CIMAT

Ciclo de Vida de Desarrollo de Software Seguro en Metodologías Ágiles

---

ESTUDIO GUIADO

ERICKA JAMIN ROBLES GOMEZ

22/07/2011

El propósito de este documento es proponer un ciclo de vida de desarrollo de software seguro y describir las herramientas automatizadas que permitan seguir los lineamientos del ciclo de vida, para realizar pruebas de seguridad en base al modelo Open Web Application Security Project (OWASP).

TABLA DE ILUSTRACIONES

AGRADECIMIENTOS

En primer lugar quiero agradecer a Dios por permitirme la oportunidad de tener salud para poder cursar ésta Maestría en Ingeniería de Software.

En seguida quiero agradecer a mi familia que fue motor de fuerza y motivación para poder concluir éstos estudios: A mi marido Jorge, mis hijos Edwin y Yoxan por supuesto a mis padres que aún en éstos momentos te doy gracias Señor por tenerlos Leobardo y Delia.

Además quiero agradecer a mis profesores la doctora Perla Velasco, Carmen Cano, el doctor Cuauhtémoc, Pepe Hernández y Juan Gabriel Hernández, gracias por su comprensión, paciencia y consejos.

Al doctor Hugo Mitre por sus revisiones, observaciones y recomendaciones que hizo sobre mi estudio.

A mis compañeros de doctorado Mery, Francisco y Mario por apoyarnos y guiarnos en las actividades de nuestro estudio diario.

A CIMAT que con el apoyo de una beca me permitió estudiar esta maestría y adquirir muchos conocimientos nuevos que con el tiempo podré transmitir a mis alumnos.

Al director Alejandro García de la empresa Softlogic, que nos permitió trabajar realizar nuestro proyecto con la industria.

A la Sra. Mamá de Mario que me prestó un cuartito para vivir en Zacatecas.

Al equipo técnico de CIMAT Perla y Martha por su amistad.

A Todos Muchas Gracias...

Ericka :)

INTRODUCCIÓN

Hasta hace poco tiempo, la seguridad era concebida con una propiedad deseable para el aseguramiento del software y formaba parte del conjunto de atributos a considerar para determinar la calidad de un software. El aumento de ataques de aplicaciones vulnerables ha conducido a un reconocimiento gradual respecto a que las protecciones a nivel de infraestructura no son suficientes para tal objetivo. Se reconoce que las protecciones y mitigaciones de seguridad de las aplicaciones deben ser especificadas a nivel del sistema y su arquitectura, e implementadas durante el desarrollo de aplicaciones y su operación.

En los últimos años esto ha evolucionado, y la seguridad del software es considerada como una característica del software que puede ser consistentemente demostrada. La garantía de producir software seguro es concebida como una disciplina, y si bien se presenta menos madura que otras relacionadas con la calidad, se cuenta con iniciativas, enfoques, metodologías y tecnologías emergentes.

Es por ello que en este documento se hace un estudio de los ciclos de vida de desarrollo seguro más comunes como lo son SSDL (Secure Software Development Lifecycle), SDL(Secure Development Lifecycle), SAMM(Software Assurance Maturity Model), BSIMM(Building Security In Maturity Model), y CLASP(Comprehensive Lightweight Application Security Process). En este documento se hace el análisis de comparativas de los ciclos de vida de desarrollo seguro para poder seleccionar el que se pueda aplicar a metodologías ágiles. Se define a detalle el ciclo de vida de desarrollo seguro SDL el cual para nuestro estudio es el que aplica para implementar en el desarrollo ágil también se hace mención de cómo implementarlo en las fases de SCRUM y se sugieren algunas herramientas para automatizar algunos procesos del ciclo de vida.

ESTADO DEL ARTE

La literatura disponible que describe el desarrollo de software en metodologías ágiles, en particular la necesidad de gestión y automatización en los procesos de las metodologías agiles, ha incrementado en los últimos años y continúa creciendo. Debido a este crecimiento no es posible llamar a cualquier bibliografía "completa" en el momento de su publicación. En este caso se aplicaron las siguientes preguntas como criterios de selección de bibliografía.

1. ¿La referencia explícitamente discute algún aspecto de metodología ágil, pruebas de seguridad, herramientas de automatización de pruebas de seguridad, automatización en metodologías ágiles, pruebas de vulnerabilidad, gestión de procesos en metodologías agiles, ciclos de vida de desarrollo de software seguro, etc.?

2. ¿La referencia está estrechamente relacionada con algún aspecto del proceso de desarrollo en las metodologías agiles, sobre todo con lo relacionado al desarrollo de software seguro y automatización de los procesos aunque no lo diga explícitamente?

3. ¿La referencia proporciona datos fiables de donde se obtuvo la información dando pauta a continuar con el trabajo de investigación sobre desarrollo de software seguro y automatización de los procesos en las metodologías ágiles?

4. ¿La referencia no pertenece a otro campo fuera de la línea de investigación del desarrollo de software seguro y la automatización de los procesos metodologías ágiles?

Es importante determinar en primer lugar si una referencia pertenece a un campo de la literatura de las metodologías agiles. Algunos artículos más importantes son los que explícitamente tienen algún término de desarrollo de software seguro en metodologías agiles y automatización de procesos en metodologías agiles.

TRABAJOS RELACIONADOS

En la actualidad existen varios estudios en los cuales se habla de los problemas en las metodologías ágiles lo cual da pauta a la elaboración de esta investigación. A continuación se detallan estos estudios:

Mishra, et. al [1]. El documento habla de un análisis crítico sobre los problemas de regulación o gestión que existen en las diferentes etapas de desarrollo dentro de las metodologías ágiles y para mi estudio habla precisamente de los existentes problemas de gestión de pruebas de seguridad y los problemas de utilizar herramientas que automaticen las actividades de los procesos. Este documento es de relevancia importante ya que da pauta al desarrollo de esta investigación. En este artículo se hace mención de la gestión de proyectos en las metodologías ágiles donde se dice que es diferente de las formas tradicionales de desarrollo de software. Temas tales como el cumplimiento de los reglamentos o normas, control de calidad del software, la estandarización de las normas tales como ISO o CMM, la seguridad y gestión del cambio, el crecimiento generalizado y la adopción de metodología en la industria. Aquí se hace mención a los problemas que existen en las metodologías ágiles que son los que motivan nuestro estudio: Problemas de reglamentación, problemas con la administración del conocimiento, administración de los roles del proyecto, problemas de seguridad, problemas de estandarización.

George Wilson [2]. Desde el punto de vista del autor considera que la automatización de las pruebas no se ha podido hacer hasta el momento, por el motivo de que el software y los requerimientos de automatización cambian constantemente y es necesario invertir mucho esfuerzo y tiempo para la solución de los nuevos problemas que se puedan presentar al tratar de remediar esto en una empresa. En su artículo menciona lo siguiente: “El software de automatización de pruebas ha estado disponible por más de un cuarto de siglo, pero la práctica sigue teniendo muchos escépticos y el mayor obstáculo para la adopción sigue siendo el nivel de mantenimiento requeridos para sostenerlo, alcanzar un nivel moderado de automatización requiere de una inversión considerable de presupuesto y de los recursos”.

Con la creciente complejidad de desarrollo de software y los departamentos de TI para las tecnologías agiles el trabajo de automatizar las pruebas tradicionales se ha vuelto demasiado complicado sostener dicha automatización.

Chaitanya Sudba Nakka[3].La publicación habla de la importancia del uso de herramientas en las diferentes etapas del desarrollo de software ágil. También menciona que no son fáciles de adquirir. El autor menciona algunos principios básicos para lograr la satisfacción del cliente lo cual es importante considerar al momento de hacer una elección de alguna herramienta.

Emilio García García, et. al [4]. Los autores hablan de los tipos de vulnerabilidades y la importancia de las herramientas en las diferentes etapas de desarrollo de software ágil. En particular en este capítulo menciona los tipos de pruebas de seguridad y la importancia de realizarlas en una etapa temprana del desarrollo del software.

David Geer [5]. En su publicación muestra los resultados de una encuesta en la cual menciona los nombres de ciclos de vida de desarrollo seguro y empresas que los utilizan. Esta publicación es de suma importancia para el desarrollo de nuestro estudio ya que es la base principal para la selección del mejor ciclo de vida de desarrollo seguro para metodologías ágiles.

Como podemos ver existe infinidad de publicaciones que hablan de la falta de gestión en el proceso de desarrollo de las metodologías ágiles, el desarrollo de software seguro y al mismo tiempo la necesidad de automatización de cada uno de estos procesos. También en algunos artículos menciona la importancia de la calidad de las pruebas de Seguridad. La mayoría de los artículos habla del ¿qué? pero no hay mucha información del ¿cómo hacerlo? Es por ello que toda esta literatura nos motiva a investigar el ¿cómo hacerlo?

MOTIVACIÓN

Los estudios y publicaciones encontrados hasta el momento hablan de la falta de gestión en las metodologías ágiles, ciclos de vida para el desarrollo de software seguro y la necesidad de al momento de implementar éstos ciclos de vida, éstos sean lo menos tediosos posible en cuanto a capacitación, implementación, documentación y mejora. También se habla de la necesidad de automatización de cada una de las actividades dentro de las metodologías ágiles.

Otra cosa muy relevante en las metodologías agiles y que varios autores lo mencionan es que dentro de las fases del desarrollo ágil, la fase de pruebas es una de las más indispensables ya que en esa fase se deberían encontrar los riesgos o vulnerabilidades del software, en esta fase el equipo de desarrollo se debe enfocar más en los atributos de calidad. Considerando esto como una actividad tardía lo ideal es que durante el desarrollo del software este tipo de riesgos se vayan mitigando y no cuando se esté a punto de lanzar el producto.

La motivación de desarrollar esta investigación la cual, está enfocada al desarrollo de software seguro en las Metodologías Ágiles. Nace a partir de que se da la oportunidad de trabajar en un proyecto en una empresa de desarrollo ágil (SOFTLOGIK), el proyecto consiste en realizar pruebas de seguridad al sistema PREP (Programa de Resultados Electorales Preliminares).

El sistema PREP es una aplicación en línea que permitirá al usuario capturar los datos de las actas de escrutinio y cómputo en las Elecciones Municipales, Estatales y Federales, con el objetivo de informar oportunamente a los Partidos Políticos sobre los resultados electorales preliminares de la elección. Este sistema permite difunde en tiempo real a través de Internet los resultados preliminares, utilizando para ello tecnologías informáticas de punta.

El Sistema PREP será utilizado por una gran variedad de usuarios, en cada casilla habrá un usuario accediendo a la aplicación Web para reportar los datos obtenidos después de cerrar la elección, por ejemplo en una elección Federal el número de usuarios sería aproximadamente de 130,000. De ahí la necesidad de realizar pruebas de seguridad al sistema ya que se requiere que el sistema tenga una operación y resolución técnica muy concreta.

Debido a la falta de experiencia y conocimiento sobre este tema se da la pauta a investigar métodos y herramientas de cómo hacer pruebas de seguridad en metodologías ágiles de una manera rápida y confiable.

Topándose con que existen mejores prácticas y métodos para el desarrollo de software seguro aplicadas a fases tempranas del desarrollo de software, no en fases tardías como se hizo con el sistema PREP, es por ello que en base a la experiencia vivida en el proyecto se detecta la falta de gestión y automatización de los procesos y se ve la necesidad de realizar esta investigación en la cual se proponga un método para el desarrollo de software seguro utilizando herramientas automatizadas basadas en el Modelo OWASP en las metodologías ágiles para que en futuros proyectos las empresas con poca experiencia en el desarrollo ágil puedan adoptar lo que se propone a continuación en este documento.

¿POR QUÉ ES NECESARIO DESARROLLAR SOFTWARE SEGURO?

Muchas de las organizaciones de desarrollo de software han iniciado a usar metodologías ágiles para construcción de sus aplicaciones. El desarrollo ágil se enfoca en los requerimientos del cliente incluyendo la seguridad. El introducir un ciclo de vida dentro de un sprint en el desarrollo ágil no retrasa los sprints, la idea es lograr la seguridad sin afectar el calendario del proyecto, los releases o las funcionalidades del producto. Los siguientes datos muestran un indicador importante del porque es recomendable utilizar ciclos de vida de desarrollo de software seguro: [6]

- 6.75 millones es el promedio del costo total de una violación a la información por incidente (Ponemon Institute Jan 2010).
- El 60% al 90% de los ciber ataques y las violaciones a la seguridad de internet son generados a través de aplicaciones de internet (varios).
- El 90% de los sitios web son vulnerables a ataques. (Verizon Business Data Breach Report-April 2009).
- Malware en sitios web legítimos se han duplicado en 6 meses (IT PRO-2008).
- 400 nuevas vulnerabilidades incrementan en un mes.

A continuación se muestra una ilustración en la cual se puede ver el incremento de los diferentes tipos de vulnerabilidades con el transcurso de los años. De ahí la importancia de adoptar un método para el desarrollo de software seguro.

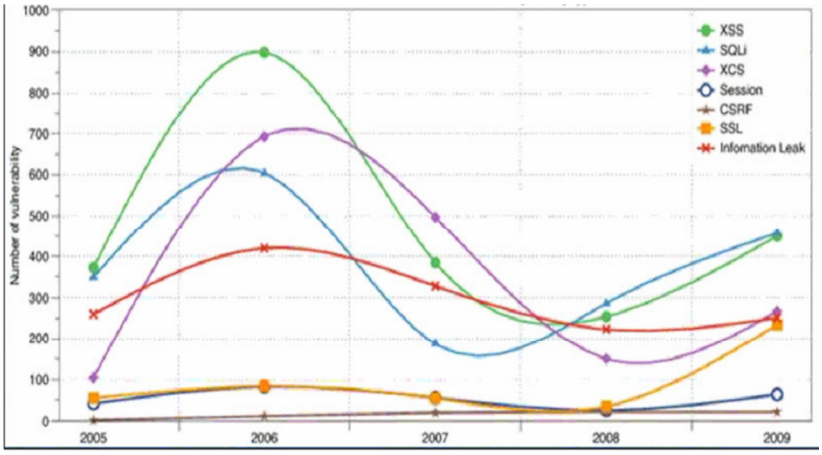

Ilustración 1 Evolución del incremento de vulnerabilidades en las aplicaciones Web en cada año

Cabe mencionar que para metodologías ágiles es crítico desarrollar software seguro ya que el costo real del software inseguro de hoy proviene de la pérdida de datos, la interrupción del proceso, y daños de ataques criminales, que son mucho más importantes que nunca en este momento. El peligro de hoy en día es latente ya que existen una gran variedad de ataques y éstos con el tiempo van incrementando. Es importante que las metodologías ágiles adopten el desarrollo de software seguro ya que un agujero en el software puede ser muy malicioso, puede ocasionar retraso con respecto a los tiempos de entrega, incremento en los costos de reparación del daño y además ahora con regulaciones federales tales como: HIPPA(Health Insurance Portability and Accountability Act), PCI(Security Standars Council) y Sarbanes-oxley en Estados Unidos de America y en México la Ley Federal de protección de datos personales por dar un ejemplo (cada una con fuertes multas por infracciones de datos electrónicos). Es obvio que la seguridad de las aplicaciones no pueden ser ignoradas, incluso con la adopción de prácticas de desarrollo ágil y menos porque se está hablando de un proceso ágil donde el software debe ser entregado a tiempo.

¿ACTUALMENTE EXISTEN COMPAÑIAS QUE IMPLEMENTEN CICLOS DE VIDA DE DESARROLLO SEGURO?

La publicación The Errata Survey llevó a cabo una encuesta a través de su página Web. La compañía anunció que la encuesta se llevaría a cabo a través del blog de The Errata Survey, en sus tweets, en la conferencia de seguridad en el BSides San Francisco y en la conferencia RSA, un evento líder en seguridad. La compañía realizó la encuesta para saber ¿qué organizaciones no implementan seguridad en el desarrollo de software? y ¿por qué?: La encuesta recibió 46 respuestas. Mientras que la encuesta tiene un margen de error de aproximadamente el 14.5%.

La Ilustración 4 muestra a todos los encuestados por The Errata Survey, 11 representan 23.9% que dijeron que SDL implica demasiado tiempo, 4 representan un 8.7 % que dijeron que es innecesario y 2% representan un 4.3% dijo que es claro, 7 representan 15.2% dijo no conocían SDL.[5]

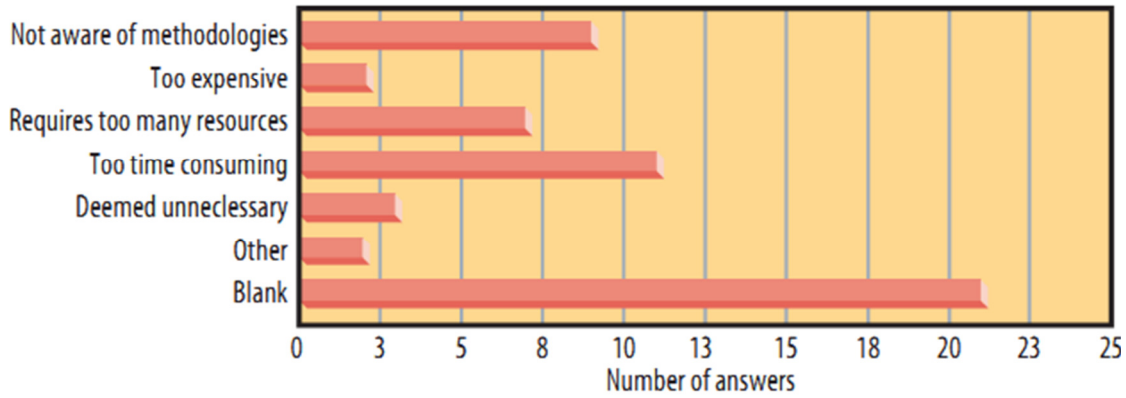

Ilustración 2 Razones para no adoptar un SDL "The Errata Survey"

De las empresas que si implementan ciclos de vida de desarrollo seguro a continuación se hace una descripción de cada uno de ellos de manera breve.

ENFOQUES DE SECURE DEVELOPMENT LIFECYCLE

Microsoft SDL (Secure Development Lifecycle)

El objetivo principal de SDL es aumentar la calidad de la funcionalidad impulsando software mediante la mejor postura de seguridad. Las actividades de seguridad son más a menudo relacionadas con las actividades de construcción basadas en la funcionalidad. Por ejemplo modelando amenazas se inicia desde las dependencias de la arquitectura con sistemas externos, mientras que la arquitectura podría reducir esas amenazas en las primeras etapas. El proceso de SDL está bien organizado y las actividades relacionadas se agrupan en etapas. Aunque estas etapas son específicas de seguridad, es sencillo para asignar a fases de desarrollo estándar. SDL hace un buen trabajo en el que especifica el método que debe utilizarse para ejecutar actividades, que, en promedio, son concretas y con frecuencia algo pragmáticas.

En la siguiente ilustración se pueden observar las actividades que se tienen que seguir en SDL y se observa como desde una etapa temprana es posible detectar o en su caso eliminar vulnerabilidades. [7]

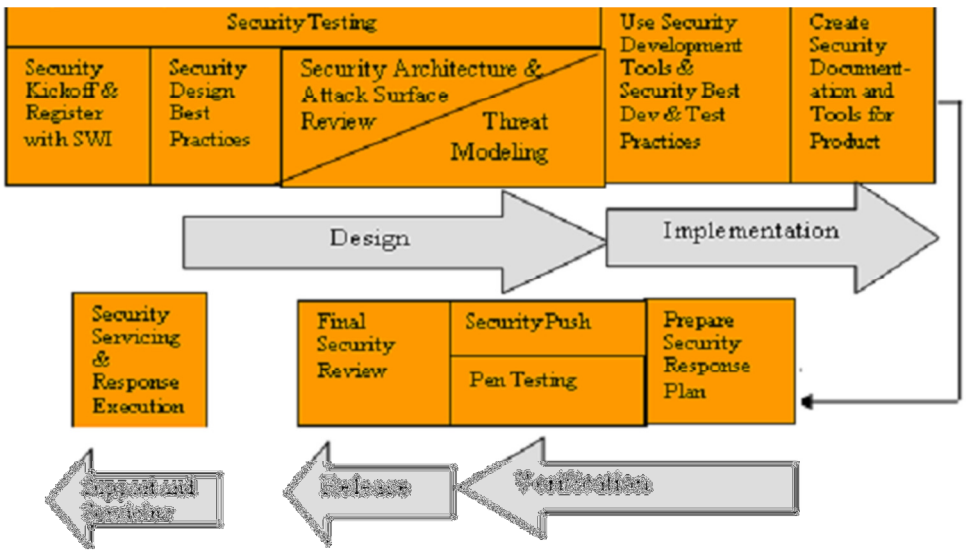

Ilustración 3 Actividades en SDL

SAMM (Software Assurance Maturity Model)

SAMM es un marco abierto para ayudar a las organizaciones a formular y aplicar una estrategia de seguridad de software que se adapte a los riesgos específicos que enfrenta la organización. Los recursos proporcionados por SAMM ayudarán a:

- La evaluación de una organización de las prácticas existentes de desarrollo de software de seguridad.

- La construcción de software de seguridad equilibrado a través de un programa de aseguramiento en sectores bien definidos llamados iteraciones.

- Demostrar mejoras concretas para un programa de garantía de seguridad.

- Definición y medición de las actividades relacionadas con la seguridad en la organización.

SAMM fue definido pensando en la flexibilidad de tal manera que puedan ser utilizados por las organizaciones pequeñas, medianas y grandes con cualquier estilo de desarrollo. Además, este modelo puede ser aplicado en toda la organización, para una línea de negocio única, o incluso para un proyecto individual. Más allá de estos rasgos, SAMM fue construido en base a los siguientes principios:

- El comportamiento de una organización cambia lentamente con el tiempo. Un software de éxito desarrollado bajo un programa de seguridad debe ser especificado en iteraciones pequeñas que ofrezcan material ganancias de la garantía, mientras que gradualmente se esté trabajando hacia objetivos a largo plazo.

- No hay una receta única que funcione para todas las organizaciones. Un software de seguridad es un marco que debe ser flexible y permitir a las organizaciones adaptar sus decisiones a tolerancia a riesgos y a la forma en que ellos construyen el software.

- Es una guía relacionada con las actividades de seguridad que debe ser prescriptiva. Todos los pasos en la construcción y la evaluación de un programa de garantía deben ser simples, bien definidos y medibles. Este modelo también proporciona plantillas de hoja de ruta para los tipo más comunes de las organizaciones.[8]

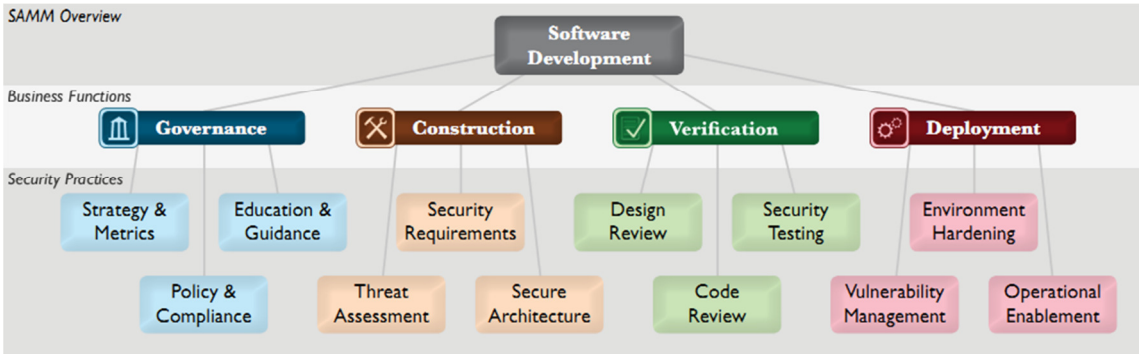

Ilustración 4 Framework SAAM

BSIMM (Building Security In Maturity Model)

El BSIMM pretende ser una guía para la construcción y la evolución de una iniciativa de seguridad de software. Como se puede ver cuando usted se familiarice con las actividades de BSIMM, inculcando la seguridad del software en una organización requiere de una planificación cuidadosa y siempre implica un cambio de organización amplio. Pero señalando claramente los objetivos y metas, y por las prácticas de seguimiento con indicadores adaptados a su propia iniciativa, que metódicamente se puede construir la seguridad del software en las prácticas de su organización de desarrollo de software.

Al utilizar el BSIMM como una guía para su propia iniciativa de seguridad de software, puede aprovechar la experiencia de muchos años capturada en el modelo.

Debe adaptar las actividades sugeridas para su propia organización (de considerar cuidadosamente los objetivos que nos documentó).

Software de seguridad siempre debe ser impulsado por la gestión de riesgo empresarial. Al llevar a cabo las actividades descritas en el BSIMM, que puede evolucionar gradualmente su programa y alcanzar la clase mundial de software de seguridad en el momento óptimo sin un coste excesivo.

En general, cualquier iniciativa de seguridad de software se ha creado con algunos de los objetivos de alto nivel en la mente. El BSIMM es apropiado para usted si sus metas de negocio global de software de seguridad incluyen:

- Decisiones informadas sobre la gestión de riesgos.
- Claridad sobre lo que es "lo que hay que hacer" para todos los involucrados en el software de seguridad.
- Reducción de costes a través de procesos estándar y repetible.
- Aumento de la calidad del código. [9]

domain	practice	business goals
Governance	Strategy and Metrics	Transparency of expectations, Accountability for results
	Compliance and Policy	Prescriptive guidance for all stakeholders, Auditability
	Training	Knowledgeable workforce, Error correction
Intelligence	Attack Models	Customized knowledge
	Security Features and Designs	Reusable designs, Prescriptive guidance for all stakeholders
	Standards and Requirements	Prescriptive guidance for all stakeholders
SSDL Touchpoints	Architecture Analysis	Quality control
	Code Review	Quality control
	Security Testing	Quality control
Deployment	Penetration Testing	Quality control
	Software Environment	Change management
	Vulnerability Mgmt and Change Management	Change management

Ilustración 5 Metas asociadas con cada práctica de BSIMM

Securosis' SSDL (Secure Software Development Lifecycle)

SSDL, creado por Securosis consultoría de seguridad, se compone de mensajes en blogs sobre el desarrollo ágil de software no se ocupa de la seguridad, y se mueve muy rápidamente a través de sub-ciclos cortos para hacer el proceso fácil, explicó. Se ocupa de cuestiones como la formación y las pruebas. SSDL hace hincapié en dar prioridad a las cuestiones de seguridad podría tener lugar en algo más que un subciclo de desarrollo de dos a cuatro semanas. [10]

CLASP (Comprehensive Lightweight Application Security Process)

Esta es la propuesta original de Secure Inc. Software y posteriormente donado a OWASP; CLASP es un proceso ligero para la construcción de software de seguridad. Incluye un conjunto de 24 actividades de alto nivel y recursos adicionales, las características clave de CLASP son las siguientes:

- El objetivo principal de CLASP es apoyar en la construcción de software, en donde la seguridad tiene un papel central. Las actividades de CLASP se definen y conciben principalmente desde la perspectiva de la seguridad teórica y, por tanto, la cobertura del conjunto de actividades es bastante amplia.

- CLASP se define como un conjunto de actividades independientes, que tienen que ser integradas en el proceso de desarrollo y de su entorno de trabajo. Dos rutas ("legado" y "Green-field") se han definido para dar una orientación sobre cómo combinar las actividades en un conjunto coherente y ordenado.

- Sin embargo, esto no es un proceso integrado. CLASP define funciones que pueden tener un impacto en la situación de seguridad del producto de software y asigna las actividades de estas funciones. Los roles son responsables de la finalización y la calidad de los resultados de una actividad.

- CLASP proporciona un amplio conjunto de recursos de seguridad que facilitan y apoyan la ejecución de las actividades. Uno de estos recursos es una lista de 104 vulnerabilidades de seguridad conocidas en el código fuente de aplicación. [7]

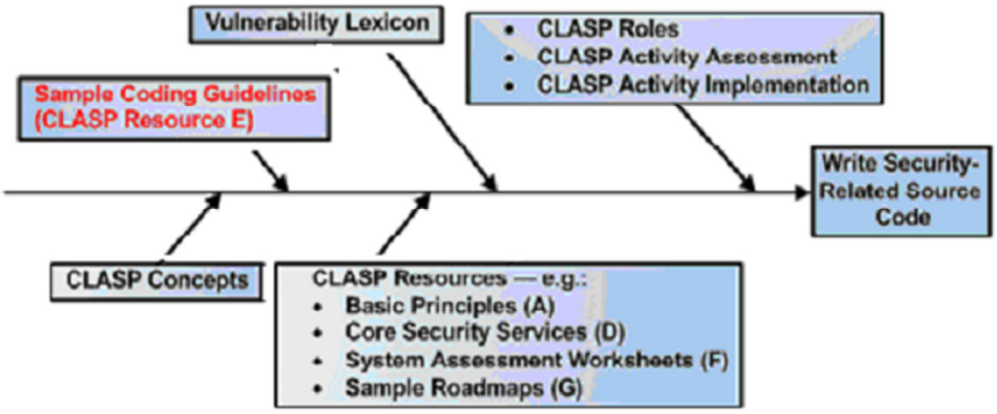

Ilustración 6 Actividades de CLASP

En el siguiente tema se explica las consideraciones tomadas en cuenta para la selección del ciclo de vida de desarrollo seguro en metodologías ágiles.

SELECCIÓN DEL CICLO DE VIDA DE DESARROLLO SEGURO

Para la selección del ciclo de vida de desarrollo de software seguro se tomaron en cuenta algunas características importantes que los desarrolladores de software ágil consideran son sus requisitos para la adopción de un ciclo de vida.

Un requisito que para toda empresa es importante por el retorno de inversión que tiene que dejar el desarrollo de software son los recursos que ocupa el implementar un ciclo de vida. Como se puede apreciar en la ilustración 7 se muestra una comparación de todos los métodos y se especifica que necesidades tiene cada uno de ellos en cuanto a recursos (personas que involucradas en el ciclo de vida de desarrollo seguro), el que menos recursos necesita es el método SDL, ya que para la implementación de este ciclo de vida solo es necesario un Tester y los líderes de equipo.

Otra consideración importante es el tiempo (esfuerzo requerido para la implementación del ciclo de vida) un ciclo de desarrollo requiere un tiempo de adaptación e implementación para medir esto se consideró el número de prácticas, actividades, fases o procedimientos que tiene cada uno de los métodos y la flexibilidad de omitir alguna actividad o adaptarla según las necesidades de la empresa, Para ello se realizó un análisis de cada uno de los métodos llegando a la conclusión de que el que requiere menos tiempo de implementar y es más flexible es SDL ya que consta de 16 procedimientos distribuidos en 5 fases. En cuanto a los demás métodos quedan descartados ya que el implementarlos requiere de más esfuerzo y además son menos flexibles en cuanto a la ejecución de sus actividades, requieren de muchas revisiones y verificaciones de código muy rigurosas cosa que no es aplicable a metodologías ágiles.

David Geer[5], nos proporciona otro dato muy importante de una encuesta realizada a empresas de desarrollo de software y es la popularidad que tiene cada uno de los métodos en las empresas que sí utilizan un ciclo de vida de desarrollo seguro. Como podemos ver en la tabla el SDL de Microsoft es el más popular.

Característica	SAMM	BSIMM	SSDL	CLASP	SDL
Recursos	Desarrolladores, arquitectos, administradores, QA Tester, Auditores de Seguridad, Dueños del negocio, personal de apoyo, operaciones	Constructores (desarrolladores, arquitectos, administradores) Tester Operaciones Administradores Ejecutivos y mandos medios(dueños del negocio, administradores del producto)	Arquitecto, desarrolladores, testers, administradores de programación	Tester oficial, administrador del proyecto, ingeniero de requerimientos, arquitecto de software	Tester y líderes de equipo
Tiempo	20 prácticas	109 actividades	6 Fases	24 actividades	5 fases 16 procedimientos
Popularidad (David Geer,2010)	12%	13%	7%	13%	23%

Ilustración 7 Caracterísiticas de selección del ciclo de vida

Posteriormente se hace un análisis más profundo de porque las empresas de desarrollo de software ágil no implementan ciclos de vida de desarrollo seguro encontrando que como requisitos piden que sea de acceso libre la descripción del ciclo de vida, que sea aplicable a metodologías ágiles, que permita la utilización de herramientas de automatización de las fases del ciclo de vida, que sea aplicable a iteraciones de desarrollo cortas.

Como se puede observar en la ilustración 8 el resultado obtenido es SDL ya que este ciclo de vida es de acceso libre, requiere pocos recursos (un Tester y líderes de proyecto), consume menos tiempo (16 procesos distribuidos en 5 fases), es aplicable a metodologías ágiles, permite la utilización de herramientas de automatización para la ejecución de los procesos del ciclo de vida, es aplicable a iteraciones de desarrollo y es el más popular.

Característica	SAMM	BSIMM	SSDL	CLASP	SDL
Acceso Libre	X	X	X	X	X
Requiere pocos recursos					X
Consume Poco Tiempo			X		X
Aplica a Metodologías ágiles	X		X		X
Utiliza herramientas para la automatización de las fases del ciclo de vida	X	X	X	X	X
Es aplicable a Iteraciones de desarrollo	X		X		X
Es el más popular					X

Ilustración 8 Resultado de la Selección del ciclo de vida

A continuación se realiza una descripción detallada de cada una de las fases y procedimientos de SDL aplicada a metodologías ágiles, para este estudio se hace la aplicación en SCRUM.

Ciclo de vida de desarrollo seguro (SDL)

SDL se basa en tres conceptos básicos: formación, mejora continua de los procesos y responsabilidad. SDL destaca especialmente la importancia de comprender la causa y el efecto de las vulnerabilidades de seguridad y *requiere* una evaluación periódica de los procesos de SDL y la introducción de cambios como respuesta a los avances tecnológicos o las nuevas amenazas.

El modelo de optimización de SDL está estructurado en torno a cinco áreas de capacidades que, en líneas generales, se corresponden con las fases del ciclo de vida de desarrollo de software:

- Formación, directivas y capacidades organizativas
- Requisitos y diseño
- Implementación
- Comprobación
- Lanzamiento y respuesta

Además, el modelo de optimización de SDL define cuatro niveles de madurez para los procedimientos y capacidades de estas áreas: básico, estandarizado, avanzado y dinámico.
El modelo de optimización de SDL comienza en el nivel de madurez básico, con pocos o sin procesos, cursos de formación y herramientas, y evoluciona hacia el nivel dinámico, que se caracteriza por la total conformidad con SDL en toda la organización. La total conformidad con SDL incluye procesos eficaces, personas altamente cualificadas, herramientas especializadas y un alto grado de responsabilidad por parte de las personas tanto internas como externas a la organización

lustración 9 Modelo de optimización SDL con los niveles de capacidades y madurez(Microsoft)

APLICABILIDAD DE SDL

Por experiencia, se recomienda someter a SDL las aplicaciones con una o varias de las siguientes características:

- Aplicaciones implementadas en un entorno empresarial
- Aplicaciones que procesan información de identificación personal (PII) u otro tipo de información confidencial
- Aplicaciones que se comunican frecuentemente a través de Internet u otras redes

ROLES, RESPONSABILIDADES Y CUALIFICACIÓN DEL PERSONAL DE SEGURIDAD

- **Roles de revisión o asesoramiento.** Estos roles se han diseñado para supervisar la seguridad y la privacidad de los proyectos y tener autoridad para aceptar o rechazar los planes de seguridad y privacidad de un equipo de proyecto.
  - *Asesor de seguridad/Asesor de privacidad.* Estos roles se asignan a expertos en la materia ajenos al equipo de proyecto. Puede tratarse de un miembro cualificado de un grupo centralizado independiente dentro de la organización que se ha constituido específicamente para este tipo de revisiones o puede tratarse de un experto ajeno a la organización. La persona elegida para esta tarea debe desempeñar dos subroles:
    - Auditor. Esta persona debe supervisar cada fase del proceso de desarrollo de software y certificar que se cumple cada uno de los requisitos de seguridad. El asesor debe tener la libertad de certificar el cumplimiento (o incumplimiento) de los requisitos de seguridad y privacidad sin que interfiera el equipo de proyecto.
    - Experto. La persona elegida para el rol de asesor debe disponer de conocimientos contrastados en materia de seguridad.
  - *Combinación de los roles de asesoramiento.* El rol de asesor de seguridad puede combinarse con el rol de asesor de privacidad si se logra identificar a una persona con los conocimientos y la experiencia apropiados.
- **Líderes del equipo.** Estos roles se deben asignar a los expertos en la materia del equipo de proyecto. Se encargan de la negociación, la aceptación y el seguimiento de los requisitos mínimos de seguridad y privacidad, además de mantener una comunicación transparente con los asesores y responsables de la toma de decisiones a lo largo de un proyecto de desarrollo de software.

- *Líder de seguridad/Líder de privacidad.* Esta persona (o grupo de personas) no solo es responsable de asegurar que se han resuelto todos los problemas de seguridad de una versión de software sino que también se encarga de coordinar y realizar un seguimiento de los problemas de seguridad del proyecto. Este rol también notifica el estado al asesor de seguridad y a otros componentes pertinentes del equipo de proyecto (por ejemplo, los responsables de desarrollo y de pruebas).

- *Combinación de roles.* Al igual que en el caso de los roles de asesor de seguridad y asesor de privacidad, las responsabilidades atribuidas al rol de líder pueden combinarse si se logra identificar a una persona con los conocimientos y la experiencia apropiados.

ACTIVIDADES DE SEGURIDAD SIMPLIFICADAS DE SDL

El proceso SDL de Microsoft es un conjunto de actividades de seguridad obligatorias, que se presentan en el orden en que deben llevarse a cabo y se agrupan por cada una de las fases de un ciclo de vida de desarrollo de software tradicional.

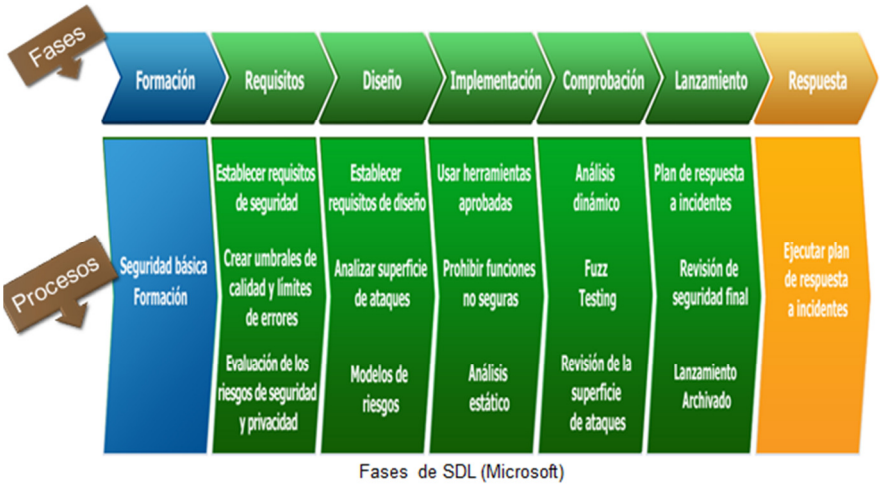

Ilustración 10 Actividades de Seguridad Simplificadas de SDL

De las organizaciones situadas en los niveles Avanzado y Dinámico del modelo de optimización de SDL, se espera que los procesos de seguridad tengan cierto grado de sofisticación. Dicho eso, da igual que un modelo de riesgos sea generado, por ejemplo, como resultado de una sesión de pizarra con el equipo de desarrollo, se redacte como un relato en un documento o se elabore con ayuda de una herramienta especializada, como la herramienta de creación de modelos de riesgos de SDL. Si bien el proceso SDL se beneficia de las inversiones en herramientas y procesos de automatización eficaces, su verdadero valor reside en los resultados exhaustivos y exactos.

SDL EN SCRUM

Después de un análisis de los procesos del ciclo de vida SDL y de la metodología ágil SCRUM se crea el siguiente meta-modelo de cómo implementar SDL en SCRUM. En la siguiente ilustración se muestra las fases de SCRUM y se propone cuales procesos de SDL ejecutar en cada una de las fases:

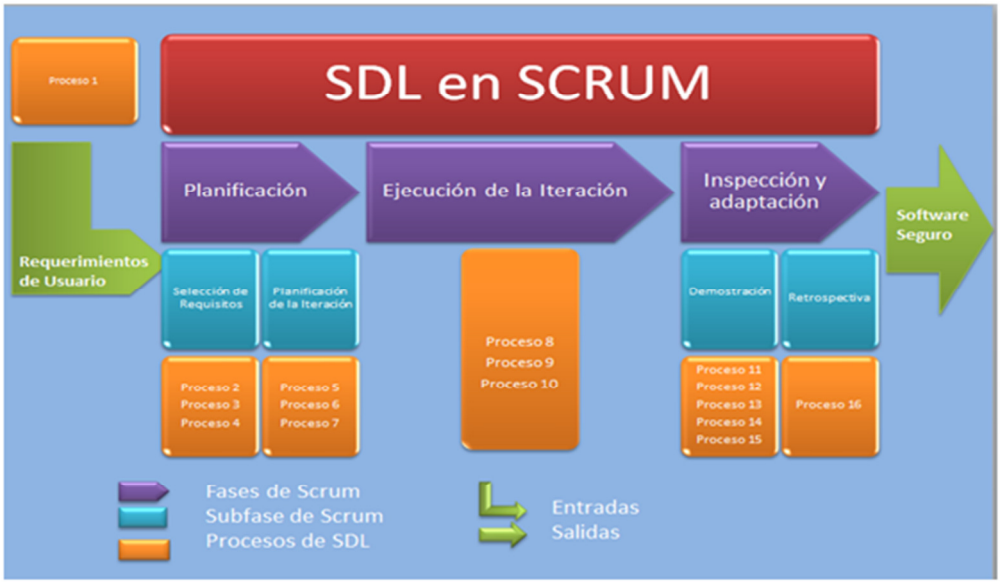

Ilustración 11 SDL en las fases de SCRUM

A continuación se va a hacer una descripción del procedimiento y se hace la mención de como ejecutarlo en SCRUM.

Proceso 1 de SDL: Formación y capacitación

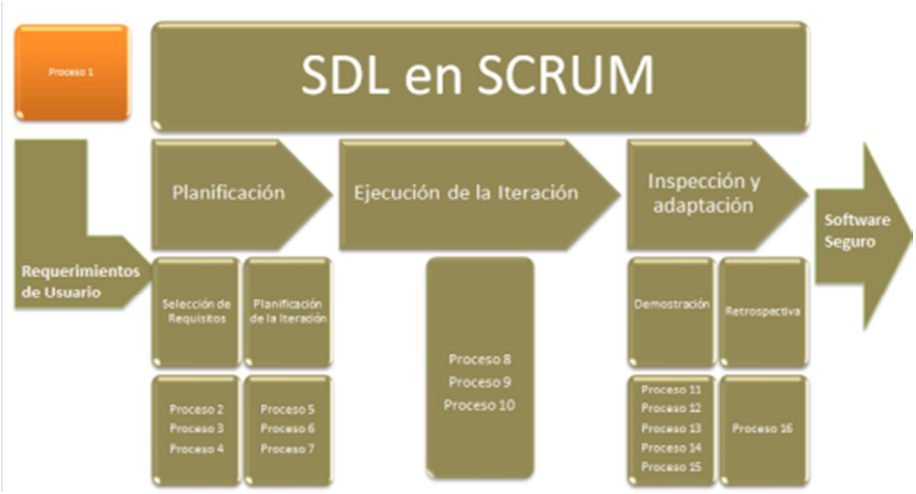

Ilustración 12 Proceso 1 SDL Formación y capacitación

El Tester y los líderes de equipo deben recibir una formación apropiada con el fin de mantenerse al corriente de los conceptos básicos y últimas tendencias en el ámbito de la seguridad y privacidad.
La formación básica en materia de seguridad de software debe abordar conceptos fundamentales como:

- Diseño seguro, incluidos los siguientes temas:
  - Reducción de la superficie de ataques
  - Defensa en profundidad
  - Principio de privilegios mínimos
  - Configuraciones predeterminadas seguras
- Modelos de riesgos, incluidos los siguientes temas:
  - Información general sobre los modelos de riesgos
  - Implicaciones de diseño de un modelo de riesgos
  - Restricciones de codificación basadas en un modelo de riesgos
- Codificación segura, incluidos los siguientes temas:
  - Saturaciones de búfer
  - Errores aritméticos de enteros
  - XSS
  - Inyección de código SQL
  - Criptografía débil
- Pruebas de seguridad, incluidos los siguientes temas:
  - Diferencias entre pruebas de seguridad y pruebas funcionales
  - Evaluación de riesgos
  - Métodos para poner a prueba la seguridad

- Privacidad, incluidos los siguientes temas:
  - Tipos de datos confidenciales
  - Procedimientos recomendados de diseño de privacidad
  - Evaluación de riesgos
  - Procedimientos recomendados de desarrollo de privacidad
  - Procedimientos recomendados de pruebas de privacidad

La formación anterior establece una línea base adecuada para los conocimientos del personal técnico. Según lo permitan el tiempo y los recursos, quizás sea necesario proporcionar formación en materia de conceptos avanzados. Algunos ejemplos son los siguientes:

- Diseño y arquitectura de seguridad avanzados
- Diseño de interfaces de usuario de confianza
- Información detallada sobre vulnerabilidades de seguridad
- Implementar mitigaciones de amenazas personalizadas

PRIMERA FASE SDL: REQUISITOS

Como se puede observar en la siguiente ilustración tenemos la fase de planificación de SCRUM y la Subfase de Selección de requisitos, en la cual se propone ejecutar los procesos dos, tres y cuatro de SDL.

Ilustración 13 Planificación->Selección de requisitos (SCRUM)
Proceso 2,3,4 (SDL)

Proceso 2 de SDL: requisitos de seguridad

En la fase de planificación en SCRUM , al principio de un proyecto, se realiza un análisis de los requisitos de seguridad y privacidad, el cual incluye la especificación de los requisitos de seguridad mínimos de la aplicación en el entorno operativo previsto, así como la especificación y la implementación de un sistema de seguimiento de los elementos de trabajo y de las vulnerabilidades de seguridad.

En esta fase se debe utilizar la plantilla de requerimientos proporcionada por Microsoft ya que permite registrar los requisitos de seguridad identificados en este proceso.

Descripción	Requisito/ Recomendación	Se aplica a los servicios en línea	Se aplica a código adminstrado	Se aplica a código nativo
Crear un modelo de amenaza de referencia	Requisito	X	X	X
Determinar las normas de respuesta de seguridad	Requisito	X	X	X
Uso aprobado de analizadores de XML	Requisito	X		X
Utilizar las últimas versiones del compilador	Requisito	X	X	X
Eliminar las dependencias de la autenticación NTLM	Requisito	X	X	X

Ilustración 14 Apendix R: SDL-Agile One-Time Requirements(Microsoft)

Proceso 3 de SDL: umbrales de calidad y límites de errores

Se usan umbrales de calidad y *límites de errores* para establecer niveles mínimos aceptables de calidad en materia de seguridad y privacidad.

Un límite de errores es un umbral de calidad que se aplica a todo el proyecto de desarrollo de software. Se usa para definir los umbrales de gravedad de las vulnerabilidades de seguridad; por ejemplo, en la aplicación no hay vulnerabilidades conocidas que estén clasificadas como "críticas" o "importantes" en el momento de su lanzamiento.

Proceso 4 de SDL: evaluación de los riesgos de seguridad y privacidad

Las evaluaciones de los riesgos de seguridad (SRA) y de privacidad (PRA) son procesos obligatorios que identifican los aspectos funcionales del software que requieren una revisión exhaustiva. Dichas evaluaciones deben incluir la siguiente información:

1. (Seguridad) Partes del proyecto que van a requerir *modelos de riesgos* antes del lanzamiento.
2. (Seguridad) Las partes del proyecto que van a requerir revisiones del diseño de seguridad antes del lanzamiento.
3. (Seguridad) Partes del proyecto (si procede) que van a requerir pruebas de penetración por parte de un grupo externo al equipo de proyecto y acordado mutuamente.
4. (Seguridad) Requisitos de análisis o de pruebas adicionales que el asesor de seguridad considera necesarios para mitigar los riesgos de seguridad.
5. (Seguridad) Ámbito específico de los *requisitos en materia de pruebas de exploración de vulnerabilidades mediante datos aleatorios*.
6. (Privacidad) ¿Cuál es el nivel de impacto sobre la privacidad? La respuesta a esta pregunta se basa en las siguientes pautas:

    - P1 Alto riesgo para la privacidad. La característica, el producto o el servicio almacena o transfiere datos de identificación personal, cambia la configuración o las asociaciones de los tipos de archivo, o instala software.
    - P2 Riesgo moderado para la privacidad. El único comportamiento que afecta a la privacidad en la característica, el producto o el servicio es una transferencia de datos anónimos única e iniciada por el usuario (por ejemplo, el usuario hace clic en un vínculo y el software abre un sitio web).
    - P3 Bajo riesgo para la privacidad. La característica, el producto o el servicio no tiene ningún comportamiento que afecte a la privacidad. No hay ninguna transferencia de datos personales ni anónimos, no se almacenan datos de identificación personal en el equipo, no se modifica ninguna configuración en nombre del usuario y no se instala ningún software.

SEGUNDA FASE SDL: DISEÑO

En la ilustración 15 tenemos se puede observar que se continúa en la fase de Planificación de SCRUM pero ahora avanzamos a la Subfase de Planificación de la iteración en la cual se propone ejecutar los procesos cinco, seis y siete de SDL. Los cuales se detallan a continuación:

Ilustración 15 Planificación->Planificación de la Iteración(SCRUM) Proceso 5,6,7 (SDL)

Proceso 5 de SDL: requisitos de diseño

Además, es de suma importancia que los equipos de proyecto entiendan la diferencia entre "características seguras" y "características de seguridad". Es perfectamente posible que se implementen características de seguridad que, en realidad, son inseguras. Las *características seguras* se definen como características cuya funcionalidad está debidamente diseñada con respecto a la seguridad, incluida una rigurosa validación de todos los datos antes de su procesamiento o una implementación criptográficamente robusta de bibliotecas para los servicios de cifrado. El término *características de seguridad* describe la funcionalidad de un programa que tiene implicaciones para la seguridad, como la autenticación Kerberos o un firewall

Las especificaciones de diseño deben describir las características de seguridad o de privacidad que van a estar directamente expuestas a los usuarios, como las que requieren la autenticación del usuario para obtener acceso a datos específicos o que necesitan el consentimiento del usuario para usar una característica con un alto riesgo para la privacidad.

La especificación funcional debe:

- Describir de manera exacta y completa el uso previsto de una característica o función.
- Describir cómo se implementa de forma segura la característica o función.

Proceso 6 de SDL: reducción de la superficie de ataques

La reducción de la superficie de ataques es una forma de reducir el riesgo dando a los atacantes menos oportunidades para aprovechar un posible punto débil o una posible vulnerabilidad. Para reducir la superficie de ataques, se cierra o se restringe el acceso a los servicios del sistema, se aplica el principio de privilegios mínimos o se usan en la medida de lo posible defensas por capas.

Proceso 7 de SDL: modelos de riesgos

Los modelos de riesgos también permiten estudiar los problemas de seguridad en el nivel de los componentes o aplicaciones. La creación de un modelo de riesgos es un trabajo de equipo que implica a los administradores de programas o proyectos, desarrolladores y evaluadores; además, es la principal tarea de análisis de seguridad que se realiza en la fase de diseño del software.

TERCERA FASE SDL: IMPLEMENTACIÓN

En la siguiente ilustración se observa los procesos que se ejecutan en las iteraciones de SCRUM para cada iteración es necesario ejecutar los procesos ocho, nueve y diez de SDL con el objetivo de reducir riesgos antes de la demostración del producto.

Ilustración 16 Ejecución de la Iteración (SCRUM) proceso 8,9,10 (SDL)

Proceso 8 de SDL: usar herramientas aprobadas

Esta lista la debe aprobar el asesor de seguridad del equipo de proyecto. En términos generales, los equipos de desarrollo deben procurar usar la versión más reciente de las herramientas aprobadas a fin de aprovechar las nuevas protecciones y funciones de análisis de seguridad.

Proceso 9 de SDL: prohibir funciones no seguras

Una vez determinada la lista de funciones prohibidas, los equipos de proyecto deben usar archivos de encabezado, compiladores más recientes o herramientas de análisis de código para comprobar si hay funciones prohibidas en el código (incluido código heredado si procede) y reemplazarlas por alternativas más seguras.

Proceso 10 de SDL: análisis estático

El equipo de seguridad y los asesores de seguridad deben ser conscientes de las ventajas y desventajas de las herramientas de análisis estático y deben estar preparados para ampliar dichas herramientas con otras o con la revisión humana, según procede.

CUARTA FASE SDL: COMPROBACIÓN

En la ilustración 17 se observa que se ha pasado a la fase de inspección y adaptación de SCRUM en la subface de Demostración es necesario recalcar que los procesos once, doce, trece, catorce y quince se deben ejecutar antes de hacer una demostración al cliente esto para cumplir con los objetivos planteados en el proceso tres de SDL.

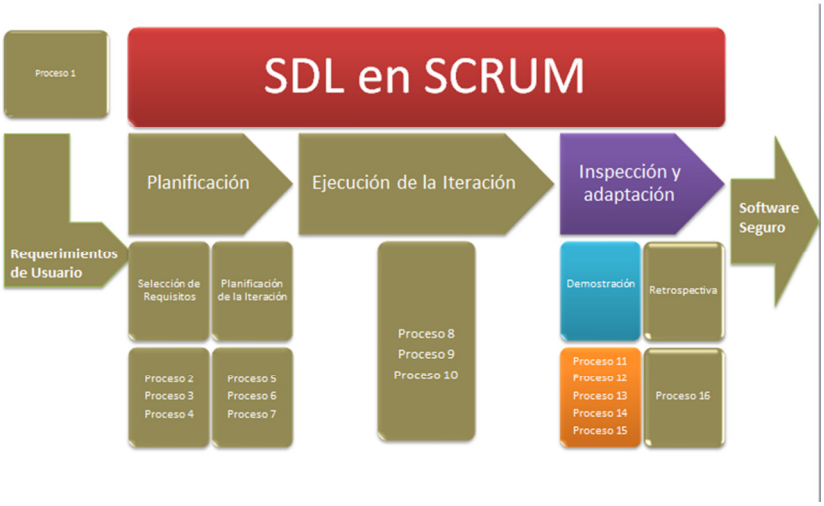

Ilustración 17 Inspección y adaptación- Demostración(SCRUM)
proceso 11,12,13,14,15 (SDL)

Proceso 11 de SDL: análisis dinámico de los programas

El proceso SDL usa herramientas en tiempo de ejecución como AppVerifier, junto con otras técnicas como pruebas de exploración de vulnerabilidades mediante datos aleatorios, para lograr los niveles de cobertura deseados de las pruebas de seguridad.

Proceso 12 de SDL: pruebas de exploración de vulnerabilidades mediante datos aleatorios

Este tipo de pruebas es una forma especializada de análisis dinámico que se usa para provocar errores en los programas introduciendo deliberadamente datos aleatorios o de formato incorrecto en una aplicación.

Proceso 13 de SDL: revisión de los modelos de riesgos y de la superficie de ataques

Por ello, es muy importante que se revisen los modelos de riesgos y la medición de la superficie de ataques de una aplicación una vez completado su código.

QUINTA FASE SDL: LANZAMIENTO

Proceso 14 de SDL: plan de respuesta a incidentes

- Un equipo de ingeniería sostenida identificado o, si el equipo es demasiado reducido para tener recursos de ingeniería sostenida, un plan de respuesta a emergencias en el que se identifica el personal de ingeniería, marketing, comunicaciones y administración que representa el primer punto de contacto en caso de una emergencia de seguridad.
- Contactos con capacidad para tomar decisiones y disponibilidad las 24 horas del día y los 7 días de la semana.
- Planes de servicios de seguridad para código heredado de otros grupos dentro de la organización.
- Planes de servicios de seguridad para código de terceros bajo licencia, incluidos nombres de archivo, versiones, código fuente, datos de contacto de terceros y permiso contractual para realizar cambios (si procede).

Proceso 15 de SDL: revisión de seguridad final

La revisión de seguridad final es una inspección deliberada de todas las actividades de seguridad realizadas con una aplicación de software antes de su lanzamiento.

El proceso 16 de SDL es ejecutado en la última fase de SCRUM como se muestra en la siguiente ilustración. Este proceso nos sirve para darnos cuenta de cómo nos fue y así realizar la mejora continua de nuestro ciclo de vida.

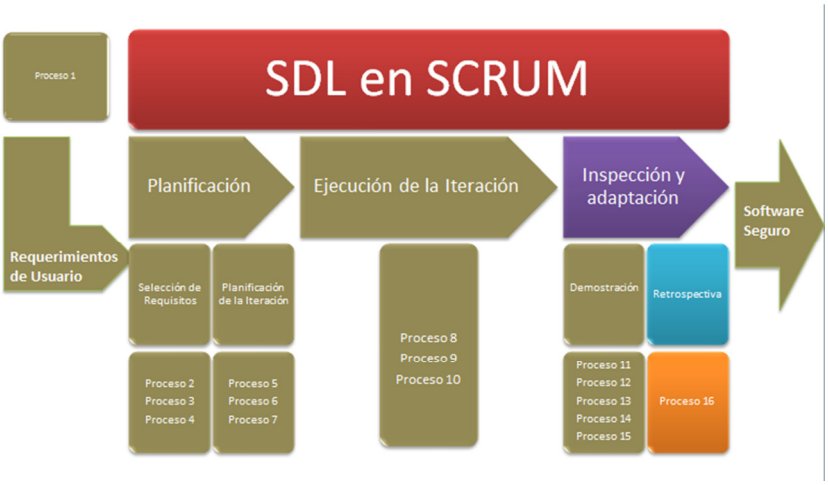

Ilustración 18 Inspección y adaptación->Retrospectiva (SCRUM) proceso 16 (SDL)

Procedimiento 16 de SDL: lanzamiento o archivado

Las versiones RTM (Release To Manufacturing) y RTW (Release To Web) dependen de la finalización del proceso SDL. [11]

CONCLUSIONES

El objetivo del presente documento es proporcionar un marco sencillo para la incorporación práctica de los procesos de seguridad en el proceso de desarrollo de software ágil. En éste documento se propone un modelo teórico para aplicar SDL en SCRUM, se describen los diversos procesos de desarrollo para la construcción de software seguro en metodologías ágiles y que lo podríamos considerar son de dominio público.

Cabe mencionar que si éstos procesos se combinan con las herramientas automatizadas y una buena orientación por parte de los directivos de la empresa incluyendo al líder de seguridad y líder de privacidad, podemos darnos cuenta que estos son los pasos necesarios para que una empresa pueda afirmar de modo objetivo que desarrolla software seguro.

Para la construcción de software seguro es necesario hacer una tarea ardua de concientización y divulgación de la información ya que los directivos de las empresas al momento de adoptar este método deben de estar bien informados de las bondades de utilizar un método de desarrollo de software seguro. Para que al momento de implementarlo toda la empresa a través de los directivos lo lleven a cabo con la mentalidad de que el adoptar éste método es beneficioso. Las empresas deben de estar conscientes de que implementar este método implica costo y esfuerzo en un principio mientras la empresa aprende pero cuando se llega a un nivel óptimo esto es automático. Además de los beneficios del retorno de inversión es muy satisfactoria y sobre todo la reputación de la empresa.

Como trabajo futuro es necesario la aplicación de éste ciclo de vida SDL en las fases de SCRUM para desarrollar un caso de estudio, también es necesario realizar una estandarización de este modelo. Para posteriormente si se obtienen resultados satisfactorios trabajar en la concientización de los directivos de las empresas de desarrollo y la divulgación de este tema. Se debe trabajar en un estudio de los atributos de calidad que le pegan a este estudio y la relación entre cada uno de ellos ya que por ser software seguro habrá algunos atributos de calidad que no se podrán implementar o que choquen entre sí.

Es necesario trabajar en herramientas que automaticen todos los procesos de éste método o la otra sería que los frameworks que se utilizan para el desarrollo de aplicaciones implícitamente contemplen funcionalidades para el desarrollo de software seguro.

Existe mucho trabajo futuro en temas de metodologías ágiles tales como control de calidad del software, estandarización de normas, la seguridad, la gestión del cambio, el crecimiento generalizado y la adopción de la metodología en la industria. Como podemos darnos cuenta es mucho trabajo lo que falta por hacer en metodologías ágiles.

BIBLIOGRAFÍA

1. Mishra, Sushma and Weistroffer, H. Roland. *"Issues with Incorporating Regulatory Compliance into Agile Development: A Critical Analysis"*. SAIS Proceedings. Paper 1, 2008.
2. George Wilson. *The Art of Throw-Away Test Automation Business agility requires disposable test assets*. Agile Record. The Magazine for Agile Developers and Agile Tester, free digital version, made in Germany, July 2010, issue3, pag 28.
3. Chaitanya Sudba Nakka. *Significance of Tools in Agile Software Development*. Agile Record. The Magazine for Agile Developers and Agile Tester, free digital version, made in Germany, July 2010, issue3, pag 61.
4. Emilio García García, Gestión de Vulnerabilidades de Seguridad: Automatización de la Revisión de Seguridad Perimetral. Comunicaciones de Telefónica I+D, publicación 34, Septiembre 2004.
5. David Gerr, *Are companies Actually Using Secure Development Life Cycles?Technology news, Published by the IEEE Computer Society, 2010.*
6. Security Development Lifecycle in Agile Scrum Methodology, AppLabs no doubt about it, 2010. AppLabs.com
7. Pandey S. K, Mustafa K. Rehman S. *CLASP and SDL Revisited*, Department of Computer Science.
8. Pravir Chandra, Software Assurance Maturity Model, A guide to building security into software development v 1.0. OWASP,
9.Gary McGraw, Ph.D., Brian Chess, Ph.D., & Sammy Migues,*Building Security In Maturity Model*. Cigital and Fortify, 2010.
10. Adrian Lane,Securosis. *Securosis'SSDL*. www.securosis.com, 2010.
11. Microsoft, Security Development Lifecycle, Implementación Simplificada del proceso SDL de Microsoft, www.microsoft.com/sdl, 2010.