Universidad de Buenos Aires

Facultades de Ciencias Económicas, Ciencias Exactas y Naturales e Ingeniería

Maestría en Seguridad Informática

Tesis

Título

Incorporación de prácticas de seguridad en metodologías Ágiles

Autor: Ing. Jorge Ezequiel Bo

Director de Tesis: Ing. Hugo Pagola

2013

Cohorte 2010

Declaración Jurada de origen de los contenidos

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Tesis vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación

Nacional e Internacional de Propiedad Intelectual.

Nombre y apellido: Jorge Ezequiel, Bo

D.N.I: 28234627

FIRMADO

Resumen

Las metodologías Ágiles, basan su éxito en dar respuesta a desarrollos en ambientes con requerimientos muy cambiantes. Esto le permite, mediante el desarrollo de ciclos cortos, obtener *software* funcionando “en demanda” de alta calidad. Sin embargo, la realidad demuestra que muchos equipos, en su afán de ser Ágiles fallan en el intento obteniendo *software* que si bien responde a los requerimientos del cliente lo hace con una calidad muy baja en cuanto a arquitectura y seguridad.

La ausencia de calidad en seguridad, se ve reflejada no solo en la cantidad de defectos con la que el *software* es liberado, sino en la ausencia de prácticas y recomendaciones de seguridad aplicadas durante el desarrollo bajo la metodología adoptada.

La presente tesis analiza el concepto de seguridad en metodologías Ágiles y los obstáculos que se presentan en el intento de construir seguridad en forma progresiva.

Mediante una investigación de campo y con la ayuda de la teoría, se demuestra en esta tesis que los obstáculos principales se deben a las dificultades que presentan las organizaciones para adaptarse a las metodologías Ágiles y a la falta de construcción de una mentalidad orientada a la seguridad en los equipos de desarrollo. Se presenta una recomendación para la construcción progresiva de seguridad en ambientes donde la adopción inmediata de un marco de desarrollo de seguridad no es adecuada.

Palabras clave: Ágil, SDL, Scrum, XP, software security, building security in.

Índice de contenidos

Prólogo

Agradezco a mis compañeros de equipo la buena predisposición para realizar las entrevistas y ser observados en sus ambientes de trabajo.

Índice de Acrónimos y Abreviaturas

TERMINO	***DEFINICION***
BSIMM	*Building Security In Maturity Model*
CERT/CC	*Computer Emergency Response Team/Coordination Center*
CWE/SANS	*Common Weekness Enumeration/ SysAdmin Audit, Networking and Security Institute*
MSDL	*Microsoft SecurityDevelopment Lifecycle*
MSF	*Microsoft Solution Framework*
OpenSAMM	*Open Software Assurance Maturity Model*
OWASP	*Open Web Application Security Project*
RUP	*Rational Unified Process*
SDL	*Security Development Lifecycle*
SSF	*Software Security Framework*
SSr.	Semi-Señor
Sr.	Señor
TI	Tecnología de Información
TDD	*Test Driven Development*
VPN	*Virtual Private Network*
XP	*Extreme Programming*

1. Introducción

La seguridad en los sistemas y redes de computación en los últimos años se ha visto comprometida principalmente por la calidad en materia de seguridad del *software* ejecutándose en los dispositivos que los conforman. Así, el surgimiento y la adopción masiva de dispositivos móviles, junto a la migración de aplicaciones para que operen en la nube (*cloud computing*) y el aumento en el intercambio de información mediante las redes sociales, ha incrementado viejos y abierto nuevos vectores de ataques originados en las vulnerabilidades de seguridad existentes en el *software* presente en estos dispositivos y aplicaciones.

Durante los años 1995 a 2005 se ha producido un incremento en las vulnerabilidades reportadas al CERT/CC [1] independientes de los mecanismos de seguridad utilizados para defensa. Reportes de vulnerabilidades [2] durante el año 2011, señalan que ha sido un año de surgimiento de vulnerabilidades sofisticadas, y pronostica un incremento de vulnerabilidades en dispositivos móviles originadas en el *software* de aplicación. Frente a esta necesidad de desarrollar *software* más seguro, surge la necesidad de la incorporación de prácticas de seguridad como parte de la metodología de desarrollo ya sea que se utilicen metodologías tradicionales o Ágiles.

Como respuesta a esta necesidad, varias iniciativas para la construcción de seguridad en metodologías de desarrollo han sido planteadas, ya sea en forma de estándares, marcos o recomendaciones de seguridad.

1.2 Antecedentes

A continuación se exponen los trabajos más relevantes que sirven como antecedentes de este trabajo.

BSIMM

El “modelo de madurez para la construcción de seguridad” BSIMM [3], presenta un estudio sobre las prácticas de seguridad utilizadas en muchas

de las iniciativas de seguridad existentes. Mediante la cuantificación de prácticas pertenecientes a iniciativas de seguridad de diferentes organizaciones describe las similitudes y las variaciones que las hacen únicas. Presenta un marco llamado “marco de seguridad del *software*” que permite unificar los términos presentes en las distintas iniciativas de seguridad analizadas y generar un vocabulario común para su comparación. Este marco describe doce prácticas organizadas en cuatro dominios, con tres prácticas de seguridad por dominio y ciento once actividades para la implementación de las prácticas; todas ellas abstracciones de los diferentes elementos encontrados en otras iniciativas. Este estudio permitió la construcción de un lenguaje común para utilizar en este trabajo y entender otros marcos de seguridad existentes.

OpenSAMM

Luego, el “modelo de madurez para el aseguramiento del *software*” OpenSAMM [4], especifica un marco de seguridad para ayudar a las organizaciones a formular e implementar una estrategia de seguridad en el *software* según los riesgos que está decidida a enfrentar. Se encuentra compuesta de cuatro unidades funcionales que categorizan operaciones que toda organización que utiliza o desarrolla *software* debe realizar en algún grado. En cada unidad funcional se encuentran tres prácticas de seguridad, las cuales definen actividades de seguridad relacionadas al aseguramiento de seguridad de la unidad funcional en la que se encuentran contenida. Su evolución es medida por tres niveles de madurez determinados por la implementación o no de ciertas actividades presentes en cada uno de ellos. Este modelo no solo permite realizar una evaluación interna para medir el nivel de seguridad, sino que ofrece guías predefinidas para definir un plan de mejora basado en aquellas prácticas de seguridad importantes según el tipo de organización a analizar.

A diferencia del marco anterior el cual es utilizado como fines de comparación con otras iniciativas, este permite realizar una evaluación interna de madurez y utilizar alguno de las guías definidas para comenzar a construir seguridad en la organización.

Su lectura permitió identificar aquellas prácticas de seguridad relevantes y un posible plan de mejora en cuanto a seguridad para una organización de *outsourcing* como la que se encuentra el proyecto presentado en este trabajo.

MSDL

El "marco de seguridad de *Microsoft*" MSDL [5], propone soluciones para la incorporación de prácticas de seguridad agnósticas a la metodología de desarrollo, ya sea Ágil u orientada al plan. A diferencia de los marcos vistos anteriormente, este se encuentra enfocado directamente en las etapas del ciclo de desarrollo del *software* y no en la construcción de seguridad en una organización que usa o desarrolla *software* en general, por lo cual su aplicación es bastante directa en respecto de los anteriores.

Se basa en tres conceptos básicos: formación, mejora continua de procesos y responsabilidad. Para las orientadas al plan, propone una incorporación de prácticas por áreas que se corresponden con las fases del ciclo de desarrollo del *software*, junto a cuatro niveles de madurez para los procedimientos de estas áreas según la madurez de la organización. Por otro lado, para las metodologías Ágiles, al no haber fases diferenciadas, propone una incorporación progresiva de las mismas prácticas en diferentes momentos del ciclo de desarrollo; al inicio del proyecto, en cada iteración y a lo largo de varias iteraciones.

Este marco nos acercó una primera idea acerca de la organización de las prácticas de seguridad alrededor de una metodología Ágil y nos dio paso al análisis de las dificultades que podrían encontrarse en su aplicación.

Otros artículos

Una publicación de SAFECODE [6] provee al practicante Ágil con una lista de historias de usuario y tareas basadas en seguridad que pueden ser introducidas en forma directa en sus ambientes Ágiles. Estas historias y sus tareas surgen de un esfuerzo en la traducción del top 10 de OWASP y el top 25 de los errores más peligrosos enumerados en CWE/SANS a ambientes Ágiles para facilitar su adopción en estos ambientes. Se espera que

mediante la introducción de estas historias predefinidas los equipos Ágiles comiencen a incorporarlas como prácticas comunes de la metodología. Esta publicación nos permitió entender el hecho de utilizar historias de usuario como nexo entre las prácticas tradicionales de seguridad y los ambientes Ágiles.

Finalmente [7], provee un análisis desde el punto de vista práctico sobre la seguridad en proyectos Ágiles. Para ello se basa en el reporte de hallazgos que surgen de entrevistas a desarrolladores Ágiles, los cuales permiten extender trabajos teóricos existentes de forma práctica. Concluye manifestando la importancia del involucramiento del cliente, la capacitación en seguridad a desarrolladores y en la mejora del proceso de construcción de seguridad continua. Este trabajo nos permitió justificar la necesidad de aportar conocimiento práctico sobre la temática de construcción de seguridad en los diferentes aspectos de que relacionan la seguridad con las metodologías Ágiles.

1.3 Planteamiento del problema

Las iniciativas de seguridad revisadas anteriormente promueven que una organización presente un cierto grado de estructura formal, de manera que su aplicación permita evolucionar la madurez de sus procesos. Las características presentes en organizaciones que son Ágiles reflejan procesos de desarrollo informales, con un bajo nivel de madurez y unidades funcionales no existentes o poco definidas, haciendo difícil la adopción de alguna de las iniciativas presentadas anteriormente

Estas metodologías Ágiles, utilizan prácticas de desarrollo que tienden a mejorar la gestión de requerimientos cambiantes, de forma que permita captar todas las funcionalidades requeridas por el cliente y generar valor agregado en cada entrega. Estas no escapan a la falta de aplicación de prácticas de seguridad para lograr funcionalidades seguras, lo que origina la siguiente pregunta de investigación:

¿Cuáles son los obstáculos que dificultan la presencia de prácticas de seguridad en proyectos ágiles?

En busca de la respuesta a esta pregunta de investigación, nos proponemos realizar los siguientes aportes al campo temático:

- Fomentar y motivar el uso de prácticas de seguridad en las metodologías Ágiles.
- Analizar la influencia de las prácticas Ágiles en la seguridad de un proyecto.
- Identificar razones que llevan al no uso de prácticas de seguridad en proyectos bajo estas metodologías.
- Proponer una alternativa para la construcción de seguridad en metodologías Ágiles, la cual consiste el uso de prácticas Ágiles no solo como punto de entrada a la construcción de seguridad, sino como punto de transición hacia la adopción de iniciativas más complejas en el futuro.

El trabajo se encuentra limitado por el siguiente alcance:

- Se analizarán proyectos correspondientes a una *software factory*.
- Se analizarán proyectos guiados por metodologías Ágiles como *Scrum* y XP en sus versiones ad-hoc.
- Se seleccionarán proyectos que no utilicen prácticas de seguridad específicas en sus desarrollos.
- Se utilizarán equipos con desarrolladores multidisciplinarios.
- Se estudiará la adopción de aquellas prácticas Ágiles orientadas a reducir los problemas de seguridad presentes en el *software*.
- Se analizaran proyectos de tipo comercio electrónico o redes sociales donde la seguridad juega un rol de importante.

1.7 Organización del trabajo

Este trabajo se encuentra dividido en los siguientes capítulos:

Capítulo 1: introducción del trabajo, antecedentes, alcance, problemática, objetivos hipótesis.

Capítulo 2: presentación del método de trabajo de esta tesis.

Capítulo 3: introducción a los fundamentos que rigen la construcción de seguridad en el *software*.

Capítulo 4: análisis de las metodologías Ágiles y su rol ante la construcción de seguridad.

Capítulo 5: análisis de las prácticas Ágiles y su impacto en la seguridad.

Capítulo 6: conclusiones y recomendaciones.

2. Método de trabajo

2.1 Introducción

En este capítulo presentaremos el método de trabajo seguido a lo largo de esta tesis. Comenzaremos por describir el contexto en cual se desarrollara el mismo, las unidades de análisis y categorías que surgieron del análisis de campo y el procedimiento llevado a cabo para identificar cada una de ellas.

2.2 Contexto

El estudio realizado se lleva a cabo sobre equipos de desarrollo de *software* que utilizan metodologías Ágiles para su desarrollo. Los equipos que se analizaron pertenecen a una *software factory* que ofrecen desarrollos *offshore* a sus clientes en la provincia de Buenos Aires, Argentina. Este *software factory* cuenta con aproximadamente 200 desarrolladores con por lo menos 3 años de experiencia en el campo de trabajo. Donde se encuentran los siguientes señoritis de semi-señor, señor, líderes técnicos, y *manager* de soluciones organizados por oficina y región.

Los equipos de trabajo se ubican en zonas de mesas comunes denominadas islas por los integrantes del equipo. Cada equipo se encuentra conformado con por lo menos tres desarrolladores, un *tester* y un líder técnico, este último posee las responsabilidades organizar, asignar y supervisar las tareas que el equipo debe realizar. Además, cada equipo cuenta con un *manager* de soluciones, quien entra en acción para resolver cualquier tipo de conflicto que pueda poner en peligro la salud del proyecto.

Las metodologías Ágiles utilizadas en sus desarrollos abarcan a *Scrum y* XP*,* sin embargo, hemos observado que una versión adaptada de *Scrum* es la implementación más usada por la mayoría de los equipos y que las prácticas sugeridas por la metodología no son aplicadas en su totalidad, lo que hemos llamado con el nombre de *Scrum ad-hoc*.

El tipo de proyectos involucrados en el estudio son de tipo comercio electrónico y redes sociales, para los cuales se debería contar con una base importante de seguridad por gestionar estos información sensible de las personas.

2.3 Procedimiento

Con la finalidad de entender los obstáculos que intervienen en la construcción de seguridad en metodologías Ágiles, se ha seguido un diseño del trabajo de tipo investigación-acción, donde los pasos aplicados involucran:

1) Observación de equipos de desarrollo donde sus proyectos requieran un alto nivel de construcción de seguridad.
2) Recolección de evidencia mediante el uso de observaciones y entrevistas.
3) Identificación de unidades de análisis, categorías y temas que determinarán el contexto y la dirección del trabajo.
4) Análisis de los temas identificados.
5) Selección de equipo para experimentación.
6) Presentación de resultados.

Las entrevistas fueron realizadas en forma personal con cada uno de los entrevistados y sus resultados registrados en un medio electrónico. El análisis de los datos es presentado a lo largo de los capítulos del trabajo en forma narrativa ordenados en relación a la literatura correspondiente a la temática abordada.

2.4 Selección del equipo de desarrollo

De los equipos observados se ha seleccionado un equipo de desarrollo Ágil que tiene 2 años de experiencia trabajando juntos en la implementación de plataformas de comercio electrónico con las siguientes características:

1) Es un equipo multifuncional.
2) Existen roles definidos.
3) No existen roles de seguridad.
4) No se aplican prácticas orientadas a mejorar la construcción de seguridad.
5) No poseen conocimientos de seguridad específicos.

La Tabla I**¡Error! No se encuentra el origen de la referencia.** muestra la configuración de este equipo.

Tabla I. Equipo de trabajo seleccionado

Nombre	Equipo A	
Metodología de desarrollo	*Scrum Ad-hoc*	
Características	Proyecto de *outsourcing* Parte de un equipo remoto localizado en USA	
Rubro	Comercio Electrónico	
Integrantes		
Código/Seudónimo		**Roles**
M1		Manager
TL1		Líder Técnico
DEVS1		Desarrollador Sr.
DEVSS1		Desarrollador SSr.
TSS1		Tester SSr.
PO1 (Localizado en el cliente)		Product Owner

2.5 Unidades de análisis y categorías encontradas

Hemos comenzado a estudiar los obstáculos presentes en la construcción de seguridad mediante la recolección de datos en el ambiente de trabajo del equipo elegido como objeto de estudio. La recolección de datos ha sido llevada a cabo mediante observaciones y entrevistas enfocadas a indagar sobre el conocimiento y experiencia que se tiene sobre la construcción de seguridad en un proyecto Ágil (ver Anexo I) obteniendo como resultado las siguientes unidades de análisis y categorizaciones que trataremos a continuación.

2.5.1 Unidad de análisis problema de seguridad

La primera unidad de análisis surgida se denomina como el “problema de seguridad”. En la

Tabla II pueden observarse algunas de las instancias identificadas correspondientes a esta unidad de análisis.

Tabla II. Instancias de unidades de análisis “problema de seguridad” observadas

Recolección de datos	Observaciones en equipo A
Unidad de análisis	Problema de seguridad
Contexto	Entrevistas y observaciones realizadas a los desarrolladores del equipo A en su ambiente de trabajo
Instancias	
Punteros nulos	
Manejo incorrecto de excepciones	
Mecanismos de autenticación incorrectos	
Incorrecta implementación de mecanismos de autorización.	
Inyección de SQL	
Inyección de scripts maliciosos (*JavaScript)*	
Mecanismos de encriptación obsoletos	
Falta de información de registro de actividad del *software (logging)*	
Pérdida de memoria (*Memory leaks*)	
Configuración incorrecta de accesos a los entornos y VPNs	
Manejo incorrecto de errores	
Falta de validación de valores de entrada y salida	
Mal resguardo de la información sensitiva	

La importancia en la identificación de esta unidad de análisis radica en la posibilidad de agrupar a sus instancias en categorías que nos permitan identificar, crear y proponer prácticas y modelos orientados a mitigar sus efectos durante la construcción del software; agregando de esta forma un primer avance en pos de la construcción de seguridad y la identificación de

obstáculos existentes.

Como resultado del proceso de categorización cualitativo de los valores arrojados por las observaciones realizadas, hemos creado la siguiente categorización tomando como base el origen de las mismas como puede apreciarse en la Tabla III**¡Error! No se encuentra el origen de la referencia.**.

Tabla III. Categorización de la unidad de análisis problema de seguridad

Unidad de análisis	**Categoría**	**Sub-categoría**
Problema de seguridad	Origen	Implementación Diseño

Definiendo cada categoría como se señala a continuación:

- **Origen**: clasifica a los problemas de seguridad según su fuente de creación.
- **Implementación**: agrupa a aquellos problemas conocidos como *bugs*. Según [8], estos son determinados por errores de implementación que pueden ser fácilmente encontrados y solucionados a simple vista o con el uso de algún analizador de código estático. Ejemplos de estos problemas son un puntero nulo, una pérdida de memoria o una validación incorrecta de parámetros de entrada/salida a funciones y métodos.
- **Diseño**: agrupa problemas originados en un nivel más profundo del *software*, generalmente a nivel de diseño y arquitectura, y según [8], son conocidos como fallas y pueden permanecer ocultas hasta que este se encuentre lo suficientemente comprometido como para revelarlas. Estos deben ser atacados mediante la aplicación técnicas de análisis de riesgos y revisiones frecuentes en el diseño y la arquitectura del sistema a desarrollar. Un ejemplo de este problema es un mal manejo de errores.

Es conveniente relacionar esta categorización encontrada con el planteo realizado en [8] sobre un problema de seguridad como puede

observarse en la Tabla IV, la cual muestra la relación existente entre lo que se define como un *bug* y una falla, ambos defectos, y un riesgo.

Tabla IV. Clasificación de problemas de seguridad

	Defecto	
	Falla ej: Gestión de errores incorrectos	**Riesgo**
Problema de Seguridad		
	Bug ej: Puntero nulo, pérdida de memoria	

Donde se entiende por:

- **Defecto**: hace referencia a dos tipos de vulnerabilidades encontradas en el *software*, las de diseño y las de implementación.
- **Falla**: es un problema en un nivel más profundo, presente o ausente a nivel de diseño en el *software.*
- ***Bug***: es una vulnerabilidad de implementación en el *software* que puede existir y nunca ser ejecutado. Se presenta en forma de un error simple de programación y puede ser fácilmente encontrado y solucionado.
- **Riesgo**: se refiere a la probabilidad de que un defecto impacte en el correcto funcionamiento de un sistema.

2.5.2 Unidad de análisis estrategia de seguridad

Luego la segunda unidad de análisis que hemos identificado se denomina “estrategia de seguridad”. En la Tabla V pueden observarse algunas de las instancias identificadas correspondientes a esta unidad de análisis.

Tabla V. Instancias de unidades de análisis “estrategia de seguridad” observadas

Recolección de datos	Observaciones en equipo A
Unidad de análisis	Estrategias de seguridad
Contexto	Entrevistas y observaciones realizadas a los desarrolladores del equipo A en su ambiente de trabajo
Instancias	

Aplicación de SDL
Análisis de seguridad por revisiones de código
Construcción de mecanismos de seguridad
Desarrollo guiado por pruebas (TDD)
Prestar atención a lo que se hace
Definición de requerimientos de seguridad

En esta observamos la aparición de instancias esperadas, como la definición de requerimientos y las revisiones de código, mientras que algunas no esperadas como la mención de la existencia e intento de aplicación de un SDL.

Esta unidad de análisis tiene su importancia no solo en la posibilidad de aplicar contramedidas para evitar posibles problemas de seguridad en la construcción del *software*, sino que nos permitirá observar las dificultades y obstáculos que se presentan al momento de su adopción.

Como resultado del proceso de categorización cualitativo de los valores arrojados por las observaciones realizadas, hemos creado la siguiente categorización tomando como base el origen de las mismas como puede apreciarse en la Tabla VI.

Tabla VI. Categorización de la unidad de análisis “estrategia de seguridad" observadas

Unidad de análisis	**Categoría**	**Sub-categoría**
Estrategia de seguridad	Iniciativa	Prácticas de seguridad
		Marcos de seguridad

Definiendo cada categoría como se señala a continuación:

- **Iniciativa**: clasifica a las estrategias según el medio o teoría que utilice para construir seguridad.
- **Prácticas de seguridad**: hace referencia a la aplicación de distintas prácticas de mitigación relacionadas o no entre sí, pero que no siguen un orden estipulado o no son parte de proceso. Ejemplo de ellas son las revisiones de código, aplicación de mecanismos de seguridad para verificación de invariantes (*guard conditions*), programación

defensiva.

- **Marcos de seguridad**: son aquellas prácticas que son aplicadas como resultado de la utilización de un modelo o marco que permite construir seguridad dentro del *software*, el cual define reglas para la aplicación de cada práctica en cada etapa del ciclo de desarrollo.

2.5.3 Unidad de análisis metodología Ágil

Luego, la próxima unidad de análisis que hemos notado es la "metodología Ágil" utilizada. La Tabla VII muestra las instancias de esta unidad recolectada mediante las observaciones y entrevistas.

Tabla VII. Instancias de unidades de análisis "metodología Ágil" observadas

Recolección de datos	Observaciones en equipo A
Unidad de análisis	Metodología ágil
Contexto	Entrevistas y observaciones realizadas a los desarrolladores del equipo A en su ambiente de trabajo
Instancias	
Scrum	
Scrum modificado (*Scrum ad-hoc*)	
Kanban	
Prácticas de XP	
Orientada a pruebas	

Esta unidad de análisis toma su importancia del hecho de permitirnos definir el escenario donde se trabajará para identificar los obstáculos presentes en la construcción de seguridad y del cual se determinara el contexto de aplicación de las estrategias de mitigación y su efectividad. Además, es de interés la necesidad de observar cual es la visión y cómo se construye seguridad dentro de las aplicaciones desarrolladas bajo las mismas en el campo de trabajo. Como resultado del proceso de categorización cualitativo de los valores arrojados, la

Tabla VIII muestra las categorías encontradas.

Tabla VIII. Categorización de la unidad de análisis metodología Ágil

Unidad de análisis	Categoría	Sub-categoría
Metodología Ágil	Aplicación	Ad-hoc Metodológica

Donde, definimos:

- **Aplicación:** clasifica a las metodologías en cuanto a su grado de rigurosidad en la aplicación.
- **Ad-hoc**: hace referencia al uso de una metodología Ágil modificada, donde solo algunas de sus prácticas son aplicadas según la necesidad del equipo de desarrollo y del proyecto.
- **Metodológica:** hace referencia al uso lo de aquellas metodologías Ágiles en forma estricta durante el desarrollo. La mayoría de las prácticas recomendadas son aplicadas a lo largo del proyecto.

Dentro de este análisis han surgido los siguientes temas relacionado las diferentes categorías obtenidas hasta el momento:

- **Visión de la seguridad en Ágil**: donde se hace referencia al concepto que se tiene cuando hablamos de seguridad en cuanto a este tipo de metodología.
- **Rol de la seguridad en Ágil**: intenta explicar la introducción de este rol en las mismas.

Finalmente, al comparar las unidades de análisis de metodología Ágil y estrategia de seguridad, notamos que existen prácticas dentro de Ágil que pueden ayudarnos a reducir los problemas de seguridad a la vez que facilita la incorporación de prácticas de seguridad específicas. Su importancia radica en el estudio de la seguridad en las metodologías Ágiles, por el solo hecho de usar sus prácticas características. La Tabla IX muestra esta última categorización.

Tabla IX. Categorización de la unidad de análisis práctica Ágil

Unidad de análisis	Categoría	Sub-categoría

Práctica Ágil	Presencia de seguridad	Inherente Segura No segura

Donde, definimos:

- **Presencia de seguridad:** clasifica a las prácticas Ágiles de acuerdo al grado de facilidad con que otorgan seguridad al desarrollo.
- **Seguridad inherente**: hace referencia al uso de prácticas Ágiles, donde su uso puede ayudar a incorporar seguridad y servir como un punto de apalancamiento para la introducción de marcos de seguridad complejos.
- **No segura:** incluye aquellas prácticas Ágiles que no contribuyen notablemente a una mejora en la construcción de seguridad.
- **Segura**: incluye aquellas prácticas Ágiles orientadas a construir seguridad.

2.6 Conclusión

En este capítulo hemos presentado el contexto de estudio, las unidades de análisis, categorías y temas que nos ayudaron a organizar el material recolectado y que determinaron el abordaje dado a la problemática propuesta en esta tesis. Estas nos permitieron generar explicaciones para el problema y la hipótesis planteadas. En el capítulo siguiente cubriremos los problemas de seguridad existentes al momento de construir seguridad dentro del *software*.

3. Estableciendo las bases de la seguridad

3.1 Introducción

En este capítulo presentaremos los fundamentos que rigen la construcción de seguridad en el *software*, los cuales nos permitirán establecer las bases para un mejor entendimiento de los tópicos tratados con posterioridad en esta tesis.

Comenzaremos por presentar un análisis de las fuentes que dificultan la erradicación de los problemas de seguridad y de la construcción de la misma en los proyectos, junto a alternativas que erróneamente se utilizan para construir seguridad en el *software* hoy en día.

3.2 Construcción de la seguridad en el software

Al estudiar la necesidad de construir seguridad en el *software*, comenzamos por investigar el origen de la misma, y encontramos evidencia en [9] que verifica que la mayoría de los ataques exitosos resultan de la explotación de configuraciones incorrectas del ambiente de despliegue del software y de problemas conocidos y no tratadas del mismo; muchas de las cuales son introducidas en las fases de diseño y codificación. Estos problemas no solo producen un funcionamiento incorrecto del software en cuanto a lo que se espera del mismo, sino que pueden ramificar en vulnerabilidades de seguridad difíciles de detectar.

La introducción de prácticas de seguridad con el objetivo de construir seguridad en el software es un complemento a las buenas prácticas de ingeniería, que permite comenzar a pensar no solo en construir la seguridad en etapas tempranas del ciclo de vida del desarrollo, sino producir software que pueda obstruir ataques en forma proactiva y responder de manera correcta en presencia de comportamientos inesperados.

3.3 Factores que dificultan la construcción de seguridad

En los últimos años hemos visto muchos avances en materia de tecnologías, como nuevos marcos de desarrollo, lenguajes de programación y sistemas operativos, que si bien han contribuido a la evolución del desarrollo de *software*, no lo han hecho de la misma manera en cuanto a la construcción de seguridad.

Dentro de los factores que afectan la evolución de la construcción de seguridad [10] encontramos los siguientes:

1. **La conectividad**: hace referencia a la facilidad con la cual puede ser alcanzado un sistema para ser atacado, atribuyendo esto al incremento de computadoras conectadas en *Internet* y la dependencia cada vez mas de sistemas informáticos para el trabajo.

2. **La extensibilidad**: se refiere a la capacidad del *software* de ser extendido mediante actualizaciones de forma que permita a su funcionalidad evolucionar incrementalmente.

3. **La complejidad**: se refiere a la cantidad de líneas de código que contiene el *software*, al uso de lenguajes de programación que no contribuyen a la prevención de errores simples, y a la creciente dependencia de nuestro *software* respecto de librerías y módulos externos en los cuales debemos confiar.

4. **La adopción**: se refiere a la expansión de los dominios del *software* en la vida privada de las personas. El alto intercambio de información mediante redes sociales y el uso masivo de dispositivos móviles, no hacen más que incrementar la exposición del usuario a los peligros de la *Web*, derivando esto en una mayor probabilidad a sufrir ataques relacionados con el robo de información sensible [11].

5. **La presión del mercado**: está determinada por la alta competitividad del software orientado hacia el consumo masivo, el cual demanda tiempos de desarrollo demasiado rápidos donde lo importante es entregar el *software* y luego iniciar nuevos proyectos con el fin de solucionar los defectos encontrados, ya sean en requerimientos funcionales o en problemas de seguridad.

6. **La falta de mano de obra:** a la alta rotación de los recursos se le suma la falta de mano de obra calificada y/o entrenada en seguridad contribuyendo al incremento de los riesgos en seguridad.

Hay que destacar que hoy en día las nuevas versiones de navegadores *Web* y sistemas operativos han incrementado notablemente las verificaciones de seguridad al momento de realizar actualizaciones o instalaciones de nuevos módulos reduciendo la capacidad de ataques por este mecanismo.

2.4 Prácticas de seguridad tradicionales

Tradicionalmente existen muchas prácticas de seguridad que se utilizan erróneamente con el objetivo de crear *software* más seguro. Entre ellas encontramos las siguientes prácticas que aunque ineficaces son aún aplicadas por la mayoría de los equipos de desarrollo:

- **La programación defensiva**: consiste en agregar código para

validar los supuestos del programador [12]. Si bien ayuda a encontrar y diagnosticar errores en el código no garantiza la seguridad del mismo, ya que esto no anticipa todas las formas en que el *software* puede ser explotado. Este punto es de mucha importancia ya que es una de las prácticas más fáciles y rápidas de aplicar por programadores sin conocimientos en seguridad, y para el cual aportan a la creación de un código fuente más complejo y con poco valor agregado en cuanto a seguridad.

• **El abuso de los mecanismos de seguridad**: estos se refieren a los mecanismos de seguridad utilizados en la mayoría del *software* existente para cumplir con los diferentes aspectos de control de acceso, autorización y monitoreo de eventos junto a cifradores criptográficos. Estos contribuyen a la formación de la visión de la seguridad [13] por parte de los desarrolladores como único medio de aplicación de seguridad en el código, y así olvidar el resto de las buenas prácticas y recomendaciones existentes. Sin embargo, la mayoría de los casos de fallos de seguridad no están relacionados con vulnerabilidades encontradas en estos mecanismos, sino con las vulnerabilidades encontradas en las características funcionales del sistema que permiten su posterior explotación. Es por ello, que los esfuerzos en materia de seguridad deben estar dirigidos hacia la identificación de problemas en el código no relacionados con el uso de mecanismos de seguridad.

• **El proceso de aseguramiento de la calidad**: hemos encontrado en [14] evidencia que asegura que la seguridad en el *software* no puede ser mejorada introduciendo mejoras en el proceso de calidad. La mayoría de los esfuerzos en esta etapa son dirigidos a encontrar diferencias entre lo especificado por los requerimientos y lo implementado, lo que permite encontrar problemas de seguridad relacionados con la violación o incumplimiento de estos requerimientos o con mecanismos de seguridad común, pero no problemas de seguridad originados en funcionalidad construida y no especificada, como muestra la Figura 1 [15]. Esto último es algo observado en la mayoría de los equipos de desarrollo que no poseen experiencia en materia de seguridad y es algo que se incrementa a medida que pasa el tiempo y no se tiene conciencia de su importancia.

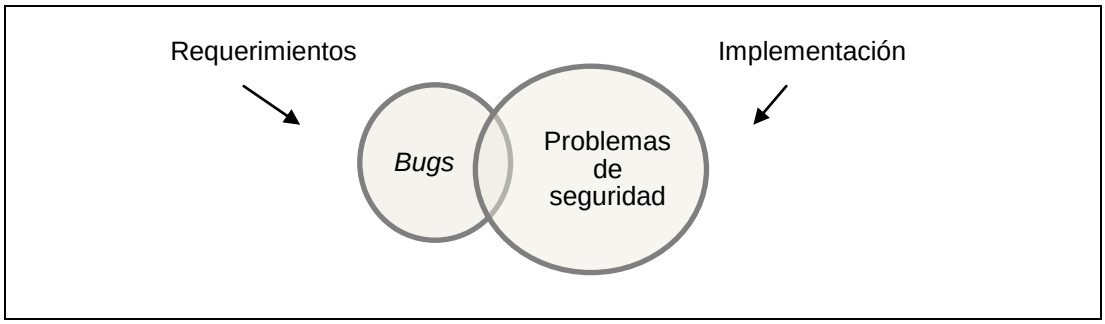

Figura 1. Requerimientos implementados y problemas de seguridad

• **Las pruebas de intrusión**: son realizadas una vez que el *software* es finalizado y muchas veces como única medida de seguridad. Este tipo de pruebas de caja negra, utilizan un conjunto de herramientas automatizadas que permiten encontrar problemas de seguridad comunes a todo software y no según sus características propias. Encontramos en [15] evidencia que sugiere que luego de la fecha de liberación del *software*, atacantes y defensores se encuentran en iguales condiciones de realizar un análisis sobre el mismo, sin embargo, la cantidad de atacantes y horas disponibles superan ampliamente a sus pares defensores. Esto les permite a los primeros encontrar mayor cantidad de vulnerabilidades a ser explotadas en menor cantidad de tiempo. Es de notar que hoy en día existen herramientas sofisticadas capaces de reducir bastante esta brecha entre ambas las partes interesadas.

3.5 Conclusión

Este capítulo presentó el concepto de construcción de seguridad en el *software*, junto a los problemas que enfrenta su búsqueda. Esto nos permitirá no solo entender el vocabulario utilizado, sino además facilitar el enfoque que se le ha dado a este trabajo en capítulos posteriores.

En el capítulo siguiente presentaremos los aspectos generales de una metodología Ágil.

4. Las metodologías Ágiles

4.1 Introducción

En este capítulo introduciremos a las metodologías Ágiles y el papel que juega ante la construcción de seguridad en el desarrollo de *software*.

Comenzaremos por caracterizar a las metodologías Ágiles, analizar la importancia de su adopción, los roles y características propias y como se ven afectados con respecto a la seguridad. Finalmente, analizaremos la idea establecida y el concepto de la seguridad que se tiene en este tipo de metodologías en el proyecto de estudio presentado en este trabajo.

4.2 ¿Qué es una metodología Ágil?

Para dar una definición exacta de lo que significa una metodología Ágil, no existe mejor forma que remitirnos al manifiesto Ágil el cual destaca los siguientes postulados:

> "**individuos e interacciones** sobre procesos y herramientas,
> ***software* funcionando** sobre documentación extensiva,
> **colaboración con el cliente** sobre negociación contractual y **respuesta ante el cambio** sobre seguir un plan" [16].

El movimiento Ágil no solo remarca la valoración de los elementos de la izquierda (aquellos en negrita) sobre los de la derecha, sino que especifica doce principios que las caracterizan [17]. Entre ellos encontramos, la priorización de la satisfacción del cliente, mediante la entrega temprana y continua del *software*, la aceptación de cambios constantes en los requerimientos, el trabajo y comunicación del equipo cara a cara, la medida del progreso mediante *software* funcionando, y arquitecturas de desarrollo simples junto a diseños intencionales y emergentes [18]. Por otro lado, en las metodologías tradicionales, donde los elementos de la derecha son predominantes, como en RUP [19] , Cascada [20] o MSF [21]; observamos características opuestas como el énfasis en contar con la planificación total y el trabajo detallado, mediante la documentación de artefactos y el uso de herramientas establecidas, la existencia de roles con responsabilidades definidas, y canales de comunicación rígidos que dificultan la adaptación adecuada a entornos donde los requisitos no pueden predecirse o son cambiantes. Así, se pone de manifiesto que la forma tradicional de construir seguridad en el *software* debe ser adaptada para encajar en estas metodologías Ágiles o bien nuevas corrientes deben surgir.

4.3 Adopción en el mundo

La adopción de metodologías Ágiles como metodología de desarrollo de *software* [22] ha ampliado su brecha durante los últimos cinco años frente a las tradicionales. La adopción de metodologías Ágiles se ha incrementado frente a las tradicionales a nivel mundial [23], siendo el tipo de metodología

que mejor refleja el proceso de desarrollo del 35% de las organizaciones de tecnologías de la información (TI). Evidencias sobre esta migración aparecen en estudios como en [24], donde se exponen métricas de calidad, costos, esfuerzo y defectos de más de 7500 proyectos de diferentes metodologías y de todo el mundo, evidenciando que los proyectos Ágiles entregan más funcionalidades con menos defectos y en menores tiempo respecto de los desarrollados bajo metodologías tradicionales. En [25] se resumen los beneficios de su aplicación en los siguientes:

- Alta productividad y bajos costos.
- Mejora del compromiso del empleado y satisfacción en su trabajo.
- Tiempos de salida al mercado más rápidos.
- Alta calidad.
- Mejora de la satisfacción del cliente.

Es de destacar las dificultades que origina la migración hacia estas formas de trabajo, donde en general se presentan situaciones de mucho rechazo ante el cambio debido al impacto que muchas de las prácticas tienen en la cultura de la organización. Como resultado de este proceso, las versiones *ad-hoc* son las más adoptadas. Es por ello, que la adopción de prácticas de seguridad en un equipo Ágil encontrará aún más resistencia en una primera etapa.

4.4 Desafíos presentados a la seguridad

Las características de las metodologías Ágiles, donde se priorizan aquellas actividades en el lado izquierdo del manifiesto, requieren adaptar las prácticas tradicionales de seguridad. La seguridad ya no debe quedar especificada en largos documentos de seguridad, ni esperar a que la empresa alcance un mayor nivel de madurez en sus procesos, sino que debe ser construida dentro del *software* y verificada mediante el correcto funcionamiento del mismo.

Para ello el nivel de experiencia requerida de los miembros de un equipo Ágil debe ser alto para poder sacar máximos beneficios de la aplicación de un marco Ágil como *Scrum,* de características iterativas y

adaptables al cambio. Esto parecería incorporar un obstáculo para la construcción de seguridad ya que [26] se exigiría a todos los miembros del equipo contar con altos conocimiento en controles de seguridad y programación segura. Si bien esto es cierto, no involucra un fracaso de equipos menos experimentados, ya que podrían verse beneficiados por otros tipos de prácticas o por la visita continua de un experto en seguridad como mencionamos anteriormente.

Notamos que la naturaleza iterativa nos otorga una oportunidad única de poder gestionar el riesgo en períodos cortos a diferencia de otras metodologías y dar una respuesta al cambio en tiempo y forma. Por ejemplo, en cada uno de las iteraciones (*sprints*) de *Scrum* pueden identificarse problemas no solo como falta de comunicación, bajo rendimiento, sino relacionados con la construcción de seguridad en el proyecto y soluciones alternativas para su mejora. Por otro lado, cabe recordar que uno de los problemas que encuentran los desarrolladores es la dificultad de cumplimentar con una lista larga de requerimientos de seguridad en las cortas iteraciones. Contar de antemano con historias de usuario y tareas de seguridad predefinidas ayudaría a incorporar y priorizar aquellos aspectos de seguridad que deben ser introducidos en cada una de las iteraciones.

Finalmente, el conocimiento en seguridad debe recaer en el equipo y no de expertos en seguridad específicamente, lo que introduce la necesidad de trabajar en equipos interdisciplinarios dispuestos a adquirir nuevos conocimientos en forma continua.

4.5 Visión de la seguridad

Al indagar sobre la importancia que se le da a la seguridad en los proyectos los participantes se promulgaron hacia diferentes alternativas. Por un lado un grupo se manifestó en favor de la importancia que se le otorga al proyecto en la organización, “la importancia se otorga en función al proyecto, siendo esto dependiente al tipo de negocio e industria donde pertenece la solución” (M1). Otros destacaron la importancia del valor de la información en cuanto al manejo de dinero o información personal. Luego, existió una inclinación sobre la inclusión de seguridad en etapas de mantenimiento o

post-liberación “es el cliente quien solicita la introducción de la misma como resultado de vulnerabilidades reportadas una vez que el producto es puesto en marcha” (DEVSSR1). Una última reflexión manifestó la “importancia del alcance y los costos en el proyecto”. (DEVSR1).

Observamos que la importancia que se le da a la seguridad depende tanto de de factores internos como externos a la organización.

Por otro lado, participantes con experiencia en proyecto tradicionales como Ágiles se han manifestado hacia la facilidad de construcción de seguridad en las primeras, justificando que las entregas cortas y productos creados al final de cada iteración hacen dificultan la percepción de seguridad.

“En mi experiencia, en los proyectos Ágiles, la seguridad es más vulnerable que en los proyectos no Ágiles, ya que al intentar realizar entregas más cortas y más tangibles, la imperceptibilidad de la seguridad queda postergada para el final” (M1). A la vez que la gran mayoría manifestó no haber utilizado seguridad en ninguno de sus proyectos.

Podemos justificar estos resultados al hecho de la falta de conocimiento de prácticas de seguridad aplicables a las diferentes etapas de un ciclo de desarrollo o bien marcos de construcción de seguridad. Muchos de los participantes manifestaron no haber participado de entrenamientos o bien haber leído sobre construcción de seguridad en el *software*.

4.6 El rol del experto en seguridad

Al adoptar una metodología Ágil como nueva forma de trabajo, nuevos roles emergen como los de *Scrum Master* y *Product Owner*, mientras que los viejos roles deben ser adaptados. Encontramos en [27] una descripción de estos roles, sus responsabilidades y habilidades que deberían tener aquellas personas con ansias de ocupar los mismos a la vez que buenas prácticas para realizar la transición hacia una metodologías Ágil si venimos de una tradicional.

¿Pero qué pasa con los roles tradicionales y en especial con el rol del experto en seguridad? Esta respuesta la he encontrado un tiempo atrás al analizar la incorporación de testers a equipos Ágiles. En este proceso, se alienta a sus miembros a realizar diferentes actividades, esto es cualquier

miembro del equipo puede tomar cualquier tipo de tarea [28] de forma que se alienta a todos los miembros del equipo a que transfieran a otros sus conocimientos. Esto es, se trata de trabajar incorporando la práctica que en XP se conoce como *whole team* [29]. Esto es, para cada equipo se debe traer aquellas personas que tengan las habilidades y actitudes requeridas por el proyecto, para lo cual en nuestro caso de interés, siempre debería haber alguien cumpliendo el rol de experto en seguridad, por lo menos interactuando intermitentemente en las diferentes iteraciones del proyecto. Del cual se espera que transmita su conocimiento al equipo en lugar de ser este el responsable por la seguridad del proyecto a diferencia de una metodología orientada al plan.

Muchos marcos de construcción de seguridad poseen una unidad funcional [30] o dominio [31] de gobernanza (*governance*) que define prácticas y actividades para facilitar este intercambio de habilidades entre los integrantes el equipo de desarrollo con este experto en seguridad, de forma que el conocimiento y las habilidades queden en el equipo y no en una persona.

Al indagar al equipo de desarrolladores, la gran mayoría manifestó no conocer la existencia de este rol en sus proyectos, mientras que solo uno de los participantes destaco la importancia de la colaboración de todos los miembros del equipo para concentrar en el mismo el conocimiento en seguridad requerido:

“El rol recae en el equipo, siendo mayor la responsabilidad en el arquitecto del mismo, encargado de TI y el líder de aseguramiento de la calidad” (M1).

4.7 Conclusión

Las metodologías Ágiles presentan muchos desafíos al momento de construir seguridad en las mismas. Como menciona [25], cada metodología Ágil a diferencia de la tradicional, que específica que hacer y cómo, deben ser adaptadas a las realidades de cada organización. De igual manera, la construcción de seguridad deberá ser adaptada para lidiar con los diferentes desafíos presentados por la versión de la metodología Ágil implementada. En el capítulo siguiente analizaremos como las prácticas usualmente utilizadas en un equipo Ágil pueden ayudarnos a construir seguridad y apalancar los esfuerzos hacia la incorporación de un marco más complejo.

5. Prácticas de desarrollo Ágiles

5.1 Introducción

En este capítulo analizaremos las prácticas de seguridad presentes en las metodologías Ágiles. Comenzaremos por describir y analizar aquellas de mayor uso y las relacionaremos conceptualmente la seguridad.

5.2 Un catálogo de prácticas

El uso de estas metodologías generalmente implica la aplicación de prácticas y valores que en su conjunto son conocidas como “prácticas Ágiles”. Estas pueden ser aplicadas en conjunto o bien individual y selectivamente de acuerdo a las necesidades y habilidades del equipo encargado del proyecto.

En el año 2009 se llevó a cabo una encuesta [32] realizada sobre 123 encuestados, la cual pretendía descubrir que estaban realizando los desarrolladores Ágiles con respecto a lo hablado en la comunidad sobre prácticas Ágiles aplicadas. Es así como la Figura 2, perteneciente a la encuesta, muestra las prácticas Ágiles más efectivas en cuanto a calidad.

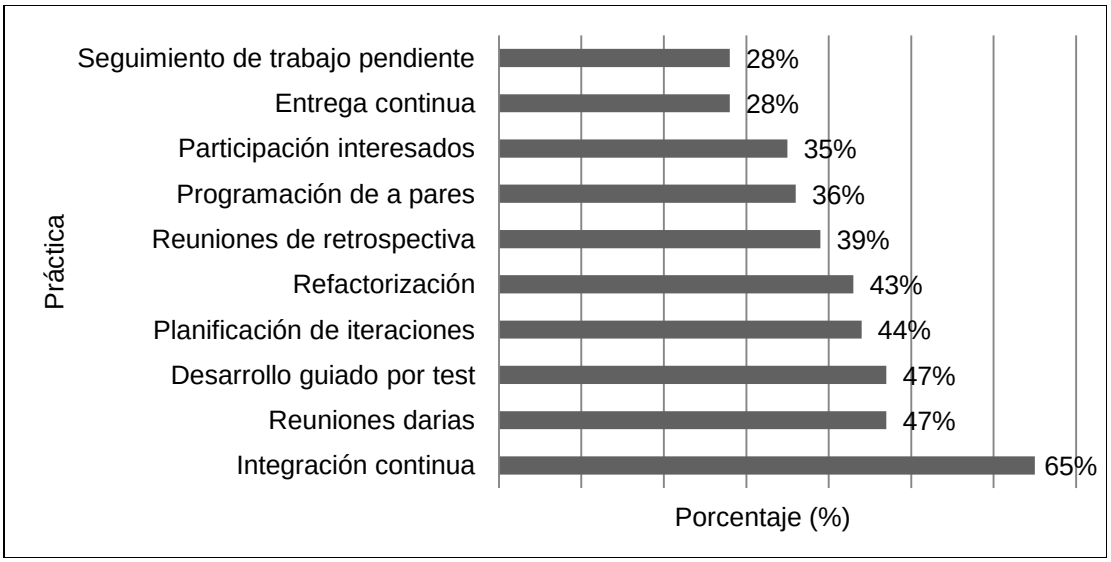

Figura 2**.** Adopción de Prácticas Ágiles

Estas prácticas, muchas de ellas originadas en XP, permiten no solo mejorar la calidad de los desarrollos, sino que algunas de ellas son un buen punto de partida para comenzar con la construcción de seguridad en aquellos equipos que no tienen la intención o es imposible incorporar un SDL a su proyecto.

5.3 Prácticas con seguridad inherentes

Hemos denominado de esta forma a aquellas prácticas que por sus características y su uso permiten enfrentar problemas de seguridad presentes en el desarrollo de *software*. Es sabido que si bien no fueron creadas para ello, creemos que estas nos ofrecen un punto de apalancamiento hacia la incorporación de prácticas específicas de seguridad. Hemos utilizado los resultados sobre la facilidad de adopción de prácticas Ágiles presentadas, para limitar el universo de prácticas que han de ser analizadas. A continuación analizaremos los aportes de cada una de ellas en materia de seguridad:

Integración continua: la integración continua es una práctica introducida como parte de XP [33]. Consiste en la compilación de todo el código fuente, junto a la ejecución de un conjunto de pruebas que acompañan al primero permitiendo verificar su correcto funcionamiento. Esta facilita la incorporación profesional de pruebas de seguridad y analizadores estáticos, donde con profesional nos referimos a la utilización de prácticas de prueba de seguridad serias y comprobadas, la cual se convierte en una herramienta muy importante y potente para cualquier equipo interesado en comenzar a construir seguridad dentro del *software.*

Desarrollo guiado por pruebas: esta práctica, consiste en la escritura de pruebas unitarias con anterioridad a la escritura del código fuente. Los pasos involucran:

1. Escritura de una pequeña prueba que no funcione
2. Escritura del código mínimo y necesario para que la prueba funcione
3. Eliminación de la duplicación introducida en el paso anterior

Esta forma de guiar el desarrollo mediante la escritura anticipada de pruebas, permite crear lo que se conoce como generación de código limpio (*clean code*) y una arquitectura emergente (*emergent architecture*), dado como resultado de la refactorización del código. Permite además, mejorar no solo el acople y la cohesión de los sistemas, ya que aquellos con bajo acople y alta cohesión son fáciles de probar. También genera confianza en el código escrito por el desarrollador, al ser cada prueba utilizada como

evidencia del correcto funcionamiento, facilitando así la lectura y modificación del mismo. Los beneficios derivados de esta práctica en aspectos de seguridad dentro del código son fundamentales. El énfasis en la generación de código limpio y la refactorización, implica una gran reducción de *bugs* que serán encontrados en tiempo de desarrollo por el mismo desarrollador, el cual no solo le permitirá perfeccionar sus técnicas de manera continua, sino que además producirá un código fácil de mantener por desarrolladores futuros más o menos experimentados. Por otro lado, esta práctica ayuda a evitar la introducción de fallas al sistema, ya que la evolución de una arquitectura emergente permite evolucionar a la misma en forma incremental, en pasos intermedios simples y acotados a los requerimientos especificados en un momento dado del ciclo de desarrollo.

Al ser la mayoría de los problemas de seguridad provienen de defectos introducidos en esta fase de desarrollo por los desarrolladores de forma involuntaria, o sea, el punto más vulnerable se encuentra en la tarea que mejor suponemos hacer, desarrollar. Hay que destacar que esta práctica es solamente aplicable a pruebas unitarias, para las pruebas de componentes o integración de módulos o sistemas interconectados deben utilizarse otras prácticas de prueba para equipos Ágiles [34].

Refactorización: la refactorización de código se refiere a la modificación de la estructura interna del código fuente del *software*, pero no en su comportamiento.

Como se mencionó en la práctica anterior, la refactorización de código fuente forma parte del primer paso a la hora de eliminar la duplicación de código, sin embargo puede aplicarse en aquellos proyectos que no utilicen desarrollo guiado por test como práctica habitual, y así obtener un punto de mejora del código.

Esta práctica no solo ayuda a la prevención de *bugs*, sino que existe la posibilidad de a partir de pequeños refactorizaciones identificar la necesidad de realizar cambios arquitectónicos a módulos del sistema [35]; estas revisiones de decisiones de diseño crean oportunidades para la identificación de posibles fallas.

Programación de a pares: esta pretende que dos programadores compartan una fracción de su tiempo programando juntos con el objetivo no

solo de obtener mejoras en el código desarrollado, sino que el tiempo invertido en programar con un compañero permitirá a ambos obtener mejores ideas sobre refinamientos a realizar en sistema; ayudará a clarificar ideas y a tomar iniciativa cuando el compañero está trabado con una tarea entre otros beneficios.

La tasa de identificación y evasión de posibles *bugs* y fallas puede ser incrementada al parear con algún programador que tenga conocimientos sólidos en prácticas seguras de codificación y/o experiencia lidiando con problemas de este tipo. Además, el pareo continuo de este experto con los diferentes miembros del equipo ayudara en el proceso de adopción de prácticas seguras de desarrollo.

Reuniones diarias: esta es una reunión que da diariamente al iniciar el día de trabajo, de forma que permita establecer el contexto en el que se estará trabajando e identificar cual trabajo se ha realizado y cual no. Al ser estas reuniones a modo informativo, creemos que no aportan muchas oportunidades para la construcción de seguridad, pero podrían ser utilizadas para conocer quienes estarán trabajando en aspectos de construcción de seguridad y coordinar con ellos futuras acciones posteriormente.

Reuniones de retrospectiva: esta es una reunión en la cual el equipo se reúne luego de haber completado el trabajo correspondiente a una iteración con el objetivo de inspeccionar y adaptar sus métodos y la forma de trabajo en equipo [36]. Encontramos esta reunión como una oportunidad para preguntarse qué es lo que se está haciendo en cuanto a materia de seguridad en el proyecto, y como realizar mejoras, en el caso de haber incorporado alguna práctica de seguridad, este es el momento de analizar sus resultados y realizar modificaciones en el rumbo.

La utilización de prácticas tradicionales de una metodología Ágil ayuda a construir seguridad en el código fuente evitando la introducción y proliferación de defectos, sin embargo, no provee prácticas para gestionar otros aspectos de la construcción de seguridad no relacionadas directamente con el desarrollo del código fuente pruebas de penetración y gestión de riesgos entre otros.

5.4 Dificultades en la aplicación

En el proyecto de estudio planteado se han observado el uso de solo las siguientes prácticas Ágiles:

Integración continua: con el objetivo de verificación del correcto proceso de compilación y enlazado de la aplicación. No se ha utilizado para ejecutar en forma continua ningún procedimiento o tarea relacionada con la construcción de la seguridad en el *software*.

Programación de a pares: su aplicación ha sido esporádica y la mayoría de las veces su objetivo ha sido la transferencia de conocimiento sobre la aplicación entre los programadores, generalmente en cuanto a decisiones de diseño y arquitectura relacionado con la implementación de los requerimientos.

Reuniones diarias y de retrospectiva: con el objetivo de comunicar el trabajo realizado y por realizar en las primeras, y la coordinación de acciones para realizar mejoras no relacionadas con la seguridad.

Hemos observado que algunos desarrolladores entrevistados, han manifestado una poca utilización de estas prácticas y aún menos con un enfoque en seguridad en sus proyectos. “Solo he aplicado la práctica de reuniones diarias por la mañana pero solo hablamos de los requerimientos e impedimentos que surgen en el proyecto nada de seguridad” (DEVSR1) .Otros han mencionado el interés por aplicarlas, pero han encontrado dificultades en el aprendizaje y la continuidad por las presiones en los tiempos de entrega.

La utilización de prácticas inherentes a una metodología Ágil nos ayuda a construir seguridad en el código fuente evitando la introducción y proliferación de defectos, sin embargo, no nos provee prácticas para gestionar otros aspectos de la construcción de seguridad no relacionadas directamente con el desarrollo del código fuente, como ser captura de requerimientos de seguridad, realización de pruebas de seguridad, pruebas de penetración, y gestión de riesgos entre otros. Es aquí donde entra en juego la necesidad de incorporar junto a estas prácticas seguras, orientadas a atacar directamente problemas específicos de seguridad.

5.5 Conclusión

La utilización de prácticas inherentes a una metodología Ágil nos ayuda a construir seguridad en el código fuente evitando la introducción y proliferación de problemas de implementación y diseño como hemos visto en este capítulo, sin embargo, no nos provee prácticas para gestionar otros aspectos de la construcción de seguridad no relacionadas o no directamente con el desarrollo del código fuente, como ser captura de requerimientos de seguridad, realización de pruebas de seguridad, pruebas de penetración, y gestión de riesgos entre otros. Es aquí donde entra en juego la necesidad de enmarcar estas metodologías dentro de un *framework* de construcción de seguridad que contemple otros aspectos el cual es el objetivo del siguiente capítulo.

6. Introducción de prácticas de seguridad en Ágiles

6.1 Introducción

En este capítulo analizaremos las facilidades que las prácticas Ágiles nos ofrecen al momento de incorporar prácticas o actividades de seguridad a nuestro proyecto y a la transición hacia a marcos de seguridad más complejos. Para ello, presentaremos aquellas prácticas de seguridad que hemos decidido implementar en nuestros proyectos según la naturaleza de este y su relación con las prácticas Ágiles. Finalmente, se presentan los resultados obtenidos tanto de la integración de prácticas como reacciones del equipo.

6.2 Selección de prácticas seguras

La introducción de prácticas seguras en el equipo del proyecto Ágil observado ha sido guiada por el marco OpenSAMM. Su elección se justifica debido a que presenta plantillas predefinidas que guían en la construcción de un programa de seguridad según las características del proyecto.

Dentro de las plantillas que encuadran en nuestros proyectos observados encontramos las de "vendedor de *software* independiente" (*independent software vendor*) y "proveedor de servicios en línea" (*online service provider*). De estas, hemos optado por elegir la primera debido a la naturaleza de *software factory* de la organización a la que pertenecen los proyectos observados.

La priorización y su evolución de las doce prácticas de seguridad que presenta OpenSAMM para sus unidades funcionales pueden verse en su especificación. La Tabla X. Prácticas y actividades seleccionadas de OpenSAMM muestra las prácticas y actividades que esta plantilla propone para sus primeras fases que haremos corresponder con las primeras iteraciones y las cuales serán apalancadas mediante las prácticas Ágiles presentadas en el capítulo anterior.

Tabla X. Prácticas y actividades seleccionadas de OpenSAMM

Práctica	Actividad
Revisión de código	RC1. Crear listas de control a partir de requerimientos de seguridad conocidos
	RC2. Realizar revisiones en el código de alto riesgo
	RC3. Utilizar herramientas de análisis de código automatizadas
	RC4. Integrar las herramientas en el proceso de desarrollo
Pruebas de seguridad	PS1. Derivar casos de prueba de requerimientos de seguridad conocidos
	PS2. Realizar pruebas de penetración al código
	PS3. Utilizar herramientas automatizadas de prueba de seguridad

	PS4. Integrar las pruebas de seguridad en el proceso de desarrollo
Requerimientos de seguridad	RS1. Derivar requerimientos de seguridad de funcionalidades de negocio
	RS2. Evaluar seguridad y cumplimiento para los requerimientos
	RS3. Evaluar seguridad y cumplimiento para los requerimientos
	RS4. Especificar requerimientos de seguridad en base a riesgos conocidos
Educación y guías	EG1. Realizar entrenamientos para generar conciencia de seguridad
	EG2. Construir guías técnicas de de seguridad
	EG3. Realizar entrenamientos específicos por roles
	EG4. Usar expertos en seguridad para dar entrenamientos
Estrategias y métricas	No se utilizará ya que en la organización no existe nada de esto
Gestión de vulnerabilidades	No se utilizará ya que el proyecto incluye solo el desarrollo

6.3 Selección de prácticas Ágiles con seguridad inherente

Al equipo observado se le ha propuesto perfeccionar el uso de sus prácticas Ágiles otorgándoles un enfoque orientado a la seguridad de forma progresiva. Estas se han utilizado para apalancar la incorporación de prácticas seguras y sus actividades mencionadas en el apartado anterior. La Tabla XI muestra las prácticas Ágiles que se han seleccionado.

Tabla XI. Prácticas Ágiles seleccionadas para apalancamiento

IC. Integración continua
RD. Reuniones diarias
RR. Reuniones de retrospectiva
WT. Whole team
DGP. Desarrollo guiado por pruebas
PP. Programación de a pares
RC. Refactorización de código

Mientras que la

Tabla XII muestra como han sido relacionadas las prácticas con las actividades.

Tabla XII. Relación entre prácticas Ágiles con seguridad inherente y actividades de prácticas seguras

Inherentes / Seguras	DGP	RC	PP	IC	RD	RR	WT
RC1							
RC2							
RC3							
RC4							
PS1							
PS2							
PS3							
PS4							
RS1							
RS2							
RS3							
RS4							
EG1							
EG2							
EG3							
EG4							

6.4 Resultados a partir de la interacción entre prácticas

Observamos que aquellas prácticas Ágiles que permiten incorporar la mayor cantidad de actividades de seguridad son:

1. Integración continua

2. Programación de a pares
3. Desarrollo guiado por pruebas
4. *Whole team*

El perfeccionamiento continuo de estas prácticas Ágiles permitirá realizar una transición hacia la incorporación de actividades y prácticas o un marco de seguridad más complejo generando menor resistencia al cambio por parte del equipo y la organización, de aquí su naturaleza de agregar seguridad inherente. Por otro lado existen un conjunto de actividades y prácticas de seguridad que no han podido ser apalancadas por estas prácticas Ágiles como ser aquellas relacionadas a estrategias y entrenamientos que deben ser implementados a un nivel de organización y no de proyecto.

6.5 Resultados sobre la metodología

Luego de dos iteraciones de dos semanas cada una de implementación de los cambios, se observaron los siguientes cambios metodológicos en el equipo seleccionado para experimentación:

Reuniones diarias: comenzaron a aparecer términos relacionados con la seguridad por parte del equipo de desarrollo.

Planificación de iteraciones: se comenzó a incorporar historias de usuario relacionadas directamente con la seguridad de los requerimientos.

Integración continua: el proceso no se utilizó solamente para verificar la correcta compilación e integración del código fuente, sino que se incluyeron analizadores de código estático para la detección de *bugs*.

Programación de a pares: dejo de ser utilizada como una práctica de iniciación de nuevos programadores, y comenzó a utilizarse para evaluar decisiones de diseño y realizar revisiones de código; atacando la posibilidad de fallas y *bugs*.

Whole team: Se sugirió el uso de esta práctica y se han aprovechado las reuniones de retrospectiva para consolidar el conocimiento en seguridad incorporado durante la iteración.

7. Conclusión

Con el ánimo de dar respuesta a la pregunta de investigación que ha originado este trabajo **¿Cuáles son los obstáculos que dificultan la presencia de prácticas de seguridad en proyectos ágiles?** , hemos encontrado los siguientes obstáculos que se enumeran a continuación:

1. Dificultades al momento de adoptar una metodología Ágil.
2. Ausencia de una mentalidad focalizada en seguridad por parte de la organización.
3. La falta de conocimientos y controles de seguridad necesarios por parte de los desarrolladores.
4. Visión de la seguridad como algo externo al desarrollo del software.

En base a lo analizado en esta tesis verificamos que si bien existen prácticas Ágiles que ayudan a construir seguridad en forma inherente, esto no es suficiente sin la incorporación de otras prácticas que incorporen seguridad en forma explícita como aquellas contenidas en algún modelo de construcción de seguridad como los vistos en este trabajo. Sin la aplicación de estas últimas, estaríamos omitiendo la construcción de seguridad en otras etapas distintas del a la de desarrollo, como ser la captura de requerimientos de seguridad, análisis de riesgos y pruebas de penetración. Luego, hemos llegado también a la conclusión de que **la aplicación de prácticas de desarrollo seguro en proyectos Ágiles está relacionada con su conocimiento**.

A lo largo de esta tesis hemos puesto en manifiesto el desconocimiento de los conceptos de construcción de seguridad en el *software* junto a su aplicación dentro de una metodología de desarrollo. Encontrando la principal causa en la falta de una mentalidad orientada a seguridad dentro de la organización, afectando a los equipos de desarrollo. De los diálogos recolectados hemos verificado la ausencia de capacitaciones en materia de seguridad, y presencia de desarrolladores calificados en la misma en los equipos de desarrollo.

7.4 Menciones

De la investigación realizada en esta tesis se ha originado el trabajo denominado ”*Building Security in Agile Projects*” el cual ha sido seleccionado para su publicación en la conferencia TIBETS 2013 a realizarse el 29 de octubre del 2013 en la ciudad de Panamá, Panamá.

7.4 Recomendaciones

En base a los hallazgos identificados a lo largo de esta tesis, hemos elaborado las siguientes recomendaciones para la aplicación de seguridad en metodologías Ágiles. Estas pretenden facilitar la incorporación de seguridad en las metodologías Ágiles por parte de una organización y se presentan en orden de complejidad creciente en cuanto a su adopción en la siguiente lista:

Evaluación de madurez: esta recomendación pretende obtener una evaluación del grado de madurez del desarrollo Ágil en el cual nos encontramos, lo que hemos denominado “fortalecer nuestra Agilidad”. De esta forma podremos identificar de que prácticas nos estamos beneficiando y cuales podremos incorporar o mejorar. Este será un buen momento para perfeccionar la aplicación de la metodología Ágil utilizada.

Desarrollo de una mentalidad en seguridad: esta recomendación tiene por objetivo comenzar a desarrollar una mentalidad en seguridad en los diferentes equipos de desarrollo. Este puede lograrse comenzando con simples capacitaciones que pueden variar desde presentar prácticas sencillas de seguridad hasta el desarrollo de conocimientos sobre algún marco de seguridad. Este despertar en seguridad debe ser el primer paso que todo equipo de desarrollo debería tomar para comenzar a construir seguridad en el código y así evitar el rehúso o rechazo al cambio que se da en forma natural. Luego, conocer nuestro estado y donde queremos llegar facilitará este proceso. Esta podría venir acompañado de la ayuda de un experto externo a la organización de ser necesario.

Aplicación de prácticas: Esta recomendación pretende comenzar a

aplicar prácticas de seguridad inherente y/o segura sobre la fase de codificación. Los desarrolladores son los primeros en querer mejorar lo que se está haciendo y el rechazo al cambio será mejor desde abajo hacia arriba. Se pueden sugerir el uso de analizadores sintácticos, técnicas de refactorización y revisión de código entre otras prácticas. Es de vital importancia.

Transferencia: Esta recomendación tiene por objetivo brindar un punto de apoyo a los equipos y que su conocimiento pueda ser transferido hacia cada integrante del equipo.

7.4 Ampliaciones y trabajos futuros

Si bien nuestro objetivo durante este trabajo ha sido identificar y analizar las dificultades presentes al momento de construir seguridad en metodologías Ágiles dentro de los equipos de desarrollo, este es un campo relativamente nuevo que requiere más análisis en otras áreas. Se aporta un listado de temas que podrían guiar nuevos trabajos de investigación que se muestran en la siguiente lista:

1. Estudio de las técnicas para realizar pruebas de seguridad.
2. Efectividad de los modelos de construcción de seguridad en diferentes metodologías Ágiles.
3. Seguridad y análisis de requerimientos mediante historias de usuario.
4. Análisis y gestión de riesgos de seguridad en metodologías Ágiles.

ANEXO I. Identificación de instancias de unidades de análisis

Recolección de datos: Observaciones en el equipo A**.**

Unidad de análisis: Problema de seguridad.

Contexto: Observaciones realizadas a los desarrolladores del equipo A en su ambiente de trabajo.

Observaciones:

Null pointers.

Mal manejo de excepciones.

Incorrecta implementación de mecanismos de autenticación.

Incorrecta implementación de mecanismos de autorización.

Inyección de SQL.

Inyección de JavaScript.

Mecanismos de encriptación obsoletos.

Falta de información de *logging*.

Memory leaks.

Configuración incorrecta de accesos a los entornos y VPNs.

Manejo incorrecto de errores.

Validación de valores de entrada y salida.

Mal resguardo de la información sensitiva

Recolección de datos: Observaciones en el equipo A**.**

Unidad de análisis: Estrategia de seguridad.

Contexto: Observaciones realizadas a los desarrolladores del equipo A en su ambiente de trabajo.

Observaciones:

Uso de algún marco de desarrollo seguro

Introducción de verificaciones de seguridad en el cógido

Programar primero realizando pruebas del código en base a resultados esperados

Poner atención en el desarrollo

Obtener una lista de requerimientos de seguridad que se esperan

ANEXO II. Sesión de trabajo con el equipo para la observación de prácticas Ágiles utilizadas

Episodio o situación: Sesión de trabajo del equipo A
Fecha: 01 de enero de 2012
Hora: 13:00
Participantes: Integrantes del proyecto del equipo A
Lugar: Empresa de trabajo
Tema del proyecto: Sistema de comercio electrónico para importante empresa.
Perfiles en el proyecto:

1. 1 Desarrollador Java Sr.
2. 1 Desarrollador Java SSr.
3. 1 Líder Técnico
4. 1 Tester SSr.
5. 1 Manager

Idioma utilizado en el proyecto para la comunicación: Ingles
Metodología de desarrollo usada: Scrum
Prácticas Ágiles puestas en marcha:

1. *Daily standup*
2. *Iteration planning*
3. *Backlog grooming*

Prácticas de seguridad en el proyecto:

1. Uso de VPN
2. Cuentas de email en el dominio del cliente
3. Uso de repositorio para el código fuente

Observaciones:

1. Se observan que los desarrolladores no presentan conocimientos fuertes en mejores prácticas de programación. Esto se ve manifestado mediante *code smells* en el código escrito, como ser métodos demasiado largos, gran cantidad de condicionales anidados, nombre de

métodos no significativos, gran cantidad de parámetros a de entrada y salida en ellos, falta de información de *logging*, falta de *guard clauses* y aplicación de patrones de diseño entre otros que permitan escribir un *clean code*.

2. El equipo se niega a realizar pruebas sobre el código desarrollado, ya que lo considera innecesario.
3. Se nota que cada miembro posee entendimientos sobre Ágil diferentes.
4. Así mismo, no existen conocimientos específicos de seguridad.
5. Se observa además que cada miembro del equipo tiene conocimientos sobre la plataforma de desarrollo que el otro no tiene, y las tareas son divididas según este conocimiento.
6. No existen roles de seguridad.

Siguientes pasos: Entrevistar a los miembros del equipo en cuestiones de seguridad y desarrollo Ágil.

ANEXO III. Guía de entrevista sobre metodologías Ágiles y seguridad

Fecha: 10 de enero de 2013

Hora: 12:00

Entrevistado: TL1

Preguntas:

1. **¿Qué entiende cuando se habla de construir seguridad dentro del software?**

   Se pretende conocer que es la construcción de seguridad en el software por parte del desarrollador, si solo hace referencia al agregado de mecanismos de seguridad como autenticadores, cifradores, y otros elementos relacionados con la protección del ambiente de despliegue del *software

   La introducción de prácticas orientadas a mitigar problemas de seguridad que puedan surgir encada etapa del ciclo de vida del desarrollo de software.

2. **¿Conoce o uso alguna vez metodologías/prácticas de desarrollo que incluyan prácticas de seguridad? *Se pretende conocer la experiencia en el uso de prácticas de seguridad. Si conoce algunas menciónelas.**

   Si, Microsoft SDL en su versión ágil y no ágil. Solo las conozco, no las he utilizado.

3. **¿Qué importancia se le da a la seguridad en su proyecto? *Se pretende entender si el hecho de construir seguridad es proyecto dependiente.**

   Es un aspecto del sistema del que no se habla, si surgen problemas se arreglan como si fueran bugs. No es algo de lo que se esté interesado.

4. **¿Existen roles de seguridad en su/s proyecto/s? *Se pretende entender como es gestionado el conocimiento de seguridad en los proyectos, si recae en alguna persona o en el equipo**

   No.

5. **¿Nota alguna diferencia en la construcción de seguridad en proyectos Ágiles vs. no Ágiles? *Se pretende diferenciar el manejo de seguridad en ambos tipos de proyectos**

   Si, en los no agiles existen determinados artefactos de seguridad que la metodología exige por lo cual deben ser implementados en su respectiva etapa. En las Ágiles esto depende de que prácticas implemente el equipo ya que no es obligatorio. Por otro lado, esto incurre en mucho *overhead* en la creación de documentos que seguramente no serán actualizados

6. **¿Ha usado prácticas en Ágil que cree le hayan ayudado a construir seguridad en el proyecto? *Se pretende conocer que prácticas de Ágil agregan seguridad**

   Sí, he utilizado TDD *y* revisiones de código, junto con integración continua. Creo que estas pueden ayudar a construir seguridad.

7. **¿Qué rol desempeña en su proyecto? ***

   Líder técnico

8. **¿Qué metodologías ágiles ha utilizado en los últimos 6 meses?**

   Scrum adaptado y Kanban

Fecha: 4 de enero de 2013

Hora: 18:09:27

Entrevistado: M1

*Preguntas***:**

1. **¿Qué entiende cuando se habla de construir seguridad dentro del software?**

   Se pretende conocer que es la construcción de seguridad en el software por parte del desarrollador, si solo hace referencia al agregado de mecanismos de seguridad como autenticadores, cifradores, y otros elementos relacionados con la protección del ambiente de despliegue del *software

   Entiendo por seguridad dentro del software el resguardo de la información, los controles para el cumplimiento de estos procesos y prácticas necesarias dentro del proceso de desarrollo.

2. **¿Conoce o uso alguna vez metodologías/prácticas de desarrollo que incluyan prácticas de seguridad? *Se pretende conocer la experiencia en el uso de prácticas de seguridad. Si conoce algunas menciónelas.**

   No

3. **¿Qué importancia se le da a la seguridad en su proyecto? *Se pretende entender si el hecho de construir seguridad es proyecto dependiente.**

   La importancia se otorga en función al proyecto, siendo esto dependiente al tipo de negocio e industria donde pertenece la solución. Otro factor importante es el valor de la información que contendrá el sistema y el manejo de dinero.

4. **¿Existen roles de seguridad en su/s proyecto/s? *Se pretende entender como es gestionado el conocimiento de seguridad en los proyectos, si recae en alguna persona o en el equipo**

   El rol recae en el equipo, siendo mayor la responsabilidad en el arquitecto del mismo, encargado de TI y el líder de calidad.

5. **¿Nota alguna diferencia en la construcción de seguridad en proyectos Ágiles vs. no Ágiles? *Se pretende diferenciar el manejo de seguridad en ambos tipos de proyectos**

   En mi experiencia, en los proyectos Ágiles, la seguridad es más vulnerable que en los proyectos no Ágiles, ya que al intentar realizar entregas más cortas y más tangibles, la imperceptibilidad de la seguridad queda postergada para el final

6. **¿Ha usado prácticas en Agile que cree le hayan ayudado a construir seguridad en el proyecto? *Se pretende conocer que prácticas de Agile agregan seguridad**

   No

7. **¿Qué rol desempeña en su proyecto? ***

   Project Manager

8. **¿Qué metodologías ágiles ha utilizado en los últimos 6 meses?**

   TDD. El entrevistado manifiesta que su metodología ha sido la de utilizar test para guiar el desarrollo, confundiendo una metodología con una práctica.

Fecha: 9 de enero de 2013

Hora: 11:12

Entrevistado: DEVSS1

*Preguntas***:**

1. **¿Qué entiende cuando se habla de construir seguridad dentro del software?**

   Se pretende conocer que es la construcción de seguridad en el software por parte del desarrollador, si solo hace referencia al agregado de mecanismos de seguridad como autenticadores, cifradores, y otros elementos relacionados con la protección del ambiente de despliegue del *software

   Prestar atención a lo que se programa para evitar errores graves como inyección de *sql* y javas*cripts* entre otros.

2. **¿Conoce o uso alguna vez metodologías/prácticas de desarrollo que incluyan prácticas de seguridad? *Se pretende conocer la experiencia en el uso de prácticas de seguridad. Si conoce algunas menciónelas.**

   No

3. **¿Qué importancia se le da a la seguridad en su proyecto? *Se pretende entender si el hecho de construir seguridad es proyecto dependiente.**

   Es el cliente quien solicita la introducción de la misma como resultado de vulnerabilidades reportadas una vez que el producto es puesto en marcha

4. **¿Existen roles de seguridad en su/s proyecto/s? *Se pretende entender como es gestionado el conocimiento de seguridad en los proyectos, si recae en alguna persona o en el equipo**

   No.

5. **¿Nota alguna diferencia en la construcción de seguridad en proyectos Ágiles vs. no Ágiles? *Se pretende diferenciar el manejo de seguridad en ambos tipos de proyectos**

   No

6. **¿Ha usado prácticas en Agile que cree le hayan ayudado a construir seguridad en el proyecto? *Se pretende conocer que prácticas de Agile agregan seguridad**
   No

7. **¿Qué rol desempeña en su proyecto? ***
   Desarrollador semi-señor

8. **¿Qué metodologías ágiles ha utilizado en los últimos 6 meses?**
   Scrum personalizado

Fecha: 9 de enero de 2013

Hora: 10:55

Entrevistado: DDEVS1

Preguntas:

1. **¿Qué entiende cuando se habla de construir seguridad dentro del software?**

   Se pretende conocer que es la construcción de seguridad en el software por parte del desarrollador, si solo hace referencia al agregado de mecanismos de seguridad como autenticadores, cifradores, y otros elementos relacionados con la protección del ambiente de despliegue del *softwtare

   Agregar los mecanismos de seguridad para que el software no sea atacado.

2. **¿Conoce o uso alguna vez metodologías/prácticas de desarrollo que incluyan prácticas de seguridad? *Se pretende conocer la experiencia en el uso de prácticas de seguridad. Si conoce algunas menciónelas.**

   No

3. **¿Qué importancia se le da a la seguridad en su proyecto? *Se pretende entender si el hecho de construir seguridad es proyecto dependiente.**

   Depende del alcance y los costos del proyecto

4. **¿Existen roles de seguridad en su/s proyecto/s? *Se pretende entender como es gestionado el conocimiento de seguridad en los proyectos, si recae en alguna persona o en el equipo**

   No que conozca

5. **¿Nota alguna diferencia en la construcción de seguridad en proyectos Agile vs. no Agiles? *Se pretende diferenciar el manejo de seguridad en ambos tipos de proyectos**

   No he tenido la oportunidad de trabajar con seguridad en ningún tipo de proyecto

6. **¿Ha usado prácticas en Agile que cree le hayan ayudado a construir seguridad en el proyecto? *Se pretende conocer que prácticas de Agile agregan seguridad**
   Quizás TDD, que ayuda a evitar *bugs* en el código

7. **¿Qué rol desempeña en su proyecto? ***
   Desarrollador Señor

8. **¿Qué metodologías ágiles ha utilizado en los últimos 6 meses?**
   Scrum personalizado

Bibliografía

[1] G. McGraw, «The security problem,» de *Software Security- Building Security In*, Crawfordsville, Addison-Wesley, 2006, p. 4.

[2] SE21, «S21 Security Blog,» 2011. [En línea]. Disponible: http://securityblog.s21sec.com/2012/03/s21sec-detects-almost-7000.html. [Último acceso: 03 01 2012].

[3] J. West, G. McGraw y S. Migues, «BSIMM,» [En línea]. Disponible: http://bsimm.com/. [Último acceso: 2013 06 20].

[4] C. Pravir, «OpenSAMM,» [En línea]. Disponible: http://www.opensamm.org/. [Último acceso: 30 06 2013].

[5] Microsoft, «Microsoft SSDL,» Microsoft, 27 04 2005. [En línea]. Disponible: http://www.microsoft.com/security/sdl/discover/sdlagile.aspx. [Último acceso: 29 09 2012].

[6] V. Asthana, I. Tarandach, N. O'Donoghue, B. Sullivan y M. Saario, «SAFECODE,» 17 07 2012. [En línea]. Disponible: http://www.safecode.org/publications/SAFECode_Agile_Dev_Security0712.pdf. [Último acceso: 2013 03 03].

[7] S. Bartsch, «Practitioners' Perspectives on Security in Agile Development,» 2011.

[8] G. McGraw, «Security Problems in Software,» de *Software Security- Building Security In*, Crawfordsville, Addison-Wesley, 2006, pp. 14-20.

[9] K. M. Goertzel, «Build Security IN,» 09 01 2009. [En línea]. Disponible: https://buildsecurityin.us-cert.gov/bsi/547-BSI.html. [Último acceso: 03 08 2012].

[10] G. McGraw, «The Trinity of Trouble: Why the problem Is Growing,» de *Software Security - Building Security In*, Crawfordsville, Addison-Wesley, 2006, pp. 5-10.

[11] M. Schwartz, «InformationWeek,» 17 01 2012. [En línea]. Disponible: http://www.informationweek.com/news/security/vulnerabilities/232400392. [Último acceso: 04 08 2012].

[12] B. Chess y J. West, «Defensive programming is not enough,» de *Secure Programming with Static Analysis*, Crawsfordsville, Addison-Wesley, 2007, p. 4.

[13] B. Chess y J. West, «Security Features != Secure Features,» de *Secure Programming with Static Analysis*, Crawfordsville, Addison-Wesley, 2007, p. 6.

[14] B. Chess y J. West, «The Quality Falacy,» de *Secure Programming with static analysis*, Crawfordsville, Addison-Wesley, 2007, p. 9.

[15] B. Chess y J. West, «The Quality Fallacy,» de *Secure Programming with static analysis*, Crawfordsville, Addison-Wesley, 2007, p. 10.

[16] M. Beedle, A. Vann Bennekum, A. Cockburn, W. Cunningham, M. Fowler, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, K. Schwaber, J. Sutherland y D. Thomas, «Manifiesto Ágil,» 2001. [En línea]. Disponible: http://agilemanifesto.org/. [Último acceso: 29 09 2012].

[17] M. Beedle, A. Vann Bennekum, A. Cockburn, W. Cunningham, M. Fowler, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. Martin, K. Schwaber, J. Sutherland y D. Thomas, «Manifesto for Agile Software Development,» 2001. [En línea]. Disponible: http://agilemanifesto.org/iso/es/principles.html. [Último acceso: 03 10 2012].

[18] Succeding With Agile - Software Development Using Scrum, Ann Arbor, Michigan: Addison-Wesley, 2010, p. 475.

[19] I. Jacobson, G. Booch y J. Rumbaugh, El proceso Unificado de Desarrollo de Software, Madrid: Addison Wesley, 2000.

[20] J. Mishra y A. Mohanty, «Software Engineering,» Pearson Education India, 2011, p. 400.

[21] Microsoft, «Microsoft Security Development Lifecycle,» Microsoft, 27 04 2005. [En línea]. Disponible: http://technet.microsoft.com/en-us/library/bb497060.aspx. [Último acceso: 29 09 2012].

[22] M. Cohn, «Agile Succeeds Three Times More Often Than Waterfall,» 2012. [En línea]. Disponible: http://www.mountaingoatsoftware.com/blog/agile-succeeds-three-times-more-often-than-waterfall. [Último acceso: 07 10 2012].

[23] P. Rodríguez, D. Musat, A. Yagüe, B. Turhan, A. Rohunen, P. Kuvaja y M. Oivo, «Adopción de

metodologías ágiles: un estudio comparativo,» *Revista Española de Innovación, Calidad e Ingeniería del Software,* vol. 6, nº 4, p. 24, 2010.

[24] M. Lunt, «Cutter Consortium,» [En línea]. Disponible: http://www.cutter.com/offers/measureup.html. [Último acceso: 07 10 2012].

[25] M. Cohn, Succeding With Agile - Software Development Using Scrum, Ann Arbor, Michigan: Addison-Wesley, 2010, p. 467.

[26] Tobias, «Secure Scrum,» 22 02 2010. [En línea]. Disponible: http://www.opensecurityarchitecture.org/cms/community/blogs/270-secure-scrum. [Último acceso: 02 01 2012].

[28] M. Cohn, «New Roles,» de *Succeding With Agile - Software Development Using Scrum*, Ann Arbor, Addison-Wesley, 2010, p. 117.

[29] L. Crispin y J. Gregory, «What Is Agile Testing, Anyway?,» de *Agile Testing - A Practical Guide For Testers And Agile Teams*, Boston, Addison-Wesley, 2009, pp. 7-10.

[30] K. Beck y C. Andres, «Primary Practices,» de *Extreme Programming Explained - Embrace Change*, Stoughton, Addison-Wesley, 2005, pp. 37-51.

[31] C. Pravir, «OpenSAMM,» 12 05 2013. [En línea]. Disponible: http://www.opensamm.org/. [Último acceso: 12 05 2013].

[32] G. McGraw, S. Migues y J. West, «The Building Security In Maturity Model,» [En línea]. Disponible: http://bsimm.com/. [Último acceso: 12 07 2013].

[33] S. Ambler, «Agile Practices Survey Results: July 2009,» http://www.ambysoft.com/surveys/practices2009.html, 2009. [En línea]. Disponible: http://www.ambysoft.com/surveys/practices2009.html. [Último acceso: 21 10 2012].

[34] K. Beck y C. Andres, Extreme Programming Explained-Embrace change, Stoughton: Addison Wesley, 2005, p. 189.

[35] L. Crispin y J. Gregory, Agile Testing, Crawfordsville: Addison Wesley, 2009.

[36] J. Bird, «What Refactoring is, and what it isn't - According to Kent Beck and Martin Fowler,» 16 12 2012. [En línea]. Disponible: http://agile.dzone.com/articles/what-refactoring-and-what-it-0. [Último acceso: 03 01 2013].

Índice de Figuras

Índice de Tablas