IMPLICACIONES DE SEGURIDAD EN METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE

JONATHAN ALEXANDER RAMIREZ AGUILERA

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2017

IMPLICACIONES DE SEGURIDAD EN METODOLOGÍAS ÁGILES DE DESARROLLO DE SOFTWARE

JONATHAN ALEXANDER RAMIREZ AGUILERA

Trabajo de grado presentado como requisito parcial

para optar por el título de Ingeniero de Sistemas

Director

LUIS EDUARDO BAQUERO REY

Magíster en Seguridad Informática

FUNDACIÓN UNIVERSITARIA LOS LIBERTADORES
FACULTAD DE INGENIERÍA Y CIENCIAS BÁSICAS
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS
BOGOTÁ D.C.
2017

TABLA DE CONTENIDO

LISTA DE TABLAS

pág.

LISTA DE GRÁFICAS

pág.

RESUMEN

La seguridad en el software es una de las características transversales más importantes para el desarrollo de aplicaciones informáticas, debido al aumento en el uso de aplicaciones y complejidad sea presentando un incremento en los ciberataques ejecutados en diferentes sistemas informáticos esto genera pérdidas de diferente índole.

En Colombia la forma en la que se abordan los proyectos de software usando metodologías ágiles con estándares de seguridad es incierto, para el desarrollo de esta investigación se utilizara la modalidad de estudio de campo por medio de una encuesta. Posterior a su implementación se presenta el análisis, resultados obtenidos y conclusiones.

El trabajo busca contextualizar como se encuentran las empresas colombianas en el uso de buenas prácticas para el desarrollo de software seguro y como hacen uso de las metodologías ágiles.

Palabras clave: Ágiles, Desarrollo, Informática, Metodología, Seguridad, Software.

ABSTRACT

Software security is one of the most important cross-cutting characteristics for the development of software applications, due to the increase in the use of applications and complexity is presenting an increase in cyber-attacks executed on different computer systems this generates losses of different nature.

In Colombia, the way in which software projects using agile methodologies with safety standards are addressed is uncertain, for the development of this research field by means of a survey study mode is used. Subsequent to its implementation is presented the analysis, results and conclusions.

The work seeks to contextualize as is found them companies Colombian in the use of good practices for the development of software secure and as make use of the methodologies agile.

Keywords: Agile, Development, Computing, Methodology, Security, Software.

INTRODUCCIÓN

Hoy en día las empresas se ven apoyadas en la mayoría de sus labores gracias a la tecnología, desde comunicaciones entre diferentes sistemas empresariales para la entrega de un resultado o funcionalidad hasta manejo de información interna, teniendo en cuenta que esto abarca usos muy amplios dependiendo el enfoque manejado por la empresa lo que supone que existan grandes riesgos en cada una de las aplicaciones de tecnología. Además, estos riesgos evolucionan con la tendencia de tecnologías como computación en la nube o aplicaciones empresariales diseñadas para equipos móviles. En Colombia los ataques informáticos impactan en las empresas por pérdidas cercanas a los 500 millones de dólares, a nivel mundial las cifras son aún mayores, ya que el 94% de las empresas del mundo reporto algún tipo de ataque cibernético. Un escenario que no puede pasar desapercibido por las personas relacionadas con el desarrollo de software las que deben empezar a diseñar el uso de metodologías que garantice el lugar que le corresponde a la seguridad sin dejar de lado la agilidad y fiabilidad con la que son desarrolladas estas aplicaciones.

El objetivo de este trabajo es presentar los impactos que se tienen en las metodologías ágiles al implementar requerimientos de seguridad en los proyectos que permitan producir aplicaciones de software seguro. Después se presenta un análisis del comportamiento de las metodologías, una comparación entre ellas para finalmente tener en cuenta cuales son los puntos a favor y en contra cuando se desarrolla software seguro en una metodología ágil para que las empresas tengan en cuenta cómo pueden ver impactados sus proyectos ante este hecho.

1. CONTEXTO GENERAL

1.1 DESCRIPCIÓN DEL PROBLEMA

Debido al incremento día a día de ataques informáticos a diversas entidades y del riesgo en términos de dinero e información que esto representa para las empresas, se tiene que contemplar una medida eficaz que permita a los proyectos de desarrollo adaptarse para tener en cuenta el desarrollo de software seguro entre sus prioridades sin impactar demasiado su ejecución a nivel de tiempo, agilidad y calidad. Por lo que se vuelve necesario una estimación de cómo puede afectar las metodologías ágiles utilizadas en las empresas los desarrollos de software seguro, para que estas lo tengan en cuenta al momento de decidir cuál metodología utilizar en el proyecto de una manera efectiva.

Así como las empresas colombianas se han visto afectadas con impactos a nivel económico cercanos a los US$500 millones anuales por este tipo de incidentes es aún más preocupante según el informe de la empresa española Telefónica (Revistadinero, 2016), donde aseguran que el 43% de las empresas en Colombia no posee planes de respuesta frente a este tipo de incidentes y todavía más sorprendente es que el 94% de las grandes empresas del mundo reportó algún tipo de ataque cibernético. Mientras que sus pérdidas alcanzaron el billón de dólares, lo que nos indica que no es un inconveniente que se presente en una zona en particular si no que es una temática de conocimiento y cultura en la que nos encontramos rezagados. Además, esto se convierte en un riesgo potencial para toda la sociedad debido al avance tecnológico en el que no encontramos en donde para realizar muchas de las actividades cotidianas que consideramos normales dependemos de aplicaciones o desarrollos de software que probablemente tengan fallas en su seguridad desde las transacciones bancarias, sistemas económicos bolsas de valores hasta sistemas diseñados para el uso de armas automatizadas, lo que nos pone en evidencia que es un riesgo con el que estamos viviendo constantemente y que debemos solucionar comenzando a priorizar la seguridad en el software al mismo nivel de importancia que la funcionalidad a desarrollar.

Un reflejo de lo anterior, es la información revelada por el Siedco (Sistema de Información Estadístico Delincuencial y Contravencional) de la Policía Nacional (Naranjo, 2017), el cual informa que las denuncias realizadas por delitos informáticos en la ciudad de Bogotá entre el periodo de 2016 – 2017 alcanza la cifra 1256, donde los ataques comunes son ‘ransomware’ secuestro de información digital a cambio de una recompensa, ’phishing’ suplantación de identidad para robar datos personales, financieros, credenciales y el ‘Malware’ software encargado de dañar o modificar un sistema operativo para robar información confidencial. La tipificación de las denuncias se observa en la siguiente tabla.

Tabla 1 Delitos informáticos en Bogotá

Denuncias por delitos Informáticos 2016 - 2017 en Bogotá

Descripción conducta	2016	2017
Hurto por medios informáticos y semejantes	576	49
Violación de datos personales	277	45
Acceso abusivo a un sistema informático	212	21
Transferencia no consentida de activos	22	3
Suplantación de sitios web para capturar datos personales	15	3
Daño informático	12	0
Interceptación de datos informáticos	8	0
Uso de software malicioso	7	3
Obstaculización ilegítima de sistema informático o red de telecomunicación	3	0
Total	**1.132**	**124**

Fuente: Periódico ADN Bogotá (Naranjo, 2017)

1.2 FORMULACIÓN DEL PROBLEMA

¿Cuáles son las implicaciones de mayor impacto en el desarrollo de software seguro con metodologías ágiles en las empresas colombianas?

1.3 JUSTIFICACIÓN

Una de las necesidades más importantes en las empresas es obtener una estimación real del dinero y tiempo que costara ejecutar determinado proyecto de software, si adicionalmente se incluye la importancia de la seguridad en la aplicación es muy probable que los resultados de la estimación sean erróneos, esto se debe por el desconocimiento sobre cual metodología usar en las empresas según el alcance del proyecto de software.

Adicionalmente construir software seguro contribuye al modelo de continuidad del negocio del SGSI (sistema de gestión de seguridad de la información) en la medida que permite identificar y analizar las vulnerabilidades que puedan existir en el software para posteriormente diseñar estrategias de recuperación y establecer procedimientos correspondientes a la moderación de los ataques al sistema.

Se requiere un estudio el cual analice la manera como se está realizando este tipo de proyectos, si hacen uso de una metodología ágil o tradicional, cual es la percepción de la importancia en la seguridad para el software construido y qué estrategias contemplan para mitigar ataques a las aplicaciones desarrolladas, dada la tendencia en el aumento de ataques informáticos que se han presentado en los últimos años alrededor del mundo y la importancia en la información que se puede extraer en cada ataque además, de la funcionalidad que cumple el sistema atacado.

Se espera que los resultados obtenidos en el presente estudio aporten un contexto claro de cómo se encuentran las empresas colombianas en el uso de buenas prácticas para realizar proyectos de software seguro con metodologías ágiles de una manera correcta y eficiente.

1.4 OBJETIVOS

1.4.1 Objetivo general

Determinar las principales implicaciones de seguridad en metodologías ágiles de desarrollo de software en el contexto colombiano.

1.4.2 Objetivos específicos

- Realizar encuesta sobre cómo se desarrollan los proyectos de software seguro en las casas desarrolladoras y empresas de tecnología cuando usan metodologías ágiles.
- Realizar prueba piloto de la encuesta.
- Identificar falencias y hacer correcciones.
- Implementar la encuesta en las empresas.
- Analizar e ilustrar los resultados obtenidos.
- Concluir las consecuencias de integrar la construcción de software seguro a las metodologías ágiles.

1.5 ALCANCES Y LIMITACIONES

1.5.1 Alcance

El estudio se realiza para que sea una referencia a las empresas que desean incluir seguridad en sus proyectos tecnológicos realizados con las metodologías ágiles para que de esta manera puedan escoger la metodología a utilizar según sea el contexto del proyecto. Las empresas consultadas se observan en el Anexo 1.

1.5.2 Limitaciones

El estudio se realizó a empresas escogidas aleatoriamente que en su momento se encontraron interesadas por los resultados de esta investigación, la encuesta trata aspectos generales en el desarrollo de los proyectos en el marco de las siguientes metodologías ágiles: Scrum, XP, Open Up y las buenas prácticas en el desarrollo de software seguro.

Adicional en las encuestas realizadas no se tendrá en cuenta el nombre de las empresas consultadas ni del encuestado, esto se realizó con el fin de que la persona brinde la información solicitada de la forma más honesta posible, solamente se solicitara el cargo que ejerce en la empresa y la ciudad o municipio en el labora.

1.6 LÍNEA DE INVESTIGACIÓN

Este trabajo se enmarca dentro de las líneas de investigación de Ingeniería de Software y la línea de Seguridad Informática, establecidas en el grupo de investigación GRIDNTIC del programa de Ingeniería de Sistemas de la Fundación Universitaria Los Libertadores.

Dando cumplimiento a los objetivos del plan estratégico al formular este proyecto para que permita dar respuesta a los problemas centrales de las líneas de investigación previamente mencionadas, Ingeniería de software debido al desarrollo del ciclo de vida del software y seguridad informática al observar como la inclusión de la seguridad informática afecta el desarrollo de los proyectos realizados con metodologías ágiles.

1.7 IMPACTO

El proyecto intenta, por una parte, contextualizar de una manera general como las empresas están implementando las metodologías ágiles cuando incluyen el desarrollo de software seguro en sus proyectos.

Por otra parte, los resultados obtenidos podrán ayudar a las empresas a identificar fases que no han tenido en cuenta al implementar proyectos de software.

2. MARCO REFERENCIAL

2.1 MARCO DE ANTECEDENTES

Un primer trabajo relacionado a la investigación corresponde a (Bisogno, 2004) quien realizó la tesis sobre “Metodología para el Aseguramiento de Entornos Informatizados – MAEI”. En este trabajo se desarrolló la metodología MAEI la cual tiene como enfoque principal realizar proyectos de software enfocados a la seguridad, para esto se en foco en tres características principales para cualquier entorno informatizado que son: integridad, confidencialidad y disponibilidad, además del estudio de varias metodologías de seguridad. Como resultado de este trabajo se obtuvo una herramienta que el ingeniero de software o la persona dedicada a realizar un proyecto de software enfocado en seguridad puede utilizar o tener en cuenta para la ejecución exitosa de este tipo de proyectos.

Un segundo trabajo de (Vianca Rosa Vega Zepeda, 2016) denominado “GUÍA PARA LA INCORPORACIÓN DE PRÁCTICAS DE SEGURIDAD EN ELMERCADO CHILENO DE DESARROLLO DE SOFTWARE” apoyado por una investigación de campo, en el cuál se utilizó la técnica de encuesta para recoger datos sobre como las empresas chilenas concebían el desarrollo de software con la seguridad informática y que herramientas, metodologías o estrategias utilizaban para verificar la seguridad de las aplicaciones realizadas. Este estudio dejó como resultados un conjunto de recomendaciones para incorporar prácticas de desarrollo de software seguro, en base a la realidad del mercado chileno, algunas de las conclusiones más relevantes de esta investigación fueron:

- Los conocimientos sobre las prácticas de seguridad en las empresas chilenas en su mayoría no las conocen o no las aplican.
- Los esfuerzos para mejorar el desarrollo se concentran principalmente en la etapa de requerimientos.
- Algunas empresas aplican prácticas de seguridad sobre todo en la etapa de requerimientos, pero no siempre saben que hay técnicas formales para esto.

De igual manera las recomendaciones generales dadas por este trabajo son las siguientes:

- Inversión en el conocimiento a través de catálogos de conocimientos locales administrados de acuerdo a las etapas del ciclo de vida del software.
- Concentrar esfuerzos para que por lo menos una persona del equipo de desarrollo se especialice en el tema de seguridad desde dos puntos de vista: atacante y el atacado, teniendo en cuenta los patrones de ataque.

- Incorporación de los casos de abuso o mal uso, puesto que se tiene familiaridad con los casos de uso, implicando reducción en los costos asociados a su incorporación.

Una investigación realizada en ACIS por (Junco, 2016) titulada Encuesta nacional de seguridad informática comprende un estudio realizado a través de internet, contó con la participación de 121 encuestados, el estudio contempla dos propósitos específicos, en primera instancia mostrar el panorama de las organizaciones colombianas frente a la seguridad de la información, y su respuesta a las demandas del entorno actual. En segunda medida, es un instrumento referente para Colombia, en la medida en llama la atención de todos los sectores interesados en los temas relacionados con la seguridad. La investigación compara resultados obtenidos en años posteriores 2014 y 2015.

Las conclusiones más relevantes obtenidas sobre esta investigación son:

- Continúa afianzando la transformación de paradigmas de la seguridad de la información en las organizaciones y su relación con los directivos de las mismas, las juntas directivas cada vez más se involucran y participan en la toma de decisiones. Esto se ajusta a la realidad mundial sobre los temas que se encuentran en el radar de los ejecutivos.
- Las empresas siguen en el camino de entender la seguridad de la información como un mecanismo para asegurar la organización. En esta visión existen aproximaciones para entenderla como un orientador de negocio. No obstante, algunos todavía ven en la seguridad de la información sólo herramientas y tecnologías de apoyo.
- Las regulaciones nacionales e internacionales son mecanismos que apoyan el fortalecimiento de los sistemas de gestión de seguridad de la información. Hoy existen en Colombia normativas como la regulación en los sectores financieros y la ley de protección de datos personales. Las regulaciones internacionales inclinan la balanza hacia la seguridad de la información y nos enfrentan a un panorama todavía denso, en materia de ataques informáticos
- Los estándares internacionales de la industria se ven reflejados en Colombia en las buenas prácticas en seguridad de la información, De ahí que ISO 27000, ITIL y Cobit se consoliden como marcos para construir arquitecturas de seguridad de la información. Por otro lado, los participantes reflejan con énfasis la necesidad de utilizar algún marco de referencia, que les permita construir modelos adaptados a las necesidades de las empresas.

2.2 MARCO TEÓRICO

2.2.1 Seguridad en el software

Según (García M. , 2013) la seguridad en el software es la aplicación de los principios de seguridad de información al desarrollo de software, la seguridad de la información se referencia a la protección de sistemas de información contra el acceso desautorizado o a la modificación de la mima, si está en una fase de almacenamiento, procesamiento o tránsito.

Para (McGraw, Software Security, 2006) la seguridad en el software es una idea de la Ingeniería de Software que busca que un producto desarrollado continúe funcionando correctamente ante ataques maliciosos, es la construcción de software que puede resistir proactivamente los ataques, además que un aspecto crítico de los problemas de seguridad, son los problemas que presenta el software mismo.

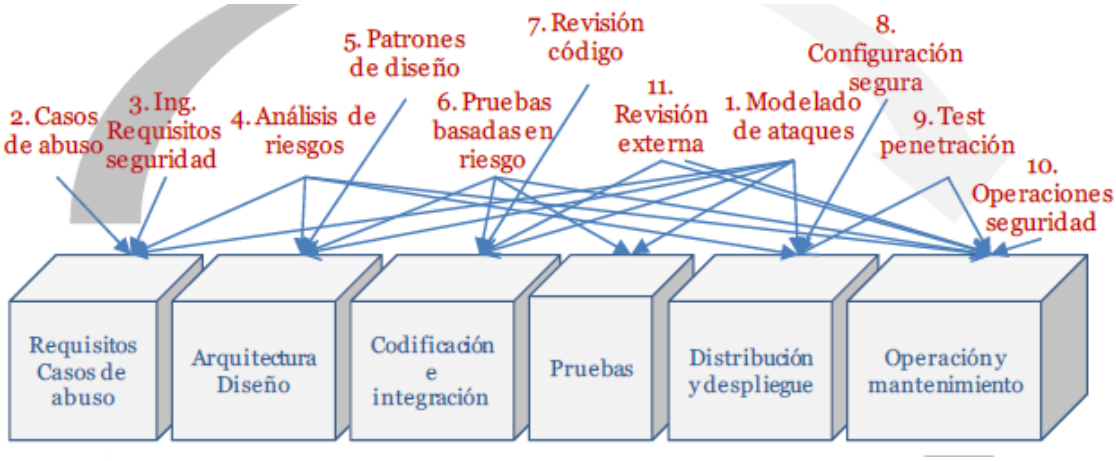
McGraw enfatiza que la seguridad no es un aspecto que pueda ser incorporado a un sistema en cualquier instante, esto implica que, al diseñar un producto de software, este diseño debe realizarse pensando en el aspecto de seguridad, dado que, una vez desarrollado el producto, no será posible incorporarla exitosamente. Interpretando lo anterior McGraw propone en su libro software security métodos que se pueden adaptar a la forma en la que se desarrolla el software, entre los más relevantes se encuentran:

- Revisión de código utilizando herramientas de análisis estático
- Análisis de riesgo arquitectónico
- Pruebas de penetración
- Pruebas de seguridad
- Desarrollo de casos de abuso

2.2.2 Seguridad en el ciclo de vida del software

La seguridad en el ciclo de vida del software es la adopción de un modelo encargado de abordar la seguridad para cada una de las fases del ciclo de vida, uno de los más adecuados referentes Gary McGraw muestra un modelo para desarrollar seguridad en el software para cada una de las etapas de ciclo de vida, como se puede apreciar en la gráfica 1.

Gráfica 1 Seguridad en el ciclo de vida del software



Fuente: Seguridad en el ciclo de vida del software (Unir, 2016) Adaptada de (McGraw, Exploiting Software: How to Break Code., 2004)

En la gráfica 1 se puede apreciar cada una de las etapas de ciclo de vida del software, adicionalmente en la parte superior están especificados los artefactos correspondientes al desarrollo de software seguro según la etapa.

2.2.3 Tipos de software en la seguridad

Comúnmente el tipo de software de seguridad más conocido es el antivirus, sin embargo, hay otras componentes de seguridad encargadas de complementar la función de los antivirus algunos de estos son:

- Programas antivirus
- Cortafuegos
- Filtro Spam
- Software para filtrar contenido
- Software contra publicidad no deseada
- Control de sitios web

2.2.4 Seguridad en el análisis de requerimientos

Definido por (García M. , 2013), en esta etapa, se deben identificar aquellos requerimientos funcionales que tendrán impacto en los aspectos de seguridad de la aplicación. Algunos de ellos son: requerimientos de compliance con normativas

locales o internacionales (ej: cifrado de mensajes enviados y recibidos, protección de información personal), tipo de información que se transmitirá o procesará (ej: Información pública o confidencial, datos personales, datos financieros, contraseñas, datos de pago electrónico, etc.) y requerimientos de registros de auditoría (ej: Qué debe registrar la aplicación en sus Logs).

2.2.5 Seguridad en el diseño

Antes de iniciar la fase de programación, hay numerosos aspectos de seguridad que deben ser tenidos en cuenta durante el diseño de la aplicación.

2.2.5.1 Diseño de autorización

Es donde se debe definir cuáles son los roles que debe tener el sistema, los privilegios que tenga cada uno de esos roles y él porque, así como los permisos para realizar acciones que modifiquen la información previa del sistema.

2.2.5.2 Diseño de autenticación

Basado en la definición de (García M. , 2013) en esta etapa se debe diseñar el modo en el que los usuarios se van a autenticar, contemplando aspectos tales como los mecanismos o factores de autenticación con contraseñas, tokens, certificados, etc. posibilidades de integrar la autenticación con servicios externos como LDAP, Radius o Active Directory y los mecanismos que tendrá la aplicación para evitar ataques de diccionario o de fuerza bruta ejemplo: bloqueo de cuentas, implementación de “captchas”, entre otras.

2.2.5.3 Diseño de los mensajes de error y advertencia

Para evitar que los mismos brinden demasiada información y que ésta sea utilizada por atacantes que logran obtener información personal desde estos mensajes.

2.2.5.4 Diseño de los mecanismos de protección de datos

En este diseño se debe contemplar el modo en el que se protegerá la información sensible en tránsito o almacenada; según el caso, se puede definir la implementación de encriptación, hashes o truncamiento de la información.

Definido por (García M. , 2013) una vez que se cuenta con el diseño detallado de la aplicación, una práctica interesante es la de realizar sobre el mismo un análisis de riesgo orientado a software. Existen técnicas documentadas al respecto, tales como Threat Modeling. Estas técnicas permiten definir un marco para identificar debilidades de seguridad en el software, antes de la etapa de codificación. Como

valor agregado, del análisis de riesgo orientado a software se pueden obtener casos de prueba para ser utilizados en la etapa de Testing/QA.

2.2.6 Seguridad en la codificación

Una vez concluido el diseño, inicia la etapa de desarrollo el momento de codificar los distintos componentes de la aplicación. Es en este punto en donde suelen incorporarse, por error u omisión, distintos tipos de vulnerabilidades. Estas vulnerabilidades podríamos dividirlas en dos grandes grupos.

2.2.6.1 Vulnerabilidades clásicas

Para (García M. , 2013) un ejemplo de estas vulnerabilidades son las presentes en el “OWASP Top 10” (Vulnerabilidades de inyección, Cross Site Scripting, errores en manejo de sesiones, etc.) como así también otras vulnerabilidades no ligadas directamente con las aplicaciones WEB, como desbordamiento de buffer, denegación de servicio, entre otros. Los ‘Frameworks’ de desarrollo de aplicaciones son una buena ayuda en este punto, ya que ofician de intermediario entre el programador y el código, y permiten prevenir la mayoría de las vulnerabilidades conocidas. Ejemplos de estos frameworks son Struts, Ruby on Rails y Zope.

2.2.6.2 Vulnerabilidades funcionales

Según lo apreciado por (García M. , 2013) son aquellas ligadas específicamente a la funcionalidad de negocio que posee la aplicación, por lo que no están previamente categorizadas y dependen directamente de la información o servicios que utilice el negocio.

Algunos ejemplos de este tipo de vulnerabilidad son los siguientes: una aplicación de banca electrónica que permite realizar transferencias con valores negativos, un sistema de subastas que permite ver los valores de otros oferentes, un sistema de venta de entradas para espectáculos que no impone límites adecuados a la cantidad de reservas que un usuario puede hacer.

Todo lo anteriormente descrito se realiza para mitigar los incidentes de seguridad en lo que nos podemos referir a:

- Acceso no autorizado a la información.
- Descubrimiento de información.
- Modificación no autorizada de datos.
- Invasión a la privacidad.
- Denegación de servicios.

2.2.7 Malware

Malware es la definición de (Malicius Sotware), para (Quintero, 2011) Se denomina malware al software malicioso, diseñado para llevar cabo acciones no deseadas y sin el consentimiento explícito del usuario.

El malware se clasifica en:

- Virus: auto-réplica, infectan otros programas.
- Gusano: se replica mediante copias de sí mismo, pero no infecta a otros programas.
- Troyano: no se replica ni infecta a otros programas de forma automática e indiscriminada.
- Adware: presenta publicidad no deseada.
- Keylogger: captura pulsaciones en el teclado, espía lo que el usuario escribe.
- Rootkit: usa técnicas para permanecer oculto en el sistema ante el usuario y las aplicaciones de seguridad.
- Backdoor: función de puerta trasera, permite al atacante conectarse y controlar la máquina infectada.
- Dialer: realiza llamadas de tarificación especial, incrementando la factura telefónica.
- Bot: los sistemas infectados son “zombies” que conforman una “botnet” o red de bots; esta red acepta órdenes de forma remota.
- Ransomware: cifra documentos y archivos, pide al usuario que pague un rescate si quiere la clave que permita acceder a los originales.
- Rogueware: falso antivirus, te hacen creer que el sistema está infectado y cobran para la supuesta desinfección.
- Crimeware: nueva denominación para el malware orientado al cibercrimen y fraude, con un claro interés de lucro.

2.2.7.1 Metodología de análisis de malware

Como es propuesto en (Casey, 2008) Cuando se descubre un malware en un sistema, hay muchas decisiones y acciones que deben tomarse, a menudo bajo una severa presión de tiempo. Para ayudar a los investigadores digitales a lograr un resultado exitoso, se puede utilizar una metodología general para tratar estos incidentes, Investigaciones que involucran malware en cinco fases:

1. Preservación forense y examen de datos volátiles
   El valor de los datos volátiles no se limita a la memoria de proceso asociada con el malware, sino que puede incluir contraseñas, direcciones IP (Protocolo

de Internet), entradas del registro de eventos de seguridad y otros detalles contextuales que pueden proporcionar una comprensión más completa del malware y su uso en un sistema. En un estado de encendido, un sistema sujeto contiene información efímera crítica que revela el estado del sistema. A veces, estos datos volátiles se denominan información con estado. Incidente de respuesta forense, o respuesta en vivo, es el proceso de adquisición de la información de estado del sistema sujeto mientras permanece encendido.

2. Examen de la memoria
   Después de adquirir una imagen de memoria física, se requiere extraer información, contenido de una manera metódica. Una captura de memoria completa puede contener evidencia crítica en un incidente de código malicioso, incluso cuando se inició el malware, los argumentos de línea de comandos utilizados, los procesos ocultos y terminados, las direcciones IP con las que se comunicaba el malware y los datos en texto cifrado en disco. Algunas herramientas forenses de memoria pueden listar archivos abiertos, conexiones de red activas y procesos en ejecución, e incluso pueden mostrar información sobre procesos que están ocultos o que ya no se ejecutan pero que aún están presentes en la memoria.

   Aunque los investigadores digitales suelen encontrar información útil en los vertederos de memoria simplemente revisando texto legible y realizar búsquedas de palabras clave, contexto adicional y metadatos sólo pueden ser obtenidos utilizando el conocimiento especializado de las estructuras de datos en la memoria. La localización de datos asociados con un proceso específico se complica por el hecho de que los sistemas operativos Windows y Linux utilizan sistemas virtuales, direcciones para crear la ilusión de más memoria que físicamente existe. Como resultado, para encontrar una determinada pieza de datos, es necesario traducir direcciones virtuales en una ubicación física. Además, la ubicación física de los datos puede estar en un archivo de página en disco en vez que en el volcado de la memoria física.

3. Análisis Forense Examen de los discos duros
   El examen forense de sistemas Windows es una parte importante del análisis de código malicioso, proporcionando contexto e información adicional que nos ayudan a entender la funcionalidad y el origen del malware. En la medida en que el análisis de sistemas encendidos puede considerarse una cirugía, el examen forense puede considerarse una autopsia de una computadora afectada por malware. La evidencia de rastreo relacionada con una pieza particular de malware se puede encontrar en los sistemas operativos y el

sistema de archivos, incluyendo archivos, entradas de registro, registros en registros de eventos y sellos de fecha asociados.

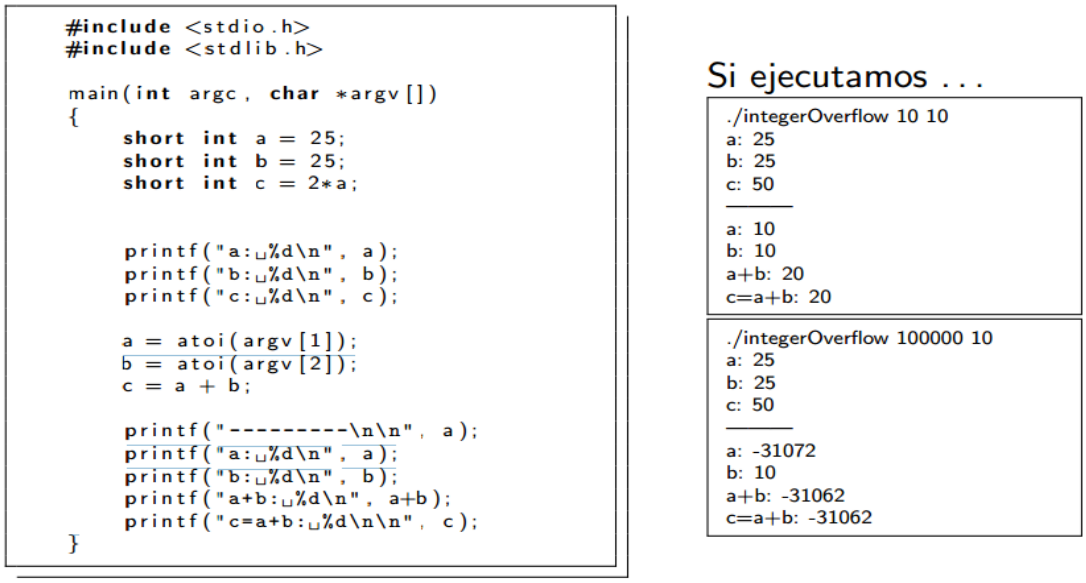
4. Análisis estático de malware
   Son las técnicas y herramientas para realizar un análisis inicial de un archivo sospechoso con el objetivo de un análisis del código ensamblador del malware para conseguir una mejor compresión y funcionamiento del mismo.

5. Análisis dinámico de malware:
   El análisis dinámico o de comportamiento implica ejecutar el código y supervisar su comportamiento, interacción y efecto en el sistema host, mientras que el análisis estático es el proceso de análisis de código binario ejecutable sin ejecutar realmente el archivo.

2.2.8 Desbordamiento de buffer

Como se aprecia en (McGraw, Exploiting Software: How to Break Code., 2004), Los Desbordamientos de búfer son un tipo de vulnerabilidad de uso de memoria. Esto es sobre todo un accidente de historia de la ciencia de computadoras. La memoria fue un recurso precioso, de esta manera gestionar la memoria era crítico. En algunos sistemas más antiguos, como la nave espacial Voyager, la memoria era tan valiosa que una vez que determinadas secciones de código de máquina ya no eran necesarias, el código era borrado para siempre del módulo de memoria, liberando espacio para otros usos. Esto efectivamente creó un programa que era autodestructivo y sólo se podía ejecutar una vez. La mayoría de los sistemas de software conectados a la red hoy en día tienen problemas abominables de memoria, especialmente cuando conectado directamente a ambientes hostiles como la Internet. La memoria es barata, pero los efectos de la mala gestión de memoria son muy caros. El mal uso de la memoria puede conducir a la corrupción interna dentro de un programa (especialmente con referencia a un control de flujo), problemas de denegación de servicio. A continuación, se muestra un ejemplo tomado de (García F. T., 2015), el problema es definido como desbordamiento de enteros, sucede cuando el tipo de datos seleccionado para almacenar cierta información no es capaz de albergar los datos del programa, ocurre al olvidar que los tipos de datos son capaces de almacenar una cantidad limitada de ellos, así como se puede observar en la siguiente gráfica.

Gráfica 2 Desbordamiento de buffer por enteros



```c
#include <stdio.h>
#include <stdlib.h>

main(int argc, char *argv[])
{
    short int a = 25;
    short int b = 25;
    short int c = 2*a;


    printf("a: %d\n", a);
    printf("b: %d\n", b);
    printf("c: %d\n", c);

    a = atoi(argv[1]);
    b = atoi(argv[2]);
    c = a + b;

    printf("---------\n\n", a);
    printf("a: %d\n", a);
    printf("b: %d\n", b);
    printf("a+b: %d\n", a+b);
    printf("c=a+b: %d\n\n", c);
}
```

Si ejecutamos . . .

```
./integerOverflow 10 10
a: 25
b: 25
c: 50
---------
a: 10
b: 10
a+b: 20
c=a+b: 20
```

```
./integerOverflow 100000 10
a: 25
b: 25
c: 50
---------
a: -31072
b: 10
a+b: -31062
c=a+b: -31062
```

Fuente: Desbordamiento por enteros (García F. T., 2015)

2.2.9 Pruebas de penetración

Según (McGraw, Software Security, 2006) Las pruebas de seguridad plantean un problema único. La mayoría de los defectos y vulnerabilidades de seguridad en el software no están directamente relacionados con la funcionalidad. En su lugar, los problemas de seguridad implican mal usos inesperados pero intencionales de una aplicación descubierta por un atacante. Si caracterizamos las pruebas funcionales como "pruebas positivas" como verificando que una característica realiza adecuadamente una tarea específica, entonces la prueba de penetración es en cierto sentido "pruebas negativas". Es decir, un probador de seguridad debe investigar directa y profundamente la seguridad (Posiblemente conducidos por casos de abuso y riesgos arquitectónicos) para determinar cómo se comporta el sistema bajo ataque.

La prueba de penetración consiste en probar un sistema en su entorno de producción final. Por esta razón, las pruebas de penetración son las más adecuadas para probar problemas de configuración y otros factores ambientales que afectan profundamente la seguridad del software. Las pruebas de conducción que se concentran en estos factores con algún conocimiento de los resultados del análisis de riesgo es el enfoque más eficaz.

No se debe olvidar que el valor real de las pruebas de penetración proviene de su papel central en la validación de la configuración y otros factores ambientales esenciales.

Como herramienta de medición, las pruebas de penetración son más poderosas cuando se integran plenamente en el proceso de desarrollo de tal manera que se usan los resultados del ciclo de vida temprano para informar sobre las pruebas y que los resultados vuelvan a las prácticas de desarrollo y despliegue.

2.2.10 Pruebas de seguridad basadas en riesgo

Como es apreciado por (McGraw, Software Security, 2006) las pruebas de seguridad deben abarcar dos estrategias: (1) probar la funcionalidad de seguridad con técnicas estándar de pruebas funcionales y (2) pruebas de seguridad basadas en el riesgo basadas en patrones de ataque, resultados de análisis de riesgo y casos de abuso. Un buen plan de prueba de seguridad abarca ambas estrategias. Los problemas de seguridad no siempre son evidentes, incluso cuando se examina un sistema directamente, por lo que es poco probable que el aseguramiento de la calidad de los problemas estándar descubra todos los problemas críticos de seguridad. La garantía de calidad es asegurarse de que suceden cosas buenas. La prueba de seguridad se trata de asegurarse de que las cosas malas no sucedan. Pensar como un atacante es esencial. Guiar las pruebas de seguridad con el conocimiento de la arquitectura del software, los ataques comunes y la mentalidad del atacante es por lo tanto extremadamente importante.

Los probadores deben llevar a cabo un enfoque basado en el riesgo, basado en la realidad arquitectónica del sistema y la mentalidad del atacante, para medir la seguridad del software adecuadamente. Mediante la identificación de riesgos en el sistema y la creación de pruebas impulsadas por esos riesgos, un probador de seguridad de software puede centrarse correctamente en áreas de código en las que es probable que un ataque tenga éxito. Este enfoque proporciona un nivel más alto de garantía de seguridad del software que es posible con las pruebas clásicas de la caja negra.

Los profesionales de la seguridad de software realizan muchas tareas diferentes para gestionar los riesgos de seguridad del software, tales como:

- Creación de casos de uso indebido de la seguridad
- Listado de los requisitos normativos de seguridad (y características y funciones de seguridad)
- Realización de análisis de riesgos arquitectónicos

- Creación de planes de pruebas de seguridad basadas en el riesgo
- Manejo de herramientas de análisis estático
- Realización de pruebas de seguridad
- Realización de pruebas de penetración en el entorno final
- Limpieza tras infracciones de seguridad

Las pruebas de seguridad deben necesariamente incluir dos enfoques diferentes:

1. Pruebas de seguridad funcional: prueba de los mecanismos de seguridad para garantizar que su funcionalidad se implemente correctamente.

2. Pruebas de seguridad adversas: realizar pruebas de seguridad basadas en el riesgo motivadas por la comprensión y la simulación del enfoque del atacante.

2.2.11 Casos de abuso

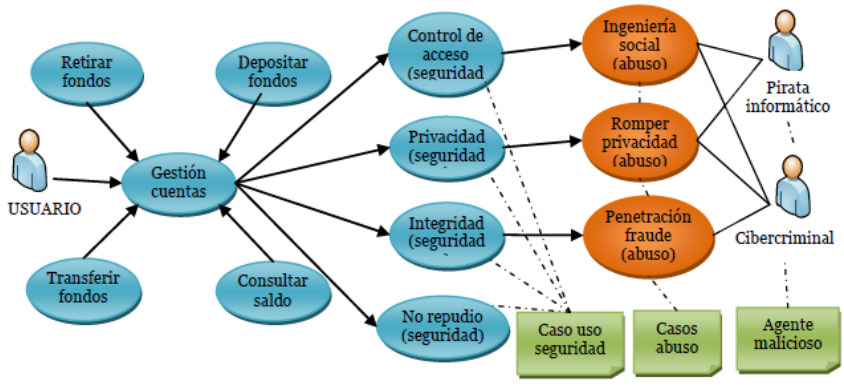
Definido por (McGraw, Software Security, 2006) el artefacto de los casos de abuso es una manera adecuada para entrar en la mente del atacante. Similar a los casos de uso, casos de abuso describen el comportamiento del sistema bajo ataque, la construcción de estos casos requiere cobertura explícita de lo que debe ser protegido, de quién y por cuánto tiempo.

Para crear los casos de abuso se debe tener en cuenta:

- identificar y documentar actores o agentes que podrían ejecutar un ataque.
- crear anti-requerimientos.
- crear un modelo de ataque.

En la gráfica 3 se puede observar el diagrama UML de un caso de abuso

Gráfica 3 Ejemplo caso de abuso



Fuente: Ejemplo caso de abuso (Unir, 2016)

2.2.12 Metodologías ágiles

Como se define por (Alaimo, 2013) las metodologías ágiles emergen en la década de los ‘90s en donde surgen varios movimientos identificados con el nombre de Metodologías Livianas (Lightweight Methodólogies). Entre estos se encuentran Extreme Programming (XP), Scrum, Software Craftsmanship, Lean Software Development, encontradas entre las más populares. Más tarde, en febrero de 2001, se reunieron en Utah (EEUU) un grupo de diecisiete profesionales reconocidos del desarrollo de software, y referentes de las metodologías livianas existentes al momento, con el objetivo de determinar los valores y principios que les permitirían a los equipos desarrollar software de forma más acertada con las necesidades del cliente y responder mejor a los cambios que pudieran surgir a lo largo de un proyecto de desarrollo. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por la rigidez y dominados por la documentación. En esta reunión se creó la Agile Alliance, una organización sin fines de lucro cuyo objetivo es el de promover los valores y principios de la filosofía ágil y ayudar a las organizaciones en su adopción. También se declaró la piedra angular del movimiento ágil, conocida como Manifiesto Ágil (Beck & Beedle, 2001).

2.2.12.1 Manifiesto ágil

El manifiesto Ágil como se puede observar en (Beck & Beedle, 2001) se resume en cuatro valores y doce principios.

Entre los valores destacan la cooperación del trabajo en equipo, eficiencia y eficacia para el desarrollo de los entregables entre otros que se observan a continuación:

1. **Valorar a las personas y las interacciones entre ellas por sobre los procesos y las herramientas:**
   Las personas son el principal factor de éxito de un proyecto de software. Es más importante construir un buen equipo que construir el contexto. Muchas veces se comete el error de construir primero el entorno de trabajo y esperar que el equipo se adapte automáticamente. Por el contrario, la agilidad propone crear el equipo y que éste construya su propio entorno y procesos en base a sus necesidades.

2. **Valorar el software funcionando por sobre la documentación detallada:**
   La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante”. Estos documentos deben ser cortos y centrarse en lo esencial. La documentación (diseño, especificación técnica de un sistema) no es más que un resultado intermedio y su finalidad no es dar valor en forma directa al usuario o cliente del proyecto.

Medir avance en función de resultados intermedios se convierte en una simple "ilusión de progreso".

3. **Valorar la colaboración con el cliente por sobre la negociación de contratos:**
   Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta mutua colaboración será la que dicte la marcha del proyecto y asegure su éxito.

4. **Valorar la respuesta a los cambios por sobre el seguimiento estricto de los planes:**
   La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también su éxito o fracaso. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores son los pilares sobre los cuales se construyen los doce principios del Manifiesto Ágil. De estos doce principios, los dos primeros son generales y resumen gran parte del espíritu ágil del desarrollo de software, mientras que los siguientes son más específicos y orientados al proceso o al equipo de desarrollo:

1. Nuestra mayor prioridad es satisfacer al cliente a través de entregas tempranas y frecuentes de software con valor.
2. Aceptar el cambio incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan los cambios para darle al cliente ventajas competitivas.
3. Entregar software funcionando en forma frecuente, desde un par de semanas a un par de meses, prefiriendo el periodo de tiempo más corto.
4. Expertos del negocio y desarrolladores deben trabajar juntos diariamente durante la ejecución del proyecto.
5. Construir proyectos en torno a personas motivadas, generándoles el ambiente necesario, atendiendo sus necesidades y confiando en que ellos van a poder hacer el trabajo.
6. La manera más eficiente y efectiva de compartir la información dentro de un equipo de desarrollo es la conversación cara a cara.
7. El software funcionando es la principal métrica de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible. Los sponsors, desarrolladores y usuarios deben poder mantener un ritmo constante indefinidamente.
9. La atención continua a la excelencia técnica y buenos diseños incrementan la agilidad.
10. La simplicidad –el arte de maximizar la cantidad de trabajo no hecho- es esencial.

11. Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto-organizados.
12. A intervalos regulares, el equipo reflexiona acerca de cómo convertirse en más efectivos, luego mejora y ajusta su comportamiento adecuadamente.

2.2.12.2 Scrum

Tal como se define por el autor (Alaimo, 2013)**,** Scrum es un marco de trabajo que nos permite encontrar prácticas emergentes en dominios complejos, como la gestión de proyectos de innovación. No es un proceso completo, y mucho menos, una metodología. En lugar de proporcionar una descripción completa y detallada de cómo deben realizarse las tareas de un proyecto, genera un contexto relacional e iterativo, de inspección y adaptación constante para que los involucrados vayan creando su propio proceso. Esto ocurre debido a que no existen ni mejores ni buenas prácticas en un contexto complejo.

El equipo de desarrollo se encuentra apoyado en dos roles: el Scrum Master y el Product Owner. El Scrum Master es quien vela por la utilización de Scrum, la remoción de impedimentos y asiste al equipo a que logre su mayor nivel de performance posible. Puede ser considerado un coach o facilitador encargado de acompañar al equipo de desarrollo. El Product Owner es quien representa al negocio, stake holders, cliente y usuarios finales. Tiene la responsabilidad de conducir al equipo de desarrollo hacia el producto adecuado.

El progreso de los proyectos que utilizan Scrum se realiza y verifica en una serie de iteraciones llamadas Sprints. Estos Sprints tienen una duración fija, pre-establecida de no más de un mes. Al comienzo de cada Sprint el equipo de desarrollo realiza un compromiso de entrega de una serie de funcionalidades o características del producto en cuestión. Al finalizar el Sprint se espera que estas características comprometidas estén terminadas, lo que implica su análisis, diseño, desarrollo, prueba e integración al producto. En este momento es cuando se realiza una reunión de revisión del producto construido durante el Sprint, donde el equipo de desarrollo muestra lo construido al Product Owner y a cualquier stake holder interesado en participar. El feedback obtenido en esta reunión puede ser incluido entre las funcionalidades a construir en futuros Sprints.

Definido por (Softeng, 2016) los beneficios que maneja esta metodología son:

- Cumplimento de expectativas: El cliente establece sus expectativas indicando el valor que le aporta cada requisito / historia del proyecto, el equipo los estima y con esta información el Product Owner establece su prioridad. De manera regular, en las demos de Sprint el Product Owner comprueba que efectivamente los requisitos se han cumplido y transmite se feedback al equipo.

- Flexibilidad a cambios: Alta capacidad de reacción ante los cambios de requerimientos generados por necesidades del cliente o evoluciones del mercado. La metodología está diseñada para adaptarse a los cambios de requerimientos que conllevan los proyectos complejos.
- Reducción del Time to Market: El cliente puede empezar a utilizar las funcionalidades más importantes del proyecto antes de que esté finalizado por completo. Mayor calidad del software: La metódica de trabajo y la necesidad de obtener una versión funcional después de cada iteración, ayuda a la obtención de un software de calidad superior.
- Mayor productividad: Se consigue entre otras razones, gracias a la eliminación de la burocracia y a la motivación del equipo que proporciona el hecho de que sean autónomos para organizarse.
- Maximiza el retorno de la inversión (ROI): Producción de software únicamente con las prestaciones que aportan mayor valor de negocio gracias a la priorización por retorno de inversión.
- Predicciones de tiempos: Mediante esta metodología se conoce la velocidad media del equipo por sprint (los llamados puntos historia), con lo que consecuentemente, es posible estimar fácilmente para cuando se dispondrá de una determinada funcionalidad que todavía está en el Backlog.
- Reducción de riesgos: El hecho de llevar a cabo las funcionalidades de más valor en primer lugar y de conocer la velocidad con que el equipo avanza en el proyecto, permite despejar riesgos eficazmente de manera anticipada.

2.2.12.3 Extreme Programming (XP)

Tal como lo explica, (González, 2010)**,** La programación extrema (en adelante XP) puede que marque un antes y un después en la ingeniería del software.

Creada por Kent Beck, Ward Cunninghamn y Ron Jeffries a finales de los noventa, la programación extrema ha pasado de ser una simple idea para un único proyecto a inundar todas las "factorías de software". Algunos la definen como un movimiento "social" de los analistas del software hacia los hombres y mujeres de negocios, de lo que debería ser el desarrollo de soluciones en contraposición a los legalismos de los contratos de desarrollo.

Para alcanzar el objetivo de software como solución ágil, la metodología XP se estructura en tres capas que agrupan las doce prácticas básicas de XP véase la gráfica 4.

Gráfica 4 Capas metodología XP

Fuente: Capas metodología XP. (González, 2010)

1) Metodología de programación: diseño sencillo, test, refactorización y codificación con estándares.

2) Metodología de equipo: propiedad colectiva del código, programación en parejas, integración continua, cuarenta horas semanales y metáfora del negocio.

3) Metodología de procesos: cliente en sitio, entregas frecuentes y planificación del juego.

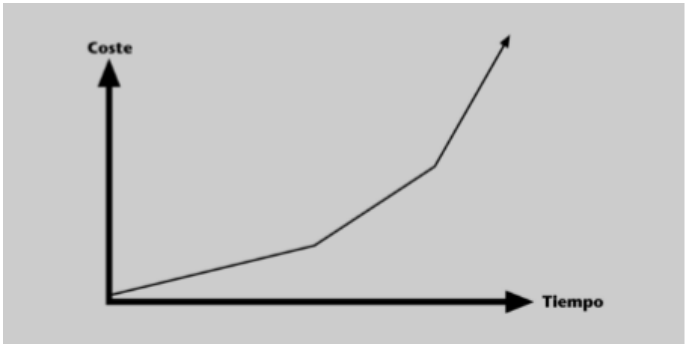
Introducir la vertiente de las relaciones sociales dentro de una metodología es lo que hace de XP algo más que una guía de buenas maneras. Convierte la programación en algo mucho más "humanizado", en algo que permite a las personas relacionarse y comunicarse para encontrar soluciones, sin jerarquías ni enfrentamientos. Los analistas y programadores trabajan en equipo con el cliente final, todos están comprometidos con el mismo objetivo, que la aplicación solvente o mitigue los problemas que tiene el cliente. La vertiente social es fundamental en otras áreas del conocimiento: por ejemplo, en las relaciones del equipo médico con el paciente, o en las de bufete de abogados con un cliente, o en las de profesorado y alumnos, cada uno tiene su función, pero el objetivo es común.

XP también humaniza a los desarrolladores. Un entorno agradable para el trabajo, que facilite la comunicación y los descansos adecuados, forma parte de esta metodología.

Pero donde XP centra la mayor innovación es en desmontar la preconcebida idea del coste del cambio de las metodologías en cascada, es decir, lo que cuesta

cambiar alguna funcionalidad de nuestro aplicativo a medida que vamos avanzando en él. La idea generalizada es que cualquier modificación a final del proyecto es exponencialmente más costosa que al principio del mismo; si algo no estaba especificado inicialmente, cuesta mucho introducirlo al final del proyecto tal como se representa en la gráfica 5.

Gráfica 5 Coste metodología en cascada

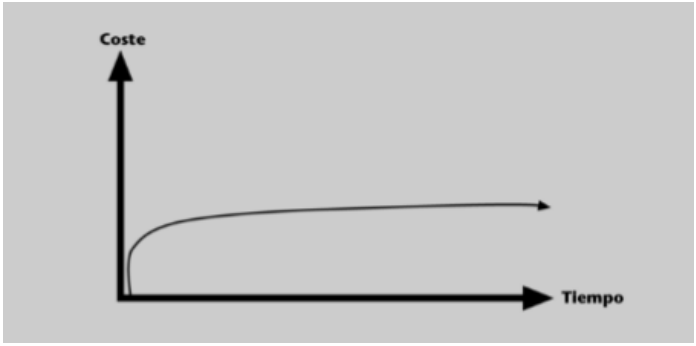


Fuente: Coste metodología en cascada. (González, 2010)

Lo que XP propugna es que esta curva ha perdido vigencia y que con una combinación de buenas prácticas de programación y tecnología es posible lograr que la curva no crezca siempre de forma exponencial.

XP pretende conseguir una curva de coste del cambio con crecimiento leve, que en un inicio es más costosa, pero que a lo largo del proyecto permite tomar decisiones de desarrollo lo más tarde posible sin que esa nueva decisión de cambio implique un alto coste en el proyecto.

Gráfica 6 Coste metodología XP



Fuente: Coste Metodología XP. (González, 2010)

XP no se olvida de la necesaria rentabilidad de los proyectos, imprescindible en una economía tan competitiva como la occidental. Por eso propone una ecuación de equilibrio entre el coste, el tiempo de desarrollo, la calidad del software y el alcance de funcionalidades del mismo.

Ventajas del Modelo XP

- Apropiado para entornos volátiles.
- Estar preparados para el cambio, significa reducir su coste.
- Planificación más transparente para nuestros clientes, conocen las fechas de entrega de funcionalidades. Vital para su negocio.
- Permitirá definir en cada iteración cuales son los objetivos de la siguiente Permite tener realimentación de los usuarios muy útil.
- La presión está a lo largo de todo el proyecto y no en una entrega final.

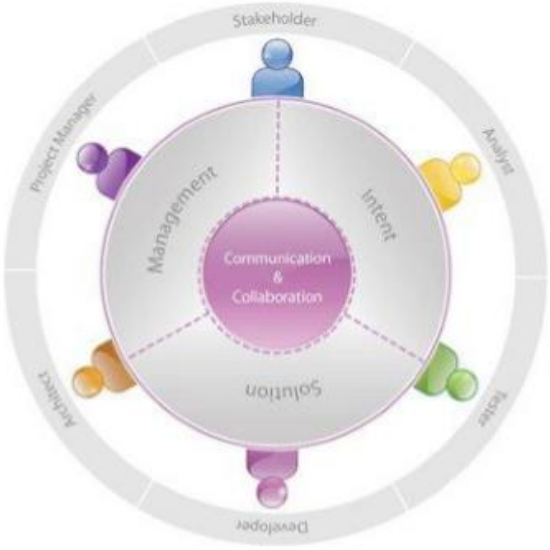
Desventajas del Modelo XP

- Delimitar el alcance del proyecto con nuestro cliente
- Para mitigar esta desventaja se plantea definir un alcance a alto nivel basado en la experiencia.

2.2.12.4 OpenUp

De acuerdo a lo mencionado por (Hernández, 2009)**,** OpenUP es un proceso ágil y unificado, que contiene el conjunto mínimo de prácticas que ayudan a los equipos a ser más eficaces en el desarrollo de software. OpenUP abraza una filosofía pragmática y ágil que se centra en la naturaleza colaborativa de desarrollo de software. Es un proceso iterativo que es Mínimo, Completo y extensible que puede utilizarse tal cual o ampliarse para tratar una amplia variedad de tipos de proyecto. Se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Está organizada dentro de cuatro áreas principales de contenido: Comunicación y Colaboración, Intención, Solución y Administración ver la gráfica 7.

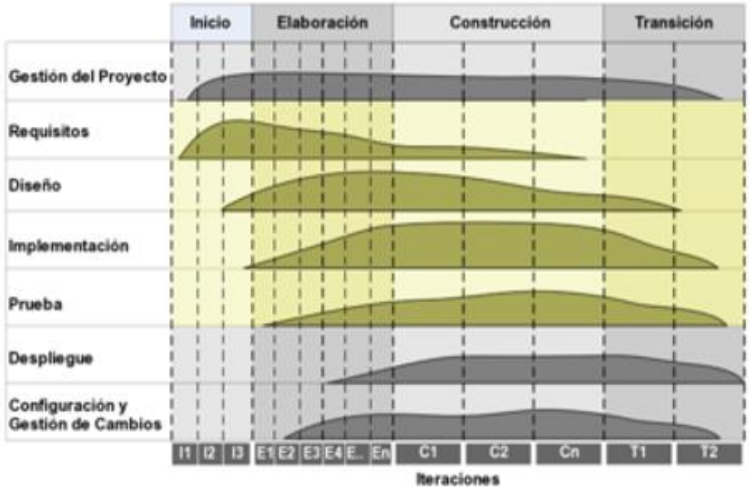
Gráfica 7 Áreas principales OpenUp



Fuente: Áreas principales OpenUp. (Hernández, 2009)

El OpenUp está organizado en dos dimensiones diferentes pero interrelacionadas: el método y el proceso. El contenido del método es donde los elementos del método (roles, tareas, artefactos y lineamientos) son definidos, sin tener en cuenta como son utilizados en el ciclo de vida del proyecto. El proceso es donde los elementos del método son aplicados de forma ordenada en el tiempo. Muchos ciclos de vida para diferentes proyectos pueden ser creados a partir del mismo conjunto de elementos del método. Como se puede observar en la gráfica 8.

Gráfica 8 Organización OpenUp



Fuente: Organización OpenUp. (Hernández, 2009)

OpenUp se caracteriza por cuatro principios básicos que se soportan mutuamente:

- Colaboración para alinear los intereses y un entendimiento compartido.
- Balance para confrontar las prioridades (necesidades y costos técnicos) para maximizar el valor para los stake holders.
- Enfoque en articular la arquitectura para facilitar la colaboración técnica, reducir los riesgos y minimizar excesos y trabajo extra.
- Evolución continua para reducir riesgos, demostrar resultados y obtener.

2.2.13 Metodologías de software seguro

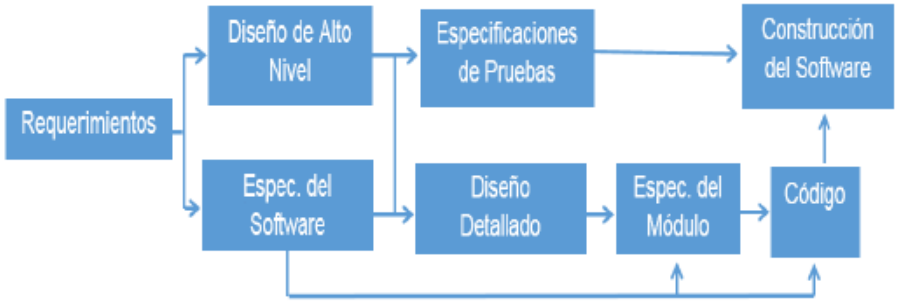
2.2.13.1 Correctness by Construction (CbyC)

Según (Abundis, 2013)**,** es un método efectivo para desarrollar software que demanda un nivel de seguridad crítico y que además sea demostrable. La empresa Praxis ha utilizado CbyC desde el año 2001 y ha producido software industrial con taza de defectos por debajo de los 0.05 defectos por cada 1000 líneas de código, y con una productividad de 30 líneas de código por persona al día.

Las metas principales de ésta metodología son obtener una taza de defectos al mínimo y una alta resiliencia al cambio; los cuales se logran debido a dos principios fundamentales: que sea muy difícil introducir errores y asegurarse que los errores sean removidos tan pronto hayan sido inyectados. CbyC busca producir un producto que desde el inicio sea correcto, con requerimientos rigurosos de seguridad, con definición muy detallada del comportamiento del sistema y un diseño sólido y verificable.

En cuanto a las fases de la metodología CbyC, estas combinan los métodos formales con el desarrollo ágil; utiliza notaciones precisas y un desarrollo incremental que permite mostrar avances para recibir retroalimentación y valoración del producto. La gráfica 9, muestra las fases propuestas por ésta metodología para el desarrollo de software

Gráfica 9 Proceso de desarrollo propuesto por el CbyC



Fuente: Proceso de desarrollo propuesto por el CbyC. (Amey, 2006)

2.2.13.2 Security Development Lifecycle (SDL)

Según (Abundis, 2013)**,** es un proceso para mejorar la seguridad de software propuesto por la compañía de Microsoft en el año 2004; con dieciséis actividades enfocadas a mejorar la seguridad del desarrollo de un producto de software.

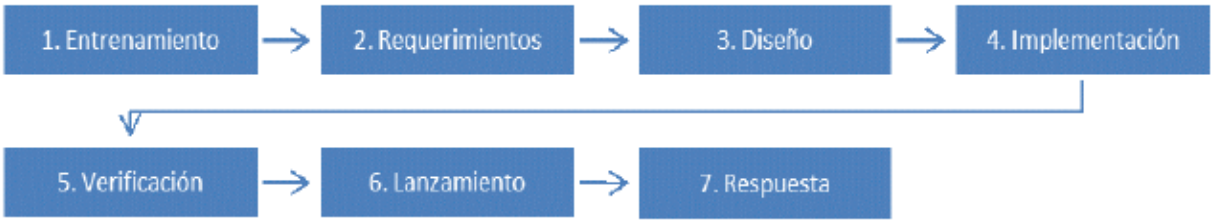
Las prácticas que propone SDL van desde una etapa de entrenamiento sobre temas de seguridad, pasando por análisis estático, análisis dinámico, fuzz testing del código hasta tener plan de respuesta a incidentes. Una de las características principales de SDL es el modelado de amenazas que sirve a los desarrolladores para encontrar partes del código, donde probablemente exista vulnerabilidades o sean objeto de ataques.

Fases de la Metodología SDL

Existen dos versiones del SDL, la versión rígida y la orientada al desarrollo ágil. Las diferencias versan en que la segunda desarrolla el producto de manera incremental y en la frecuencia de la ejecución de las actividades para el aseguramiento de la

seguridad. La versión rígida del SDL es más apropiada para equipos de desarrollo y proyectos más grandes y no sean susceptibles a cambios durante el proceso. SDL ágil es recomendable para desarrollos de aplicaciones web o basadas en la web. En la gráfica 10, se encuentra el flujo de las fases en la metodología de SDL cabe resaltar la existencia de una etapa previa a los requerimientos, enfocada al entrenamiento en seguridad y la última etapa del proceso que se encarga de darle seguimiento al producto en caso de algún incidente de seguridad.

Gráfica 10 Proceso del desarrollo software del SDL



Fuente: Proceso del desarrollo software del SDL. *(Microsoft, 2010)*

2.3 MARCO LEGAL

De manera práctica no hay leyes colombianas que traten el tema de normas o prácticas que se deberían implementar en las empresas para realizar software seguro, aun así la ley 1273 de enero de 2009 entre sus artículos trata las diferentes maneras en las que se puede atacar un sistema informático la explicación de manera breve y los castigos que repercuten sobre las personas que realicen este tipo de acciones entre las más destacadas contextualizado al tema de software seguro se encuentran en (Congreso_Colombiano, 2009)

- Una de las maneras de infringir en el delito OBSTACULIZACIÓN ILEGÍTIMA DE SISTEMA INFORMÁTICO O RED DE TELECOMUNICACIÓN es por medio de un ataque de denegación de servicios usando como método la sobrecarga de buffer.
- Se infringe en el delito INTERCEPTACIÓN DE DATOS INFORMÁTICOS, cuando se leen las cabeceras de paquetes, para desvelar la identidad de uno o más usuarios implicados en la comunicación ilegalmente intervenida (intercepción de identidad).
- Comúnmente realizan el delito DAÑO INFORMÁTICO**.** Cuando hacen uso de malware gusano, troyano y keylogger. Esta actividad también acarrea el delito de USO DE SOFTWARE MALICIOSO estos tienen una pena de prisión de cuarenta y ocho (48) a noventa y seis (96) meses y en multa de 100 a 1000 salarios mínimos legales mensuales vigentes

- El delito HURTO POR MEDIOS INFORMÁTICOS y SEMEJANTES comúnmente sucede cuando el atacante realiza la práctica ransomware para exigir una recompensa por la liberación del equipo instalado.
- TRANSFERENCIA NO CONSENTIDA DE ACTIVOS. Este tipo de delito puede realizarse cuando el atacante ejecuta inyección SQL o XML para alterar la información de un proceso bancario, por ejemplo.

Los artículos anteriormente expuestos detallan los principales ataques que se presentan en los sistemas informáticos en la mayoría de las veces debido a que los desarrollos de las aplicaciones realizadas tienen bajos estándares de seguridad en su software.

2.4 GLOSARIO

Antispam: es lo que se conoce como método para prevenir el correo basura. Tanto los usuarios finales como los administradores de sistemas de correo electrónico utilizan diversas técnicas contra ello.

CbyC: Corrección por la construcción (Correctness by Construction) es una metodología para desarrollar software seguro, con metas de obtener una taza de defectos mínimo y una alta resiliencia al cambio.

Cortafuegos: El cortafuegos (firewall en inglés) es una parte de un sistema o una red que está diseñada para bloquear el acceso no autorizado, permitiendo al mismo tiempo comunicaciones autorizadas.

CVE: Es un diccionario o catálogo público de vulnerabilidades, administrado por MITRE, que normaliza su descripción y las organiza desde diferentes tipos de vista (vulnerabilidades web, de diseño, implementación, etc.).

Metodología ágil: Las metodologías ágiles son una serie de técnicas para la gestión de proyectos que han surgido como contraposición a los métodos clásicos de gestión como CMMI. Aunque surgieron en el ámbito del desarrollo de software, también han sido exportadas a otro tipo de proyectos.

Product Owner: Como Product Owner representa al cliente, y es el encargado de negociar con el equipo, con el Scrum Master por medio como facilitador, la prioridad del trabajo a realizar. Esto desde una perspectiva del retorno de inversión para el negocio.

Proyecto de software: Es el proceso de gestión para la creación de un sistema o software, la cual encierra un conjunto de actividades, una de las cuales es la estimación; estimación es una actividad importante que no debe llevarse a cabo de forma descuidada. Existen técnicas útiles para la estimación de costes de tiempo.

Y dado que la estimación es la base de todas las demás actividades de planificación del proyecto y sirve como guía para una buena Ingeniería de Sistemas y Software.

Scrum Master: Es la persona conocedora del proceso Scrum que se encarga de orientar al equipo y al propietario del producto para que sigan el proceso determinado por Scrum. Si las personas involucradas en el proyecto ya conocen Scrum su labor puede no ser necesaria.

SDL: Ciclo de vida de desarrollo de seguridad (Security Development Lifecycle) es un proceso de control de seguridad orientado al desarrollo de software.

Seguridad informática: se refiere a las características y condiciones de sistemas de procesamiento de datos y su almacenamiento, para garantizar su confidencialidad, integridad y disponibilidad.

Software seguro: se refiere a un software que ha sido pensado desde las primeras etapas de diseño de este para tener una robustez amplia en lo que se refiere a seguridad informática.

Sprint: es la unidad básica de trabajo para un equipo Scrum. Esta es la característica principal marca la diferencia entre Scrum y otros modelos para el desarrollo ágil. Es una simple iteración llevada a cabo por los miembros del equipo

Stake holders: Se puede definir como cualquier persona o entidad que es afectada o concernida por las actividades o la marcha de una organización; por ejemplo, los trabajadores de esa organización, sus accionistas, las asociaciones de vecinos afectadas o ligadas, los sindicatos, las organizaciones civiles y gubernamentales que se encuentren vinculadas, etc.

3. METODOLOGÍA DE INVESTIGACIÓN

3.1 DISEÑO METODOLÓGICO

El enfoque de la investigación es teórico, donde se en marca dos tipos de investigación, tipo exploratoria dada la poca información encontrada de acuerdo al foco de la investigación y consignando el cumplimiento del objetivo general para determinar las implicaciones de la seguridad en proyectos trabajados con metodologías ágiles y tipo descriptiva en la medida que se contextualizara a través de los resultados el conocimiento de los encuestados sobre la seguridad de software, análisis, implementación, mantenimiento, continuidad del negocio y como esto se ve reflejado en la ejecución de los proyectos de software.

3.2 MUESTRA E INSTRUMENTOS

Para la muestra de investigación se realizara una muestra dirigida a empresas casas desarrolladoras de software o con área de tecnología y realización de proyectos de software del sector público o privado, dada la cantidad de empresas existentes y buscando imparcialidad en los resultados, se seleccionara de manera aleatoria las empresas a las que se les aplicara la encuesta de igual manera teniendo en cuenta que los encuestados tienen perfiles de TI importantes en las organizaciones y que la realización de la misma es voluntaria se considerara una muestra de mínimo cincuenta y cinco respuestas completas.

Procedimiento de selección: no probabilístico, muestra de sujetos voluntarios, encuestados que desearon llenar la encuesta, muestra de expertos, encuestados a realizar la encuesta tienen que tener perfiles de TI o afines.

El tamaño de la muestra se estimó teniendo en cuenta el listado de empresas del anexo No 1, con un valor de nivel de confianza del 95% considerando un tamaño poblacional de empresas de 1000 y un error muestral del 8%, utilizando la siguiente ecuación:

$$n = \frac{N.Z^2.p.q}{(N-1).e^2 + Z^2.p.q}$$

$$n = \frac{1000.2^2.10.90}{(1000-1).8^2 + 2^2.10.90}$$

$$n \approx 54$$

Aproximando el valor, se calculó un total de 54 empresas como mínimo para la muestra.

Como instrumento para la recolección de información sobre el tema de la investigación ver anexo No 4 se realizó una encuesta la cual se aplicó a las empresas determinadas, se escogió como medio de difusión la plataforma de encuestas SurveyMonkey debido a la amplia variedad de opciones que ofrece para realizar la encuesta, así como su personalización visual, además de que la ejecución de la encuesta se puede realizar desde computadores o dispositivos móviles.

3.3 FASES DEL PROYECTO

1. Recolección de información desarrollo de software seguro, metodologías ágiles.
2. Diseño de encuesta.
3. Prueba piloto de la encuesta.
4. Análisis prueba piloto.
5. Ajustes al instrumento de recolección.
6. Selección de la muestra.
7. Aplicación del instrumento de recolección.
8. Tabulación de resultados.
9. Análisis e interpretación de la información recolectada.
10. Conclusiones del análisis respectivo.
11. Revisión y presentación de la investigación.

3.4 RECURSOS E INSUMOS

Fue necesario como recursos para el proyecto:

- Una persona encargada de recolectar, analizar e interpretar la información relevante sobre metodologías ágiles y el desarrollo de software seguro, así como de concluir los resultados obtenidos en las encuestas.
- Un computador en el cual se desarrolló la documentación y presentaciones necesarias para el proyecto.

Como insumos se utilizaron principalmente los siguientes:

- Tesis de grado: “Metodología para el Aseguramiento de Entornos Informatizados” – MAEI.

- Investigación de campo: “GUÍA PARA LA INCORPORACIÓN DE PRÁCTICAS DE SEGURIDAD EN ELMERCADO CHILENO DE DESARROLLO DE SOFTWARE”
- Libro: Software security, Exploiting Software: How to Break Code - Gary McGraw
- Libro: PROYECTOS ÁGILES CON SCRUM - Diego Alaimo.
- Libro: Introducción a la metodologías ágiles - Jorge Fernández.

3.5 PRUEBA PILOTO

Se realizó apertura de la encuesta desde la plataforma por medio de un enlace web desde la cual se ingresó para realizarse, se envió alrededor de 50 personas en las instalaciones de la bolsa de valores de Colombia, en el primer día se consiguió un total de 16 respuestas, transcurridos nueve días se cerró el recopilador desde la plataforma, consiguiendo una muestra de 18 respuestas completas de la encuesta ver anexo No 2, adicional se les solicito a los participantes retroalimentación sobre el entendimiento de las preguntas así como su dificultad, entre otras observaciones.

Teniendo que con la prueba piloto no se puede obtener resultados importantes debido a que únicamente se consultó una empresa, se escogieron once de las preguntas de la encuesta para socializar la prueba piloto. Consultar anexo No 3.

3.5.1 Resultados prueba piloto

De acuerdo a los resultados obtenidos se concluye que menos de la mitad de las personas encuestadas no tienen un conocimiento claro sobre la temática, esto se debe a que el objetivo de la prueba piloto consiste en depurar cada a una de las preguntas en base a su comprensión, tipos de respuesta y observar si hay algún sesgo entre las respuestas.

3.5.2 Ajustes al instrumento

Con base a la retroalimentación suministrada por cada colaborador y con la ayuda del área de Estadística de la universidad, se corrigieron varios puntos los cuales son descritos a continuación:

- No se solicitará el nombre o datos personales del encuestado esto se realiza para garantizar la sinceridad de la persona que responde la encuesta.
- Para las preguntas que contengan la opción de todas las anteriores se realiza ajuste para permitir que el encuestado deba seleccionar un mínimo de

respuestas esto de acuerdo a la cantidad de opciones, esto se realiza debido a que la opción todas la anteriores supone una facilidad para el encuestado derivando en un sesgo en los resultados que se puedan obtener.

- Se reduce el número de preguntas por que en la retroalimentación obtenida los colaboradores percibieron que la encuesta es demasiado larga, esto supone un riesgo para su ejecución por lo tanto se minimiza las preguntas desestimando las que presentan un aporte menor de información relevante para la investigación.

4. DESARROLLO INVESTIGACIÓN

4.1 DIFUSIÓN Y ESTRUCTURA DE LA ENCUESTA

Como sea descrito en el proyecto la manera de recolectar la información fue desde una plataforma especializada desde donde se envió una invitación por correo electrónico y un link para su acceso, además de contar con preguntas condicionadas a la metodología que el encuestado dice utilizar, por último y si lo desea la persona puede ingresar su correo electrónico para conocer los resultados obtenidos una vez la investigación finalice.

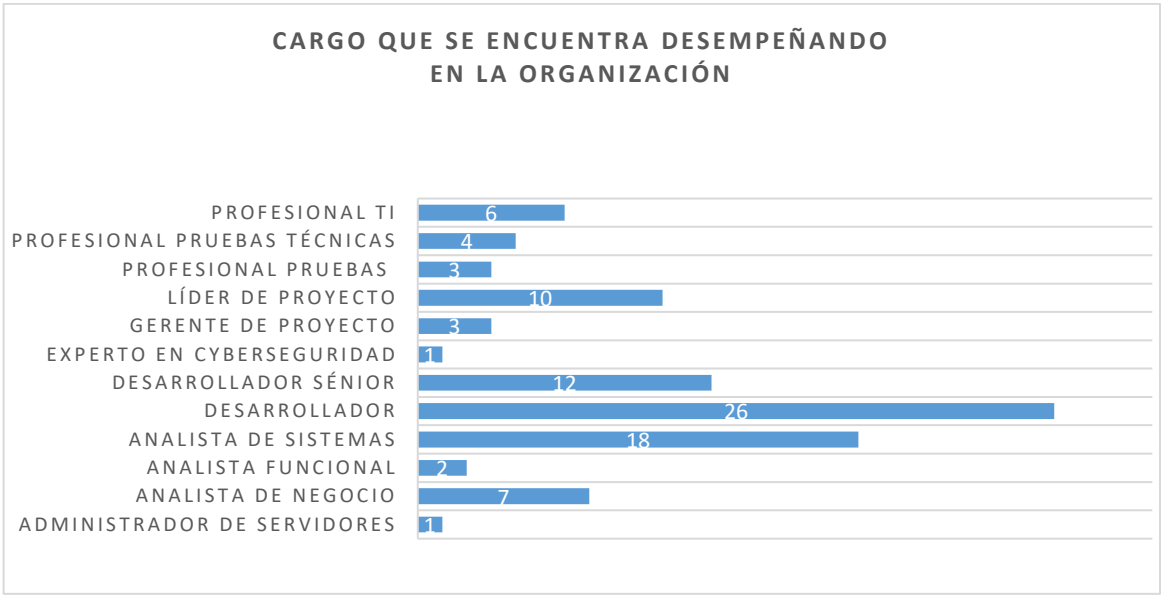
4.2 RECOLECCIÓN DE INFORMACIÓN

Luego de realizar las correcciones y revisar la prueba piloto, se inició el día 15 de noviembre del 2016 la invitación por correo electrónico a un total de 35 personas de las que se obtuvo 12 respuestas completas y 2 parciales, después de observar la baja participación se opta por realizar la divulgación de la encuesta por medio de enlace web contactando a las personas encuestadas de una manera menos formal se percibe una ejecución mayor con un total 79 respuestas de las cuales 49 fueron respuestas completas, 30 parciales para un total de 61 respuestas completas. Se realiza el cierre de la encuesta el día 28 de enero de 2017.

4.1 ANÁLISIS DE PREGUNTAS

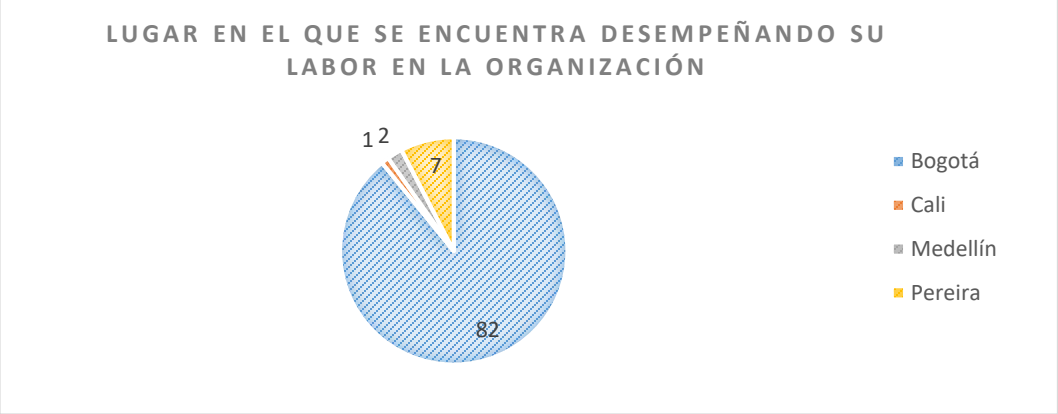
4.1.1 Pregunta 1: Localización y cargo del encuestado

Gráfica 11 Cargos desempeñados por los encuestados



Tal como se puede observar en la gráfica 11 el total de los encuestados fueron 93 aunque en 32 oportunidades no terminaron la encuesta, el cargo que realizo más veces la encuesta fue el de desarrollador seguido por analistas de sistema y desarrollador sénior respectivamente lo cual satisface en gran medida las respuestas que se obtendrán en este estudio. Se hace claridad que las preguntas fueron evaluadas por los 61 encuestados que completaron la encuesta.

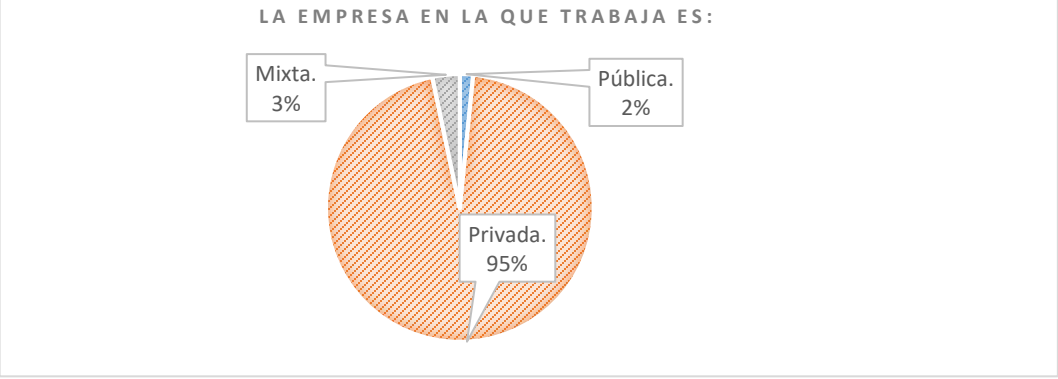
Gráfica 12 Ciudades donde se obtuvo respuesta



De acuerdo a la gráfica anterior, se observa que la tendencia de residencia de los encuestados es Bogotá continuada por Pereira con siete respuestas, una en la ciudad de Cali y Medellín de donde se esperaba mayor participación solo se obtuvieron 2 respuestas.

4.1.2 Pregunta 2: Clasificación de la empresa

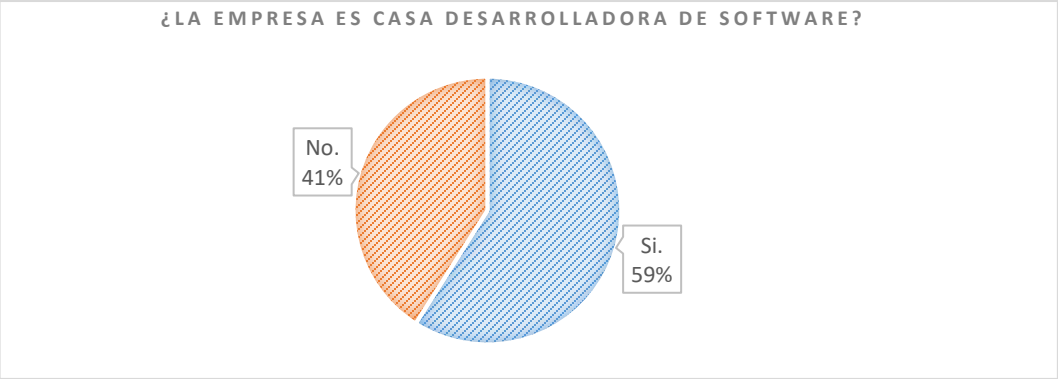
Gráfica 13 Clasificación de la empresa



La grafica denota que el 95% de los encuestados trabajan en empresas privadas de esta manera las respuestas obtenidas tienen un mayor enfoque a empresas que pertenecen al sector privado.

4.1.3 Pregunta 3: ¿La empresa es casa desarrolladora de software?

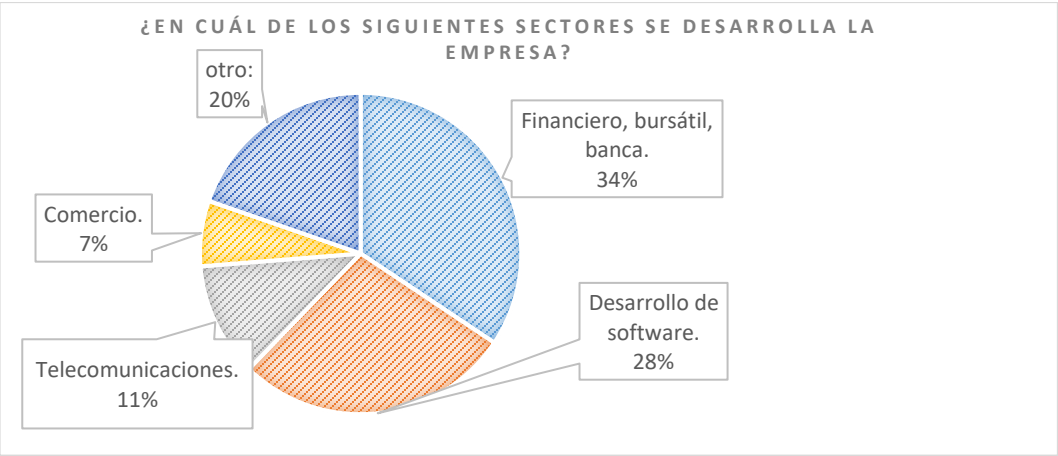
Gráfica 14 Ocupación de la empresa



Como se puede apreciar en la gráfica anterior, los encuestados consideran en un mayor porcentaje que la empresa en donde trabajan es casa desarrolladora de software con un 59% y el otro 41% laboran en empresas que a pesar que desarrollan proyectos de software este no es el sustento económico de la empresa.

4.1.4 Pregunta 4: Sector en el que se desarrolla la empresa

Gráfica 15 Sector de la empresa

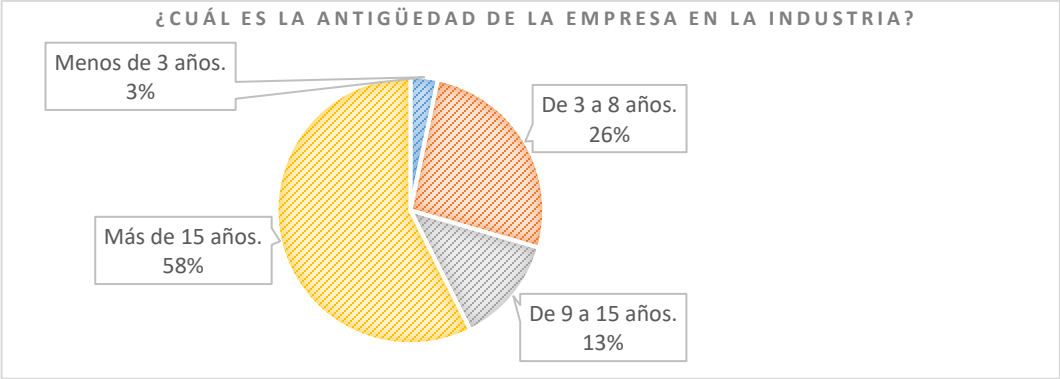


Se muestra en la gráfica 15, que los encuestados pertenecen a empresas en el sector financiero, bursátil, banca correspondiente a 21 empresas y desarrollo de software con 17 el resultado de esta respuesta hace que se considere que las empresas a las que les interesa en mayor medida la seguridad son estos dos sectores, uno por el manejo que deben tener con información sensible como el sector banca y el segundo porque prestan su conocimiento en desarrollo de software a otras empresas de otros sectores.

Igualmente se observa que el 20% con la respuesta de otro sector corresponde a Pymes con diferentes actividades económicas como publicidad, salud, marketing y entretenimiento.

4.1.5 Pregunta 5: Antigüedad de la empresa

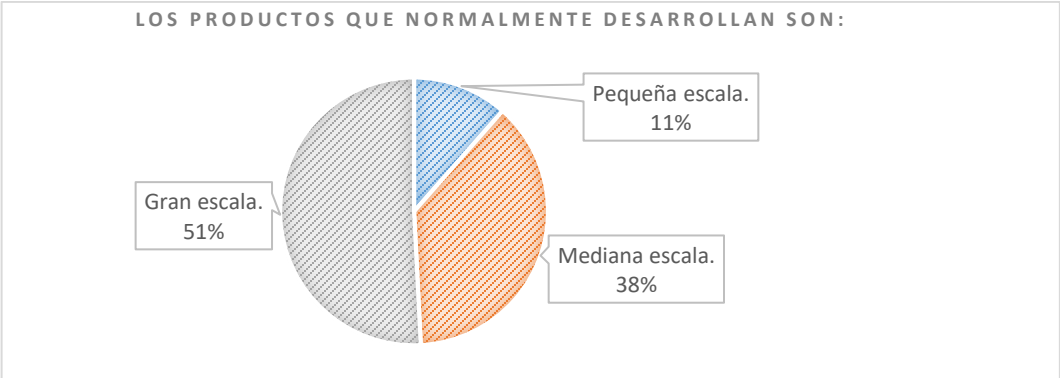
Gráfica 16 Antigüedad de la empresa



En la gráfica 16 se visualiza que las empresas en las que se realizó la encuesta tienen un tiempo de fundación en su mayoría de más de 15 años un dato importante pues se espera que la experiencia y antigüedad de las empresas sean proporcionales a la madurez de los procesos de trabajo, así como de la calidad y seguridad del software que desarrollen.

4.1.6 Pregunta 6: Escala de los productos desarrollados

Gráfica 17 Escala de los desarrollos

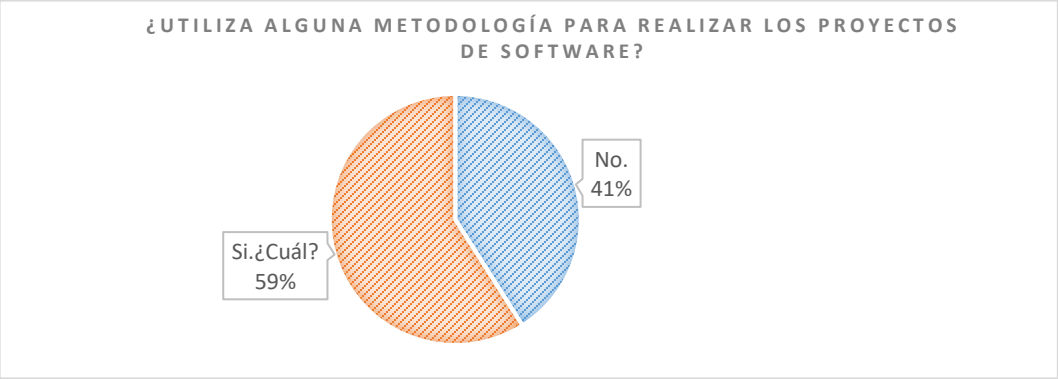


Según la gráfica 17, la información suministrada indica que los proyectos de desarrollo que se ejecutan en las empresas encuestadas son de mediana y gran

escala con un 89% entre las dos, mientras que solo el 11% trabaja proyectos de pequeña escala, es de anotar que entre más complejo sea el sistema es probable que no se contemplen en su totalidad los requerimientos de seguridad que requiere el sistema en el desarrollo, lo que deja una brecha de seguridad importante.

4.1.7 Pregunta 7: Uso de metodología para proyectos de software

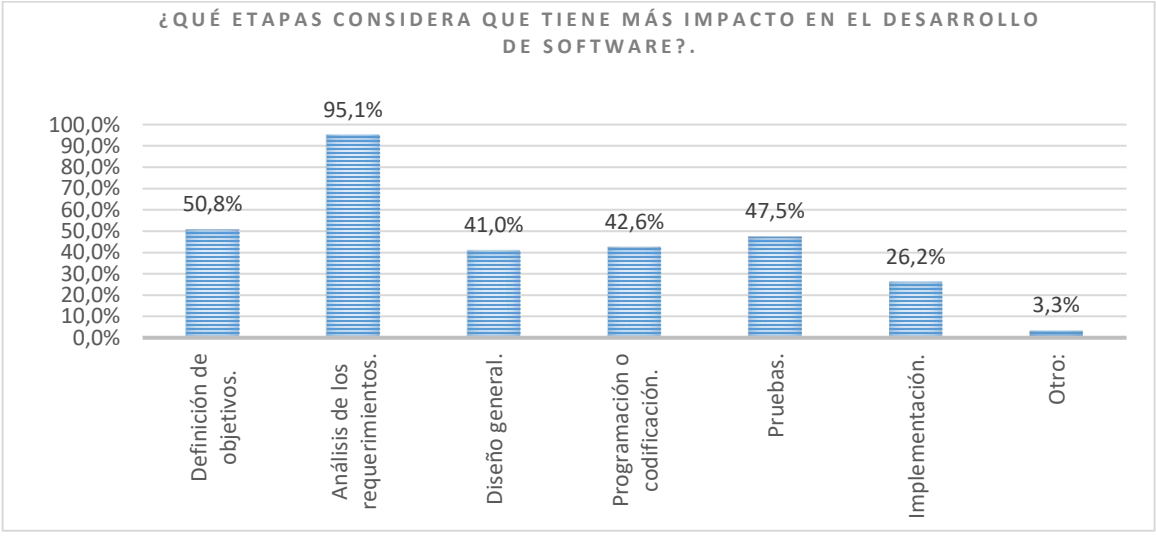
Gráfica 18 Uso de metodología en proyectos de software



En la gráfica 18, visualiza que el 41% de los encuestados no utiliza ninguna metodología para realizar los desarrollos mientras que el 59% restante indico que utilizan diversas metodologías desde la tradicionales como Rup, cascada hasta las ágiles como SCRUM, XP y Agile.

4.1.8 Pregunta 8: Etapas de impacto en el desarrollo de software

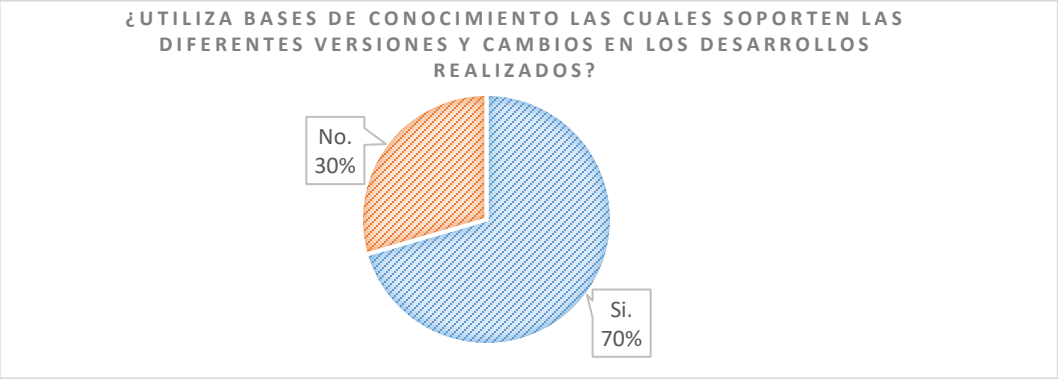
Gráfica 19 Etapas de impacto en el desarrollo de software



En la gráfica 19, los encuestados seleccionan que la etapa más importante en el desarrollo de software es el análisis de requerimientos ya que este es el punto de partida para definir el desarrollo, así como las pruebas que se deben ejecutar para garantizar el funcionamiento de los solicitado.

4.1.9 Pregunta 9: Uso de bases de conocimiento

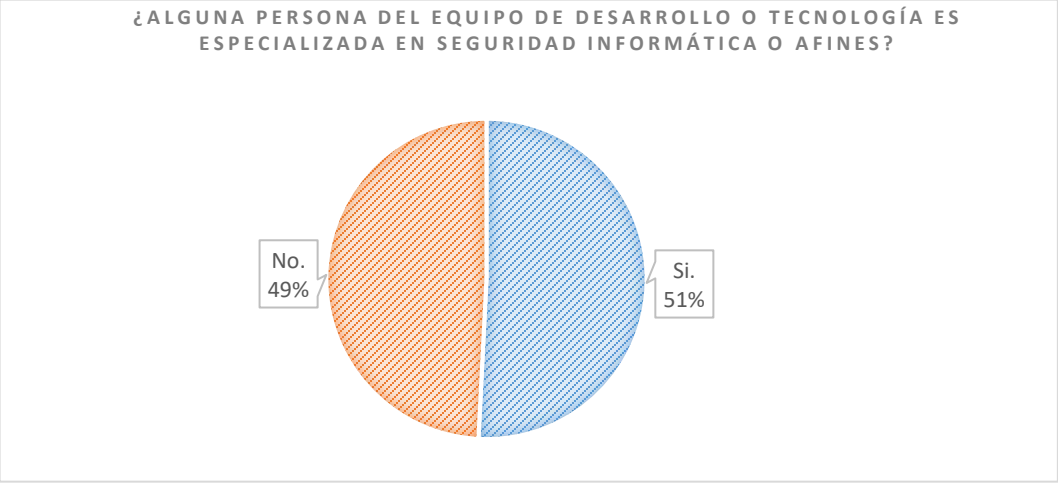
Gráfica 20 Uso de bases de conocimiento



Como se puede observar en la gráfica 20, presenta que el 70% de las empresas consultadas utilizan bases de conocimiento para conservar la documentación e información de los desarrollos realizados, esto suele ser una buena práctica para detectar fallos u optimizaciones para el sistema.

4.1.10 Pregunta10: Trabajadores especializados en seguridad informática

Gráfica 21 Personas especializadas en seguridad

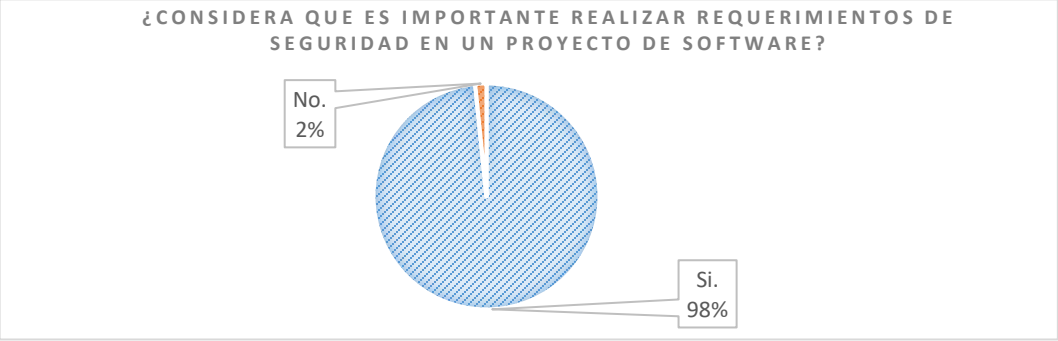


Tal como se puede observar en la gráfica 21, el 51% consideraron que en sus empresas laboran personas especializadas en seguridad informática un dato

importante pues según los resultados obtenidos en la prueba piloto se observó que hay una tendencia a interpretar como trabajadores especializados en seguridad informática a arquitectos de software o desarrolladores sénior, esperando que este no sea el caso esta es una buena cantidad de empresas que dan la importancia que merece el tema del software seguro.

4.1.11 Pregunta11: Requerimientos de seguridad

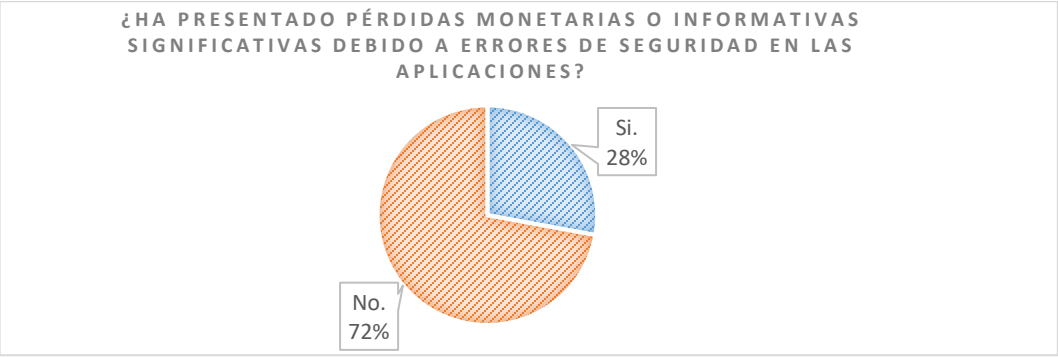
Gráfica 22 Requerimientos de seguridad



De lo expuesto en la gráfica 22 se observa que el 98% de los encuestados consideran que es importante realizar requerimientos de seguridad de software mientras que el 2% considera que no es importante realizar este tipo de requerimientos.

4.1.12 Pregunta 12: Perdidas monetarias por errores de seguridad en aplicaciones

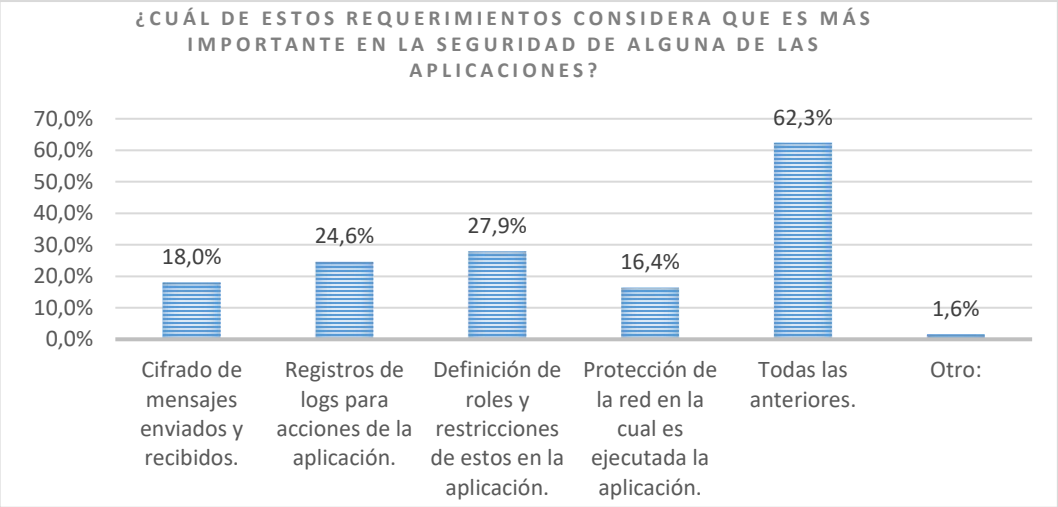
Gráfica 23 Perdidas monetarias por seguridad



Se puede observar en la gráfica 23, que un 72% contesto que no presentan pérdidas monetarias o informativas a causa de la seguridad en las aplicaciones utilizadas por la empresa en las que trabajan, en cambio un 28% contestan que si han presentado estas pérdidas, lo cual reafirma la importancia que tiene desarrollar software seguro para no permitir la vulneración de datos sensibles en los sistemas.

4.1.13 Pregunta 13: Requerimientos de seguridad

Gráfica 24 Requerimientos de seguridad

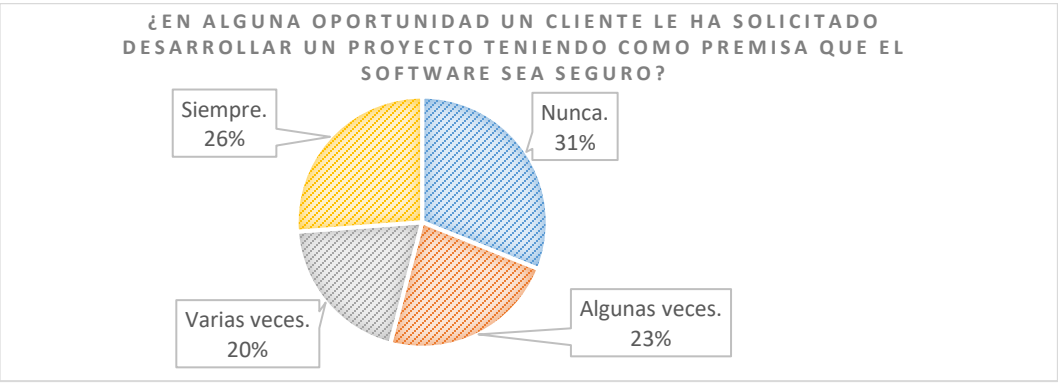


Como se puede apreciar en la gráfica 24, se generó un sesgo en la respuesta ya que el 62,3% contesto la opción de todas las anteriores seguida del requerimiento para la definición de roles y restricciones con 27,9 % en tercer lugar el requerimiento para los registros de logs con 24,6%, entre el cuarto y quinto lugar los requerimientos para el cifrado de mensajes con 18% y protección de red con 16,4% respectivamente.

En último lugar se tiene una única respuesta con el 1,6% en la opción otro en la que el encuestado contemplo requerimientos sobre cifrado de passwords, cuentas de usuario para diferentes roles y formularios que restrinjan al usuario acceso a los datos mediante scripts.

4.1.14 Pregunta 14: Desarrollo de software seguro como premisa

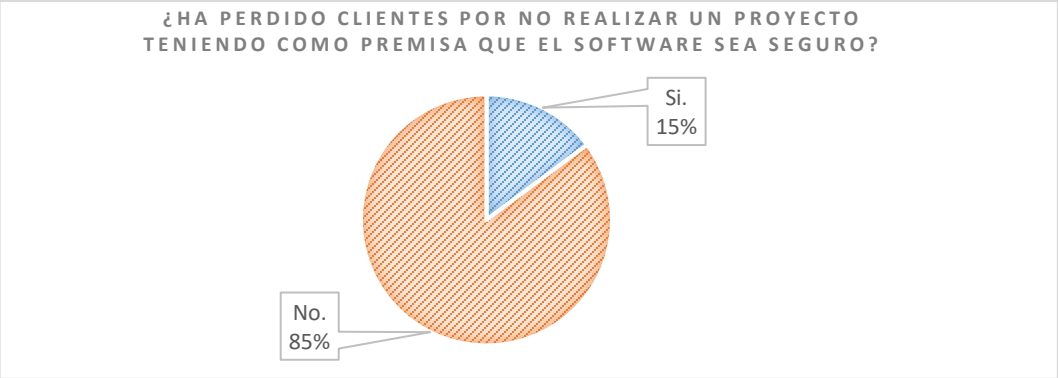
Gráfica 25 Desarrollo de software seguro como premisa



En la gráfica de la pregunta 14 se evidencia que los clientes tienden a tener en cuenta la funcionalidad de la aplicación sobre la seguridad, el 31% de las personas responde que nunca se le ha solicitado desarrollar un proyecto teniendo como premisa el software seguro continuado con un 26% en el que siempre se solicita que el software sea seguro, terminando con un 23% en el que algunas veces se ha solicitado y un 20% en el que varias veces se ha tenido como premisa que el software sea seguro.

4.1.15 Pregunta 15: Perdida de clientes por proyecto de software seguro

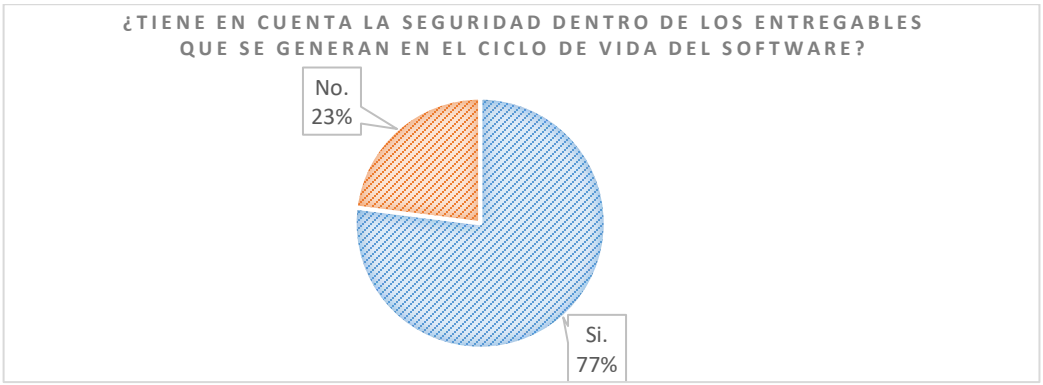
Gráfica 26 Perdida de clientes por software seguro



Según la gráfica 26 la pérdida de clientes por no realizar el proyecto a que el software sea seguro es bajo, solo el 15 % respondió afirmativamente, mientras que el 85% restante respondió que no han perdido clientes por esta causa.

4.1.16 Pregunta 16: Seguridad en entregables del ciclo de vida

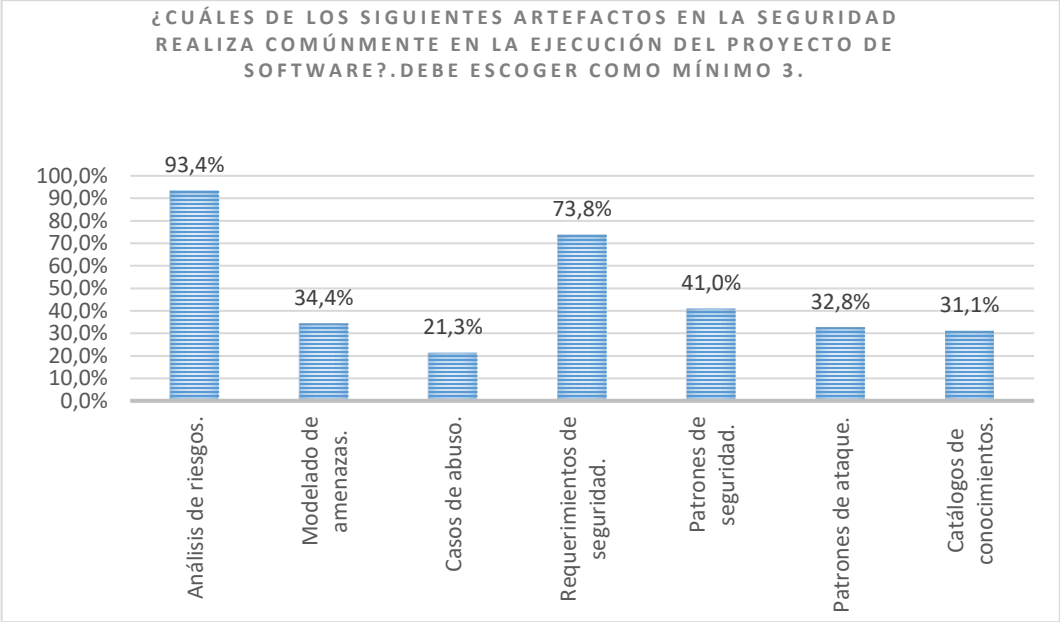
Gráfica 27 Seguridad en entregables del ciclo de vida



En la gráfica 27 se puede ver el 77% de los encuestados tienen en cuenta la seguridad en los entregables generados en el ciclo de vida del software en cambio un 23% no la tiene en cuenta.

4.1.17 Pregunta 17: Artefactos en la seguridad de software

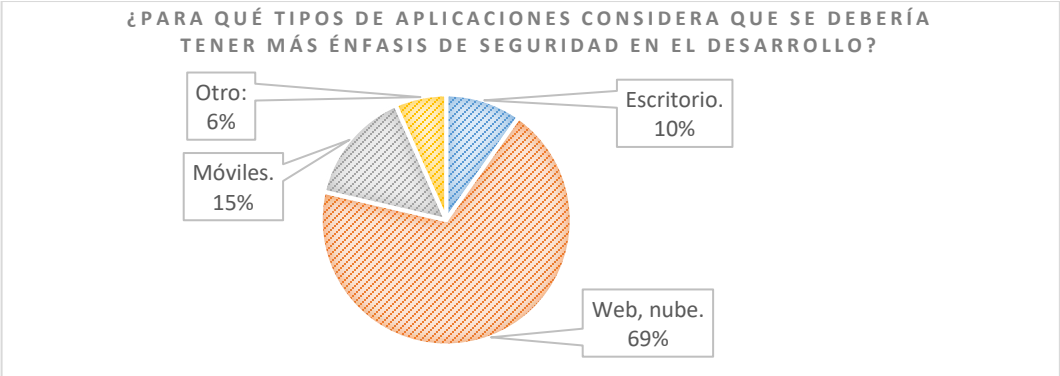
Gráfica 28 Artefactos software seguro



De acuerdo a la gráfica anterior los artefactos de seguridad que son más usados son Análisis de riesgos con 93,4%, requerimientos de seguridad con 73,8% y patrones de seguridad con 41%. Adicionalmente los artefactos modelo de amenazas, catálogos de conocimientos y casos de abuso son los menos usados por los encuestados.

4.1.18 Pregunta 18: Tipos de aplicaciones software seguro

Gráfica 29 Aplicaciones, énfasis en software seguro



Según la gráfica 29, los encuestados consideran que se debe tener más énfasis en seguridad hacia los desarrollos web, nube con un 69%, móviles con un 15 % de escritorio con 10% y otro con 6%.

4.1.19 Pregunta 19: Top 10 OWASP

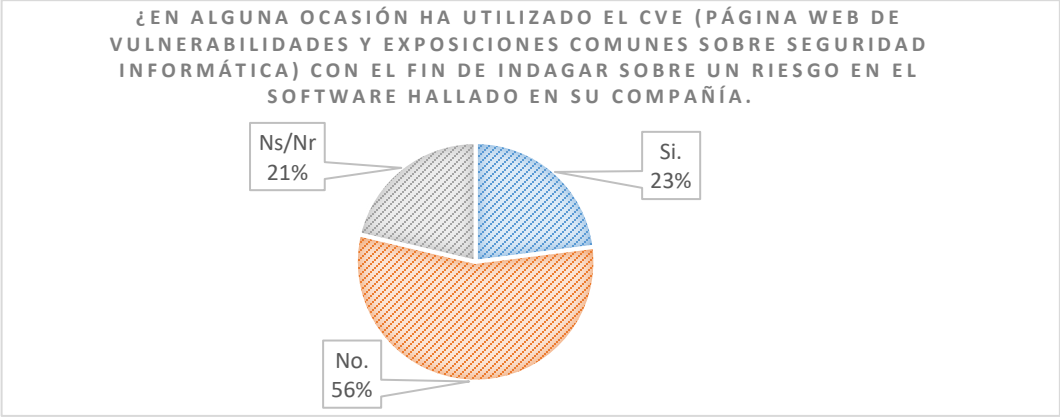
Tabla 2 Top 10 OWASP

Teniendo en cuenta su conocimiento en el Top 10 de OWASP publicado, ¿Cuáles considera que son los ataques más riesgosos para su empresa? Debe escoger como mínimo 3.		
Opciones de respuesta	**Respuestas en porcentaje**	**Cantidad de respuestas**
Inyección.	42,6%	26
Pérdida de autenticación y gestión de sesiones.	45,9%	28
Secuencia de comandos en sitios cruzados (XSS).	18,0%	11
Referencia directa insegura a objetos.	19,7%	12
Configuración de seguridad incorrecta.	54,1%	33
Exposición de datos sensibles.	62,3%	38
Ausencia de control de acceso a funciones	26,2%	16
Falsificación de peticiones en sitios cruzados (CSRF)	26,2%	16
Utilización de componentes con vulnerabilidades conocidas.	23,0%	14
Redirecciones y reenvios no validados.	36,1%	22

Según la tabla 2 los ataques más riesgosos para las empresas teniendo en cuenta el top 10 de owasp, el ataque más representativo para los encuestados es exposición de datos sensibles con 62,3% seguido de la configuración de seguridad incorrecta con 54,1%, continuando con ataques como pérdida de autenticación de y gestión de sesiones con 45,9%, Inyección 42,6% y redirecciones y reenvíos no validos 36,1%, las otras opciones que se consideraron como menos riesgosas son ausencia de control de acceso a funciones 26,2%, falsificación en sitios cruzados 26,2%, utilización de componentes con vulnerabilidades conocidas 23%, referencia directa insegura a objetos 19,7% y secuencia de comandos en sitios cruzados 18%.

4.1.20 Pregunta 20: Uso del CVE

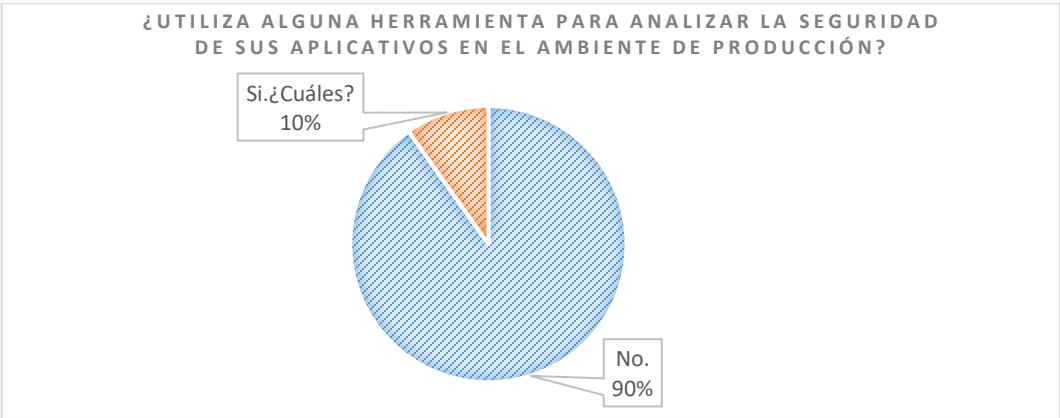
Gráfica 30 Uso del CVE



La gráfica 30 establece que el 56% de los encuestados no han utilizado el CVE en ninguna oportunidad el 23% si lo ha utilizado para indagar sobre un riego de software hallado en la compañía mientras que un 21% contesto que no sabe o no responde.

4.1.21 Pregunta 21: Uso de herramientas para analizar la seguridad

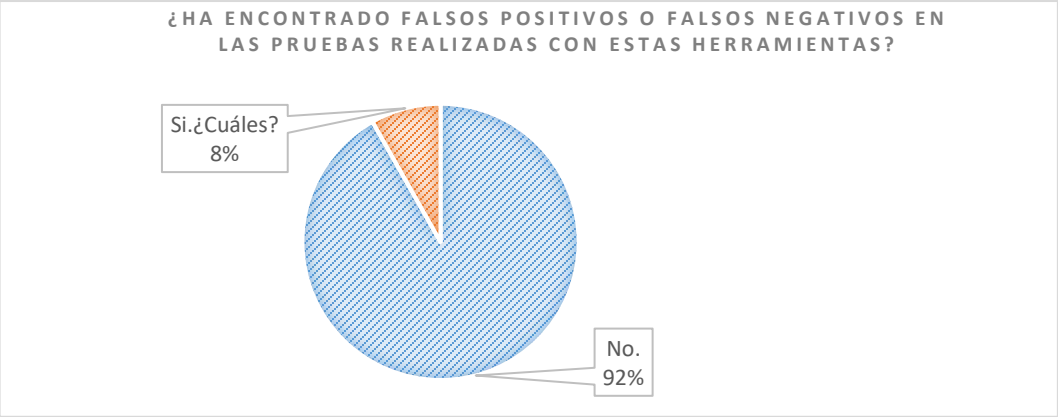
Gráfica 31 Herramientas análisis de seguridad



Según la gráfica 31, el 90% contesto que no utiliza ninguna herramienta para analizar la seguridad de sus aplicativos en ambiente productivo y un 10% si utiliza herramientas para este fin, usando Nessus, Nagios, nmap, OpenVas y en una respuesta se especifica que la herramienta usada es propia de la compañía.

4.1.22 Pregunta 22: Falsos positivos o falsos negativos encontrados

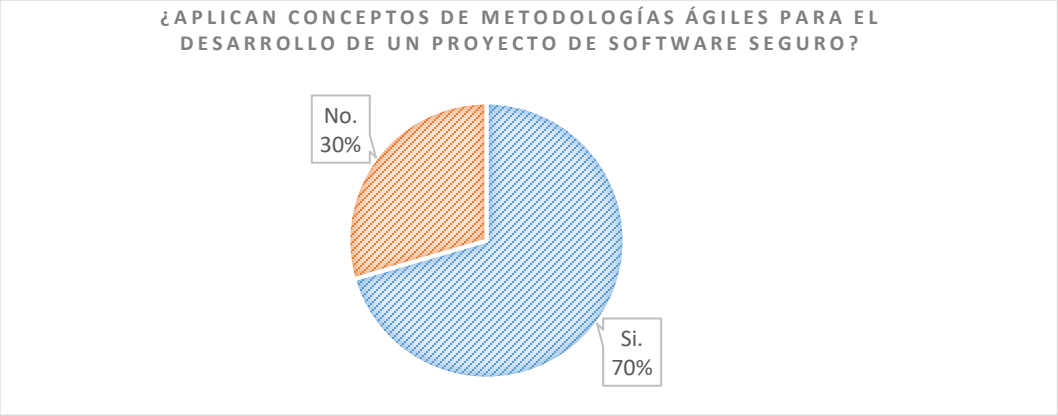
Gráfica 32 Falsos positivos o falsos negativos



De acuerdo con la gráfica 32 sean encontrado falsos positivos o negativos en las pruebas realizadas con las herramientas especificadas en la pregunta 21 en 5 oportunidades que representa el 8% en las que se describe que han sido problemas relacionados con la parte contable y ataques a páginas web con denegación de servicios los demás encuestados responden que no encontraron falsos positivos o negativos con un 92%.

4.1.23 Pregunta 23: Conceptos metodologías ágiles

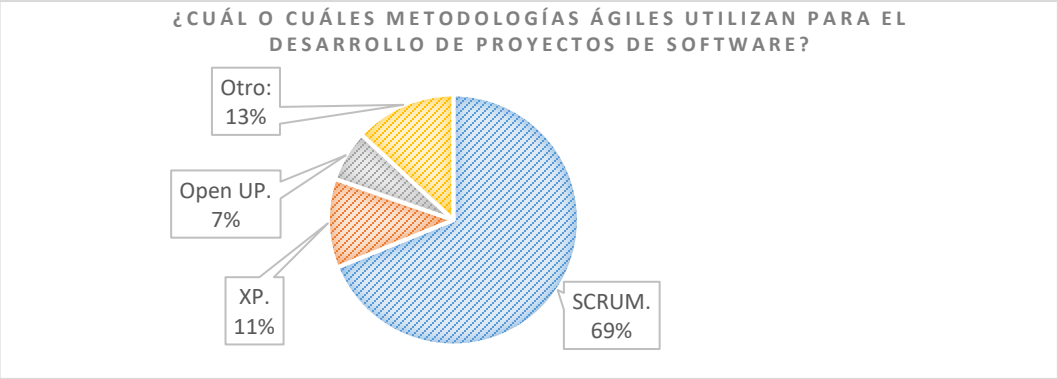
Gráfica 33 Conceptos metodologías ágiles



La gráfica 33 muestra que el 70% de los encuestados si utiliza en alguna medida conceptos de metodologías ágiles para el desarrollo de un proyecto de software y el restante correspondiente al 30 % no aplican conceptos de metodologías ágiles.

4.1.24 Pregunta 24: Metodologías ágiles utilizadas

Gráfica 34 Metodologías ágiles utilizadas



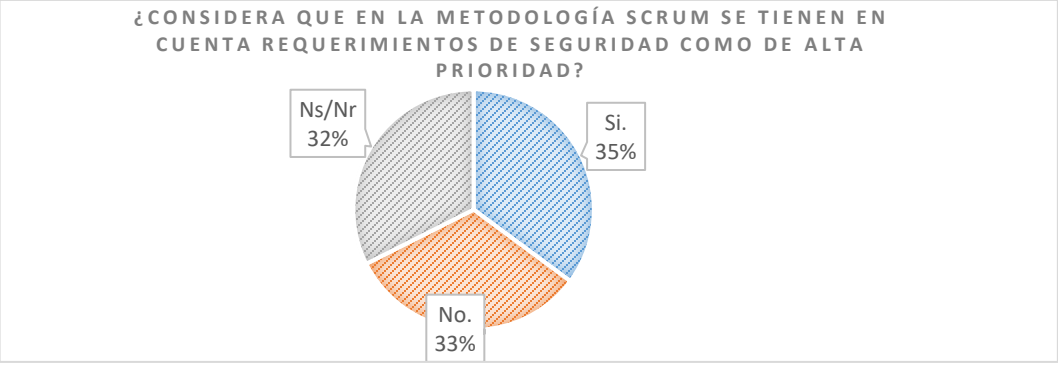
Como se puede observar en la gráfica 34, el 69% utiliza como metodología ágil SCRUM con 69% en segundo lugar XP con 11% y en ultimo OpenUp con 7%, para la opción de otro con 13% los encuestados usan las siguientes metodologías: Scrum con Rup, Rup únicamente, propia y ninguna. Según la respuesta escogida por la persona la encuesta se dirigirá a una serie de preguntas enfocadas a la metodología que se dio respuesta tal como se puede observar en la tabla 3.

Tabla 3 Respuesta encuestados por metodología

Opciones de respuesta	Cantidad de respuestas
SCRUM.	42
XP.	7
Open UP.	4
Otro:	8

4.1.25 Pregunta 25: Requerimientos de seguridad metodología Scrum

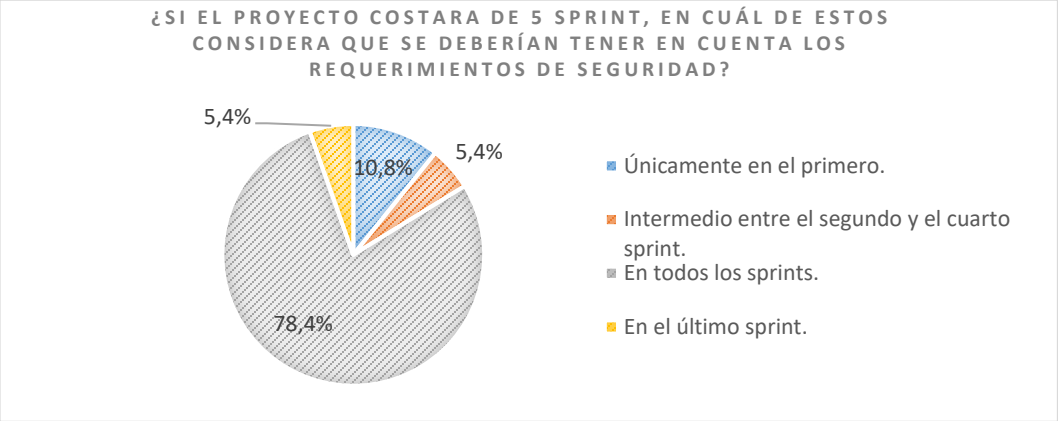
Gráfica 35 Requerimientos de seguridad metodología Scrum



Según la gráfica anterior, el 35% de las personas consideran que la metodología Scrum tiene en cuenta requerimientos de seguridad como de alta prioridad, seguido por el 33% que no lo considera y un 32% que no sabe o no responde.

4.1.26 Pregunta 26: Sprint en el desarrollo de software seguro

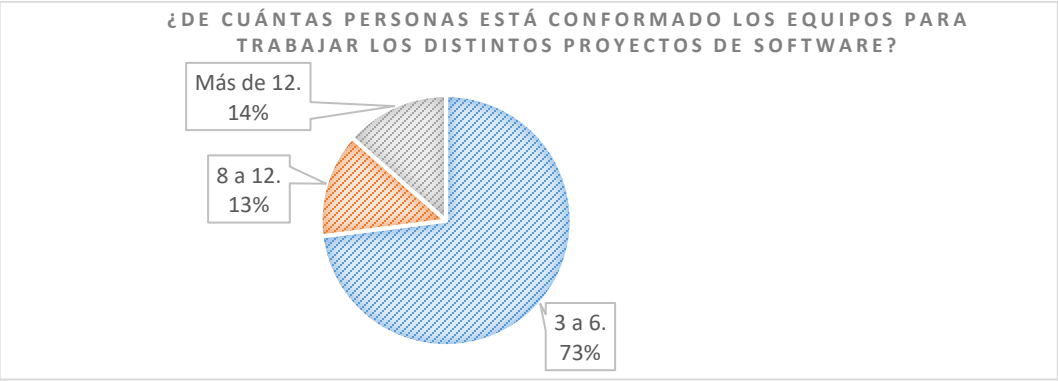
Gráfica 36 Sprint en software seguro



Como se puede observar en la gráfica 36, el 78,4% de los encuestados responden que se debe tener en cuenta los requerimientos de seguridad en todos los sprints, el 10,8% que se deben tener en cuenta únicamente en el primero y con el mismo porcentaje 5,4% se deben tener en cuenta para el último sprint o intermedio entre el segundo y el cuarto.

4.1.27 Pregunta 27: Conformación equipos Scrum

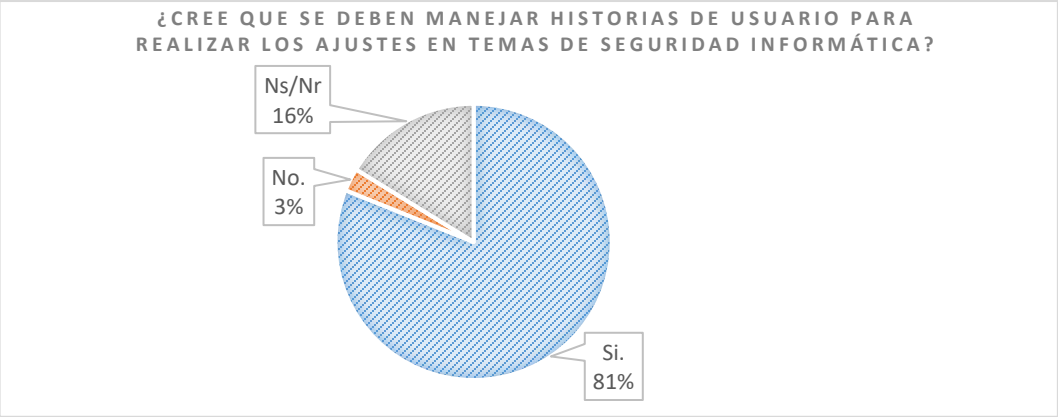
Gráfica 37 Conformación equipos Scrum



La gráfica 37 muestra que los grupos de trabajo normalmente están conformados por 3 a 6 personas 73%, 8 a 12 13% y más de 12 personas con 14%, la respuesta con mayor número de respuestas corresponde al número de integrantes con el que comúnmente se desarrolla la metodología Scrum.

4.1.28 Pregunta 28: Manejo de historias de usuario

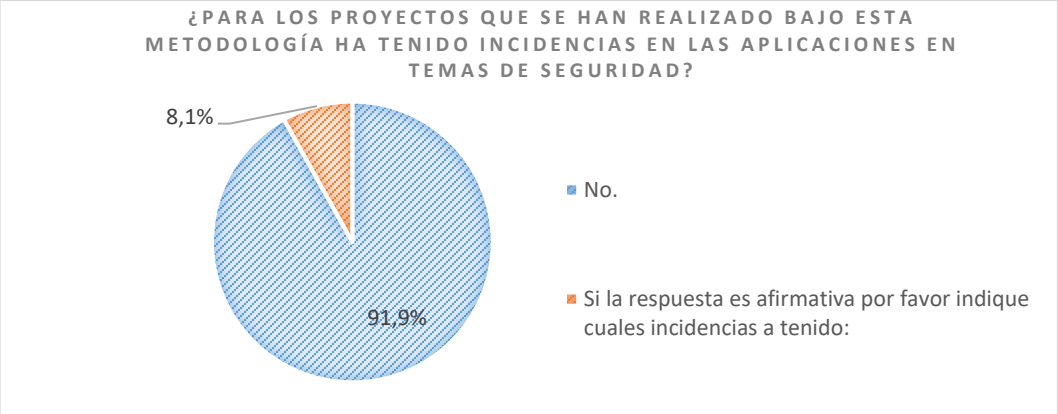
Gráfica 38 Manejo de historias de usuario



Se puede apreciar en la gráfica 38, que el 81% de los encuestados consideran que se deben manejar historias de usuario independientes a la funcionalidad para tratar temas de seguridad mientras que un 3% considera lo contrario y el restante 16% No sabe o no responde.

4.1.29 Pregunta 29: Incidencias en seguridad, Scrum

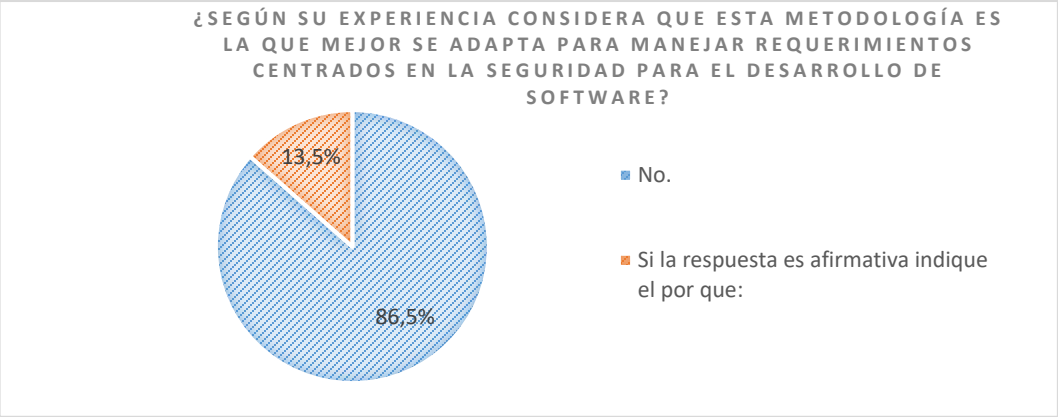
Gráfica 39 Incidencias en seguridad, Scrum



En la gráfica 39 se observa que, el 91,9% de encuestados no han presentado incidencias en las aplicaciones desarrolladas bajo la metodología Scrum y un 8,1% si les ha ocurrido sus incidencias son por ataques a portales web o ingreso al sistema operativo que ejecutaba la aplicación.

4.1.30 Pregunta 30: Experiencia con metodología Scrum

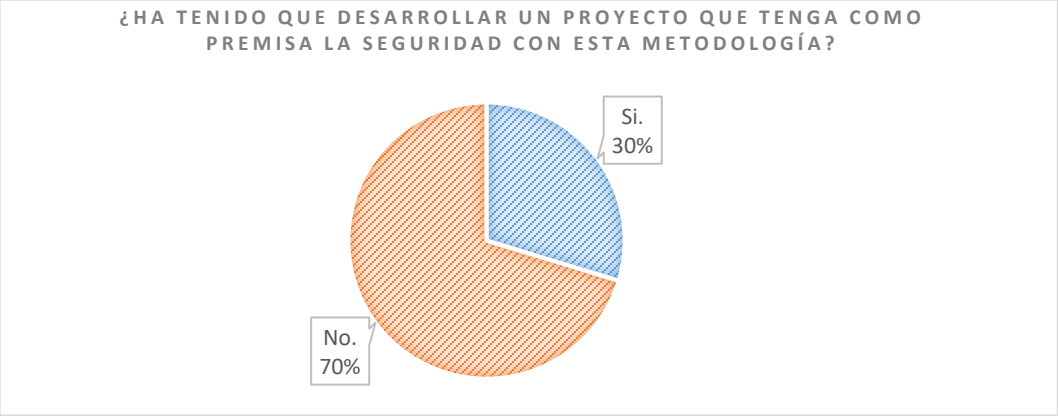
Gráfica 40 Experiencia metodología Scrum



Como se puede apreciar en la gráfica 40, los encuestados opinan en un 86,5% que la metodología Scrum no es la que mejor se adapta para manejar requerimientos centrados en la seguridad, el otro 13,5% restante si considera que es la que mejor se adapta por las siguientes razones: consigue un análisis constante del desarrollo, metodología que más se adapta a los entornos de desarrollo con altos estándares de seguridad.

4.1.31 Pregunta 31: Desarrollo software seguro con Scrum

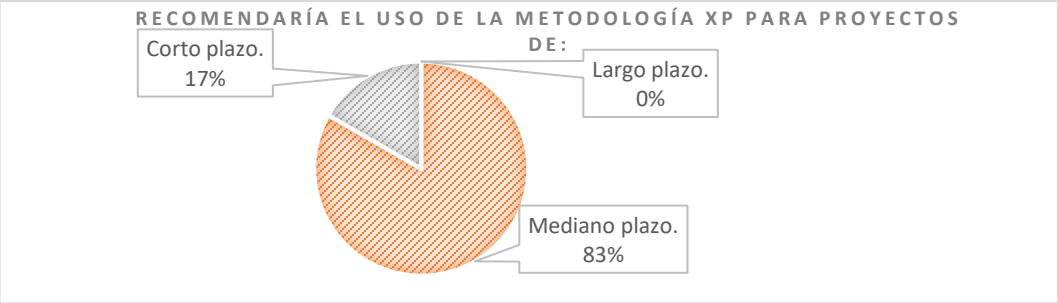
Gráfica 41 Desarrollo de software seguro con Scrum



Respecto a la gráfica anterior se evidencia que el 70% de las personas no han tenido que desarrollar un proyecto que tenga como premisa la seguridad con la metodología Scrum, en cambio al 30% restante que si han sido incluidos en este tipo de proyectos de desarrollo.

4.1.32 Pregunta 32: Uso metodología XP en proyectos

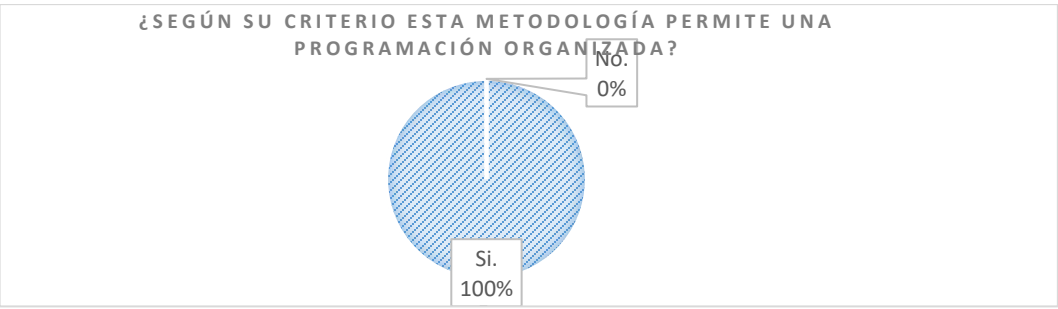
Gráfica 42 Uso metodología XP proyectos



Según la gráfica 42, el 83% considera que el uso de la metodología XP es más efectiva para proyectos a mediano plazo, mientras que el restante considera que se debería usar a corto plazo, ninguno de los encuestados considero que XP se deba usar a largo plazo.

4.1.33 Pregunta 33: Criterio metodología XP

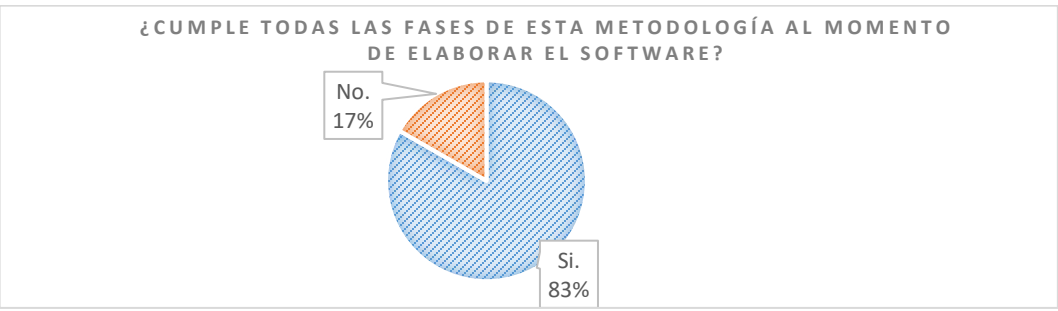
Gráfica 43 Criterio metodología XP



En la gráfica 43, los 7 encuestados que usan XP en su trabajo la evalúan como una metodología que permite una programación organizada.

4.1.34 Pregunta 34: Fases Metodología XP

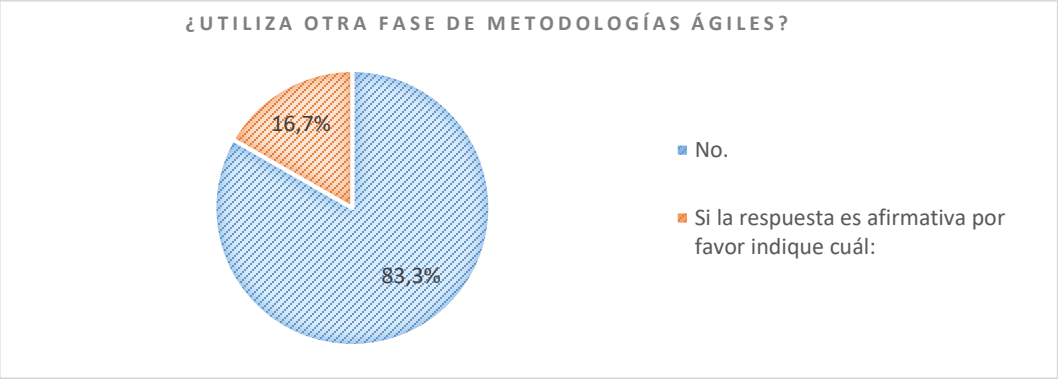
Gráfica 44 Fases metodología XP



Como se puede notar en la gráfica 55, el 83% correspondiente a 5 encuestados afirmaron que cumplen todas las fases de la metodología XP para desarrollar el software y un 17% correspondiente a 1 encuestado contesto que no cumple con todas las fases.

4.1.35 Pregunta 35: Uso de otras metodologías ágiles a XP

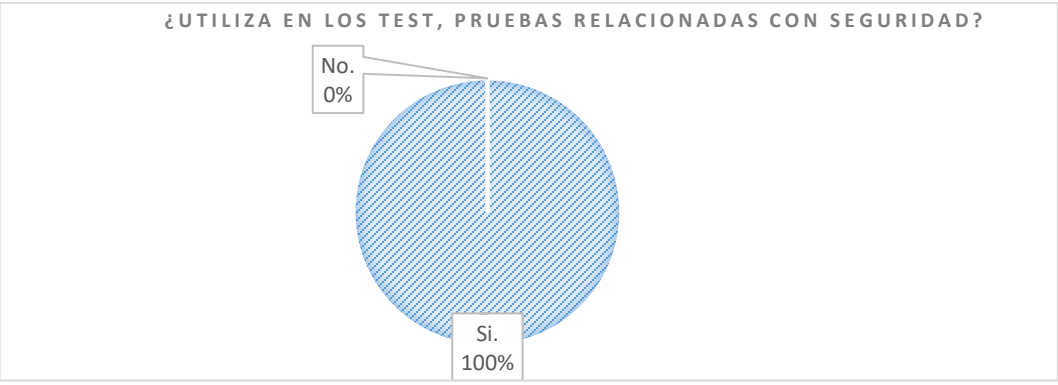
Gráfica 45 Uso de otras metodologías ágiles a XP



Según la gráfica anterior, el 16,7% correspondiente a uno de los encuetados indico que utiliza otra fase de metodologías ágiles adicional a XP se trata del uso de agile interactuando con XP los demás encuestados el 83,3% respondieron que no utilizan ninguna otra fase diferente a XP.

4.1.36 Pregunta 36: Uso de test, relacionados con seguridad

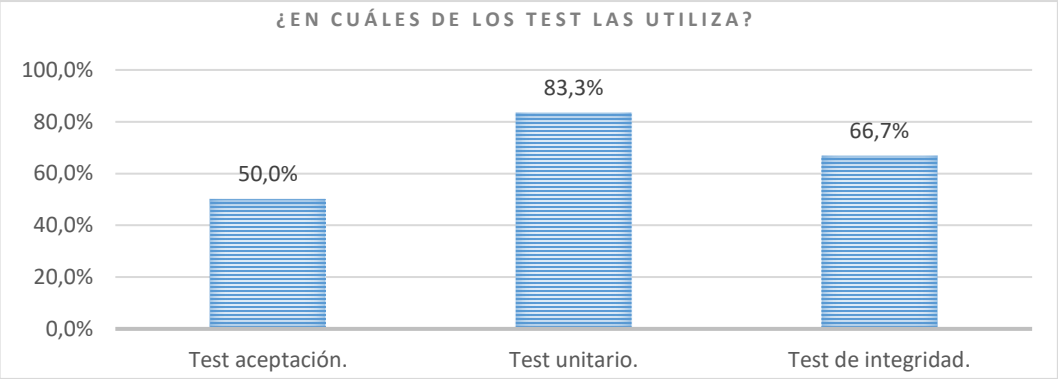
Gráfica 46 Uso de test, relacionados con seguridad



Según la gráfica 46, todos los encuestados realizan pruebas relacionadas a la seguridad de software en los test correspondientes a la metodología XP.

4.1.37 Pregunta 37: Uso de seguridad en los Test XP

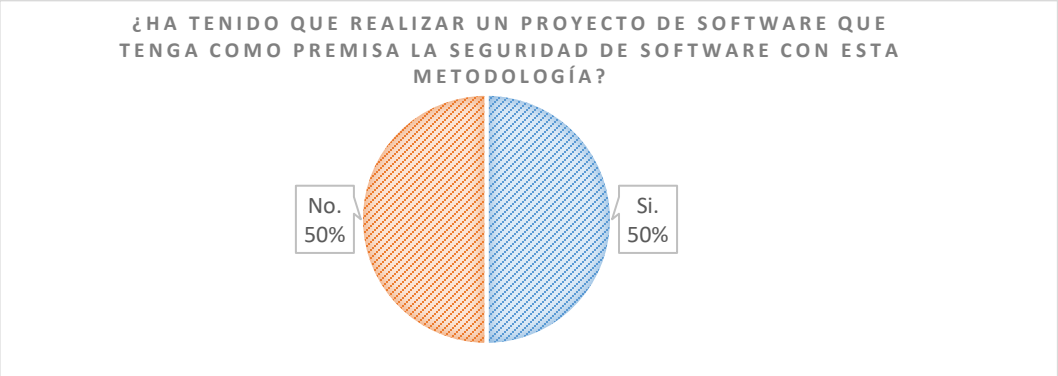
Gráfica 47 Uso de seguridad en los test XP



Como se puede analizar en la gráfica 47, los encuestados usan de manera recurrente el test unitario para pruebas relacionadas con la seguridad en un 83% continuando con test de integridad 67,7% y terminando en test de aceptación con 50%.

4.1.38 Pregunta 38: Desarrollo de software seguro con XP

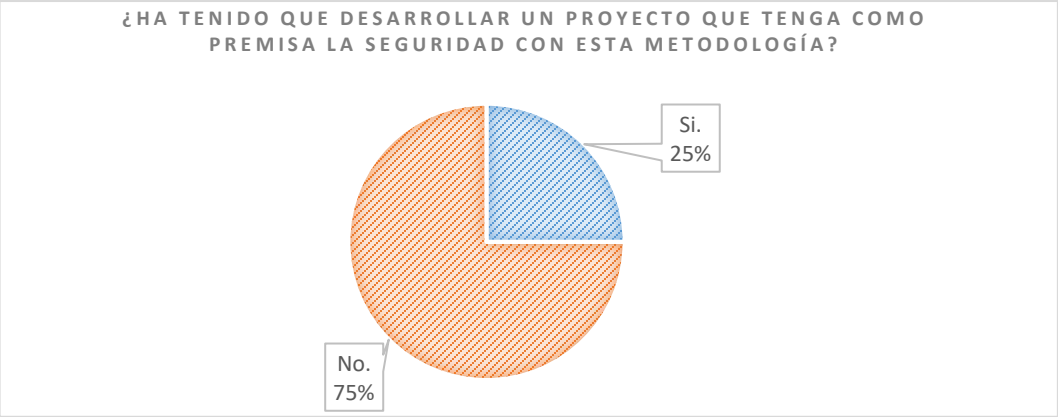
Gráfica 48 Desarrollo de software seguro con XP



Según la gráfica 48, la mitad de las personas que utilizan metodología XP han tenido que realizar un proyecto de software con la premisa que el software sea seguro y las otras 3 personas no lo han tenido que realizar.

4.1.39 Pregunta 39: Desarrollo de software seguro con OpenUp

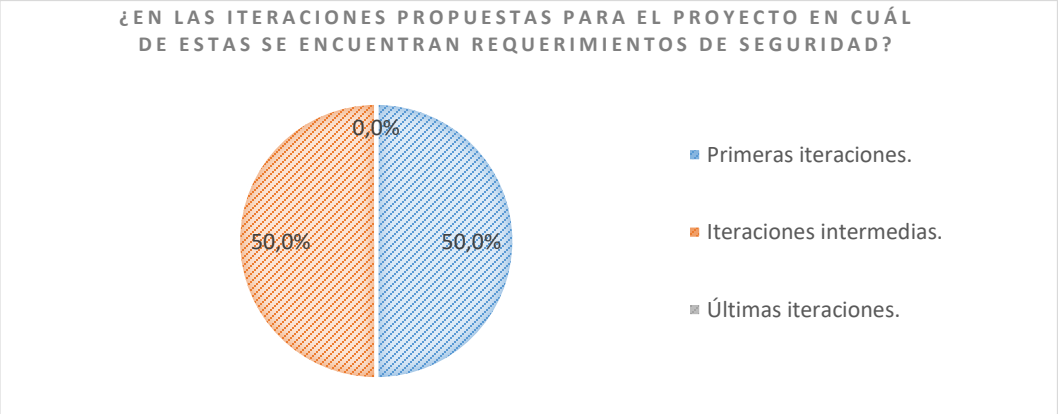
Gráfica 49 Desarrollo de software seguro con OpenUp



Se observa en la gráfica 49 y correspondiendo con la tabla 3, cuatro personas contestaron la encuesta enfocada en OpenUp en la que el 75% no ha realizado desarrollos que tengan como premisa la seguridad con OpenUp y solo el 25% correspondiente a una sola persona si lo ha hecho.

4.1.40 Pregunta 40: Requerimientos de seguridad con OpenUp

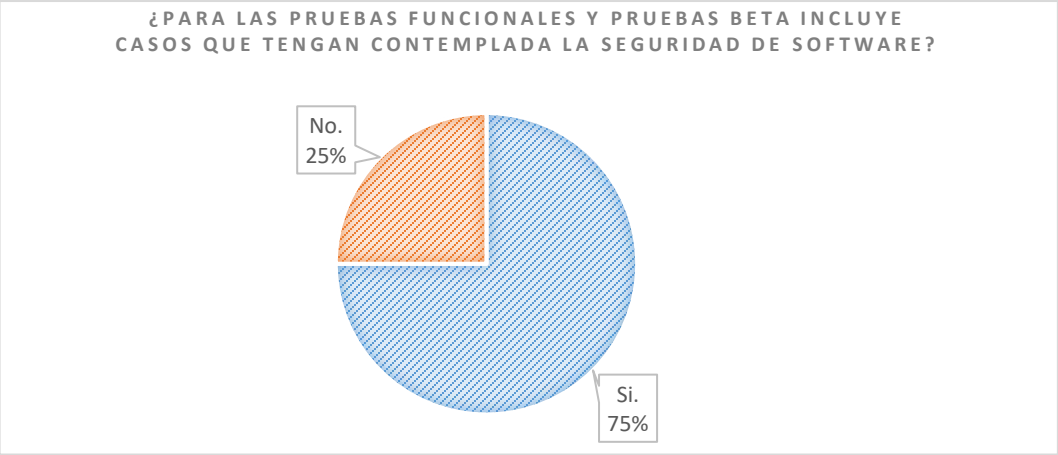
Gráfica 50 Requerimientos de seguridad con OpenUp



Tal como se puede ver en gráfica anterior 2 personas correspondientes al 50% consideran que los requerimientos de seguridad se encuentran en las interacciones intermedias y la otra mitad opina que debe ser en las últimas interacciones.

4.1.41 Pregunta 41: Pruebas de seguridad con OpenUp

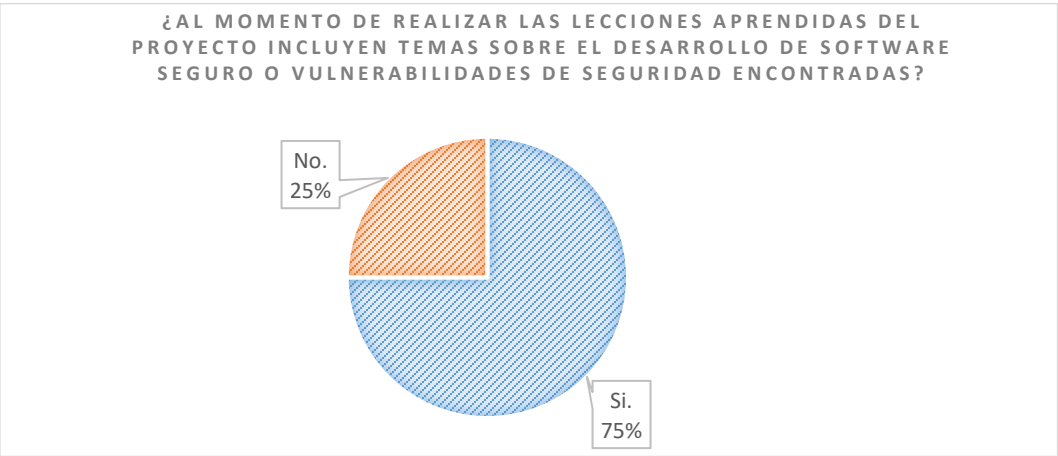
Gráfica 51 Pruebas de seguridad con OpenUp



Se detalla en la gráfica anterior, que únicamente una persona incluye en las pruebas funcionales y pruebas beta casos que contemplen la seguridad de software, los 3 restantes correspondientes al 75 % los cuales indicaron lo contrario.

4.1.42 Pregunta 42: Lecciones aprendidas con OpenUp

Gráfica 52 Lecciones aprendidas con OpenUp



Como se puede observar en la gráfica 52, el 75% de los encuestados correspondiente a 3 personas afirman que al realizar las lecciones aprendidas incluyen temas sobre el desarrollo de software seguro o vulnerabilidades de seguridad encontradas, mientras que 1 encuetas correspondiente a 1 persona no realiza esta práctica.

4.1.43 Pregunta 43: Recomendación encuesta

En la última página de la encuesta se encuentran dos preguntas más en las que el encuestado deberá responder si recomendaría la encuesta a sus colegas y si quiere recibir los resultados de la encuesta en general.

La respuesta de la tabla 4 indica que tan probable es que el encuestado recomiende la encuesta en una escala de 0 a 10 donde 0 es nada probable y 10 muy probable.

Tabla 4 Recomendación encuesta

¿Qué tan probable es que recomiendes esta encuesta a tus amigos o colegas?												
Opciones de respuesta	**Nada probable - 0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Muy probable - 10**	**Total, respuestas**
	3	1	1	3	0	10	4	12	9	4	6	53

Se puede observar en la tabla 3, que los encuestados consideran en recomendar la encuesta, entre 53 respuestas se obtuvo 45 favorables entre los rangos 5-10 y 8 desfavorables indicando que no es probable que recomienden esta encuesta.

4.1.44 Pregunta 44: Retroalimentación por correo

Por último, se les solicito a los encuestados el correo de manera opcional al cual se enviaran los resultados de la encuesta una vez se concluya el análisis, 35 de los encuestados suministraron el correo electrónico al que se les enviara los resultados generales de la encuesta.

4.2 ANÁLISIS GENERAL DE RESULTADOS

Según las 61 respuestas obtenidas al finalizar la encuesta, se obtuvo los siguientes resultados:

1 El estudio se centró en las ciudades de Bogotá, Pereira, Medellín y Cali con participación en su gran mayoría por empresas privadas de los sectores de banca y desarrollo de software de las cuales más de la mitad llevan trabajando de 9 años o más en la industria.
2 El 88,5% de las empresas realizan desarrollos de mediana y gran escala, a pesar de esto se observa que el 41% de estas empresas no manejan ninguna metodología para realizar estos desarrollos.
3 Las empresas consideran que las etapas más importantes en el desarrollo de software son análisis y definición de objetivos.

4 A pesar de que se han presentado perdidas de información debido a problemas en la seguridad del software, los clientes en el 31% de las veces nunca solicitan que el software sea seguro y en el 85% de las veces esto no amerita la perdida de los clientes si no es realizado, se considerara más importante la funcionalidad sobre la seguridad.

5 Las empresas consideran en mayor medida que las aplicaciones que tienden hacer más vulnerables son las que están disponibles en línea comunicándose con otras aplicaciones a través de internet.

6 La metodología ágil que más usan en las empresas encuestadas es Scrum.

7 Las empresas que desarrollan proyectos con metodología Scrum la utilizan de manera correcta y ven la importancia de realizar los proyectos enfocados a el desarrollo de software seguro, pero el 86,5% considera que esta no se adapta a requerimientos de seguridad adicional a que solamente el 30% de estas empresas han tenido que desarrollar proyectos con enfoque hacia la seguridad del software.

8 Las empresas que desarrollan proyectos con metodología XP consideran que es una metodología organizada que se debe usar en proyectos de mediano plazo, se observa que la seguridad la enfocan en la ejecución de Test unitarios e informan que el 50% que usan esta metodología la han ejecutado con proyectos que tienen enfoque en el desarrollo de software seguro.

9 Las empresas que desarrollan proyectos con metodología OpenUp, únicamente el 25% la utilizan para proyectos con premisa en la seguridad de software, manejan temas relacionados con seguridad en la fase de lecciones aprendidas y consideran que los requerimientos de seguridad deben estar entre las interacciones intermedias y últimas del proyecto.

5. CONCLUSIONES Y TRABAJOS FUTUROS

5.1 CONCLUSIONES

La presente investigación se llevó a cabo con la intención de proporcionar el estado de cómo se está realizando los proyectos de software cuando se tiene en cuenta el desarrollo de software seguro haciendo uso de metodologías ágiles, los resultados de la encuesta concluyen que según las empresas encuestadas el 41% de estas no tienen definida alguna metodología para desarrollar software y en la gran mayoría un 85% para más exactitud, los clientes optan por dar más importancia a la funcionalidad de la aplicación que a la seguridad de la mismas, por parte de las empresas esta consideran que es necesario el desarrollo de requerimientos de seguridad e importante pero en la gran mayoría de las ocasiones se pasan desapercibidos debido a que no se tiene conciencia de que el desarrollo de software seguro es transversal a la elaboración de la aplicación y se debe especificar requerimientos específicos durante todo el ciclo de vida del software.

Adicionalmente, la metodología ágil más usada es Scrum debido a la adaptación que se le puede dar, a pesar de esto los encuestados consideran que esta metodología no es lo suficientemente robusta para dar manejo a requerimientos o historias de usuario centradas en seguridad de software en vez de esto la observan como una metodología para realizar proyectos a corto o mediano plazo además del riesgo por falta de experiencia en su uso, informada por los encuestados, ya que solo el 30% de los que utilizan la metodología han desarrollado proyectos que implican desarrollo de software seguro como premisa, para las metodologías Open Up y XP se tiene una relación similar pues la experiencia en la práctica no es significativa y aunque las consideran como metodologías eficientes y competentes se observa que no optan por adaptarlas al desarrollo de software seguro.

Las implicaciones de seguridad en el desarrollo de proyectos con metodologías ágiles son las siguientes:

- Los artefactos de seguridad que se generan se centran en mayor medida en análisis de riesgos y pruebas basadas en el riesgo.
- Los encuestados indican que según su experiencia la metodología Scrum no se adapta para el desarrollo de requerimientos centrados en la seguridad.
- Al hacer uso de metodologías ágiles no se reconoce de manera clara en cuál de las etapas, dependiendo la metodología usada se debe comenzar a incluir temas acordes a la seguridad del software.
- En el 65% de las oportunidades no se escoge metodologías ágiles como XP, Scrum u Open Up para desarrollar proyectos centrados en software seguro, dada la incertidumbre de su uso para este tipo de proyectos.

En términos generales, los resultados obtenidos evidencian los inconvenientes que las empresas presentan en temas de desarrollo de software seguro, capacitación

de personal especializado en el tema, estimación y definición para la ejecución de proyectos que contengan un componente de seguridad alto, desconocimiento de herramientas para la verificación de fallas relacionadas a la seguridad en el sistema y diseño de pruebas enfocadas a detectar errores de seguridad.

5.2 TRABAJOS FUTUROS

Teniendo en cuenta el trabajo realizado, se sugiere un trabajo posterior donde se involucren un mayor número de empresas, sectorizándolas por empresas desarrolladoras de software, empresas comerciales, entidades del estado, instituciones financieras; tanto públicas como privadas. Así mismo, incluir otros aspectos relacionados con el desarrollo de software seguro, que en este documento no se contemplaron.

De igual manera contemplar la seguridad según los leguajes de programación usados.

BIBLIOGRAFÍA

- Abundis, C. J. (Diciembre de 2013). Metodologías para desarrollar software seguro . Zacatecas, Zacatecas, Mexico.
- Alaimo, D. M. (2013). *PROYECTOS ÁGILES CON SCRUM .* Buenos Aires: Kleer.
- Amey, P. (2006). *US-CERT*. Obtenido de https://www.us-cert.gov/bsi/articles/knowledge/sdlc-process/correctness-by-construction
- Beck, K., & Beedle, M. (2001). *Manifesto for Agile Software Development*. Obtenido de Manifesto for Agile Software Development: http://www.agilemanifesto.org/
- Bisogno, M. V. (12 de Octubre de 2004). Metodología para el Aseguramiento de Entornos Informatizados. *Metodología para el Aseguramiento de Entornos Informatizados*. Buenos aires, Argentina.
- Casey, E. (2008). *Malware Forensics.* Syngress.
- Congreso_Colombiano. (2009). Ley No. 1273. En C. colombia, *Código penal.* Bogota: REPÚBLICA DE COLOMBIA - GOBIERNO NACIONAL.
- García, F. T. (2015). Tema X: Desbordamientos de memoria. Zaragoza, España.
- García, M. (10 de 6 de 2013). *UNIDAD 4 - SEGURIDAD EN INGENIERIA DE SOFTWARE.* Obtenido de blogspot: http://magdalyithunid4.blogspot.com.co/
- González, J. F. (2010). *Introducción a las metodologías ágiles.* UOC.
- Hernández, Y. A. (2009). CONFIGURACIÓN DE LA METODOLOGÍA OPENUP V1.0. Habana, Cuba: Universidad de las Ciencias Informáticas .
- Junco, A. R. (2016). Encuesta nacional de seguridad informática . *SISTEMAS*, 18-37.
- McGraw, G. (2004). *Exploiting Software: How to Break Code.* Addison-Wesley Software Security Series.
- McGraw, G. (2006). *Software Security.* Addison-Wesley Professional.
- Microsoft, C. (2010). *www.microsoft.com.* Obtenido de www.microsoft.com: https://www.microsoft.com/es-ES/download/confirmation.aspx?id=12379

- Naranjo, J. (2017). La sombra del delito informático. *ADN*, 4.
- Quintero, B. (3 de 2011). Information security encyclopedia - malware. Madrid, España.
- Revistadinero. (10 de Marzo de 2016). Los ataques informáticos impactan a las empresas colombianas en US$500 millones. pág. 1.
- Softeng. (23 de Abril de 2016). *SOFTENG*. Obtenido de Softeng: https://www.softeng.es/es-es/empresa/metodologias-de-trabajo/metodologia-scrum.html
- Unir. (2016). Tema 2 Seguridad en el ciclo de vida del software. España.
- Vianca Rosa Vega Zepeda, N. C. (3 de Abril de 2016). INCORPORACIÓN DE PRÁCTICAS DE SEGURIDAD EN EL MERCADO CHILENO. *INCORPORACIÓN DE PRÁCTICAS DE SEGURIDAD EN EL MERCADO CHILENO*. Chile.

Anexos

Anexo 1. Listado de empresas consultadas.

Anexo 2. Encuesta prueba piloto.

Anexo 3. Resultados prueba piloto.

Anexo 4. Encuesta implicaciones de seguridad.