1.1. Pregunta de investigación

¿Es posible añadir seguridad al proceso de desarrollo de software que utiliza metodologías ágiles como Scrum?

1.2. Objetivo general

Propuesta de un modelo de Scrum seguro utilizando buenas prácticas de seguridad.

1.3. Objetivos específicos

- Documentar el estado del arte de Scrum seguro al momento de realizar este documento.
- Documentar las buenas prácticas de seguridad utilizadas en el ciclo de desarrollo de software.
- Identificar las necesidades de seguridad en el desarrollo de software en la compañía.
- Diseñar un modelo de Scrum seguro que se adapte a la compañía considerando las metodologías de seguridad y artículos revisados.
- Aplicar y evaluar el modelo propuesto.
- Generar conclusiones del modelo resultante considerado las opiniones del equipo de Scrum.

2. MARCO TEÓRICO

Las metodologías de desarrollo de software ágil como Scrum y XP contribuyen a un proceso eficiente al aceptar cambios en los requerimientos en cualquier etapa del proyecto y entregar partes del producto priorizadas en cada sprint para ser evaluado por el usuario final [5]. Sin embargo, estas metodologías no definen la seguridad como parte del proceso de desarrollo; lo cual ha llevado a desarrollar aplicaciones con poca o ninguna consideración de la seguridad.

El software debe resistir, recuperarse y reducir el área de impacto de ataques identificados o no, para ser un software seguro. Las metodologías de software seguro agregan este componente de seguridad al proceso de desarrollo de software, permitiendo al desarrollador identificar las amenazas y las vulnerabilidades en etapas tempranas del proceso de desarrollo aplicando buenas prácticas de seguridad.

2.1. Metodologías ágiles de desarrollo de software

Por definición, "*Las metodologías de desarrollo de software proveen un marco de trabajo para planificar, ejecutar, y gestionar el proceso de desarrollo de sistemas de software*" [1]. Adicionalmente, proveen una guía para el equipo de desarrollo, pruebas, jefe de proyecto; definiendo el ciclo de vida del software desde la captura de requerimientos, análisis, diseño, implementación, pruebas, puesta en producción, mantenimiento hasta el retiro del software.

Las metodologías de software ágil son utilizadas para producir software con calidad en cortos periodos de tiempo. La utilización de enfoque iterativo e incremental, integración continua de código, y aceptación de cambios en los requerimientos de negocio ha permitido a las metodologías ágiles remplazar las metodologías tradicionales en negocios en donde las especificaciones no son claras o van cambiando con el tiempo, dependiendo de las necesidades del cliente [10].

En el año 2001, se reunió un grupo de desarrolladores en donde se estableció el manifiesto ágil, estableciendo cuatro principios básicos [11]:

1. *Se valora a los individuos y las interacciones sobre los procesos y las herramientas.*
2. *Se valora a las aplicaciones que funcionan sobre la documentación exhaustiva.*
3. *Se valora la colaboración del cliente sobre las negociaciones contractuales.*
4. *Se valora la respuesta al cambio sobre el seguimiento de un plan.*

Existe un amplio número de metodologías ágiles propuestas por la industria para el desarrollo de software como Kanban, eXtreme programming, Scrum, entre otras.

2.1.1. Kanban

Es una metodología de gestión de proyectos para el desarrollo de software que enfatiza la teoría "*Just in Time*" justo a tiempo. El objetivo de Kanban es definir con precisión que trabajo debe hacerse a través de priorizar las tareas, definir el flujo de trabajo y el plazo de entrega, reduciendo el riesgo de tareas incompletas y agregando flexibilidad entre tareas del proyecto. Los principios que define Kanban son:

- Límite del trabajo en progreso: define el límite de tareas que pueden estar en cada etapa del proceso de esta forma se pasa una tarea a la siguiente etapa solo si la etapa siguiente tiene capacidad disponible. Contribuye a identificar los cuellos de botella y de esta forma gestionarlos.
- Visualizar el flujo de trabajo: se crea un tablero con las etapas del proceso y las tareas a desarrollar. De esta forma cada vez que una tarea cambia de etapa se refleja en el tablero.
- Reducción del desperdicio: las tareas a realizar son las necesarias nada más.
- Calidad garantizada: las tareas completadas deben ser hechas con calidad.

Los desarrolladores del proyecto inician añadiendo componentes que agreguen valor al proyecto y no implementan atributos innecesarios; no empiezan con especificaciones que no puedan codificar, implementar, probar o publicar. Kanban elimina el desperdicio en cada etapa del proceso de desarrollo.

El flujo de trabajo de Kanban como se indica en la Figura 1 debe ser visible. Este flujo visualiza todos los pasos necesarios para completar una tarea. Cada tarea debe ser identificada con una nota. Además, cada etapa del proceso debe tener un límite para de esa forma restringir el trabajo del equipo y completar las tareas que están en progreso y de esta forma eliminar el desperdicio [12].

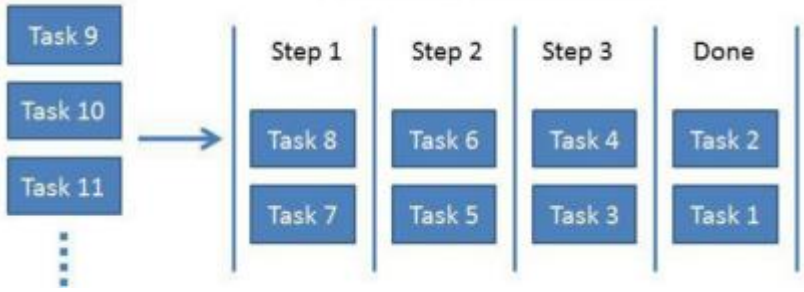

Figura 1.- Flujo de trabajo de Kanban
[12] pág. 4

2.1.2. eXtreme Programming

Es un marco de trabajo para desarrollo de software ágil. Su principal objetivo es producir software de alta calidad con un equipo de desarrollo con sobresaliente calidad de vida. Las características de eXtreme programming son: desarrollo iterativo e incremental, pruebas unitarias continuas, programación en parejas, frecuente comunicación con el cliente, refactorización de código, propiedad del código compartida y simplicidad en el código.[13] Los valores requeridos de esta metodología son: simplicidad, comunicación, retroalimentación, respeto y valor.

- Simplicidad: la solución más simple es la mejor opción.
- Comunicación: el equipo y el cliente tienen comunicación.
- Retroalimentación: se realiza una entrega del producto en cada iteración recibiendo retroalimentación del producto por parte del cliente y del proceso por parte del equipo de desarrollo.
- Respeto: cada uno de los miembros del equipo contribuye con entusiasmo y con las soluciones más simples, clientes y desarrolladores respetan la experiencia de cada miembro uno de ellos.
- Valor: decir la verdad sobre las estimaciones y el progreso del proyecto. No existe el temor al error porque todos trabajan como un equipo [14].

En la Figura 2 se visualiza el proceso de desarrollo con eXtreme Programming. En esta figura se puede identificar los componentes de la metodología: historias de usuario, aceptación de nuevos requerimientos, iterativo, pruebas de aceptación, mejora de las estimaciones con las iteraciones y pequeñas entregas del producto al cliente [13].

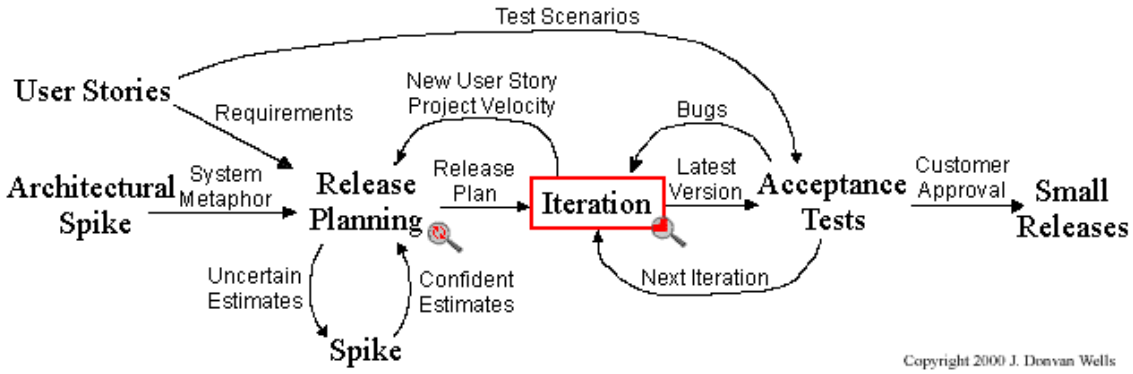

Figura 2.- Modelo de eXtreme Programming
[13] Flow Chart

2.1.3. Scrum

Es un marco de trabajo para el desarrollo y mantenimiento de productos complejos empleado en el desarrollo de software, usa un enfoque iterativo e incremental para

optimizar la previsibilidad y controlar el riesgo. Scrum está fundamentado en la teoría de control de procesos empírico definiendo que el conocimiento viene de la experiencia y la toma de decisiones basada en el conocimiento [15].

Scrum define tres elementos principales en su marco de trabajo para alcanzar el mejor resultado en un proyecto: los roles, eventos y artefactos. Los roles construyen el producto de la mejor forma posible enfocándose en la meta del sprint. Los roles en Scrum están compuestos por el Scrum master, Product Owner y equipo de desarrollo. El Product Owner es el representante del cliente ante el equipo de desarrollo; responsable de priorizar y definir los ítems del product backlog. El Scrum master es responsable de proteger el proceso, remover los impedimentos del equipo y facilitar el proceso al equipo de desarrollo y Product Owner. El equipo de desarrollo es multidisciplinario y autoorganizado, añade funcionalidad al producto.

Los eventos crean regularidad y minimizan la necesidad de reuniones durante el sprint. Los eventos de Scrum son: planificación del sprint, reunión diaria, revisión del sprint y la retrospectiva del sprint. Cada evento de Scrum tiene un límite de tiempo definido, facilitando la inspección y adaptación del proyecto si es requerido. La planificación del sprint permite definir el trabajo que va a realizar el equipo durante el sprint; ¿Qué? y ¿Cómo? Son las preguntas que se resuelven en esta reunión. Scrum diario permite identificar el estado del proyecto y los impedimentos existentes. La reunión de revisión del sprint presenta el producto desarrollado durante el sprint al cliente y así obtener retroalimentación del producto. La reunión de retrospectiva del sprint identifica y define las mejoras al proceso de desarrollo.

Finalmente, los artefactos son utilizados para transparentar el proyecto. Los artefactos de Scrum como product backlog y sprint backlog están diseñados para transparentar la información del trabajo realizado por el equipo en el proyecto y dar información para inspección y adaptación. El product backlog es una lista ordenada de los requerimientos del cliente, definida por el Product Owner. El sprint backlog es un conjunto de ítems del product backlog que el equipo se comprometió a entregar al final del sprint. En la Figura 3 se visualiza el modelo de Scrum descrito en esta sección.

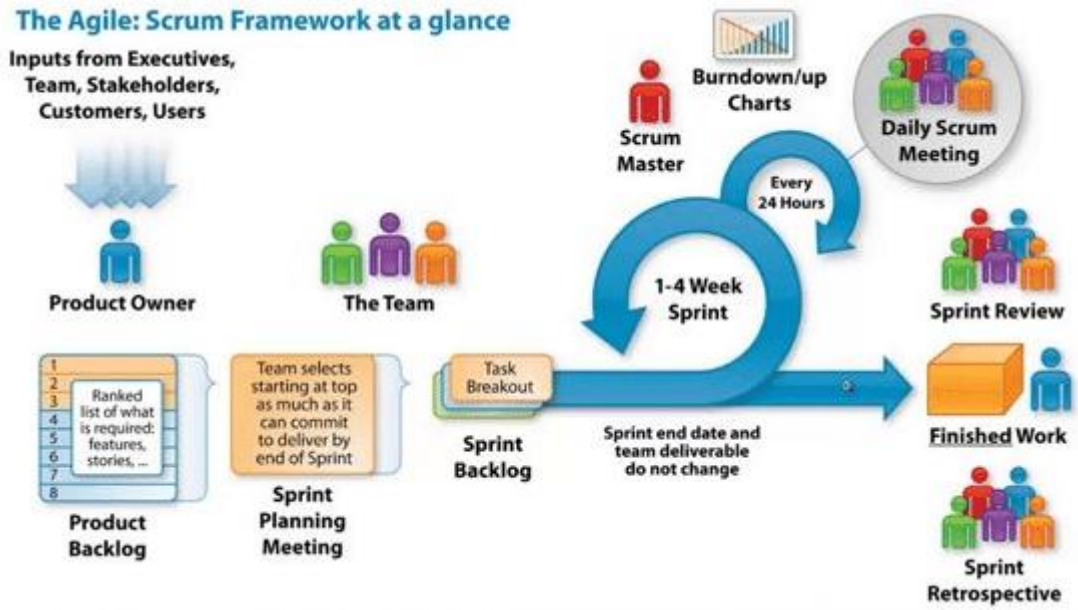

Figura 3.- Modelo de Scrum
[16] Fig. 1

2.1.4. ScrumBan

Es un sistema híbrido entre Scrum y Kanban que define pequeñas iteraciones y un tablero visual. La planificación es bajo demanda cuando el product backlog llega a un límite mínimo entonces las tareas planeadas son agregadas a la fase “To Do” del tablero. El tablero básico está compuesto por estas fases “To Do”, “Doing”, “Done” que permiten ir monitoreando el estado de las tareas. Además, se puede definir un límite en la capacidad de trabajo de cada una de estas fases y de los desarrolladores.

En este sistema, el Product Owner prioriza las tareas en el product backlog y el desarrollador va tomando los ítems desde el más prioritario, dependiendo de la capacidad definida para cada fase. Las reuniones diarias, planificación, revisión, retroalimentación son utilizadas también por este sistema para obtener más información del equipo, del cliente y mejorar el proceso continuamente [17].

En la Figura 4 se presenta el modelo de ScrumBan con el tablero de visualización del proceso de desarrollo.

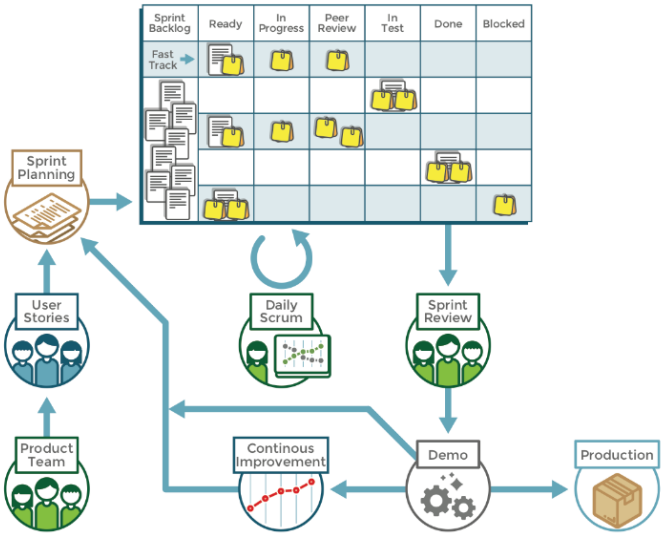

Figura 4.- Modelo de proceso de ScrumBan
[18] Fig. 3

2.1.5. Scrum XP Hybrid

Las metodologías Scrum y eXtreme Programming pueden ser utilizados juntos para maximizar el beneficio del uso de metodologías ágiles. eXtreme Programming esta principalmente enfocado en prácticas de ingeniería mientras que Scrum en la gestión del proceso. eXtreme Programming contribuye con desarrollo dirigido por pruebas, enfoque en pruebas automáticas, programación en pares, diseño simple, refactorización mientras que Scrum gestiona el trabajo a realizar a través de su equipo, eventos y artefactos.

La combinación de ambas metodologías crea mejores resultados como podemos ver en la Figura 5.

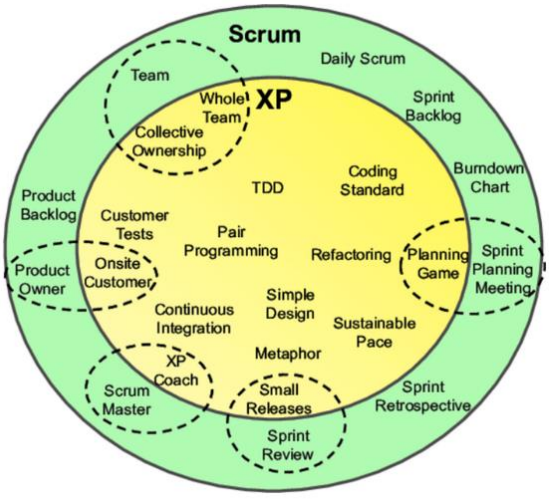

Figura 5.- Modelo Scrum XP Hybrid
[19] Fig. 1

Las razones para implementar metodologías ágiles en las compañías, según el reporte del 2019 que realiza CollabNet VersionOne, son: acelerar la entrega de software, facilidad de cambio de requerimientos, incremento de la productividad, alineamiento entre el negocio y las tecnologías de información, mayor calidad de software, visibilidad del proyecto, reducción de costos y mejora de la moral del equipo. La Figura 6 es parte del reporte de CollabNet [20] sobre el uso de metodologías ágiles al 2019.

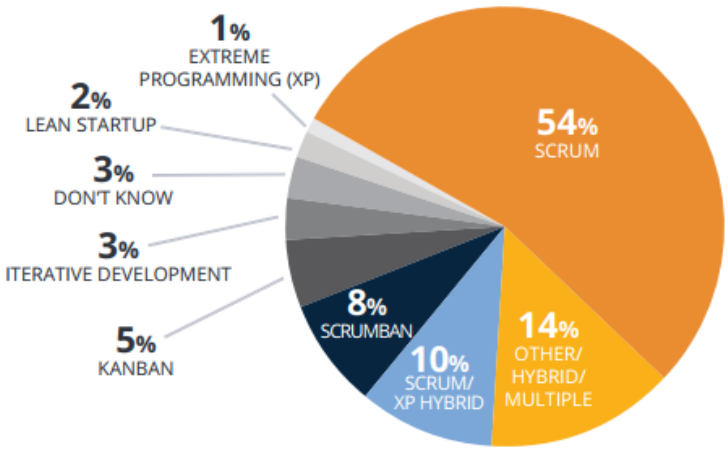

Figura 6.- Reporte de uso de Metodologías Ágiles
[20] pág. 9

Siendo Scrum la metodología ágil más utilizada, se propone analizar y modificar este marco de trabajo para agregar la seguridad como parte del proceso de desarrollo de software.

2.2. Metodologías de desarrollo de software seguro

A pesar del hecho de reconocer que la seguridad de redes, sistemas y software es un componente crítico para el funcionamiento de las organizaciones, cada día se van descubriendo nuevas vulnerabilidades en los sistemas. Los desafíos de la implementación de la seguridad están relacionados con el triángulo de gestión de proyectos (alcance, tiempo, presupuesto), agrega complejidad a la aplicación (más compleja al uso del usuario y de implementar) y valor de retorno al negocio (demostrar que la seguridad reduce costos a la compañía) [21].

La seguridad es una característica del software que refleja la capacidad del sistema para protegerse a sí mismo de ataques accidentales o deliberados.

La Figura 7 agrupa los conceptos de seguridad definido en la certificación CSSLP [21]. La confidencialidad es el acceso autorizado a la información del sistema, al código

del sistema, ejecución del sistema. La integridad es la capacidad del software de prevenir alteraciones o eliminaciones de los datos no autorizadas. La disponibilidad es que los datos y el sistema estén disponibles cuando el usuario lo necesite. No repudio es la capacidad del sistema para identificar las acciones que hizo un usuario en el sistema.

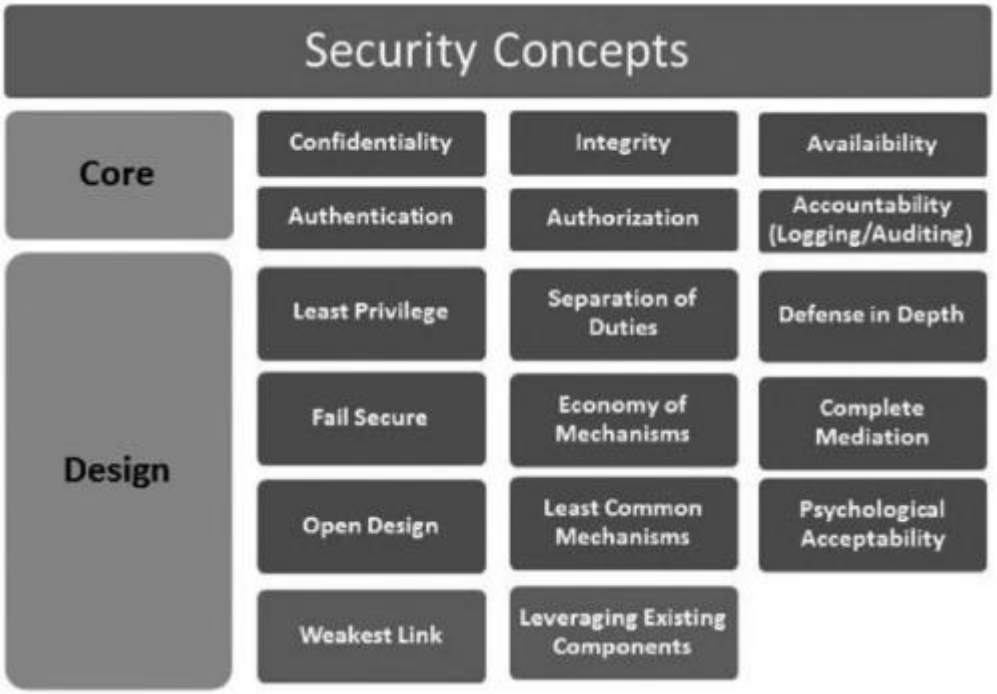

Figura 7.- Conceptos de seguridad
[21] Fig.1.4

Los principios de seguridad pueden guiar el diseño e implementación de software sin fallas de seguridad. Si en el proceso de desarrollo no se considera o se está violando un principio de seguridad, esta violación es un síntoma de una potencial falla en el proceso de desarrollo por lo cual es necesario revisar cuidadosamente el proceso para que la falla sea controlada [5].

- **Separación de privilegios**: el proceso de desarrollo necesita identificar a los desarrolladores y clientes considerandos sus responsabilidades.
- **Menos privilegios**: cada usuario del sistema debe utilizar el menor número de privilegios que le permita hacer su trabajo.
- **No acceso por defecto**: la acción por defecto es no acceso al sistema. El permiso es asignado solo bajo ciertas condiciones.
- **Economía del mecanismo**: mantener el diseño simple.
- **Psicológicamente aceptable**: las interfaces deben ser fáciles de usar para el usuario. De esta forma el usuario puede aplicar los mecanismos de seguridad de forma fácil y correcta.

- **Diseño abierto**: el diseño no debe ser un secreto, y el mecanismo de seguridad no puede depender del desconocimiento del diseño por parte del atacante.
- **Mecanismo menos común**: el mecanismo para compartir recursos no debe ser compartido.
- **Mediación completa**: cada acceso a objetos del sistema debe ser verificado para autorizarlo o no.

Las metodologías y buenas prácticas de desarrollo de software seguro se encargan de incluir actividades de seguridad al proceso de desarrollo. A continuación, se mencionan algunas metodologías y buenas prácticas de desarrollo de software seguro.

2.2.1. Microsoft SD3+C Principles.

Es un modelo general para el desarrollo de software seguro [22] que define 4 aspectos para desarrollar software de forma segura:

- **Seguridad por diseño:** la seguridad debe estar presente en cada etapa del ciclo de desarrollo.
- **Seguridad por defecto:** la configuración por defecto del software es por diseño lo más segura posible.
- **Seguridad en despliegue:** los componentes de privacidad y seguridad el software puede ser gestionado en el proceso de despliegue.
- **Comunicación:** Los desarrolladores de software deben estar preparados para descubrir vulnerabilidades y comunicar a los clientes abierta y responsablemente.

2.2.2. Microsoft Security Development Lifecycle SDL

Agrega privacidad y seguridad en todas las fases del proceso de desarrollo definiendo una guía, mejores prácticas y herramientas a ser seguidas para construir productos y servicios más seguros [3].

1. Entrenamiento en seguridad.
2. Definir los requerimientos de seguridad.
3. Define métricas y reportes de cumplimiento.
4. Modelado de amenazas.
5. Establecer requerimientos de diseño.
6. Definir el uso de estándares de criptografía.
7. Administrar los riesgos de seguridad del uso de componentes de terceros.
8. Utilización de herramientas aprobadas.
9. Pruebas de seguridad de análisis estático.
10. Pruebas de seguridad de análisis dinámico.

11. Pruebas de penetración.
12. Proceso estándar de respuesta a incidentes.

2.2.3. NIST 800-64

Describe un proceso de cinco fases [2]:

- **Fase de inicio**
  - Inicio de la planificación de la seguridad.
  - Categorización del sistema de información.
  - Análisis de impacto de negocio.
  - Análisis de impacto de la privacidad.
  - Aseguramiento del uso de procesos de desarrollo de sistemas de información seguro.
  - Planificación de entrenamiento de seguridad.
- **Fase de adquisición/desarrollo**
  - Evaluación inicial del riesgo.
  - Seleccionar y documentar los controles de seguridad.
  - Diseño de una arquitectura segura.
  - Implementación de control de seguridad en el diseño del sistema.
  - Documentación de seguridad.
  - Análisis de fortalecimiento de la seguridad.
  - Documentos iniciales para la Certificación y Acreditación de Sistemas.
- **Fase de implementación/evaluación**
  - Creación de un plan detallado para la certificación y acreditación del sistema.
  - Integrar seguridad en los entornos establecidos.
  - Evaluar la seguridad del sistema.
  - Acreditación de la seguridad. Autorizar a la aplicación el almacenamiento, procesamiento y transferencia de la información.
- **Fase de mantenimiento/operaciones**
  - Revisar la preparación operativa para manejar modificaciones no planificadas al sistema.
  - Realizar actividades de administración y control de la configuración para garantizar la consideración de posibles impactos de seguridad debido a cambios específicos en el sistema.
  - Realice un monitoreo continuo para garantizar la efectividad de los controles de seguridad a lo largo del tiempo.

- **Fase de eliminación**
    - Construir y ejecutar un plan de eliminación / transición.
    - Asegurar la protección de la información (copia de seguridad) y los métodos de recuperación.
    - Requisitos legales relacionados con la retención de registros, al desechar sistemas.
    - Política de saneamiento de medios para evitar la divulgación de información no autorizada.
    - Política de eliminación de hardware y software.
    - Política de cierre o desmontaje del sistema.

2.2.4. Comprehensive Lightweight Application Security process

Está diseñado para integrar actividades de seguridad en el proceso de desarrollo de software integrado por cinco niveles llamados vistas que interactúan entre ellos [23]:

- **Vista de conceptos:** definición de cómo el proceso funciona y como las vistas interactúan entre ellas.
- **Vista basado en roles:** creara roles de seguridad para el proyecto y que puedan ser utilizados en las siguientes vistas.
- **Vista de evaluación de la actividad:** evalúa cada una de las 24 tareas de seguridad propuestas por CLASP para aplicar o no en el proyecto, considerando costos, aplicabilidad y los riesgos de no hacer nada.
- **Vista de la implementación de la actividad:** Ejecutar las tareas de seguridad identificadas en la vista anterior.
- **Vista de vulnerabilidad:** se realiza una gestión de riesgos de las vulnerabilidades identificadas en las etapas anteriores.

2.2.5. Building Security in Maturity Model

Es un estudio de las iniciativas de desarrollo de software seguro existentes, compuesto de cuatro dominios y 12 prácticas. Los resultados proporcionan una forma de medir y evaluar las iniciativas de seguridad [24].

- **Gobernanza:** Organiza, gestiona y mide una iniciativa de seguridad de software, además del desarrollo del personal.

- **Inteligencia:** Creación de conocimiento corporativo utilizado en las actividades de seguridad de software de la compañía, incluye una guía de seguridad y un modelo de amenaza.
- **Punto de contacto de desarrollo de software seguro:** Análisis y aseguramiento de los artefactos y procesos del desarrollo de software.
- **Despliegue:** incluye las tareas de seguridad de red y mantenimiento de software.

2.2.6. Agile Development Using Microsoft Security Development Lifecycle

Microsoft propone una mejora al proceso de SDL considerando metodologías ágiles. Para completar este proceso Microsoft propone tres grupos de requerimientos a ser aplicados [25]:

Requerimientos en cada sprint: requerimientos esenciales de seguridad que ningún software podría ser publicado si no lo cumple:

- Ejecutar las herramientas de análisis diariamente o por compilación.
- Modelo de amenazas de todos los nuevos requerimientos.
- Miembros del equipo completen al menos un curso de seguridad por año.
- Utilizar encriptación en nuevo código.

Requerimientos del listado: tareas que necesitan ser ejecutadas regularmente durante la vida del proyecto, pero no son tan críticas para ser ejecutadas en cada sprint. Estos requerimientos están categorizados en tres grupos que son:

Tareas de verificación

- Análisis de superficie de ataque.
- Análisis binario.

Tareas de revisión del diseño

- Control de cuenta de usuario.
- Revisar el diseño de encriptación.

Tareas de planificación de respuesta

- Crear documentos de soporte de privacidad.
- Actualizar los contactos de respuesta de seguridad.

Requerimientos de una sola vez: requerimientos necesarios al iniciar un proyecto o empezar utilizando SDL.

- Crear línea base del modelo de amenazas.
- Identificar el experto de seguridad en el equipo.
- Utilizar la última versión del compilador.

2.2.7. SSE-CMM System Security Engineering – Capability Maturity Model

Modelo de madurez de ingeniería de seguridad de sistemas para evaluar los niveles de madurez de seguridad de una organización en tecnologías de información. Este modelo define las actividades de seguridad del ciclo de vida de un producto de software [26]. Este modelo divide la seguridad en 3 áreas básicas:

- **Riesgo:** identifica y prioriza los peligros del producto o sistema desarrollado.
- **Ingeniería de seguridad:** implementa soluciones a los peligros identificados
- **Aseguramiento:** establece confianza en las soluciones de seguridad.

Entre las áreas de proceso de ingeniería de seguridad del Sistema definidas por SSE-CMM tenemos [27]:

- PA01 Administer Security Controls
- PA02 Assess Impact
- PA03 Assess Security Risk
- PA04 Assess Threat
- PA05 Assess Vulnerability
- PA06 Build Assurance Argument
- PA07 Coordinate Security
- PA08 Monitor Security Posture
- PA09 Provide Security Input
- PA10 Specify Security Needs
- PA11 Verify and Validate Security

El modelo también incluye once áreas de proceso relacionadas con proyectos y practicas organizacionales:

- PA12 – Ensure Quality
- PA13 – Manage Configuration
- PA14 – Manage Project Risk
- PA15 – Monitor and Control Technical Effort
- PA16 – Plan Technical Effort
- PA17 – Define Organization’s Systems Engineering Process
- PA18 – Improve Organization’s Systems Engineering Process
- PA19 – Manage Product Line Evolution
- PA20 – Manage Systems Engineering Support Environment
- PA21 – Provide Ongoing Skills and Knowledge

- PA22 – Coordinate with Suppliers

2.2.8. VAHTI

Es parte de un grupo de normas de seguridad del gobierno de Finlandia es el estándar de facto para el desarrollo y mantenimiento de sistemas [28].

2.2.9. OWASP Secure Software Development Lifecycle

Es una metodología de desarrollo de software seguro. S principal objetivo es definir un estándar de seguridad para el ciclo de desarrollo de software. Este proyecto define procesos, herramientas, guías y listas de verificación de actividades en cada fase [4]:

- **Fase de requerimientos:** evaluación de riesgo y documentos de requerimientos.
- **Fase de diseño:** guías de revisión de diseño de seguridad y guías de modelo de amenazas.
- **Fase de implementación:** guías de seguridad de código.
- **Fase de validación:** guías de pruebas de seguridad y niveles de actividades.
- **Fase de mantenimiento y despliegue:** guías de gestión de vulnerabilidades y respuesta a incidentes.

2.2.10. SAMM Software Assurance Maturity Model

Es una guía para construir seguridad en el desarrollo de software. Los recursos que este modelo propone en [29] son:

1. Evaluar las prácticas de seguridad de software existentes en una organización.
2. Construir un programa de aseguramiento de seguridad de software.
3. Demostrar mejoras a las prácticas de aseguramiento de seguridad
4. Definir y medir las actividades relacionadas con la seguridad en una organización.

La Figura 8 indica el modelo propuesto por esta metodología.

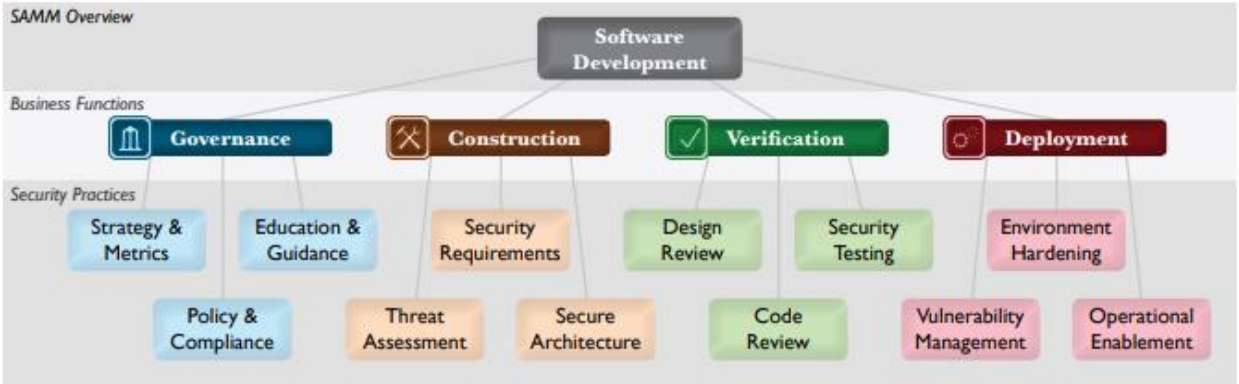

Figura 8.- Modelo SAMM
[29] pág. 3

3. REVISIÓN SISTEMÁTICA DE LITERATURA

Esta sección presenta una revisión metodológicamente de las investigaciones existentes sobre la temática expuesta [30].

3.1. Fase de búsqueda

El propósito de este trabajo es la generación de un modelo de seguridad con metodologías ágiles para mejorar el proceso de construcción de software.

3.1.1. Cadena de búsqueda

La cadena de búsqueda que se define para realizar la búsqueda en la literatura académica se presenta en la Tabla 1. La investigación se limita a identificar modelos de desarrollo seguro en metodologías de desarrollo de software ágil.

Tabla 1.- Cadena de búsqueda

Population	Software development
Intervention	Agile OR Scrum
Outcome	Secure
Context	Academia
Search strategy	Population AND Intervention AND Outcome

3.1.2. Criterios de inclusión y exclusión

La Tabla 2 define los criterios de inclusión y exclusión para seleccionar los artículos académicos que serán parte de la investigación. Se incluyen los documentos oficiales o propuestas de estándares por parte de compañías como Microsoft, OWASP, NIST, SCRUM. Además de restringe la investigación a modelos propuestos desde el 2015 en adelante.

Tabla 2.- Criterios de inclusión y exclusión

Inclusión	Metodologías ágiles Documentos oficiales de Microsoft, NIST, OWASP, Scrum y Scrum Alliance
Exclusión	Investigación antes del 2015

3.1.3. Formas de extracción

En la Tabla 3 se definen las fuentes de datos que fueron consultadas en la presente investigación se consideró tres revistas científicas muy utilizadas para publicar información en área informática

Tabla 3.- Formas de extracción

Fuente	**Enlace**
Elsevier	https://www.sciencedirect.com
ACM Digital Library	https://dl.acm.org/
IEEE Xplore Digital Library	https://ieeexplore.ieee.org/Xplore/home.jsp

3.1.4. Criterios de selección

Los criterios de selección de las investigaciones resultantes para ser considerados en este documento son:

- Idioma: inglés o español.
- Haber propuesto un modelo de seguridad o evaluación de un modelo.
- Ser únicos. No sean documentos duplicados.
- Desde el año 2015 en adelante.

3.2. Fase de ejecución

La Tabla 4 describen los resultados totales obtenidos con las cadenas de búsqueda.

Tabla 4.- Cadenas de búsqueda resultados totales

Cadena de búsqueda	**Termino asociado**	**Elsevier**	**ACM**	**IEEE**	**Total**
Secure	Scrum Software Development	2	6	7	15
Secure	Agile Software Development	4	82	21	110
Número total de trabajos encontrados					115

La Tabla 5 resume los documentos encontrados en la investigación y los agrupa en:

Estudios relevantes: son estudios que presentan un modelo de seguridad con una metodología ágil.

Estudios no relevantes: son estudios que la herramienta de búsqueda presenta pero que no relaciona completamente las cadenas de búsqueda. En este caso pueden ser trabajos de metodologías agiles o de seguridad que no están relacionados.

Estudios repetidos: Son modelos propuestos y en una siguiente fase de investigación se realizó la evaluación del modelo. Se considero el documento que presenta el modelo.

Incompleto: Son propuestas de modelos, pero no dan mucho detalle de si diseño. Así que es complejo entender el modelo en su totalidad.

Tabla 5.- Cadena de búsqueda resultados relevantes

Fuente	**Resultado de la Fuente**	**Estudios relevantes potenciales**	**No relevantes**	**Repetidos**	**Incompletos**	**Estudios relevantes**
Elsevier	6	2	4	0	0	2
ACM	88	12	2	4	2	4
IEEE	28	11	1	6	0	4
Número total de trabajos relevantes						11

3.3. Fase de extracción

La Tabla 6 lista los documentos identificados por relevancia para la investigación y la Tabla 7 identifica los artículos académicos que proponen un nuevo modelo y utilizan un estándar de seguridad.

Tabla 6.- Extracción de resultados

Titulo	**Autor**	**Año**	**Referencia**	**Relevancia**
Towards a Secure Scrum Process for Agile Web Application Development	Patrik Maier Zhendong Ma Roderick Bloem	2017	[9]	Alta
ScrumS: a model for safe	Rene Esteves María	2015	[6]	Alta

agile development	Luis Antonio Rodríguez Nelson Alves			
Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency	Kale Rindell Sami Hyrynsalmi Ville Leppaen	2016	[28]	Alta
Secure Scrum: Development of Secure Software with Scrum	Christoph Pohl Hans-Joachim Hof	2015	[31]	Alta
A comparison of security assurance support of agile software development methods	Kale Rindell Sami Hyrynsalmi Ville Leppanen	2015	[32]	Medio
Secure Software Engineering for Agile Methodology: Preliminary Investigation	Luthfi Ramadani Nur Ichsan Utama	2015	[33]	Medio
Integrating the Extreme Programing Model with Secure Process for Requirement Selection	Amit Singh	2018	[34]	Medio
Secure Paradigm For Web Application Development	B Subedi Abeer Alsadoon P W C Prasad A Elchouemi	2016	[35]	Medio
Towards a Secure Agile Software Development Process	S Hassan Adelyar Alex Norta	2016	[5]	Medio

Tabla 7.- Modelos y Metodologías de Seguridad

Título del articulo académico	**Propone un nuevo modelo incluyendo seguridad**	**Metodologías de seguridad identificadas**	**Metodología de seguridad analizada**
Towards a Secure Scrum Process for Agile Web Application Development	Si	• SSE-CMM System Security Engineering Capability Maturity Model • OWASP • Microsoft SDL • NIST 800-64 • SREP Security requirements engineering process • Cigital Touchpoint	SSE-CMM
ScrumS: a model for safe agile development	Si	Ninguna	No
Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency	Si	• VAHTI (guías de seguridad del gobierno de Finlandia)	VAHTI
Secure Scrum: Development of Secure Software	Si	• OWASP	No
A comparison of security assurance support of agile	No	• VAHTI (guías de seguridad del gobierno de	No

software development methods		Finlandia) • Microsoft SDL	
Secure Software Engineering for Agile Methodology: Preliminary Investigation	No	• SAMM Software Assurance Maturity Model • BSIMM Building Security in Maturity Model • SSDL Secure Software Development Lifecycle • SSE-CMM System Security Engineering Capability	No
Integrating the Extreme Programing Model with Secure Process for Requirement Selection	Si	Ninguna	No
Secure Paradigm For Web Application Development	Si	Ninguna	No
Towards a Secure Agile Software Development Process	No	Ninguna	No

Los autores en [9], proponen un modelo de Scrum seguro, agregando actividades de seguridad esenciales en proyectos de desarrollo de aplicaciones web y mapeando estas actividades contra el modelo de seguridad SSE-CMM. Este modelo, está diseñado para

aplicaciones web porque son un objetivo popular de ataques de seguridad por la facilidad de ataques remotos, varios tipos de vulnerabilidades del software y amplia superficie de ataques. En la Figura 9 se muestran las actividades de seguridad propuestas en este modelo.

Requerimientos

- Iniciar la planificación de la seguridad.
- Análisis de riesgo ágil - Dependencias externas.
- Análisis de los requerimientos de seguridad.
- Análisis de riesgo ágil – Identificar niveles de confianza.
- Análisis de riesgo ágil – Identificar activos.
- Análisis de riesgo ágil – Determinar puntos de Acceso.

Diseño

- Análisis de riesgo ágil – Identificar amenazas.
- Análisis de riesgo ágil – Priorización de las amenazas con DREAD.
- Análisis de riesgo ágil – Estrategias de mitigación.

Implementación

- Documento de controles de seguridad
    - Análisis de código estático.
    - Análisis de código dinámico.
    - Programación en pares.
    - Verificador de dependencias.

Verificación

- Análisis de riesgo ágil priorizado para definir los casos de prueba.
- Pruebas de penetración de pares.
- Pruebas de penetración.
    - Problemas detectados son priorizados con DREAD y controlados por la actividad Análisis de riesgo ágil – Estrategias de mitigación.
- Revisión de código.

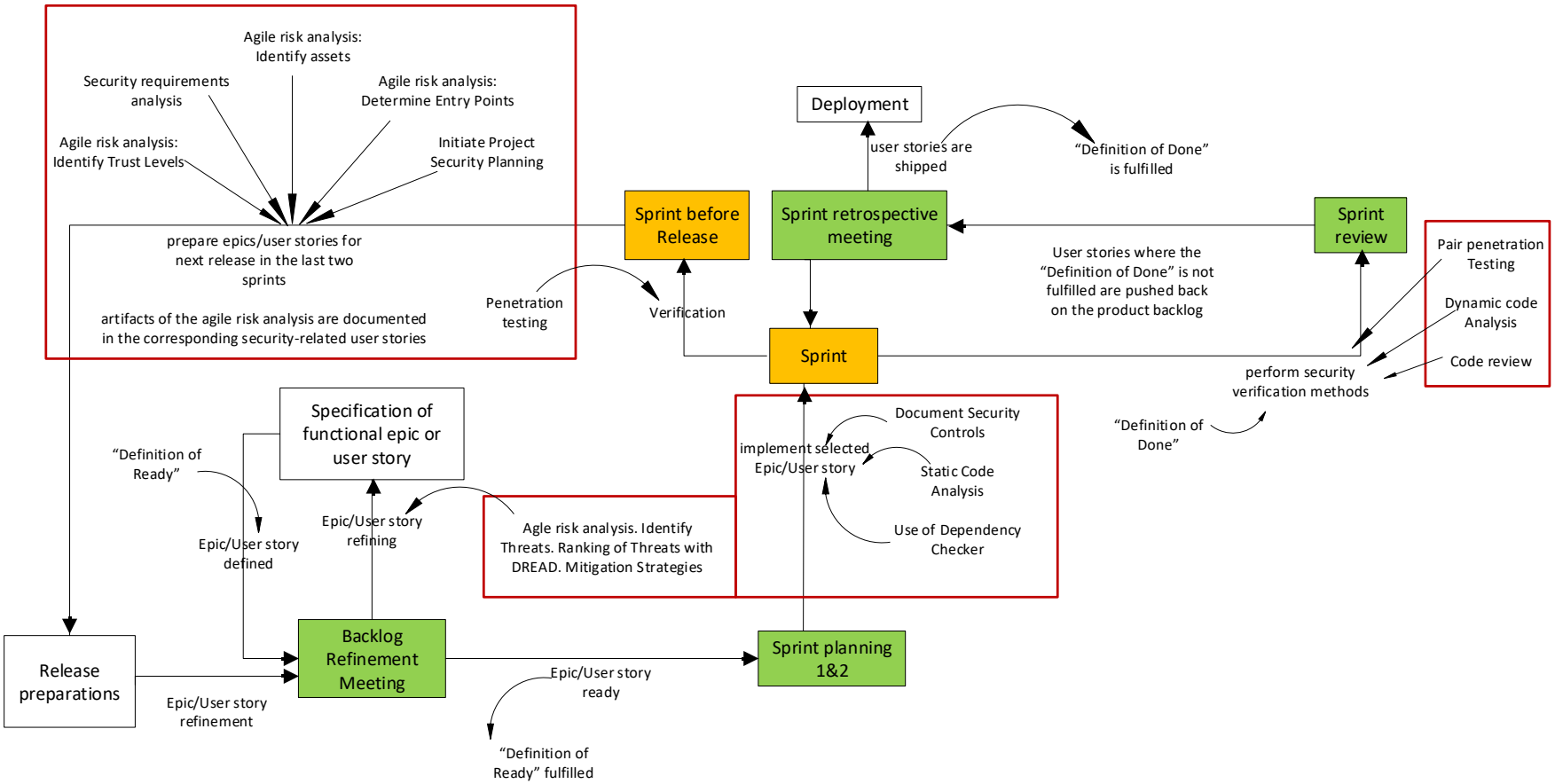

Figura 9.- Modelo de Scrum Seguro
[9] Fig. 2

Este documento [6] propone un modelo de Scrum seguro que agrega técnicas de seguridad maduras al ciclo de vida de Scrum. El proceso corre en paralelo al proceso de Scrum por lo cual no tiene impacto en el proceso original como se puede visualizar en la Figura 10.

Al inicio del proyecto se define las herramientas, ambientes de pruebas y producción, procesos y políticas de seguridad para el proyecto que es una de las actividades de un ciclo seguro. Además, utiliza el sprint cero que por definición es utilizado para configurar las tareas iniciales del desarrollo, para identificar las historias de usuario que necesitan seguridad.

Cada historia de usuario que sea agregada al sprint actual necesita ser evaluada para:

- Identificar el activo a proteger.
- Elaborar las historias de usuario relacionas con la seguridad del activo: identificar las amenazas al activo, por simplicidad se considera solo amenaza externa o interna.
- Análisis de vulnerabilidades: un árbol de ataques es diseñado en este proceso para identificar el camino del atacante a los activos. Asimismo, contribuye a identificar las vulnerabilidades del software y crear medidas de seguridad que pueden ser utilizadas en otras historias de usuario.
- Inventario de Riesgos: una vez identificados el activo, las amenazas, las fuentes de amenazas y las vulnerabilidades el siguiente paso es listar los riesgos dentro del sistema. Esto es construido para agregar controles de seguridad y de esta forma eliminar, mitigar o controlar el riesgo de seguridad en el sistema.

Los controles de seguridad deben ser agregados en cada historia de usuario a ser desarrolla en el sprint. En cada planificación del sprint es recomendado definir la especificación de completado para el product backlog ítem que tiene un control de seguridad.

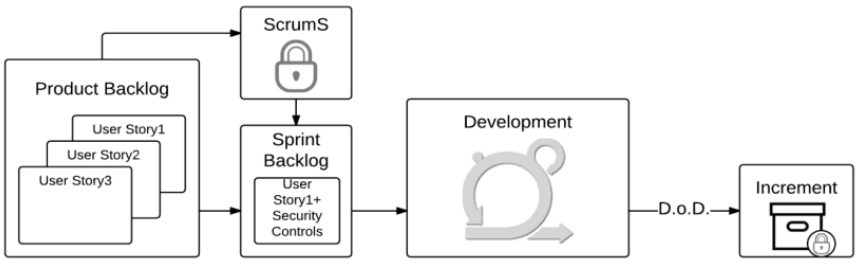

Figura 10.- Modelo de Scrum Seguro
[6] Fig. 1

El siguiente modelo propuesto por [28] es nombrado VAHTI-SCRUM. Este modelo incluye un nuevo rol para el desarrollador de seguridad, implementación enfocada en los artefactos de seguridad y anticipación y planeación de las tareas de seguridad.

- El rol del desarrollador de seguridad está enfocado en la seguridad del producto, crea o revisa la documentación requerida para aprobar las auditorias de seguridad.
- La implementación enfocada en los artefactos de seguridad genera documentos relacionados con la seguridad.
    - Certificados de entrenamiento de seguridad requeridos para el equipo del proyecto.
    - Documentación de la arquitectura, plan de gestión de riesgos, planes de pruebas y reportes de pruebas que son entregados al auditor de seguridad. Como parte del aseguramiento de la seguridad el auditor genera reportes con esta información.
- En la planeación de las tareas de seguridad, deberían estar presentes en el sprint y en las reuniones diarias de Scrum. Sin embargo, estas tareas de seguridad podrían o no estar presentes en un sprint, si existen demasiadas tareas de seguridad estas podrían tener su propio sprint.

En la Figura 11 se puede visualizar el modelo propuesto en donde las pruebas de seguridad, revisión y auditoria son parte del sprint backlog.

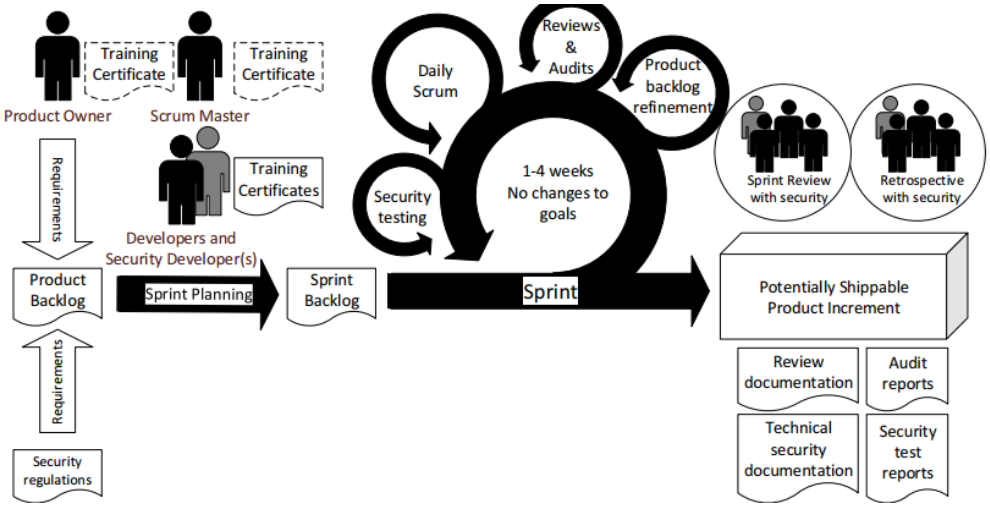

Figura 11.- Modelo de Scrum Seguro
[28] Fig. 2

El modelo propuesto en [31] propone la modificación de Scrum para integrar cuatro componentes identificación, implementación, verificación y definición de completado como se visualiza en la Figura 12.

Componente de identificación: Identifica historias de usuario relevantes para la seguridad y puede ser usado en varias etapas del proceso de Scrum.

- El equipo de desarrollo prioriza las historias de usuario utilizando el valor de pérdida que podría tener si existe un problema de seguridad.
- El equipo de Scrum y los interesados en el proyecto evalúan los casos de mal uso y los priorizan en base al riesgo.

Al final del proceso el equipo de Scrum y los interesados en el proyecto están conscientes de los riesgos de seguridad del product backlog ítem.

Los product backlog ítem (pbi) de seguridad pueden tener un marcado diferente para ser identificados y una descripción del problema de seguridad. Uno o varias pbi puede estar enlazados al mismo problema de seguridad por lo que se podrían agrupar.

Componente de implementación: Mantiene presente al equipo de Scrum de los problemas de seguridad durante el sprint.

- Los pbi de seguridad deben estar presentes en el sprint backlog para que el equipo de desarrollo tenga presente los problemas de seguridad. Cada uno de estos pbi pueden ser dividas en tareas de seguridad y ser implementados.

Componente de verificación: Asegura que los miembros del equipo puedan probar el software con un enfoque en seguridad.

Componente de definición de completado: Define cuando los problemas de seguridad están completados.

La verificación puede ser completada por el desarrollador al entregar su tarea o puede necesitar de una tarea extra de verificación para ser desarrollado durante el sprint por otro miembro del equipo.

Cada componente de Scrum seguro permite la inclusión de recursos externos para desarrollar estas tres funciones definidas:

- Mejorar el conocimiento: entrenamiento de seguridad al equipo de Scrum para ayudar a entender y especificar de mejor forma los requerimientos de seguridad del proyecto.
- Resolver desafíos: existen desafíos al momento de evaluar o implementar seguridad en el proyecto que no necesitan ser desarrollados ya que existen productos de terceros o especialistas que conocen el área.
- Proveer con un punto de vista externo: las formas de identificar o hacer pen testing puede ser desafiante además de necesitar profesionales que estén al tanto de como identificar vulnerabilidades en un sistema.

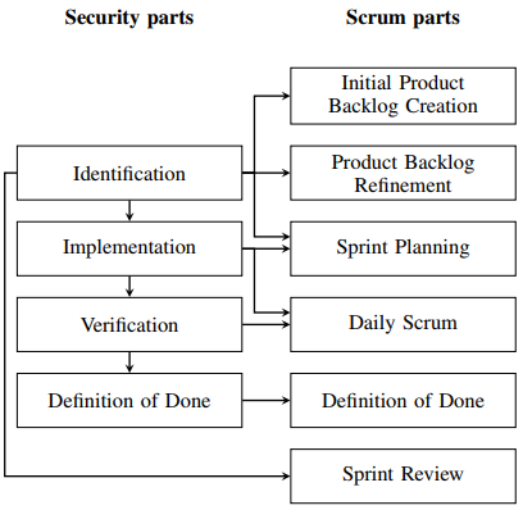

Figura 12.- Modelo de Scrum Seguro
[31] Fig. 1

El modelo propuesto en [34] agrega seguridad a una metodología ágil como Xtreme Programming. Identifica que el problema de seguridad en este modelo empieza desde que el desarrollador solicita al usuario que defina las historias de usuario.

Para la creación de historias de usuario se pide la autenticación y se valida el usuario para crear la especificación con el apoyo del desarrollador, al momento de validar esta historia de usuario se verifica el dueño de esta historia y si está de acuerdo con los requerimientos definidos además de completar con un análisis de riesgos y verificación del requerimiento como se indica en la Figura 13.

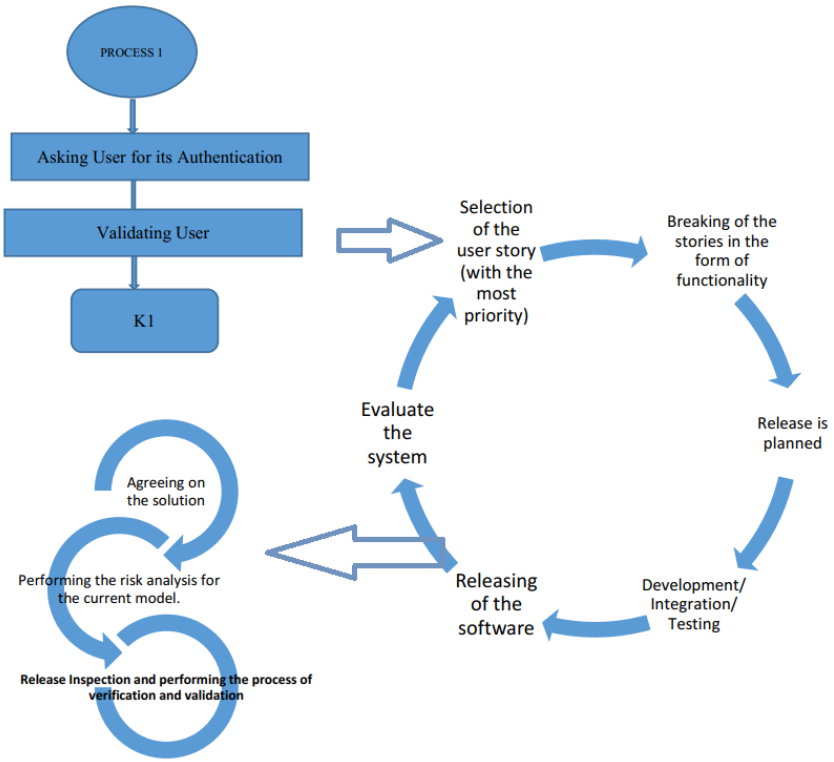

Figura 13.- Modelo de eXteme Programming seguro
[34] Fig. 4

Los autores en [35] proponen un modelo de seguridad para el desarrollo de aplicaciones web, en la Figura 14, se puede visualizar las tres etapas para el desarrollo: concepción, construcción y transición.

Concepción: es la fase inicial de proyecto en donde el diseño debe considerar los requerimientos de seguridad identificados, agrupados y priorizados.

Construcción: el proceso de construcción del software es iterativo e incremental en donde en cada iteración el equipo será encargado de tomar las historias de usuarios priorizadas y además los requerimientos de seguridad relacionados. La evaluación de riesgos de las historias de usuario identifica los riesgos de seguridad.

Transición: es la etapa antes de desplegar el software en el entorno de producción en donde se debe probar nuevamente los riesgos de seguridad identificados. Las pruebas pueden incluir puertos por defecto, contraseñas por defecto, pruebas de penetración.

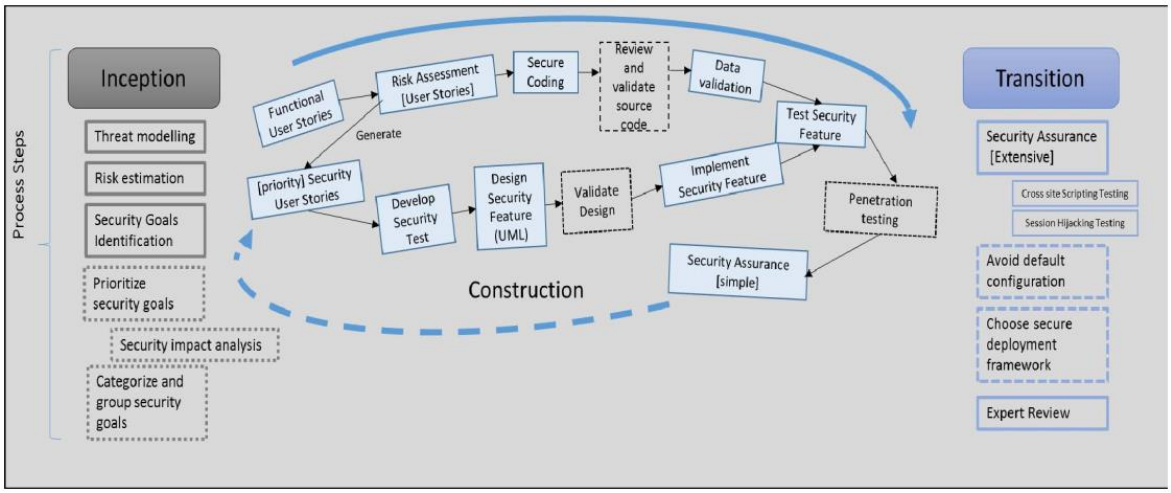

Figura 14.- Modelo de Desarrollo de Aplicaciones Web seguras
[35] Fig. 1 Modificado

En la Tabla 8 se evalúan los modelos para identificar la metodología, estándar de seguridad, roles y artefactos propuestos para agregar seguridad al proceso de desarrollo.

Tabla 8.- Análisis de los modelos en la literatura

Titulo	**Metodología**	**Utiliza un estándar de seguridad**	**Nuevos Artefactos**	**Nuevos Roles**
Towards a Secure Scrum Process for Agile Web Application Development	Scrum	Si	Si	No
ScrumS: a model for safe agile development	Scrum	No	Si	No
Case Study of Security Development in an Agile Environment: Building Identity	Scrum	Si	Si	Si

Management for a Government Agency				
Secure Scrum: Development of Secure Software with Scrum	Scrum	No	Si	No
Integrating the Extreme Programing Model with Secure Process for Requirement Selection	XP	No	Si	No
Secure Paradigm For Web Application Development	Iterativo e Incremental	No	N/A	N/A

3.4. Resultados de la revisión de la literatura

Los resultados encontrados en la literatura identifican y diseñan modelos para mejorar el proceso de desarrollo seguro en metodologías agiles, aunque la información descrita en cada uno de ellos no es muy clara y se enfocan en proponer un rol o agrupar eventos en procesos de seguridad más amplios. Se concluye que se utilizara los modelos propuestos de cada uno de estos modelos para generar un modelo que integre un nuevo rol, agrupar eventos de Scrum por pasos de seguridad y definir tareas identificadas en los modelos o en las mejores prácticas de seguridad.

4. METODOLOGÍA

4.1. Investigación – Acción

La metodología investigación – acción examina un problema, genera una solución y evalúa los resultados de la solución diseñada. Esta metodología es utilizada en proyectos en conjunto entre la universidad y la empresa privada por su flexibilidad para hacer ajustes durante el avance del estudio, consiguiendo resultados relevantes e inmediatos para las dos organizaciones. Además, esta metodología se caracteriza por ser iterativa e incremental que, en entornos de innovación, mejora continua, proactividad y ambiente participativo se ajustan a las condiciones de una empresa de desarrollo de software [36].

El manejo de ciclos y la flexibilidad para realizar ajustes durante el proceso de investigación y pruebas de la metodología, permite su uso en el presente proyecto facilitando la investigación, análisis y valoración del modelo propuesto.

Existen cinco componentes de la metodología que se van ajustando durante el desarrollo del proyecto hasta que se complete la solución al problema [36]:

Diagnóstico y detección

- Conocer el problema
- Identificar personas y procesos involucradas
- Identificar los resultados actuales
- Análisis y validación de los datos

Planificación para resolver el problema

- Crear un plan para resolver el problema

Ejecución del plan

- Aplicar el plan propuesto al problema

Evaluación del plan

- Recolectar datos medibles para evaluar el plan propuesto

Aprendizaje

- Identificar los ajustes necesarios y retroalimentar al ciclo

4.2. Caso de estudio: Empresa Privada

4.2.1. Contexto

La empresa privada en la que se evaluara el modelo es una sucursal en Ecuador de una compañía proveedora de software como servicio para el área de salud en clínicas de diálisis en los Estados Unidos. La compañía fue establecida en Ecuador en el año 2010.

Al momento cuenta con veinticinco personas distribuidas en las áreas de desarrollo, pruebas, operaciones y gestión administrativa. La Figura 15 indica el organigrama de la compañía.

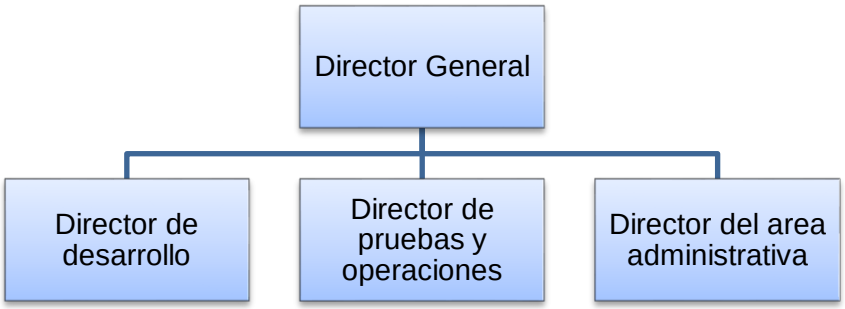

Figura 15.- Organigrama de la Compañía

La compañía desarrolla y mantiene aplicaciones de software con metodologías ágiles como Scrum y Kanban. Estas aplicaciones son desarrolladas utilizando tecnologías web, servicios y aplicaciones de escritorio con equipos multidisciplinarios y distribuidos en Ecuador y Estados Unidos.

Las aplicaciones desarrolladas almacenan información de pacientes, personal operativo y administrativo de las clínicas. Las aplicaciones deben cumplir con estándares para el manejo de información de pacientes y el intercambio electrónico de información clínica entre aplicaciones requerido por el gobierno de los Estados Unidos.

4.3. Diseño de la investigación

La Figura 16 indica el diseño del modelo de investigación - acción realizado para mejorar el proceso de desarrollo.

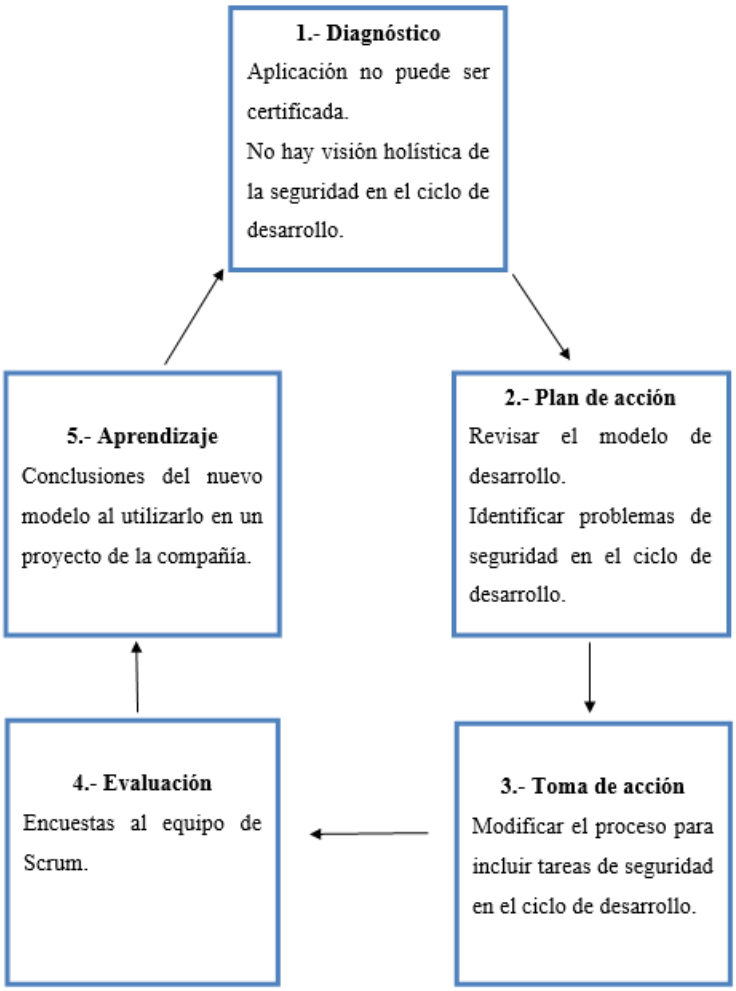

Figura 16.- Diseño de la investigación

Diagnóstico. – el problema fue identificado al momento de evaluar el software para ser certificado en el manejo de información médica. La certificación de registro electrónico de salud de los Estados Unidos, CEHRT [37] define estándares para el intercambio de información entre sistemas de salud y la confidencialidad de los datos de los pacientes. Además, la certificación requiere que toda aplicación sea construida con un modelo de seguridad durante el ciclo de desarrollo. Al diagnosticar el problema se detectó que los modelos de desarrollo utilizados por la compañía no contienen elementos de seguridad para reducir o controlar las siguientes vulnerabilidades:

1. OWASP Top 10 [38]
2. CWE/SANS Top 25 Most Dangerous Software Errors [39]

La Tabla 9 indica los requerimientos de seguridad de la compañía y el estándar de seguridad solicitado.

Tabla 9.- Matriz de requerimientos de la compañía

Tipo de Proyecto	Estándares	Requerimiento	Información

WEB	OWASP TOP 10 SANS TOP 25	• Incluir seguridad en el modelo de desarrollo • Código fuente • Servidores Web • Aplicaciones .Net • Transferencia de información entre aplicaciones	• Información de salud personal • Información de usuarios

Plan de acción. – se necesita modificar los modelos de desarrollo y mantenimiento para tener una visión holística de la seguridad en la construcción del software para cumplir los requisitos de la certificación.

Toma de acción. – se debe modificar el modelo de desarrollo que utiliza Scrum para agregar componentes de seguridad.

Evaluación. – el modelo tiene una lista de verificación para evaluar el proceso de seguridad durante el ciclo de desarrollo. El evento de retroalimentación de Scrum es utilizado para evaluar el proceso de desarrollo y de seguridad.

Aprendizaje. – el modelo será evaluado en un proyecto de la compañía.

5. IMPLEMENTACIÓN DEL DISEÑO DE INVESTIGACIÓN

5.1. Diagnóstico

El proceso de certificación para el almacenamiento y transferencia de información clínica de pacientes requiere que las aplicaciones de software sean desarrolladas con una visión holística de la seguridad en el proceso de desarrollo. El incluir los conceptos de seguridad en la fase de desarrollo de software le permite al producto desarrollado ser más confiable, resiliente y recuperable. Cada fase de desarrollo debe considerar y agregar estos conceptos de seguridad, la perdida de atención en una de las fases provocaría que todos los esfuerzos tomados en las fases anteriores o posteriores sean en vano. Después de analizar Scrum y Kanban metodologías utilizadas en el proceso de desarrollo y mantenimiento de software se identificó que estos procesos no incluyen ninguna característica o requerimiento de seguridad de forma explícita o implícita.

5.2. Plan de acción

Los procesos de desarrollo y mantenimiento de software necesitan agregar seguridad de forma explícita para poder ser medibles y evaluados. Es necesario identificar las necesidades de seguridad de cada etapa del ciclo de desarrollo y definir cuales tareas son necesarias en cada sprint, al inicio del proyecto, al final del proyecto y cada entrega del producto.

5.3. Toma de acción

Los tiempos para la evaluación del modelo, las oportunidades de reducir fallas, errores y defectos en proyectos nuevos y los tiempos de entrega del proyecto más flexible durante el proceso de desarrollo influyen en la decisión de modificar el modelo de desarrollo utilizado por la compañía.

El modelo de seguridad es paralelo al modelo de desarrollo y similar el proceso de calidad de software. Durante la etapa del ciclo de vida del software se ejecutan los eventos de seguridad propuestos para reducir las fallas de seguridad.

El modelo propuesto extiende los elementos identificados por Pohl [31] y utiliza el ciclo de Deming de la Figura 17 para definir los componentes de planificación, hacer, verificar y ajustar. El modelo agrupa los eventos de Scrum en componentes y define los principios de seguridad necesarios para cada componente. Además, propone la creación de un nuevo rol en el equipo de desarrollo que sea el experto de seguridad y encargado de coordinar todo el proceso de seguridad en Scrum.

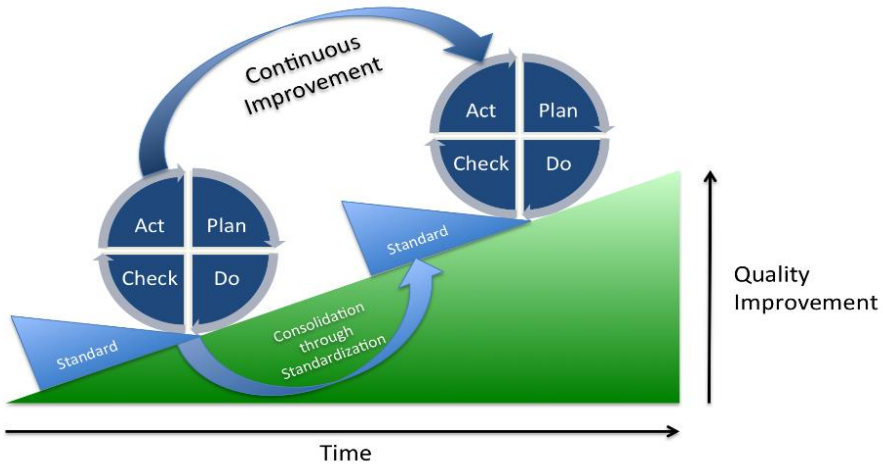

Figura 17.- Ciclo de mejora continua de Deming
[40]

Cada evento de Scrum tiene tareas de seguridad que deben ocurrir utilizando los modelos actualmente propuestos en la literatura científica 1.2.4, y las buenas prácticas de seguridad definidas en 1.5.1. Para la identificación de tareas de seguridad se propone el uso de etiquetas como lo sugiere Pinto [6] en cada PBI de esta forma la trazabilidad del requerimiento de seguridad durante el sprint está garantizada.

El modelo de seguridad utiliza el ciclo de Deming de mejora continua para identificar los componentes de seguridad en el proceso de Scrum. En la Figura 18 se identifica las cuatro etapas del ciclo de Deming con los eventos de Scrum. Los componentes del modelo de seguridad son planificación, identificación e implementación, verificación y ajuste [5].

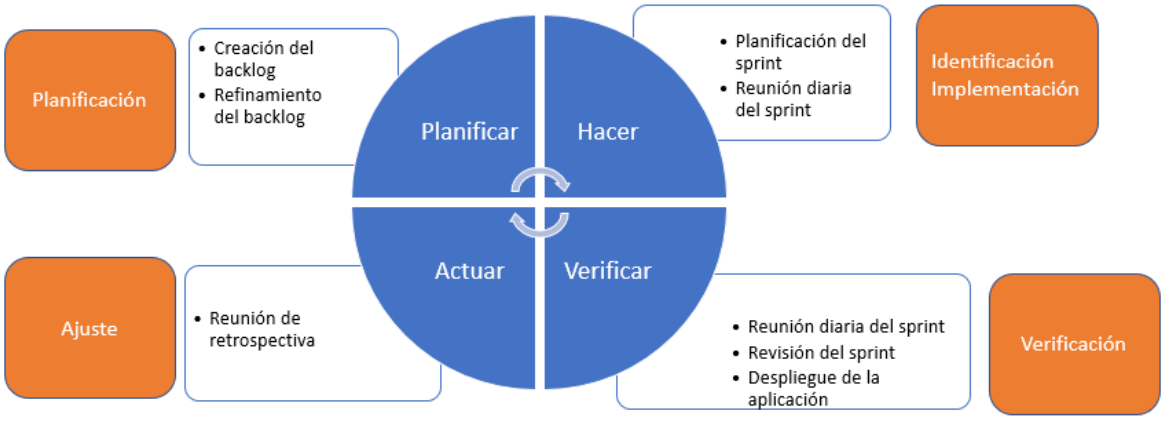

Figura 18.- Modelo propuesto de Scrum seguro

Planificación

El componente de planificación define los objetivos de la seguridad, código seguro y entrenamiento buscando siempre un equilibrio entre la funcionalidad, usabilidad y seguridad. Este componente debe ocurrir en el Sprint 0, que es un sprint que no necesita

entregar ningún producto al cliente, y las reuniones de creación y refinamiento del backlog.

Objetivos de seguridad. - Se debe identificar, priorizar y categorizar los objetivos de seguridad del proyecto estos definen la confidencialidad, integridad y disponibilidad de los datos y del sistema. Estos objetivos pueden actualizarse durante la implementación y verificación del producto.

Definición de código seguro. - Se define las herramientas de desarrollo a utilizar en el proyecto, algoritmos de encriptación, estándares de seguridad, y marcos de trabajo a ser utilizados durante el desarrollo. Las políticas de seguridad de la compañía proveen una guía para diseñar software seguro considerando la integridad, confiablidad y disponibilidad en el software

Entrenamiento. – Se define el tipo de entrenamiento de seguridad y negocio que requiere el equipo para el éxito del desarrollo del proyecto. El equipo de Scrum necesita entender las necesidades del negocio para codificar, diseñar e implementar el software. El equipo de desarrollo diseña e implementa el software así el equipo debe identificar las vulnerabilidades y amenazas que se pueden presentar durante la vida del producto de software.

Identificación e Implementación

En la etapa de identificación e implementación se realiza la gestión de riesgos y la implementación de las tareas de seguridad. En Scrum ocurre al inicio de cada sprint o durante el sprint.

La etapa de identificación ocurre durante la planificación del sprint, en esta etapa existen dos eventos que definen que hay que hacer y cómo hay que hacer el software. En esta reunión es necesario identificar los requerimientos del usuario que tienen condiciones de seguridad visibles u ocultas y realizar la gestión de riesgo del requerimiento asegurándose de definir los casos de prueba para validar el requerimiento de seguridad.

Los requerimientos de seguridad pueden ser extensos y no necesariamente se pueden concluir en un sprint por lo cual es necesario establecer estrategias de trazabilidad y despliegue continuo que no afecte la entrega del producto al final del sprint. Una estrategia para mantener la trazabilidad de los requerimientos de seguridad es utilizar una etiqueta en las tareas de seguridad y una de despliegue continuo es activar o desactivar funcionalidad.

Gestión de riesgos de seguridad del PBI. - Se identifica, evalúa, prioriza las amenazas y vulnerabilidades para mitigar, eliminar o aceptarlas. Para el modelo de amenazas se utiliza STRIDE (suplantación de identidad, manipulación, repudio, divulgación de

información, negación del servicio y elevación de privilegios) como el modelo de amenaza. Además, se utilizan DREAD (daños, reproducibilidad, aprovechamiento, usuarios afectados y descubrimiento) para la clasificación del riesgo.

Los casos de pruebas son un requisito para validar los requerimientos de seguridad en el proceso de pruebas y aseguramiento de calidad.

Las herramientas para utilizar en este proceso pueden ser: modelo de amenazas, casos de abuso, y evaluación de superficie de ataque.

Los requerimientos de seguridad y las formas de controlarlas son [21]:

- Confiabilidad: encriptación, hashing, esteganografía, marca de agua digital y enmascaramiento.
- Integridad: validación de datos de entrada, chequeo de redundancia cíclica y hashing.
- Disponibilidad: replicación, tolerancia a fallos, diseño escalable, tiempo objetivo de recuperación y tiempo máximo de inactividad.
- Autenticación: SSO (Identificación de una sola vez) y Factor de autenticación múltiple.
- Autorización: roles o actividad, modelado de subsistema confiable, suplantación, modelo de delegación y reducir la superficie de ataque.
- Responsabilidades: registro histórico de las acciones del usuario (Auditoria), registro de flujo del programa (Solucionar problemas). Los campos mínimos de auditoria son: usuario, acción, objeto modificado y fecha de modificación.
- Gestión de sesión: no vulnerable a ataques de fuerza bruta, predecible y destruir la sesión al salir de la aplicación.
- Gestión de errores y excepciones: todas las excepciones deben ser controladas, mensajes de error amigables, auditoria y monitoreo de las excepciones periódicamente.
- Gestión de parámetros de configuración: archivos de configuración encriptado, contraseñas no quemadas en el código, monitoreo de la inicialización y eliminación de variables globales.
- Operaciones: política de actualización de software en producción, claves de encriptación y sesión deben ser protegidas y diferentes al entorno de desarrollo, gestión de incidentes para detectar la causa raíz del problema, respaldos, políticas de mantenimiento y soporte.

- Despliegue: identificar el entorno de producción, la aplicación es desplegada en una zona desmilitarizada, puertos, protocolos, privilegios en el entorno de producción y arquitectura de producción.

En la etapa de implementación los requerimientos de seguridad van a ser implementados y la documentación técnica creada por el equipo de desarrollo. Además, mantener la trazabilidad de las tareas ayudara al equipo a tener presente de la tarea de seguridad durante el sprint.

Implementación de tareas de seguridad. - Se deben utilizar las mejores prácticas de desarrollo disponibles para mejorar la calidad del código y reducir riesgos de funciones inseguras.

- Programación en parejas
- Uso de estándares de código para el desarrollo de código seguro
    - Validación de los datos de entrada. - se debe definir donde hacer las validaciones, que validar y como validar.
    - Manejo de Errores. – mensaje de errores amigables al usuario y registro de las excepciones.
    - Manejo de sesión. – el manejo de sesión es compartido en el navegador.
    - Parámetros de configuración. – definir como compartir y mantener los parámetros de configuración entre aplicaciones y páginas.
    - Criptografía. – uso de criptografía simétrica y asimétrica.
    - Concurrencia. – manejo de hilos
- Desarrollo guiado por pruebas
- Programación orientada a objetos, patrones diseño, principios SOLID

Documentación técnica de la seguridad. - El desarrollador que implementa las tareas de seguridad también contribuye con documentación técnica sobre la implementación. Además, esta información es utilizada para las pruebas de seguridad.

Verificación

La etapa de verificación es la encargada de asegurarse que la seguridad ha sido implementada correctamente y para eso las tareas identificadas son: programación en parejas, revisión de código, documentación técnica de la seguridad, reportes de seguridad y la validación por un experto de seguridad. Esta etapa ocurre durante el sprint, en la reunión diaria y en la reunión de revisión del producto a entregar al cliente.

Revisión de código. - Para la revisión de código se utilizan los estándares de seguridad definidos en la planificación del proyecto. Además, se debe realizar análisis de código estático y dinámico para detectar posibles errores en el código. En el análisis dinámico se pueden utilizar pruebas de caja negra o caja blanca.

Documentación técnica de la seguridad. - El desarrollador de tareas de seguridad es responsable de crear la documentación técnica de la seguridad y diseño implementado. Esta documentación será utilizada para la validación del pbi y para agregar en el wiki del proyecto.

Reportes de seguridad. - La realización de reportes de auditoria de seguridad es un entregable que puede o no ser completado al final de un sprint. Este es dependiente de las tareas de seguridad identificadas durante el sprint.

Pruebas de seguridad y aseguramiento de la calidad

- Pruebas funcionales y no funcionales
- Pruebas de caga negra y caja blanca
- Documentación de errores, defectos y vulnerabilidades.
- Validación de la superficie de ataque
- Análisis de dependencias

Despliegue de la aplicación

Gestión de configuración del software. - Se realiza la validación de la configuración por defecto de la aplicación al ser instalada con los permisos mínimos para funcionar correctamente.

Validación por un experto de seguridad. - Un experto en seguridad ayudara a identificar posibles problemas del producto de software desarrollado durante el sprint.

- Pruebas de penetración
- Pruebas de fuzzing
- Escaneo de vulnerabilidades, contenido y privacidad
- Pruebas de validación de datos de entrada
- Pruebas de inyección de código
- Pruebas de no repudio
- Pruebas de control de falsificación de identidad
- Pruebas de manejo de errores y excepciones
- Pruebas de manejo de privilegios
- Pruebas de validación criptográfica

Plan de incidentes. - Se debe poder registrar los incidentes de seguridad en el sistema como ataques al sistema, auditoria de usuarios maliciosos, manejo de permisos y errores del sistema. El equipo necesita registrar esta información para poder dar seguimiento y mejorar o eliminar características del sistema que están causando el problema.

Ajuste

Durante la reunión de retroalimentación del proceso de Scrum es importante también identificar mejoras al proceso de seguridad. Evaluando las tareas realizadas, las necesidades de entrenamiento del equipo y las etapas del proceso seguro para mejorar y expandir el conocimiento en la compañía.

Rol de seguridad

El modelo propone la inclusión de un nuevo rol en el equipo de Scrum para gestionar la seguridad en el ciclo de desarrollo. El Security specialist es el encargado de coordinar el proceso de seguridad en un equipo de Scrum. El principal objetivo del Security specialist es entrenar y hacer responsable al equipo de desarrollo de la implementación de la seguridad durante el ciclo de desarrollo. Son responsabilidades del Security specialist:

- Asegurar que el proceso de seguridad se cumpla en el sprint.
- Mayor experiencia en temas de seguridad de software.
- Entrenar al equipo en temas de seguridad de software.
- Coordinar el proceso de seguridad durante el sprint.
- Mantener al proceso de seguridad dentro del proceso de mejora continua.
- Programación en pares con otros desarrolladores para transmitir el conocimiento al equipo y desarrollar al equipo en temas de seguridad.
- Definir las listas de verificación de cada etapa.

Síntesis de la propuesta del modelo

El modelo propuesto es detallado en la Figura 19. Además de proponer la agrupación de los eventos de Scrum en etapas de planificación, identificación, implementación y verificación, crea un nuevo rol encargado de verificar que las actividades de seguridad propuesta por el modelo se lleven a cabo.

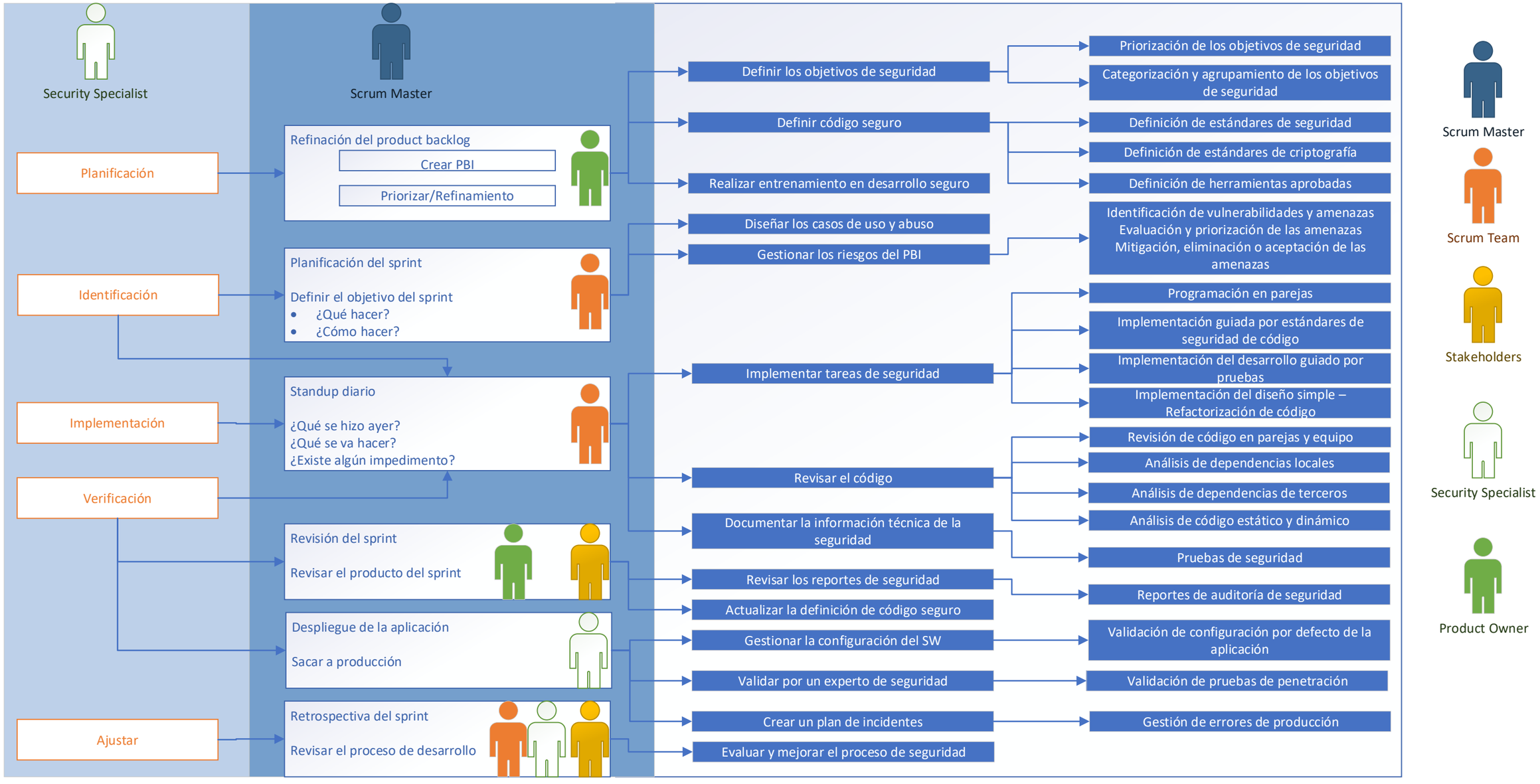

Figura 19.- Síntesis modelo Scrum seguro

5.4. Evaluación

El modelo de Scrum seguro será aplicado a un proyecto de la compañía y será evaluado durante las reuniones de retroalimentación de Scrum para detectar mejoras en el proceso y evaluar si el rol y modelo propuesto están contribuyendo a la inclusión de la seguridad en el proceso de desarrollo.

El proceso de seguridad definido requiere de listas de verificación para evaluar el modelo durante el ciclo de desarrollo. las Tabla 10, Tabla 11, Tabla 12, y Tabla 13 presentan las listas de verificación para las etapas de seguridad requeridas en este modelo. Cada una de estas tablas consulta si una tarea fue o no realizada y si hay comentarios con respecto a la tarea. Si una tare no aplica para el proyecto a verificar no es necesario responder las preguntas.

Tabla 10.- Lista de verificación de la etapa de planificación

Etapa de Planificación			
Descripción de la tarea	Si	No	Comentarios

Existe una definición de estándares de seguridad para:
• Autenticación
• Autorización
• Manejo de sesión
• Validación de entradas de datos
• Manejo de errores
• Registro de acciones en la aplicación
  ○ Eventos de seguridad (auditoría)
  ○ Eventos de la aplicación (diagnóstico del flujo de la aplicación, trazabilidad)
• Gestión de cambio: estrategia para identificar, controlar, garantizar y reportar los cambios en el software.			

Herramientas aprobadas para el proyecto que cumplan los requerimientos de seguridad • Lenguaje de programación • Programación Orientada a Objetos • Software de criptografía (autenticación, integridad, confidencialidad) • Base de datos y forma de mantener la información de esta (encriptada)			

○ Seguridad de los datos del sistema ○ Desarrollo con datos enmascarados • Herramientas de Integración continua ○ Entornos de desarrollo, pruebas ○ Seguridad de los repositorios de código			
Entrenamiento de seguridad para el equipo • Gestión de Riesgos ○ Objetivos del negocio ○ Identificar riesgos del negocio y técnicos ○ Priorizar los riesgos ○ Mitigación y validación de riesgos • OWASP Top 10 • SANS Top 25			

Tabla 11.- Lista de verificación de las etapas de identificación e implementación

Etapa de Identificación e Implementación			
Descripción de la tarea	Si	No	Comentarios
Se han identificado amenazas o vulnerabilidades en los requerimientos			
Existe casos de abuso o modelo de amenazas			
Se han mitigado, eliminado o reducido las amenazas			
Están definidos las mejores prácticas de programación • Programación en pares • Diseño guiado por pruebas • Diseño simple			
Existe integración continua para permitir la contribución continua de código en equipo • Existe un repositorio de código fuente • Construcción automática • Pruebas unitarias que se ejecuten en cada construcción del software • Manejo de versiones del software • Seguridad en el código fuente • Técnicas para modificación del comportamiento del sistema sin cambiar el código			

Tabla 12.- Lista de verificación en la etapa de verificación

Etapa de Verificación			
Descripción de la tarea	Si	No	Comentarios
Hubo revisión de código durante el sprint			
Se utiliza herramientas para el análisis estático de código			
Existe un análisis de las dependencias de terceros			
Existe un análisis de dependencias locales			
Los casos de pruebas fueron verificados			
Fueron ejecutadas pruebas de seguridad por un agente interno o externo Los resultados fueron entregados al gerente del proyecto			
La configuración por defecto para cada usuario es la mínima requerida			
Fueron creados y entregados los reportes de seguridad al gestor del proyecto, Scrum master por el experto de seguridad del proyecto.			

Tabla 13.- Lista de verificación de etapa de ajuste

Etapa de Ajuste			
Descripción de la tarea	Si	No	Comentarios
Existen comentarios sobre el proceso de seguridad			
Existen tareas a trabajar por el equipo para mejorar el proceso de seguridad			
Fueron identificados ítems de seguridad del proyecto			

5.5. Aprendizaje

Para la evaluación del modelo propuesto la compañía ha designado el siguiente proyecto de desarrollo: “Implementación de autenticación externa y doble factor de autenticación”. La aplicación actualmente implementa la tecnología de un solo inicio de sesión para ingresar a diferentes aplicaciones web, definiendo por lo tanto un proveedor de identidad y varios proveedores de servicio que puede utilizar una misma cuenta. El

proyecto concluirá cuando se haya implementado un segundo factor de autenticación por usuario, clínica. Además de soportar un proveedor de identidad externo.

El proyecto está dividido en varios sprint, pero se consideran solo los 3 primeros debido al tiempo de desarrollo de tesis y los recursos asignados al proyecto. Cada sprint está compuesto de 9 días laborables y un día para las reuniones de retroalimentación, presentación y planificación. El equipo de desarrollo estará compuesto de dos desarrolladores, dos aseguradores de calidad, un Scrum master y un administrador de proyecto.

Las encuestas al equipo de desarrollo serán utilizadas para valorar el modelo. El equipo es el responsable de la seguridad por lo tanto es responsabilidad del equipo mejorar o eliminar tareas agregadas al modelo de seguridad propuesto.

- **Sprint 0**

En el sprint 0 se llevaron a cabo tareas de diseño no de implementación. El objetivo de este sprint es definir los requerimientos del usuario considerando la seguridad como un requerimiento crítico del proyecto.

Evento de seguridad: Planificación

Objetivos de seguridad del proyecto

Autenticación: solo usuarios autorizados pueden ingresar al sistema y tienen la capacidad de ingresar a los otros proveedores de servicios.

Auditoría: Registrar las acciones del usuario desde que se autenticó. Si existe un error en la aplicación desplegar mensajes de error genérico si el usuario no está autenticado. Si el usuario está autenticado, desplegar un mensaje amigable para identificar el problema y se brinde mejor ayuda para soporte.

Autorización: el manejo de los usuarios está basado en permisos por actividades y el registro de eventos de seguridad. Los cambios de información en la URL deben verificar los permisos del usuario para analizar si el usuario tiene permiso o no.

Definición de estándares de seguridad

El proyecto es desarrollado sobre un entorno web. Los estándares de seguridad identificados para el proyecto son los mismos definidos para las aplicaciones de la compañía.

- Autenticación de tipo formulario
- Manejo de sesión para la información que es requerida durante el software mientras el usuario está autenticado
- Definición de información que puede ser manejada como parámetro en la URL

- Manejo de tiempo de sesión para forzar la autenticación después de que la página web no ha registrado ningún evento por parte del usuario
- Mensajes de error amigables para el cliente además de registrar el error real en la base de datos de errores.
- Las peticiones del navegador al servidor web debe ser verificadas para autorizar los permisos del usuario al sistema.
- Manejo de cache de servidor o cliente.
- El usuario no tiene permisos por defecto en la aplicación. Solo un usuario autorizado puede dar permisos a otros usuarios.
- Autorización de usuarios por direcciones IP.
- Validación de la información en los formularios implementada a nivel del cliente con JavaScript y en el servidor. En el servidor es requerido, en el cliente es opcional, pero sería ideal ya que, de esa manera se mejora la experiencia del usuario con el sistema.
- Actualización de librerías dependiendo de la frecuencia de publicación para mantener el sistema actualizado.

Definición de estándares de criptografía

Durante el inicio del proyecto se identificó la necesidad de generar un código numérico aleatorio no secuencial que el usuario lo utilizaría para la autenticación.

Definición de herramientas aprobadas

Herramientas aprobadas por la compañía para implementar

- Desarrollo web
    - Desarrollo en el lado del cliente, servidor, base de datos
- Integración continua
- Herramientas de análisis de código estático y dinámico
- Herramientas de terceros comprados por la compañía o software libre aprobado por la industria previa autorización con el experto de seguridad, arquitecto de software, Scrum master, gestor del proyecto y CISO de la compañía.

La Tabla 14 resume las tareas ejecutadas en el Sprint 0. Identificando las tareas realizadas y si fue definida como un ítem de seguridad.

Tabla 14.- Tareas del Sprint 0

Tareas	¿Es un ítem de seguridad?
Se crearon los mockups y la definición de los requerimientos del proyecto	NO
Se definió el estándar de encriptación para	SI

generar el código que se utilizará como factor de autentificación	
Actualización del proceso de ingreso al sistema para la creación de la cookie de autenticación solo después del ingreso del código de autenticación	SI
Transferencia de datos entre páginas antes de la creación de la cookie de autenticación.	SI
Validación en dos niveles en el cliente y en el servidor. En el cliente por usabilidad en el servidor por seguridad	SI
Despliegue de errores genéricos cuando el usuario no está autenticado en el sistema.	SI

La Tabla 15 verifica si el modelo de seguridad fue aplicado en la etapa de planificación. Además de presentar un estado de la seguridad en el proyecto.

Tabla 15.- Lista de verificación de la etapa de planificación Sprint 0

Etapa de Planificación			
Descripción de la tarea	Si	No	Comentarios

Existe una definición de estándares de seguridad para:
• Autenticación
• Autorización
• Manejo de sesión
• Validación de entradas de datos
• Manejo de errores
• Registro de acciones en la aplicación
  ○ Eventos de seguridad (auditoría)
  ○ Eventos de la aplicación (diagnóstico del flujo de la aplicación, trazabilidad)
• Gestión de cambio: estrategia para identificar, controlar, garantizar y reportar los cambios en el software.
	
	X
	X
	X
	
	X
	X
	
	X		

Herramientas aprobadas para el proyecto que cumplan los			

requerimientos de seguridad
• Lenguaje de programación
• Programación Orientada a Objetos
• Software de criptografía (autenticación, integridad, confidencialidad)
• Base de datos y forma de mantener la información de esta (encriptada)
  ○ Seguridad de los datos del sistema
  ○ Desarrollo con datos enmascarados
• Herramientas de Integración continua
  ○ Entornos de desarrollo, pruebas
  ○ Seguridad de los repositorios de código
	X
	X
	X
	X
	X		

Entrenamiento de seguridad para el equipo
• Gestión de Riesgos
  ○ Objetivos del negocio
  ○ Identificar riesgos del negocio y técnicos
  ○ Priorizar los riesgos
  ○ Mitigación y validación de riesgos
• OWASP Top 10
• SANS Top 25
	X		

- **Sprint 1**

Durante este sprint se llevó a cabo la implementación de la página de administración de doble factor de autenticación basado en una preferencia. Además de la modificación de la página de inicio de sesión para verificar el factor de autenticación como segundo paso dentro del proceso de autenticación del sistema.

Evento de seguridad: Identificación e implementación

Identificación

Se realizó la gestión de riesgos del requerimiento de los ítems marcados como seguros utilizando los casos de abuso para identificar amenazas y vulnerabilidades:

- Los parámetros de la url no pueden ser utilizados para transferir información desde la página de inicio de sesión.
- La cookie de autenticación debe ser creada solo después de verificar el código de autenticación.

- El código de autenticación no debe ser secuencial, debe tener un tiempo de validez.
- El código de autenticación debe ser amigable con el usuario, 6 a 7 dígitos.
- Si el código de autenticación no llega al canal de comunicación, existe un riesgo de disponibilidad del servicio.
- El código de autenticación debe tener un límite máximo de reenvío.

Implementación

El desarrollador pudo implementar el código considerando la especificación y los casos de abuso definidos. Los casos de prueba son validados por el desarrollador y el equipo de pruebas.

Evento de seguridad: Verificación

Revisión de código para:

- Verificar los estándares de código.
- Uso de las librerías para la implementación de páginas con arquitectura orientada a servicios, manejo de errores y controles HTML.
- Creación de ticket de autenticación solo después de verificar el factor de autenticación.
- Transmisión de información entre páginas sin utilizar los parámetros de la url o campos ocultos.
- Uso correcto de la librería para generación del código de autenticación.

Análisis de código estático con sonarqube

- Identificar la complejidad de código
- Verificar las dependencias de librerías de terceros
- Verificar el reporte de seguridad

Actualización del wiki del producto

Evento de seguridad: Ajustar

Revisión de los reportes de seguridad de sonarqube

Los ítems de seguridad pueden ser actualizados durante el sprint para considerar casos de prueba no identificados durante el diseño.

La Tabla 16 indica las tareas ejecutadas en el sprint 1. Identificando las tareas realizadas y si fue definida como un ítem de seguridad.

Tabla 16.- Tareas del Sprint 1

Tareas	¿Es un ítem de seguridad?
Creación de una nueva página en el sistema de administración para habilitar el	NO

factor de autenticación bajo demanda	
Creación de una página que solicite el factor de autenticación	SI
Diseño del código de autenticación	SI
Generar el código de autenticación y envío del código por el canal primario de comunicación del usuario	SI
Actualizar la página para permitir el reenvió del código de autenticación por un máximo de 3 intentos	SI

La lista de verificación para la etapa de planificación no fue realizada en el sprint1 ya que fue realizado en el sprint 0 y no se presentaron requerimientos o cambios en las especificaciones del modelo que provoquen un cambio en la planificación.

La Tabla 17 realiza la verificación de las etapas de identificación e implementación del modelo propuesto para el Sprint 1.

Tabla 17.- Lista de verificación etapa identificación e implementación Sprint 1

Etapa de Identificación e Implementación			
Descripción de la tarea	Si	No	Comentarios
Se han identificado amenazas o vulnerabilidades en los requerimientos	X		
Existe casos de abuso o modelo de amenazas		X	Casos de abuso que genero la creación de los casos de pruebas
Se han mitigado, eliminado o reducido las amenazas	X		
Están definidos las mejores prácticas de programación • Programación en pares • Diseño guiado por pruebas • Diseño simple	X		
Existe integración continua para permitir la contribución continua de código en equipo • Existe un repositorio de código fuente • Construcción automática • Pruebas unitarias que se ejecuten en cada	X		

construcción del software • Manejo de versiones del software • Seguridad en el código fuente • Técnicas para modificación el comportamiento del sistema sin cambiar el código		X	El producto del sprint 1 no fue desplegado en producción.

La Tabla 18 indica el estado de la seguridad en la etapa de verificación del Sprint 1.

Tabla 18.- Lista de verificación en la etapa de verificación Sprint 1

Etapa de Verificación			
Descripción de la tarea	Si	No	Comentarios
Hubo revisión de código durante el sprint	X		
Se utiliza herramientas para el análisis estático de código		X	
Existe un análisis de las dependencias de terceros		X	
Existe un análisis de dependencias locales		X	
Los casos de pruebas fueron verificados	X		
Fueron ejecutadas pruebas de seguridad por un agente interno o externo Los resultados fueron entregados al gerente del proyecto		X X	
La configuración por defecto para cada usuario es la mínima requerida	n/a		
Fueron creados y entregados los reportes de seguridad al gestor del proyecto, Scrum master por el experto de seguridad del proyecto		X	

La Tabla 19 identifica y señala si el proceso de ajuste fue analizado en el Sprint 1.

Tabla 19.- Lista de verificación de etapa de ajuste Sprint 1

Etapa de Ajuste			
Descripción de la tarea	Si	No	Comentarios
Existen comentarios sobre el proceso de seguridad	X		¿Quién es el encargado de definir que el proyecto necesita un Security specialist? ¿La etapa de planificación puede ser modificada durante el

			proyecto? ¿El Security specialist es el responsable de la seguridad del producto?
Existen tareas a trabajar por el equipo para mejorar el proceso de seguridad	X		Riesgo de seguridad en aplicaciones web.
Fueron identificados ítems de seguridad del proyecto	X		

- **Sprint 2**

En el sprint 2, el proyecto ha sido modificado debido a las necesidades del cliente para integrar la aplicación actual con un proveedor de identidad externo a la compañía. Esta modificación incluye el consumo de API REST y el cambio de la autenticación por formulario definida en el proyecto para utilizar OWIN y así permitir un proveedor de identidad no acoplado a la autorización.

Se mantienen los objetivos de seguridad del proyecto para la autenticación, autorización y registro de auditoria modificando:

- Autenticación: las tareas de autenticación vienen definidas por el proveedor de identidad externo como políticas de seguridad, tipo de encriptación de la contraseña, uso de sesión o tokens para la transferencia de información de autenticación y el tiempo de validez. Para desacoplar la autenticación de la autorización y soportar distintos proveedores de identidad es necesario cambiar de tecnología de autenticación en la aplicación.
- Manejo de sesión de autenticación por parte del proveedor de identidad y del proveedor del servicio.
- Analizar si las políticas de seguridad del proveedor cumplen con las mínimas requeridas por la compañía y si son compatibles.

Definición de herramientas aprobadas

- Los estándares de criptografía para la encriptación de contraseña, factor de autenticación y transferencia de datos son definidos por el proveedor de identidad.

Definición de herramientas aprobadas

- Empresa proveedora de identidad.
- API REST para el consumo por el proveedor del servicio

Evento de seguridad: Identificación e implementación

Identificación

Se realizó la gestión de riesgos del requerimiento de los ítems marcados como seguros utilizando los casos de abuso para identificar amenazas y vulnerabilidades:

- Consumo del token de autenticación provisto por el proveedor de identidad.
- Modificación de la autenticación del proveedor de servicios para manejar otra tecnología que desacople la autenticación con la autorización.
- Lugares de almacenamiento de las claves de autenticación para el consumo del API REST.
- Definición de los campos del usuario a compartir con el proveedor de identidad.
- Sincronización de información del proveedor de identidad con el proveedor de servicio, por diseño solo cuatro campos. Nombre de usuario, apellido, nombre y correo.
- Despliegue de errores del proveedor de identidad en la aplicación.

Implementación

El desarrollador pudo implementar el código considerando la especificación y los casos de abuso definidos. Los casos de prueba son validados por el desarrollador y el equipo de pruebas.

Evento de seguridad: Verificación

Revisión de código para

- Verificar los estándares de código.
- Uso de las librerías para la implementación de páginas con arquitectura orientada a servicios, manejo de errores y controles HTML.
- Uso correcto de la librería de errores para despliegue de mensajes de errores por parte del proveedor de identidad.

Análisis de código estático con sonarqube

- Identificar la complejidad de código
- Verificar las dependencias de librerías terceros
- Verificar el reporte de seguridad

Actualización del wiki del producto

Evento de seguridad: Ajustar

Se necesita reducir la complejidad de código y detectar posibles vulnerabilidades después de utilizar la herramienta de análisis de código estático.

La solución tiene muchas alertas al construir la aplicación, se necesita reducir el número de alertas que despliega el compilador.

Las pruebas unitarias de una API ayudan a verificar el manejo de errores.

Conflictos al actualizar las páginas de manejo de perfil porque actualizan la información del proveedor del servicio y del proveedor de identidad. Identificando un problema de sincronización de información.

La Tabla 20 indica las tareas ejecutadas en el Sprint 2. Identificando las tareas realizadas y si fue definida como un ítem de seguridad.

Tabla 20.- Tareas del Sprint 2

Tareas	¿Es un ítem de seguridad?
Consumo del API REST del nuevo proveedor de identidad.	No
Análisis de las páginas que manejan autenticación en la aplicación para consumir el API.	No
Análisis para la migración de información de autenticación del sistema al proveedor de identidad.	Si
Modificación de la aplicación para implementar OWIN y soportar diferentes proveedores de identidad y separa la autenticación de la autorización.	Si
Sincronización de información entre las dos aplicaciones.	Si
Despliegue de mensajes de error del proveedor de servicio e identidad.	Si
Actualización del wiki del proyecto para agregar la funcionalidad de proveedor de identidad externo	Si
Verificación de los archivos de logs creados para depurar el flujo del código además de los errores que se registren correctamente. La aplicación desplegara un error amigable en caso de errores.	Si
Pruebas de carga sobre el sistema. Para llevar el registro de pruebas antes y después de la implementación de esta	No

nueva funcionalidad	

La Tabla 21 muestra la lista de verificación de la etapa de planificación del Sprint 2. El proyecto fue modificado por lo cual la etapa de planificación fue requerida.

Tabla 21.- Lista de verificación de la etapa de planificación Sprint 2

Etapa de Planificación			
Descripción de la tarea	Si	No	Comentarios

Existe una definición de estándares de seguridad para:
• Autenticación
• Autorización
• Manejo de sesión
• Validación de entradas de datos
• Manejo de errores
• Registro de acciones en la aplicación
  o Eventos de seguridad (auditoría)
  o Eventos de la aplicación (diagnóstico del flujo de la aplicación, trazabilidad)
• Gestión de cambio: estrategia para identificar, controlar, garantizar y reportar los cambios en el software.
	X
	X
	X
	X
	X
	X
	X
	X
			Proveedor externo de identidad.
			Desacoplar la autenticación de la autenticación.
			Manejo de errores aplicación y proveedor de identidad.
			Reportes de autenticación modificados.
			Actualizaciones del API comunicación al proveedor del servicio.

Herramientas aprobadas para el proyecto que cumplan los requerimientos de seguridad
• Lenguaje de programación
• Programación Orientada a Objetos
• Software de criptografía (autenticación, integridad, confidencialidad)
• Base de datos y forma de mantener la información de esta (encriptada)
  o Seguridad de los datos del sistema
  o Desarrollo con datos enmascarados
	X
	X
	X
	X
	X		

• Herramientas de Integración continua
  ○ Entornos de desarrollo, pruebas
  ○ Seguridad de los repositorios de código			

Entrenamiento de seguridad para el equipo
• Gestión de Riesgos
  ○ Objetivos del negocio.
  ○ Identificar riesgos del negocio y técnicos.
  ○ Priorizar los riesgos
  ○ Mitigación y validación de riesgos.
• OWASP Top 10
• SANS Top 25
		X
		
		X
		X
			Evaluación del proveedor de identidad.
			
			El manejo de la autenticación pasa a ser responsabilidad del proveedor de identidad.

La Tabla 22 muestra la lista de verificación de las etapas de identificación e implementación del Sprint 2.

Tabla 22.- Lista de verificación etapa identificación e implementación Sprint 2

Etapa de Identificación e Implementación			
Descripción de la tarea	Si	No	Comentarios
Se han identificado amenazas o vulnerabilidades en los requerimientos	X		Comunicación aplicación, proveedor de identidad. Actualización de la autenticación en el sistema. Autorización afectada con la modificación de la autenticación.
Existe casos de abuso o modelo de amenazas	X		Casos de abuso que llevan a la creación de casos de pruebas.
Se han mitigado, eliminado o reducido las amenazas	X		.

Están definidos las mejores prácticas de programación • Programación en pares • Diseño guiado por pruebas • Diseño simple	 X X X		Uso de estándares de consumo de API. Revisión de documentación mejores prácticas para proveedores de identidad externos SAML, Open ID Connect.
Existe integración continua para permitir la contribución continua de código en equipo • Existe un repositorio de código fuente • Construcción automática • Pruebas unitarias que se ejecuten en cada construcción del software • Manejo de versiones del software • Seguridad en el código fuente • Técnicas para modificación el comportamiento del sistema sin cambiar el código	 X X X X X	 X	 .

La Tabla 23 muestra la lista de verificación para la etapa de verificación del modelo del Sprint 2.

Tabla 23.- Lista de verificación en la etapa de verificación Sprint 2

Etapa de Verificación			
Descripción de la tarea	Si	No	Comentarios
Hubo revisión de código durante el sprint	X		
Se utiliza herramientas para el análisis estático de código	X		
Existe un análisis de las dependencias de terceros		X	
Existe un análisis de dependencias locales	X		
Los casos de pruebas fueron verificados	X		Se incluye pruebas de regresión para verificar la autorización.

			Pruebas de los flujos de autenticación.
Fueron ejecutadas pruebas de seguridad por un agente interno o externo Los resultados fueron entregados al gerente del proyecto		X X	
La configuración por defecto para cada usuario es la mínima requerida	X		
Fueron creados y entregados los reportes de seguridad al gestor del proyecto, Scrum master por el experto de seguridad del proyecto.		X	

La Tabla 24 muestra la lista de verificación de la etapa de ajuste del Sprint 2.

Tabla 24.- Lista de verificación de etapa de ajuste Sprint 2

Etapa de Ajuste			
Descripción de la tarea	Si	No	Comentarios
Existen comentarios sobre el proceso de seguridad	X		¿Cuál es el plan para implementar las herramientas de análisis estático y dependencias? ¿Cómo evaluar el proceso de educación para el equipo en temas de seguridad? ¿Cuál es el conocimiento que debe tener un desarrollador para ser un Security specialist?
Existen tareas a trabajar por el equipo para mejorar el proceso de seguridad	X		Plan de entrenamiento en temas de seguridad y uso de herramientas para el análisis de código.
Fueron identificados ítems de seguridad del proyecto	X		Las políticas de seguridad del proveedor de identidad deben cumplir o ser mejores que las requeridas por la certificación

			y las políticas de seguridad de la compañía

Resultados

Para la evaluación del proceso de desarrollo seguro propuesto en la compañía se realizó una encuesta al equipo que trabajo en el proyecto de doble factor de autenticación e implementación de un proveedor de identidad externo.

Las preguntas fueron orientadas a identificar elementos del modelo de Scrum seguro propuesto que no están claros para el equipo. Se realizo al equipo de Scrum (desarrolladores, probadores, aseguradores de calidad, Scrum master y el Product Owner). Todos ellos se vieron afectados por el proceso en el cual la seguridad se incluyó en el ciclo de vida del software.

La siguiente encuesta fue realizada:

1. ¿Cuál es el rol del Security specialist durante el sprint?
   1.1. Responsable de la seguridad del producto desarrollado.
   1.2. Soporte al equipo para reducir riesgos de seguridad.
   1.3. Coordinador de las tareas de seguridad.
   1.4. Entrenador del equipo en temas de seguridad.
   1.5. Revisor de código desarrollado.
   1.6. Encargado de realizar las tareas de seguridad.
2. Revisando el modelo de seguridad propuesto, ¿Qué etapa del ciclo de desarrollo es las más compleja de realizar en el modelo de Scrum seguro?
   2.1. Requerimientos
   2.2. Diseño
   2.3. Implementación
   2.4. Pruebas
   2.5. Despliegue
3. ¿Qué pilar de seguridad es el más fácil de identificar durante el análisis de riesgo de seguridad?
   3.1. Confidencialidad
   3.2. Integridad
   3.3. Disponibilidad
4. Durante la identificación y análisis de riesgos de seguridad de un requerimiento del usuario, ¿utiliza las siguientes herramientas?
   4.1. Casos de uso

4.2. Casos de abuso

4.3. Modelo de amenazas con STRIDE (suplantación de identidad, manipulación, repudio, divulgación de información, negación del servicio y elevación de privilegios)

4.4. Clasificación de riesgos con DREAD (daños, reproducibilidad, aprovechamiento, usuarios afectados y descubrimiento)

5. ¿Cuál de los riesgos de seguridad en aplicaciones web identificados por la industria es utilizado para reducir riesgos de seguridad?

   5.1. OWASP top 10

   5.2. SANS Top 25

6. ¿Qué herramientas de análisis de código estático es utilizado en la compañía?

7. Durante la etapa de codificación de aplicaciones, ¿Cuál de los siguientes pasos son revisados y realizados en su día a día?

   7.1. Creación de pruebas unitarias

   7.2. Revisión de los logs de información y de errores

   7.3. Mejores prácticas de programación

   7.4. Estándares de programación

   7.5. Políticas de seguridad de la compañía

8. Cada etapa del ciclo de desarrollo requiere de conocimiento en el área para aplicar las tareas de seguridad. ¿Un miembro de Scrum debe ser experto en todas las áreas del ciclo de desarrollo?

A continuación, se registran las respuestas del equipo de Scrum.

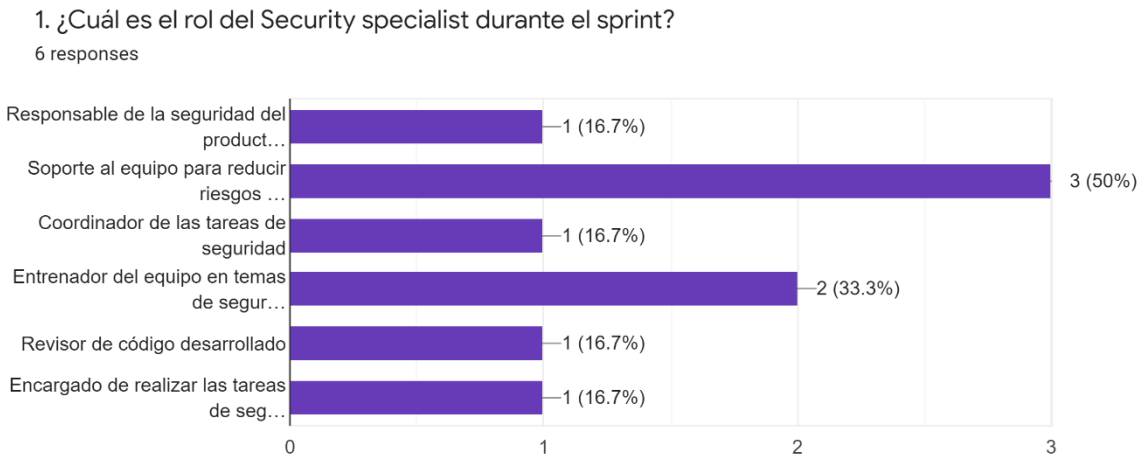

Figura 20.- Respuestas de la pregunta 1

2. Revisando el modelo de seguridad propuesto, ¿Qué etapa del ciclo de desarrollo es las más compleja de realizar en el modelo de Scrum seguro?

6 responses

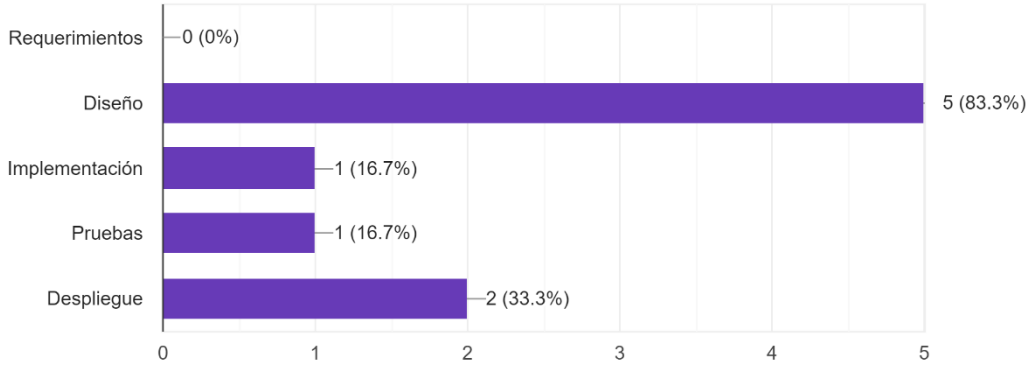

Figura 21.- Respuestas de la pregunta 2

3. ¿Qué pilar de seguridad es el más fácil de identificar durante el análisis de riesgo de seguridad?

6 responses

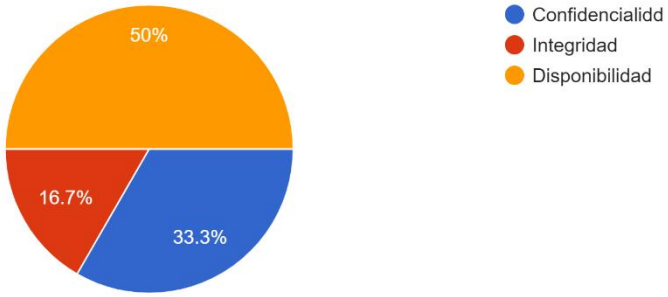

Figura 22.- Respuestas de la pregunta 3

4. Durante la identificación y análisis de riesgos de seguridad de un requerimiento del usuario, ¿utiliza las siguientes herramientas?

5 responses

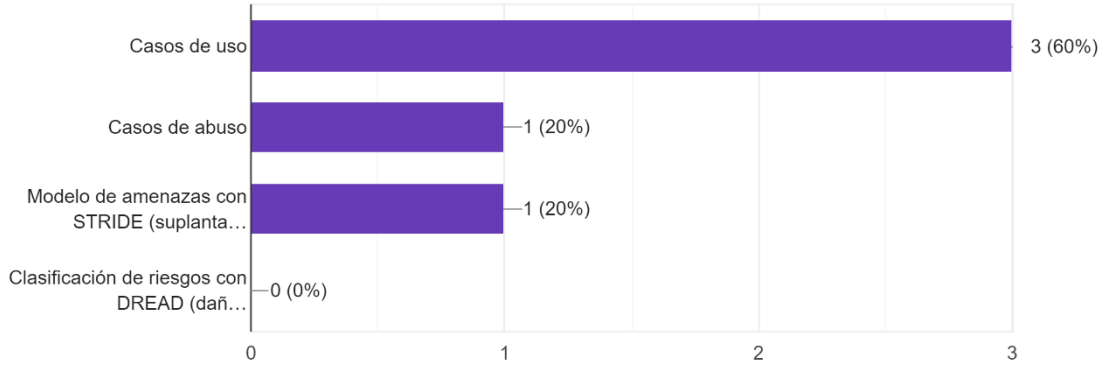

Figura 23.- Respuestas de la pregunta 4

5. ¿Cuál de los riesgos de seguridad en aplicaciones web identificados por la industria es utilizado para reducir riesgos de seguridad?

5 responses

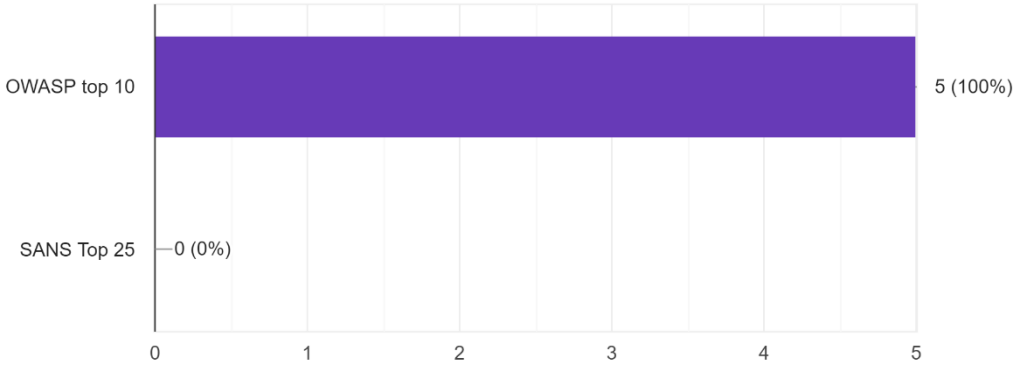

Figura 24.- Respuestas de la pregunta 5

6. ¿Qué herramientas de análisis de código estático es utilizado en la compañía?

6 responses

Resharper

SonarQube

Unit tests

Sonarqube

SonarQube en prototipo

Figura 25.- Respuestas de la pregunta 6

7. Durante la etapa de codificación de aplicaciones, ¿Cuál de los siguientes pasos son revisados y realizados en su día a día?

6 responses

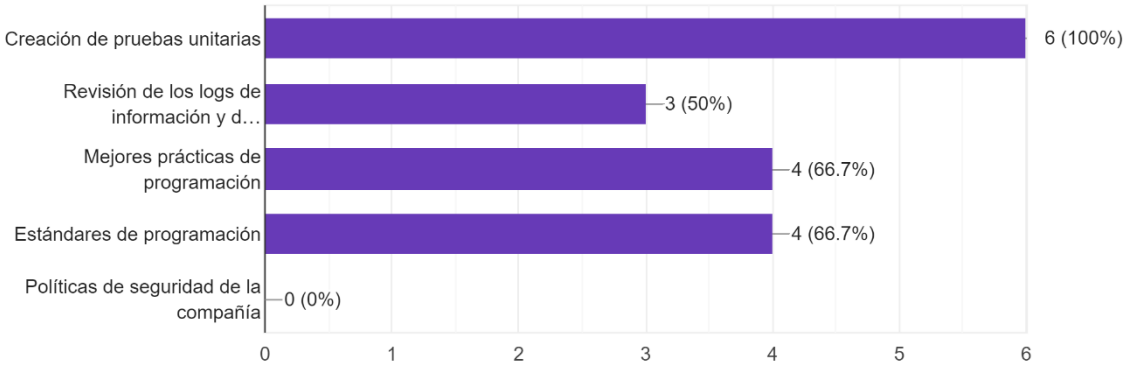

Figura 26.- Respuestas de la pregunta 7

8. Cada etapa del ciclo de desarrollo requiere de conocimiento en el área para aplicar las tareas de seguridad. ¿Un miembro de Scrum debe ser experto en todas las áreas del ciclo de desarrollo?

6 responses

No

Si, de esa forma el equipo puede apoyarse en esa persona en caso de dudas en las tareas

El experto en seguridad puede transmitir su conocimiento al equipo y contar con ellos para implementar la seguridad en el sistema. No se requiere que todos sepan de seguridad o que el experto en seguridad sepa de todo el desarrollo, con conocer la parte general y ser parte del equipo de desarrollo basta.

No, porque el ciclo de desarrollo tiene desarrolladores en SQL, Backend, Frontend, IT Deploy. Creo que conocer sobre seguridad en cada área en que es experto cada miembro del equipo ayudaria

Figura 27.- Respuestas de la pregunta 8

6. DISCUSIÓN

En el presente estudio se modificó el modelo de Scrum para agregar componentes de seguridad en cada uno de los eventos. La discusión está enfocada en los resultados de la encuesta, las etapas de seguridad identificadas y el estado del arte.

La encuesta realizada al equipo identifica la necesidad de mayor entrenamiento en:

- Identificación de riesgos de seguridad en la etapa de diseño del proyecto.
- Desarrollo de casos de abuso, modelo de amenazas y clasificación de riesgos.
- Uso de SANS top 25 para reducir errores de software.
- Conocimiento de las políticas de seguridad de la compañía.

Además, ayudó a identificar que el equipo:

- Conoce el rol del security specialist, pero no está muy seguro de que sea una sola persona la que conozca todos los temas de seguridad en todas las áreas de desarrollo.
- Identifica la disponibilidad como el pilar de seguridad más conocido.
- Identifica las herramientas de análisis estático.
- Sigue las buenas prácticas de desarrollo con pruebas unitarias, estándares de programación y revisión de logs.

Durante la creación del product backlog y la priorización de estos el principal objetivo del Product Owner es obtener la mayor funcionalidad posible del producto a ser desarrollado por un equipo de Scrum. Además, es muy común que las necesidades del cliente no estén muy claras al inicio del proyecto por eso el sprint 0 es utilizado para empezar el product backlog.

El modelo propuesto por [6] identifica las actividades que deben ocurrir en la planificación de un proyecto junto con [25] en metodologías ágiles. El evento de planificación propuesto en este modelo une las propuestas de los dos modelos y toma ventaja de estos eventos de Scrum para definir lineamientos de seguridad a considerar durante el desarrollo del proyecto.

Las reuniones de planificación, diaria, revisión del sprint y la planificación del despliegue del producto son las ideales para identificar y verificar la seguridad del producto desarrollado. Identificar los requerimientos del usuario es una de las principales tareas del equipo de Scrum para estimar de forma adecuada y completar la meta del sprint. Al modificar el modelo de Scrum para considerar la seguridad se define que el equipo también debe identificar los requerimientos de seguridad del producto a desarrollar durante el sprint. La identificación de estos requerimientos de seguridad con el uso de los casos abuso y la identificación de usuarios no indebidos descritos en [41]

contribuyen a detectar fallas de seguridad en el producto y crear más casos de uso como respuesta a esos casos de abuso.

La gestión de riesgos de los requerimientos y la implementación de tareas de seguridad por un equipo de Scrum, que está enfocado en entregar requerimientos funcionales y que la metodología mismo prioriza como más importante las necesidades del usuario final, se vuelve compleja y difícil de gestionar sin un cambio de mentalidad del equipo y un rol que coordine al equipo en este nuevo proceso. Además de identificar que las tareas a desarrollar por el equipo necesitan de compromiso del equipo y de la compañía para desarrollar procesos que sean automáticos y ágiles.

Existe extensa literatura para procesos de revisión de código, pruebas en desarrollo, pruebas por el asegurador de la calidad para mejorar la calidad del desarrollo de software.

El proceso de desarrollo necesita ser mejorado y adaptado según las necesidades de la compañía y principalmente del equipo de desarrollo. Utilizar las reuniones de retrospectiva para mejorar el proceso de desarrollo es la forma ideal en donde el equipo de Scrum debe ser responsable de mejorar el proceso.

A partir del modelo diseñado coincidimos con [28] en la necesidad de un nuevo rol para la coordinación de las tareas de seguridad durante el ciclo de desarrollo. Este nuevo rol está encargado de coordinar las etapas del desarrollo de software seguro y del entrenamiento del equipo de Scrum para certificar que en el sprint se consideró la seguridad. Así, el rol fue extendido con respecto a la propuesta inicial de un auditor de seguridad a un rol activo durante el ciclo de desarrollo y con una meta importante de entrenar y desarrollar al equipo en temas de seguridad.

Además, se modificó el modelo propuesto por [31] en la definición de etapas del modelo seguro para incluir la etapa de planificación propuesto por [6]. Dando como resultado el modelo propuesto para considerar los eventos del Sprint 0 y priorización del backlog.

Para la definición de las tareas en cada etapa de consideró la propuesta de [6] para la etapa de planificación y [9], [35] y [28] para las etapas de identificación, implementación y verificación. Para la etapa de ajuste se consideró el propio modelo de Scrum para retroalimentar el modelo y mejorarlo. Así el modelo generado es una combinación de los modelos propuestos en la literatura actual.

7. CONCLUSIONES Y RECOMENDACIONES

7.1. Conclusiones

En esta tesis se diseñó un modelo de Scrum seguro utilizando buenas prácticas de seguridad. Utilizando los modelos propuestos en la literatura científica actual y las buenas prácticas de seguridad propuestas por compañías del sector. En la creación del modelo se identificaron y se definieron etapas, tareas, listas de verificación y un nuevo rol de seguridad para cumplir con una visión holística de la seguridad en el ciclo de desarrollo.

Se revisó los modelos de desarrollo utilizado en las áreas de desarrollo y mantenimiento y se identificó que la seguridad no es parte del proceso de Scrum y Kanban utilizados en la compañía.

El modelo requiere de un equipo entrenado y de un especialista de seguridad para reducir los riesgos de seguridad en el software desarrollado. Los casos de abuso, modelos de ataque, herramienta de análisis de código estático, programación en parejas son temas para conocer y aprender en un equipo de Scrum. Es en donde el criterio de artesanos del software es esencial para la mejorar la calidad del software.

El equipo de Scrum al construir el software y decidir que es o no riesgoso para el sistema necesita adueñarse del tema de seguridad como parte de su día a día en la entrega del producto desarrollado.

7.2. Recomendaciones

En un trabajo futuro se podría validar el modelo generado con un estándar de seguridad de software. Esta validación contribuiría a masificar el uso del modelo en las compañías de desarrollo de software y poder realizar un análisis de costo beneficio considerando el entrenamiento al equipo y la reducción de riesgos de seguridad en el software.

REFERENCIAS BIBLIOGRÁFICAS

[1] L. R. Vijayasarathy and C. W. Butler, "Choice of Software Development Methodologies: Do Organizational, Project, and Team Characteristics Matter?," *IEEE Software*, vol. 33, no. 5, pp. 86-94, 2016, doi: 10.1109/MS.2015.26.

[2] R. Kissel, K. Stine, M. Scholl, H. Rossman, J. Fahlsing, and J. Gulick. "NIST." https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-64r2.pdf (accessed 02/19, 2019).

[3] Microsoft. "Microsoft Security Development Lifecycle." https://www.microsoft.com/en-us/securityengineering/sdl/ (accessed 02/19, 2019).

[4] O. Project. "OWASP Secure Software Development Lifecycle Project." https://www.owasp.org/index.php/OWASP_Secure_Software_Development_Lifecycle_Project (accessed 25-03, 2019).

[5] S. H. Adelyar and A. Norta, "Towards a Secure Agile Software Development Process," in *2016 10th International Conference on the Quality of Information and Communications Technology (QUATIC)*, 6-9 Sept. 2016 2016, pp. 101-106, doi: 10.1109/QUATIC.2016.028.

[6] R. E. Maria, J. Luiz Antonio Rodrigues, and N. A. Pinto, "ScrumS: a model for safe agile development," presented at the Proceedings of the 7th International Conference on Management of computational and collective intElligence in Digital EcoSystems, Caraguatatuba, Brazil, 2015.

[7] L. B. Othmane and A. Ali, "Towards Effective Security Assurance for Incremental Software Development the Case of Zen Cart Application," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 31 Aug.-2 Sept. 2016 2016, pp. 564-571, doi: 10.1109/ARES.2016.86.

[8] E. Condon, M. Cukier, and T. He, "Applying Software Reliability Models on Security Incidents," in *The 18th IEEE International Symposium on Software Reliability (ISSRE '07)*, 5-9 Nov. 2007 2007, pp. 159-168, doi: 10.1109/ISSRE.2007.29.

[9] P. Maier, Z. Ma, and R. Bloem, "Towards a Secure SCRUM Process for Agile Web Application Development," presented at the Proceedings of the 12th International Conference on Availability, Reliability and Security, Reggio Calabria, Italy, 2017.

[10] J. A. Livermore, "Factors that impact implementing an agile software development methodology," in *Proceedings 2007 IEEE SoutheastCon*, 22-25 March 2007 2007, pp. 82-86, doi: 10.1109/SECON.2007.342860.

[11] K. Beck *et al.* "Manifesto for Agile Software Development." https://agilemanifesto.org/ (accessed 07/25, 2019).

[12] H. Lei, F. Ganjeizadeh, P. K. Jayachandran, and P. Ozcan, "A statistical analysis of the effects of Scrum and Kanban on software development projects," *Robotics and Computer-Integrated Manufacturing*, vol. 43, pp. 59-67, 2017/02/01/ 2017, doi: https://doi.org/10.1016/j.rcim.2015.12.001.

[13] "Extreme Programming." http://www.extremeprogramming.org/ (accessed 07/23, 2019).

[14] M. C. Paulk, "Extreme programming from a CMM perspective," *IEEE Software*, vol. 18, no. 6, pp. 19-26, 2001, doi: 10.1109/52.965798.

[15] K. Schwaber and J. Sutherland. "The Scrum Guide." https://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf (accessed 04, 2019).

[16] www.testingexcellence.com. "Overview of Scrum Agile Development Methodology." https://www.testingexcellence.com/overview-of-scrum-agile-development-methodology/ (accessed 07/21, 2019).
[17] P. Gambill. "Scrumban: A different way to be Agile." https://www.deloittedigital.com/us/en/blog-list/2013/scrumban--a-different-way-to-be-agile.html (accessed 07/27, 2019).
[18] A. Suresh. "SCRUM AND KANBAN – HOW SCRUMBAN CAN HELP." https://www.practicallogix.com/scrum-and-kanban-how-scrumban-can-help/ (accessed 07/27, 2019).
[19] P. Rogers. "Better Together — XP and Scrum." https://medium.com/agile-outside-the-box/better-together-xp-and-scrum-c69bf9bffcff (accessed 07/30, 2019).
[20] C. V. One, "13th Annual State of Agile Survey," p. 16
[21] M. Paul, *Official (ISC)2 Guide to the CSSLP CBK*. CRC Press, 2013.
[22] S. Lipner, "The trustworthy computing security development lifecycle," in *20th Annual Computer Security Applications Conference*, 6-10 Dec. 2004 2004, pp. 2-13, doi: 10.1109/CSAC.2004.41.
[23] OWASP. "CLASP Concepts." https://www.owasp.org/index.php/CLASP_Concepts (accessed 2019).
[24] BSIMM. "BSIMM Framework." https://www.bsimm.com/framework.html (accessed 07/02, 2019).
[25] Microsoft, "Security Development Lifecycle for

Agile Development," Microsoft Corporation, 2009.
[26] L. Huang, X. Bai, and S. Nair, "Developing a SSE-CMM-based security risk assessment process for patient-centered healthcare systems," presented at the Proceedings of the 6th international workshop on Software quality, Leipzig, Germany, 2008.
[27] S. S. E. C. M. M. S.-C. Project. "Systems Security

Engineering

Capability Maturity Model SSE-CMM." http://all.net/books/standards/ssecmmv3final.pdf (accessed 08/01, 2019).
[28] K. Rindell, S. Hyrynsalmi, and V. Leppänen, "Case Study of Security Development in an Agile Environment: Building Identity Management for a Government Agency," in *2016 11th International Conference on Availability, Reliability and Security (ARES)*, 31 Aug.-2 Sept. 2016 2016, pp. 556-563, doi: 10.1109/ARES.2016.45.
[29] P. Chandra. "Software Assurance Maturity Model." https://opensamm.org/downloads/SAMM-1.0.pdf (accessed 08/01, 2019).
[30] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering – A systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7-15, 2009/01/01/ 2009, doi: https://doi.org/10.1016/j.infsof.2008.09.009.
[31] C. Pohl and H.-J. Hof, "Secure Scrum: Development of Secure Software with Scrum," *eprint arXiv:1507.02992*, p. arXiv:1507.02992, 2015.
[32] K. Rindell, S. Hyrynsalmi, V. Lepp, #228, and nen, "A comparison of security assurance support of agile software development methods," presented at the Proceedings of the 16th International Conference on Computer Systems and Technologies, Dublin, Ireland, 2015.

[33] L. Ramadani and N. I. Utama, "Preliminary Investigation," in *2015 Second International Conference on Computing Technology and Information Management (ICCTIM)*, 21-23 April 2015 2015, pp. 134-139, doi: 10.1109/ICCTIM.2015.7224607.

[34] A. Singh, "Integrating the Extreme Programing Model with Secure Process for Requirement Selection," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 29-31 March 2018 2018, pp. 423-426, doi: 10.1109/ICECA.2018.8474598.

[35] B. Subedi, A. Alsadoon, P. W. C. Prasad, and A. Elchouemi, "Secure paradigm for web application development," in *2016 15th RoEduNet Conference: Networking in Education and Research*, 7-9 Sept. 2016 2016, pp. 1-6, doi: 10.1109/RoEduNet.2016.7753243.

[36] P. Reason and H. Bradbury, S. P. Ltd, Ed. *The Sage Handobook of Action Research Participative Inquiry and Practice*, Second ed. London: Sage Publications Ltd, 2008.

[37] CMS.gov. "Certified EHR Technology." https://www.cms.gov/Regulations-and-Guidance/Legislation/EHRIncentivePrograms/Certification.html (accessed 06/07, 2019).

[38] OWASP. "OWASP Top 10." https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project (accessed 06/07, 2019).

[39] SANS. "CWE/SANS TOP 25 Most Dangerous Software Errors." https://www.sans.org/top25-software-errors/ (accessed 06/07, 2019).

[40] J. Vietze, "Depiction of the PDCA cycle," vol. 800x473, PDCA_Process.png, Ed., ed: https://commons.wikimedia.org/wiki/, 2013.

[41] G. Sindre and A. L. Opdahl, "Eliciting security requirements by misuse cases," in *Proceedings 37th International Conference on Technology of Object-Oriented Languages and Systems. TOOLS-Pacific 2000*, 20-23 Nov. 2000 2000, pp. 120-131, doi: 10.1109/TOOLS.2000.891363.