UNIVERSIDAD
NACIONAL
DE COLOMBIA

Propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles

Luisa Fernanda Betancur Cartagena

Universidad Nacional de Colombia
Facultad de Minas, Departamento de Ingeniería de la Organización
Medellín, Colombia
2016

Propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles

Luisa Fernanda Betancur Cartagena

Tesis presentada como requisito parcial para optar al título de:

Magister en Ingeniería Administrativa

Director: Ph.D. en Ciencias Económicas Iván Alonso Montoya Restrepo

Co-Director: Ms. en Ingeniería de Sistemas Jhoany Alejandro Valencia Arias

Línea de Profundización

Universidad Nacional de Colombia

Facultad, Departamento de Ingeniería de la Organización

Medellín, Colombia

2016

A mis padres y mi esposo.

Agradecimientos

Infinitas gracias a Juan Rafael Álvarez Director Administrativo en Fluidsignal Group S.A., Miembro del comité Directivo de S-SQUARE y Nuva por su apoyo en un momento definitivo para el logro de este trabajo, asimismo la dedicación de su bien más preciado, el tiempo.

¡Gracias!, a los profesores Iván Alonso Montoya Restrepo y Jhoany Alejandro Valencia Arias quienes me acogieron y dirigieron con tanta motivación y compromiso. Fomentaron en mí la confianza y capacidad de lograr este gran reto a través de su credibilidad.

Resumen

Este trabajo tiene como objetivo general plantear una propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles, esto se logra inicialmente con la revisión del estado y tendencia actual, la identificación de modelos vigentes propuestos para el desarrollo seguro, la caracterización de principios y prácticas ágiles usadas en la industria para el desarrollo de software y los aspectos de seguridad de la información deseables en proyectos de tecnología con base en la norma ISO 27002. Finalmente, se realiza un análisis del cumplimiento de agilidad y seguridad de las prácticas identificadas en donde se obtienen las más ágiles y seguras, que en conjunto un ejercicio prospectivo sobre las variables estratégicas: Valores, Principios, Objetivos de control, Prácticas y Metodologías ágiles, generan los escenarios probables que permiten orientar una organización en las acciones concretas a emprender para encontrar la senda hacia un futuro más favorable en la implementación del agilismo.

Palabras clave: Metodologías Ágiles, Seguridad, Desarrollo de Software, ISO27002, prácticas ágiles de desarrollo de software.

Abstract

This study concentrated on giving an strategic proposal of secure practices for software development with Agile methodologies, this is achieved by reviewing the current state of use and trend of agile methodologies, presenting current models for secure software development using agile methodologies, also establishing the agile principles and practices used in the industry for software development and selecting the desirable security aspects in software development projects based on the standard ISO 27002. Finally, an analysis is performed to determinate the compliance of agility and security of current practices where the most agile and secure practices are obtained, which together with a prospective exercise on strategic variables associated with the environment such as: values, principles, control objectives and practices agile, allows the identification of the most likely north to give to an organization to find the path to a more favorable implementation agilismo from the knowledge of its potential future action scenarios.

Keywords: Agile Methodologies, Security, Software Development, ISO27002, Sofware development agile practices.

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

Introducción

Dadas las constantes presiones de nuevas características y mejores beneficios para el cliente, múltiples industrias se ven abocadas a incorporar en sus métodos de desarrollo las metodologías ágiles de forma que el tiempo de puesta del producto en manos de los clientes se minimice, generando beneficios para el usuario y la industria (Tassey, 2002).

Los métodos ágiles de desarrollo aceptan el cambio constante y consideran como prescindible gran parte de la documentación técnica, pruebas y el desarrollo de toda funcionalidad que no aporte un valor directo observable para el usuario final y/o cliente (Azham, Ghani & Ithnin, 2011), además, los requisitos de seguridad o no funcionales y las actividades encaminadas a verificar el riesgo de las aplicaciones suelen ser considerados de baja prioridad y "enemigos" naturales de la agilidad (Chivers, 2005).

A pesar de lo expuesto anteriormente, algunas industrias reciben presiones externas, principalmente regulatorias, que exigen la implementación de los temas ágiles en el desarrollo de software con altos estándares de cumplimiento de controles que garanticen un proceso con seguridad y calidad, entre ellas industrias como la farmacéutica, la aeronáutica y la financiera, son altamente reguladas debido al alto impacto que tienen sus productos social y económicamente, en donde se les obliga a implementar prácticas y controles dentro del ciclo de vida de desarrollo de software que permitan asegurar la calidad del producto.

A partir de estas situaciones, las organizaciones acuden a la adaptación y conjunción de marcos para la gestión y gobierno de las tecnologías y seguridad de la información, resultando en una "mezcla" de prácticas, que en muchos casos pueden resultar en implementación de acciones costosas y no efectivas si no se tiene conocimiento y experiencia en la aplicación y selección de dichos marcos (Mesquida & Mas, 2015).

Con base en lo anterior, surge la necesidad de plantear una propuesta estratégica, en donde a partir de la selección de las prácticas más ágiles y seguras y la identificación de los escenarios estratégicos más probables, se pueda dar norte a una organización sobre acciones concretas a emprender para encontrar la senda hacia un futuro más favorable en la implementación del agilismo.

Por consiguiente, este trabajo tiene como objetivo general plantear una propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles. A su vez contempla como objetivos específicos: revisar estado actual de uso y tendencia de las metodologías ágiles, identificar los modelos actuales propuestos para el desarrollo seguro en metodologías ágiles, determinar principios y prácticas ágiles usadas en la industria para el desarrollo de software, determinar los aspectos de seguridad de la información deseables en proyectos de desarrollo de software con base en ISO 27002- categoría de control número 14. Adquisición, Desarrollo y Mantenimiento De Sistemas y realizar análisis prospectivo del cumplimiento de agilidad y seguridad de las prácticas actuales.

Este documento está organizado de la siguiente manera. El Capítulo 1 presenta los conceptos teóricos que incluyen la revisión del estado actual de uso y tendencia de las metodologías ágiles, principios y prácticas ágiles usadas en la industria, los aspectos de seguridad de la información deseables en proyectos de desarrollo de software y los modelos actuales propuestos para el desarrollo seguro en metodologías ágiles, respectivamente. En los Capítulos 2 y 3, se presentan los principios de agilidad y seguridad seleccionados para fundamentar la propuesta, así mismo se realiza el análisis del cumplimiento de agilidad y seguridad de las prácticas actuales y se ejecuta el ejercicio prospectivo para la identificación de los escenarios posibles en los cuales se analizan las variables estratégicas asociadas al entorno como lo son: Valores, Principios, Objetivos de control, Prácticas y Metodologías ágiles, respectivamente. Finalmente, en el Capítulo 4 se presentan las conclusiones y el trabajo futuro propuesto.

1. Marco teórico

En este capítulo se presentan los conceptos teóricos que incluyen la revisión del estado actual de uso y tendencia de las metodologías y prácticas ágiles usadas en la industria, los aspectos seguridad de la información deseables en proyectos de desarrollo de software con base en la norma ISO 27002- categoría de control número 14. Adquisición, Desarrollo y Mantenimiento de Sistemas y los modelos actuales propuestos para el desarrollo seguro en metodologías ágiles.

1.1 Estado actual de uso y tendencia de las metodologías ágiles

El desarrollo de un producto de software se contempla como un proceso, el cual requiere la aplicación de técnicas y procedimientos que permitan asegurar la calidad del producto. A esos conjuntos de pasos y etapas se les conoce como métodos de desarrollo de software, que son recursos sistemáticos para guiar a los desarrolladores en la construcción de software confiables y de calidad, de manera productiva, disciplinada y predecible (Martínez, 2012).

Los métodos permiten representar las etapas del ciclo de vida (requisitos, análisis, diseño, codificación, prueba, implementación y mantenimiento) al igual que ordenar actividades y designar funciones. Al mismo tiempo, la principal intención de la ingeniería del software es mejorar "la calidad de sus productos" para lograr que estos sean competitivos, y así se ajusten a los requerimientos establecidos por el cliente (usuarios finales) (Moreno, González, & Echartea, 2008).

A su vez, la norma ISO/IEC 25010 define "la calidad del producto software como el grado en que dicho producto satisface los requisitos de sus usuarios aportando de esta manera valor. Igualmente, el modelo de calidad del producto definido por esta norma se encuentra compuesto por ocho características de calidad: adecuación funcional, eficiencia de desempeño, compatibilidad, usabilidad, fiabilidad, mantenibilidad, portabilidad y seguridad" (ISO/IEC 25000, 2015).

Generalmente la ingeniería de software enfatiza en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, este esquema tradicional ha demostrado ser efectivo y necesario en proyectos de gran tamaño donde por lo general se exige un alto grado de formalismo en su desarrollo, sin embargo, la realidad de los proyectos se ubica en un entorno en donde se exige reducir drásticamente los tiempos de desarrollo manteniendo una alta calidad (Izaquita, 2011).

En efecto, el Grupo Standish publica desde 1994 el informe anual CAOS **(CHAOS Reports)**, en el cual presenta el estado de la industria de desarrollo de software. En el informe de 2015 presentó el desempeño y resultados de más de 50.000 proyectos en todo el mundo, que van desde pequeñas mejoras a los sistemas masivos hasta implementaciones de re-ingeniería. La definición de éxito utilizada en el informe incluye factores como: tiempo, presupuesto, foco, objetivo, valor y satisfacción del cliente (Hastie & Wojewoda, 2015).

En la Tabla 1 se presenta el resumen de los resultados de los proyectos analizados en los últimos cinco años dentro del reporte en mención. Las cifras presentadas indican trabajo por hacer en torno a la consecución de los resultados exitosos de los proyectos de desarrollo de software ya que la sumatoria de resultados fallidos y retados, en promedio muestra que el 71% de los proyectos no entregaron valor satisfactorio al usuario final. Se entiende exitoso aquel proyecto que se termine a tiempo y dentro del presupuesto, con todas las características y funciones especificadas como en un principio. Proyecto retado, el que termina y se encuentra en funcionamiento, pero presenta exceso de presupuesto, sobrestimación de tiempo, y ofrece menos características y funciones de las originalmente especificadas y, finalmente, proyectos fallidos o con problemas, aquel que fue cancelado en algún momento durante el ciclo de desarrollo:

Tabla 1-1 Resultados de los proyectos de software en los últimos cinco años.
Pág. 1 (Hastie & Wojewoda, 2015).

Resultado	2011	2012	2013	2014	2015	Promedio
Exitoso	29%	27%	31%	28%	29%	**29%**
Retado	49%	56%	50%	55%	52%	**52%**
Fallido	22%	17%	19%	17%	19%	**19%**
Retado + Fallido	**71%**	**73%**	**69%**	**72%**	**71%**	**71%**

Asimismo, el reporte presenta que los proyectos pequeños tienen una probabilidad mucho mayor de éxito que los más grandes, como se muestra en la Tabla 2 el 62% de los proyectos pequeños contaron con éxito en su desempeño con respecto al 8% de los de tamaño gigante y grande (Tabla 2). De modo que

el resultado de éxito de proyectos es acorde al tamaño. Se entiende el costo medio de un proyecto de desarrollo de software para una compañía grande alrededor de US$2.322.000; para una empresa mediana a partir de US$1.331.000; y para una empresa pequeña desde US$ 434.000 (Hastie & Wojewoda, 2015).

Tabla 1-2 Resultado de éxito de proyectos acorde al tamaño.
Pág. 1 (Hastie & Wojewoda, 2015).

Tamaño	Exitoso	Retado	Fallido
Gigante	2%	7%	17%
Grande	6%	17%	24%
Medio	9%	26%	31%
Moderado	21%	32%	17%
Pequeño	**62%**	16%	11%
Total	100%	100%	100%

Por consiguiente, en relación al desarrollo de proyectos de software existe una variedad de propuestas metodológicas que promueven y ofrecen el uso de herramientas y artefactos que se tienen en cuenta a medida que se va elaborando la aplicación para garantizar el éxito del proyecto. Estas pretenden reducir costos, aprovechar los recursos disponibles, medir el avance y mejorar la calidad del sistema (Martínez, 2012).

Las mtodologías de desarrollo de software se pueden clasificar en dos grupos, las basadas en planes o tradicionales, que hacen énfasis en la documentación y control del proyecto y las ágiles dirigidos para equipos de desarrollo pequeño, se enfocan en el factor humano y consideran al cliente como un elemento principal dentro del proyecto (Martínez, 2012).

En la Tabla 3 se presenta un resumen comparativo entre metodologías ágiles y las denominadas tradicionales, lo cual permite un mejor entendimiento de cada una:

Tabla 1-3 Comparación entre Metodologías Tradicionales y Metodologías Ágiles.
Pág. 19 (Izaquita, 2011).

Metodologías Tradicionales	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.	Basadas en heurísticas provenientes de prácticas de producción de código.

Metodologías Tradicionales	Metodologías Ágiles
Resistencia al cambio debido al impacto que representa en el plan de trabajo.	Especialmente preparados para el cambio durante el proyecto.
Impuestas externamente.	Impuestas internamente (por el equipo).
Proceso controlado con políticas y normas.	Proceso menos controlado, con pocas políticas.
Existe un contrato prefijado.	No existe contrato tradicional o al menos es bastante flexible.
El cliente interactúa con el equipo de desarrollo mediante reuniones.	El cliente es parte del equipo de desarrollo.
Grupos grandes y posiblemente distribuidos.	Grupos pequeños (menos de 10 personas) y trabajan en el mismo sitio.
Más artefactos.	Pocos artefactos.
Más roles.	Pocos Roles.
La arquitectura del software es esencial y se expresa mediante modelos.	Menos énfasis en la arquitectura del software más énfasis en la construcción iterativa e incremental.

A continuación se revisará en detalle los principios y características de cada una de las metodologías con el propósito de lograr un mejor entendimiento.

1.1.1 Métodos basados en planes

Los métodos basados en planes se caracterizan por seguir una planificación estrictamente definida desde el inicio del proyecto. Estos métodos dividen el desarrollo en fases, en las cuales se busca ejecutar actividades y tareas acorde a lo planeado. Entre estas fases se destacan el análisis y diseño del proyecto así como el desarrollo de la aplicación. Otra característica de estos métodos es que exigen la elaboración de documentación en cada una de las fases del desarrollo. Estos métodos basados en planes presentan inconvenientes, como la rigidez y la extensa documentación que retarda la entrega del producto (Martínez, 2012). En esta categoría los más utilizados son (Izaquita, 2011):

Lineal en cascada o *Waterfall*: el proceso se hace en pasos sucesivos o etapas definidas siguiendo un orden determinado y completando cada etapa antes de comenzar la siguiente.

En espiral: El proceso se hace en sucesivos pasos o etapas pero no es necesario terminar una etapa para comenzar con la siguiente, en un momento dado se pueden tener varias etapas activas a la vez o se pueden iniciar micro–procesos iterativos para ir completando diferentes partes del proyecto.

Como lo menciona Alaimo (2013) los problemas detectados en los modelos para el desarrollo de software tradicionales o de tipo cascada se fundamentan, por un lado, en el entorno altamente cambiante propio de la industria, y por el otro, en el proceso mismo de desarrollo donde el resultado depende de la actividad cognitiva de las personas más que de las prácticas y controles empleados, por tanto indica que las metodologías en cascada resultaron "pesadas" y prohibitivas para responder satisfactoriamente a los cambios de negocio en los proyectos de software.

Por consiguiente, las metodologías ágiles buscan proporcionar efectividad en proyectos con requisitos que cambian frecuentemente durante el desarrollo del proyecto y reducir los tiempos del desarrollo con respecto a las metodologías tradicionales, sin dejar de cumplir con la calidad requerida por la organización en su producto y/o servicio, lo que ha generado una gran acogida en el entorno global (Cañadillas, 2010).

En efecto, el Grupo Standish en el informe anual CAOS (*CHAOS Reports*) del 2015 presenta los resultados de desempeño de los proyectos ágiles (*Agile*) y tradicionales (*Waterfall*) registrados entre el 2011 y 2015, indicando que a través de todos los tamaños de proyecto los enfoques ágiles dieron lugar a proyectos más exitosos, como se muestra en la Tabla 4 (Hastie & Wojewoda, 2015):

Tabla 1-4 Desempeño de los proyectos metodologías ágiles Vrs Cascada.
Pág. 1 (Hastie & Wojewoda, 2015).

Tamaño	Método	Exitoso	Retado	Fallido
Todos los tamaños del proyecto	Ágil	**39%**	**52%**	**9%**
	Cascada/ tradicional	**11%**	**60%**	**29%**
Proyectos grandes	Ágil	18%	59%	23%
	Cascada/ tradicional	3%	55%	42%
Proyectos medianos	Ágil	27%	62%	11%
	Cascada/ tradicional	7%	68%	25%

Tamaño	Método	Exitoso	Retado	Fallido
Proyectos pequeños	Ágil	58%	38%	4%
	Cascada/ tradicional	44%	45%	11%

A continuación, se revisará los fundamentos de las metodologías ágiles, la evolución de las mismas y se detallará Scrum como la metodología más usada en la actualidad.

1.1.2 Métodos ágiles

Como una alternativa a los inconvenientes que los métodos basados en planes exhiben, surgen los métodos ágiles. Estos métodos constituyen un enfoque en el desarrollo de aplicaciones que se orientan a proyectos en contextos cambiantes y adaptativos y que requieren la entrega de productos en tiempos reducidos. Así, se busca lograr la calidad, procurando la simplicidad y la creación de aplicaciones de software flexibles (Martínez, 2012). Se enfocan en satisfacer al cliente mediante la entrega temprana y continua de software con valor (Beck et al., 2001).

La piedra angular del movimiento ágil es conocida como *Manifiesto por el desarrollo ágil de software.* Este se de los siguientes 4 valores y 12 principios (Cañadillas, 2010):

Ilustración 1-1 Valores del agilismo (Beck et al., 2001).

1. **Se valora el individuo y las interacciones del equipo de desarrollo sobre el proyecto y las herramientas**: la gente es el principal factor de éxito en un proyecto de software, es más importante construir un buen equipo que construir el entorno, se debe crear primero el equipo y esperar a que este configure su entorno de acuerdo con sus necesidades de desarrollo.
2. **Desarrollar software que funcione más que conseguir una buena documentación**: 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 fundamental. 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. **La colaboración con el cliente más que la negociación de un contrato:** se propone que exista una interacción constante entre el cliente y el equipo de desarrollo, esta colaboración será la que marque la marcha del proyecto y asegure su éxito.
4. **Responder a los cambios más que seguir estrictamente un plan:** la habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (requisitos, tecnología, equipo) determinan el éxito o fracaso del mismo, por lo que la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los siguientes principios del manifiesto ágil, los cuales representan las características que diferencian un proceso ágil de uno tradicional (Beck et al., 2001):

Ilustración 1-2 Principios del Agilismo (Beck et al., 2001).

El manifiesto por el desarrollo ágil de software es el resultado del trabajo colaborativo de un grupo formado por diecisiete personas, entre desarrolladores de software, escritores y consultores, quienes lo construyeron y suscribieron en 2001. La firma y publicación del Manifiesto en ese año no implica que esa sea la fecha de origen de las metodologías ágiles o que antes de ese año no existieran, sino el reconocimiento de la necesidad y la expresión de un lineamiento común capaz de hacer posible algún tipo de agrupación entre ellas (Navarro, Fernández, & Morales, 2013). La Tabla 5 muestra un resumen de la forma cómo han evolucionado estas metodologías:

Tabla 1-5 Evolución de las metodologías ágiles. Pág. 16 (Izaquita, 2011).

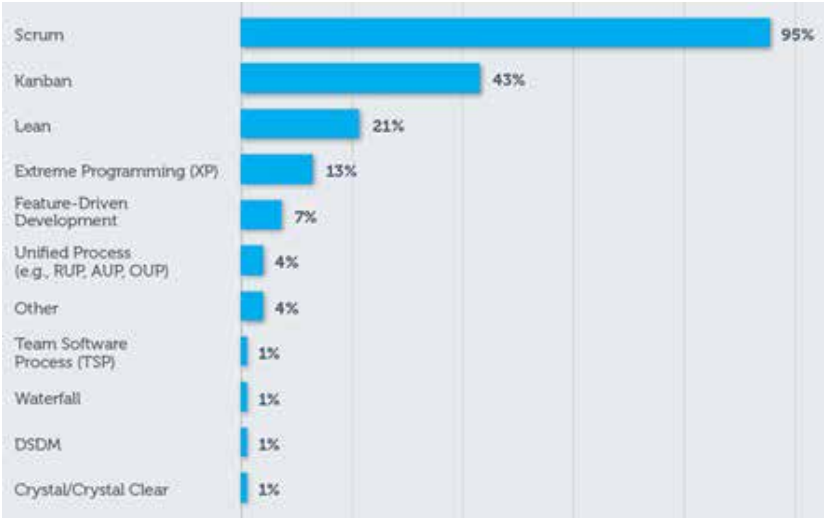
Metodología	Acrónimo	Creación	Tipo de modelo	Característica
Evolutionary Project Management	EVO	Gilb 1976	Marco de trabajo adaptativo	Primer método ágil Existente
Microsoft Solutions Framework	MSF	Microsoft 1994	Lineamientos, disciplinas, prácticas	Marco de trabajo para el desarrollo de soluciones
Scrum	Scrum	Sutherland 1994 Schwaber 1995	Proceso – Marco de trabajo administrativo	Complemento de otros métodos, ágiles o no
Rapid Development	RAD	McConnell 1996	Encuesta de técnicas y modelos	Selección de mejores prácticas
Dynamic Solutions Delivery Model	DSDM	Stapleton 1997	Marco de trabajo/ modelo de ciclo de vida	Creado por 16 expertos RAD
Cristal Methods	CM	Cockbum 1998	Familia de metodologías	Metodología ágil con énfasis en modelo de ciclos
Agile RUP	dX	Booch, Martin, Newkirk 1998	Marco de trabajo/ Disciplina	XP dado vuelta con artefactos RUP
eXtreme Programming	XP	Beck 1999	Disciplina en prácticas de ingeniería	Método ágil radical
Feature-Driven Development	FDD	De Luca & Coad 1998 Palmer & Felsing 2002	Metodología	Método ágil de diseño y construcción
Adaptative Software Development	ASD	Highsmith 2000	Prácticas + ciclo de vida	Inspirado en sistemas adaptativos complejos
Lean Development	LD	Charette 2001, Mary y Tom Poppendieck	Forma de pensar – modelo logístico	Metodología basada en procesos productivos
Agile Modeling	AM	Ambler 2002	Metodología basada en la práctica	Suministra modelado ágil a otros métodos

En general las metodologías ágiles de desarrollo se caracterizan por ser iterativas. No intentan minimizar los cambios, sino estar preparados para aceptarlos. Son adaptativas en lugar de repetibles. Priorizan a los individuos y a las interacciones por sobre los procesos. Incorporan retroalimentación sobre el proceso y priorizan la colaboración con el cliente. Buscan minimizar la sobrecarga al proceso metodológico (Izaquita, 2011).

Con base en las metodologías ágiles existentes la Scrum Alliance® fundada en 2001, organización sin ánimo de lucro que fomenta y apoya la adopción generalizada y la práctica efectiva de Scrum con más de 400.000 miembros alrededor del mundo, muestra la tendencia de uso de dichas prácticas en su reporte estado de 2015, para el cual encuestó aproximadamente 5.000 personas a lo largo de 108 países. El 29% de estas personas labora en el sector de tecnologías de la información encontrándose en primer lugar y un 12% en el sector financiero ocupando el tercer lugar de los 15 sectores identificados en el reporte. Estas cifras presentan la tendencia de uso de SCRUM en el desarrollo de software dentro de los diferentes sectores y en particular su importancia en el sector financiero. Los resultados más representativos indican que actualmente el 82% de los encuestados ya usa Scrum, y otro 11% lo están experimentando (Scrum Alliance, 2015).

Así mismo, el informe presenta que del total encuestado un 95% planea continuar con el uso de Scrum en su organización como filosofía principal para el desarrollo de software tal como se puede observar en la Ilustración 3 Tendencia del uso de prácticas ágiles.

Ilustración 1-3 Tendencia del uso de prácticas ágiles (Scrum Alliance, 2015).



Del informe se observa que las otras metodologías más comúnmente usadas son Kanban, Lean y XP (*Extreme Programming*), respectivamente, en donde el 54% de los encuestados dijeron que usan Scrum en combinación con otras prácticas, mientras que el 42% informó de uso exclusivo de Scrum, esto indica que la mayoría de equipos ágiles han adaptado prácticas ágiles de distintas metodologías para generar un proceso de desarrollo propio que se ajusta a sus necesidades (Scrum Alliance, 2015). Estas adaptaciones parecen estar centradas en mezclas de prácticas enfocadas en el desarrollo de software y la administración de proyectos, en donde el éxito no está relacionado con una metodología en particular, sino con el concepto de metodologías ágiles en general; de hecho, el eje común a los temas tratados es la presentación de experiencias exitosas de implantación (Navarro et al., 2013).

A continuación una breve mirada a los principales aspectos de Scrum como metodología ágil de desarrollo de software de mayor uso en la actualidad.

1.1.3 Scrum

Su nombre no corresponde a una sigla, sino a un concepto deportivo, propio del rugby, relacionado con la formación requerida para la recuperación rápida del juego ante una infracción menor. Su primera referencia en el contexto de desarrollo data de 1986, cuando Takeuchi y Nonaka utilizan el *Rugby Approach* para definir un nuevo enfoque en el desarrollo de productos, dirigido a incrementar su flexibilidad y rapidez, a partir de la integración de un equipo interdisciplinario y múltiples fases que se traslapan entre sí (Navarro et al., 2013).

Define un proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos mediante la aceptación de la naturaleza caótica del desarrollo de software y la utilización de prácticas tendientes a manejar la predictibilidad y el riesgo a niveles aceptables (Izaquita, 2011). Es un marco de trabajo diseñado para lograr la colaboración eficaz de equipos Scrum en proyectos, que emplea un conjunto de reglas y artefactos y define roles que generan la estructura necesaria para su funcionamiento (Navarro et al., 2013).

Los llamados equipos Scrum son auto-gestionados, multifuncionales y trabajan en iteraciones. La autogestión les permite elegir la mejor forma de hacer el trabajo, en vez de tener que seguir lineamientos de personas que no pertenecen al equipo y carecen de contexto. Los integrantes del equipo tienen todos los conocimientos necesarios (por ser multifuncionales) para llevar a cabo el trabajo (Navarro et al., 2013).

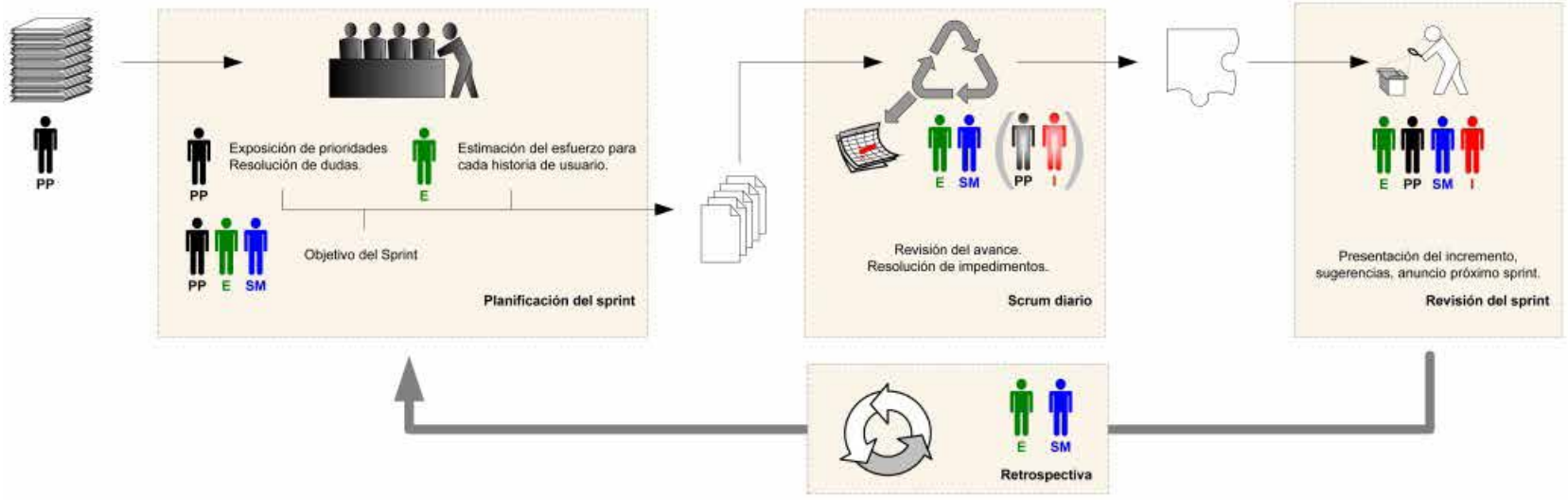
El marco de trabajo Scrum se fundamenta en los equipos Scrum, roles, eventos, artefactos y reglas asociadas. Cada componente dentro del marco de trabajo sirve a un propósito específico y es esencial para el éxito de Scrum; a continuación se definen sus principales fundamentos:

Ilustración 1-4 Roles Equipo Scrum (Menzinsky & Palacio, 2015).

Ilustración 1-5 Artefactos Scrum (Menzinsky & Palacio, 2015)

Ilustración 1-6 Eventos Scrum (Menzinsky & Palacio, 2015).

Ilustración 1-7 Marco de trabajo Scrum (Menzinsky & Palacio, 2015).



En resumen a medida que han pasado los años, y con el advenimiento de las economías globalizadas y los entornos web, el contexto de negocio de los sistemas ha pasado de ser relativamente estable a convertirse en un contexto altamente volátil, donde los requerimientos expresados hoy, en pocas oportunidades son válidos unos meses más tarde. Bajo esta nueva realidad, las metodologías tradicionales resultaron prohibitivas para responder satisfactoriamente a los cambios de negocio (Alaimo, 2013).

Como respuesta a las necesidades que plantea el entorno, las metodologías ágiles funcionan bien dentro de un contexto específico caracterizado por equipos pequeños de desarrollo, ubicados en el mismo sitio, con clientes que pueden tomar decisiones acerca de los requerimientos y su evolución, con requerimientos que cambian con frecuencia (semanal, mensual), con alcance del proyecto o presupuesto variable, con pocas restricciones legales y con pocas restricciones en el proceso de desarrollo. Por fuera de este ambiente, es común que se presenten inconvenientes derivados de la falta de participación del cliente, los contratos con precio fijo, los proyectos con arquitectura o diseño intensivo o con documentación intensiva, la tasa de cambios lenta y los equipos distribuidos (Navarro et al., 2013).

A continuación se presentarán las prácticas ágiles más usadas en la industria para el desarrollo de software.

1.2 Prácticas ágiles usadas en la industria para el desarrollo de software

Buscar el compromiso de calidad, alcance y tiempo en el desarrollo de proyectos se puede llevar a cabo a través de prácticas de desarrollo y teorías de gestión diseñadas para tal fin, como son las metodologías ágiles, es por eso que las prácticas de desarrollo de software son un conjunto de actividades o procedimientos que al ser incorporadas dentro de un proyecto logran apoyar la ejecución y mejoramiento del proceso, permitiendo tener software con calidad. Es decir, tanto las metodologías y como las prácticas tienen en común un objetivo principal, y es responder eficaz y rápidamente a los cambios obteniendo resultados positivos de ello (Cañadillas, 2010).

En efecto la *Agile Alliance*, organización sin fines de lucro con membresía global, comprometida con el avance de los principios y prácticas de desarrollo ágil con el propósito de hacer de la profesión asociada al software más productiva, humana y sostenible, plantea una guía de prácticas ágiles, donde describe para cada una de ellas los beneficios esperados, errores más comunes, orígenes e historia de la práctica, además, vincula estudios académicos en los cuales existe constancia de los efectos de la práctica en su uso, con el propósito de ayudar a la comunidad de profesionales ágiles (Agile Alliance, 2015b).

Ilustración 1-8 Mapa de prácticas ágiles (AgileAlliance, 2015b)

La *Agile* Alliance presenta 60 prácticas agrupadas en las siguientes 9 áreas de interés como se observa en la Ilustración 8: Programación extrema XP (*eXtreme programming*), Equipos (*Teams*), Lean, Scrum, Administración del producto PM (*Product Management*), DevOps, Diseño (*Design*), Pruebas (*Testing*) y Fundamentos (Fundamentals). Esta guía es un documento vivo, que evoluciona con el tiempo para reflejar los comentarios de la comunidad ágil sobre prácticas nuevas o emergentes.

Con base la Ilustración 8, Mapa de prácticas ágiles (Agile Alliance, 2015b), se observa que estas prácticas engloban tanto el propio desarrollo del software como la gestión del proyecto. Así, las metodologías ágiles están centradas en la parte de desarrollo software o en la gestión de los procesos del proyecto, según la distinta metodología aplicada. Se pueden mezclar incluso la aplicación de distintas metodologías y prácticas con el fin de agilizar todo el proceso, desde la gestión hasta la programación de código del software (Cañadillas, 2010).

Además del punto de vista metodológico existen ciertas prácticas dirigidas a la programación ágil, como lo son las asociadas a las áreas de diseño (Design), desarrollo (DevOps) y pruebas (Testing), grupos en los cuales se va a centrar este estudio, porque detallan a más bajo nivel técnicas de desarrollo de software de manera ágil, ver Tabla 6.

Tabla 1-6 Prácticas para el desarrollo de software ágil. Pág. 1 (Agile Alliance, 2015a).

ID	Práctica	Descripción
P1- Área de interés: Pruebas	**Pruebas de usabilidad - *Usability Testing***	Consiste en la observación de un usuario final representante de interactuar con el producto, teniendo en cuenta una meta a alcanzar, pero no hay instrucciones específicas para el uso del producto. Los miembros del equipo observan las acciones del usuario sin intervenir, registran lo que sucede (ya sea de manera informal, por ejemplo, tomar notas, o más ampliamente, el uso de vídeo, de seguimiento ocular, capturas de pantalla o software especializado). El análisis se centrará en las dificultades encontradas por el usuario, lo que ilustra las diferencias entre hipótesis del equipo y el comportamiento real.
P2 - Área de interés: Pruebas	**Pruebas Exploratorias - *Exploratory Testing***	Destaca autonomía, habilidad y creatividad del que prueba. Son pruebas intuitivas, con base en unas tareas específicas asociadas a unos objetivos concretos. Se completa con un reporte de errores sobre un producto realizado por un probador en particular en un tiempo determinado y con una misión a cumplir.

ID	Práctica	Descripción
P3 - Área de interés: Pruebas	**Pruebas de unidad - *Unit Testing***	Es un fragmento de programa corto escrito y mantenido por los desarrolladores en el equipo de producto, que revisa una pequeña parte del código fuente del producto y comprueba los resultados. El resultado de una prueba de unidad es binaria: "pasa" si el comportamiento del programa es consistente con las expectativas grabadas, o "fracasa" de manera contraria.
P4 - Área de interés: Pruebas	**Desarrollo basado en pruebas - *TDD Test-driven development***	Se refiere a un estilo de programación en el que tres actividades están estrechamente entrelazados: la codificación, las pruebas (en forma de pruebas de unidad) y diseño (en forma de reconstrucción de código).
P5 - Área de interés: Pruebas	**Objetos simulados - *Mock Objects***	Una técnica de uso común en el contexto de la elaboración de las pruebas unitarias automatizadas. Consiste en crear instancias de una versión de prueba específica de un componente de software (por lo general una clase), que en lugar de los comportamientos normales proporciona resultados precalculados, y a menudo también comprueba que se invoque como se esperaba por los objetos que se está probando. Por ejemplo, la versión de "Mock / simulacro" de un componente de base de datos tiene dos funciones a) dar respuestas "enlatadas" a las consultas de bases de datos, en lugar de conectarse a una base de datos real, y b) verificar que se está accediendo a la base de datos de la manera esperada y se estipula en el examen.
P6 - Área de interés: Pruebas	**Pruebas de aceptación - *Acceptance Testing***	Es una descripción formal de la conducta de un producto de software que será aceptada por el usuario, generalmente expresado como un escenario de uso o ejemplo. De manera similar a una prueba de unidad, una pruebas de aceptación se entiende generalmente para tener un resultado binario, pasar o fallar; un fracaso sugiere "a pesar de que no demuestra" la presencia de un defecto en el producto.
P7 - Área de interés: Pruebas	**Desarrollo impulsado por la prueba de aceptación - *ATDD Acceptance Test Driven Development***	Análogo al desarrollo basado en pruebas, esta práctica consiste en el uso de pruebas de aceptación automatizados con la restricción adicional de que estas pruebas se pueden escribir antes de la implementación de la funcionalidad correspondiente. En la situación ideal (aunque rara vez se logra en la práctica), el experto dueño del producto, cliente o dominio es capaz de especificar nuevas funcionalidades escribiendo nuevas pruebas de aceptación o de casos de prueba, sin necesidad de consultar a los desarrolladores.

ID	Práctica	Descripción
P8 - Área de interés: Pruebas	**Desarrollo impulsado por el comportamiento - BDD (Behaviour Driven Development)**	Esunasíntesisyelperfeccionamientodelasprácticasderivadasde TDD (Test Driven Development) y ATDD (Acceptance Test Driven Development). BDD aumenta TDD y ATDD con las siguientes tácticas: Aplicar el principio del "Cinco por qué" a cada historia de usuario propuesto, por lo que su objetivo está claramente relacionado con los resultados del negocio. Pensar "de afuera hacia adentro", en otras palabras, implementar c aquellos comportamientos que contribuyen más directamente a estos resultados de negocio, con el fin de minimizar los residuos. Describir los comportamientos en una única notación disponible para los expertos de dominio, probadores y desarrolladores, a fin de mejorar la comunicación. Aplicar estas técnicas de todo el camino hasta los niveles más bajos de abstracción del software.
P9 - Área de interés: Pruebas	**Plantilla Dada-Cuando-Entonces** ***Given – When – Then template***	Es una plantilla destinada a guiar la redacción de las pruebas de aceptación para una historia de usuario: (Dada) algún contexto (Cuando) algún tipo de acción se lleva a cabo (Entonces) un conjunto particular de consecuencias observables que debe obtener.
P10 - Área de interés: Pruebas	**Plantilla Role, Característica, Razón** ***Role-feature-reason template***	Es una de las ayudas más comúnmente recomendados para los equipos y los dueños del producto que empiezan a escribir historias de usuario: Como un yo quiero Así que eso
P11 - Área de interés: Desarrollo	**Control de versiones** ***Versión Control***	El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedan recuperarse versiones específicas más adelante.
P12 - Área de interés: Desarrollo	**Construcción automática** ***Automated Build***	La "construcción" en el lenguaje de desarrolladores consiste en el proceso que, a partir de archivos y otros activos bajo la responsabilidad de los desarrolladores, da como resultado un producto de software en su forma "consumible" final. Esto puede incluir la compilación de archivos fuente, la producción de los instaladores, la creación o actualización de esquema de base de datos o de datos, entre otros. La construcción es "automática" en la medida en que estas medidas son repetibles y no requieren la intervención humana directa, y se puede realizar en cualquier momento sin ninguna información que no sea lo que se almacena en el repositorio de control de código fuente.

ID	Práctica	Descripción
P13 - Área de interés: Desarrollo	**Integración continua** ***Continous Integration***	Los equipos que practican la integración continua buscan dos objetivos: ■ Minimizar la duración y esfuerzo requerido por cada episodio de integración. ■ Estar en capacidad de entregar "en cualquier momento", una versión del producto adecuado para la liberación. El logro de estos objetivos requieren un procedimiento reproducible y automatizado. Esto se logra a través de herramientas de control de versiones , políticas y convenciones de equipo y herramientas diseñadas para ayudar a lograr la integración continua. Para la mayoría de los equipos, la integración continua la práctica asciende a lo siguiente: ■ Uso de una herramienta de control de versiones . ■ Un proceso de construcción y lanzamiento de un producto automatizado. ■ Instrumentación del proceso de construcción para activar las pruebas unitarias y de aceptación "cada vez que un cambio se publica al control de versiones ". ■ Uso de una herramienta tal como un servidor de integración continua, que automatiza el proceso de integración, las pruebas y la notificación de los resultados de pruebas.
P14 - Área de interés: Desarrollo	**Despliegue continuo Continuous Deployment**	Extensión de la integración continua, con el objetivo de reducir al mínimo tiempo de espera, el tiempo transcurrido entre el desarrollo escribir una nueva línea de código y en ser este nuevo código utilizado por los usuarios en vivo, en producción.
P15 - Área de interés: Diseño	**Tarjetas CRC Clases, Responsabilidades, Colaboradores** ***CRC Cards***	Es una actividad con la intención de esbozar rápidamente varias ideas diferentes para el diseño de alguna característica de un sistema orientado a objetos, dos o más miembros del equipo escriben en tarjetas los nombres de las clases más relevantes que intervienen en la función. Las tarjetas contienen la lista de las responsabilidades de cada categoría y los nombres de los colaboradores El siguiente paso es validar – o invalidar como sea el caso – cada idea de diseño jugando un escenario plausible de la computación, cada desarrollador asume el papel de una o más clases. El diálogo puede ir de la siguiente manera, por ejemplo: "¡Hola, Controlador de autenticación Soy una solicitud Web y me gustaría el contenido de este recurso." – "Muy bien, déjame tener sus credenciales para que pueda darles, junto con el nombre de la operación que está intentando llevar a cabo, a nuestra lista de control de acceso, que voy a redirigir a una de nuestro ver componentes en función del resultado, etc. "
P16 - Área de interés: Diseño	**Sesión rápida de diseño** ***Quick Design Session***	Cuando un equipo favorece "diseño simple", los desarrolladores suelen manejar las decisiones de diseño locales momento a momento, pero están en alerta a las opciones de diseño que pueden tener consecuencias de largo alcance. Cuando surge esa elección, dos o más desarrolladores se reúnen para una sesión de diseño rápido en la pizarra, posiblemente con el uso de ayudas de diseño tales como tarjetas CRC.

ID	Práctica	Descripción
P17 - Área de interés: Diseño	**Reglas de simplicidad** ***Rules Of Simplicity***	Son un conjunto de criterios, en orden de prioridad, propuesto por Kent Beck para juzgar si algo de código fuente es "suficientemente simple", si el código: ■ Se verifica mediante pruebas automáticas y todas estas pruebas pasan. ■ No contiene duplicación. ■ Expresa por separado cada idea distinta o responsabilidad. ■ Se compone de un número mínimo de componentes (clases, métodos, líneas) compatible con los tres primeros criterios.
P18 - Área de interés: Diseño	**Reconstrucción** ***Refactoring***	Consiste en la mejora de la estructura interna del código fuente de un programa existente, preservando al mismo tiempo su comportamiento. El "refactoring" sustantivo se refiere a una transformación de la conducta conservadora en particular, como "Extract Method" o "Introducir parámetros".
P19 - Área de interés: Diseño	**Diseño Simple** ***Simple Design***	El equipo al basar su estrategia de diseño de software utiliza los siguientes principios: ■ El diseño es una actividad continua, que incluye reconstrucción y heurística. ■ La calidad del diseño se evalúa con base en las reglas de código de la simplicidad. ■ Todos los elementos de diseño, como "patrones de diseño", arquitecturas basadas en *plugins*, etc. son vistos como costos, por tanto los beneficios y costos de diseño deben estar justificados; ■ Las decisiones de diseño deben ser diferidos hasta que el "último momento", con el fin de recoger la mayor cantidad de información posible sobre los beneficios de la opción elegida antes de incurrir en los costos.
P20 - Área de interés: Diseño	**Lenguaje ubicuo Ubiquitous Language**	Consiste principalmente en esforzarse por utilizar el vocabulario de un dominio de negocio dado, no solo en las discusiones acerca de los requisitos para un producto de software, también en las discusiones sobre el diseño como bien y hasta el final en "código fuente del producto en sí".

Con la evolución de las metodologías ágiles han ido apareciendo multitud de prácticas que se incorporan al proyecto en función de la necesidad del momento, sin embargo la inclusión sobre la marcha puede provocar problemas y retrasos inesperados, lo recomendado es hacer una búsqueda de la calidad en el proceso de programación que proporcione seguridad y valor en el producto final, en donde estas prácticas sean un soporte para el logro de este objetivo de calidad (Cañadillas, 2010).

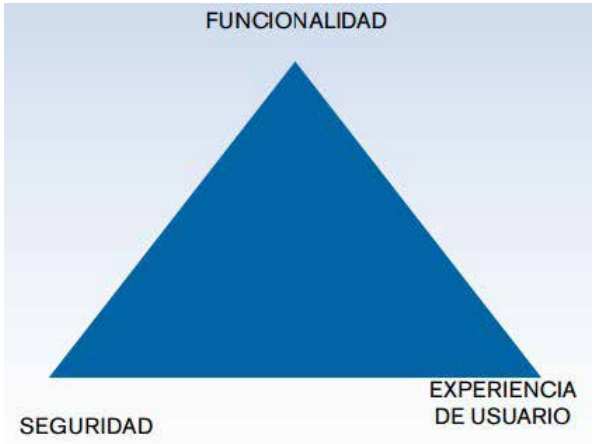
A continuación se presentarán los aspectos seguridad de la información deseables en proyectos de desarrollo de software con base en la norma ISO 27002- categoría de control número 14. Adquisición, Desarrollo y Mantenimiento de Sistemas y los modelos actuales propuestos para el desarrollo seguro en metodologías ágiles.

1.3 Fundamentos de seguridad.

La criticidad y tendencia de uso de los actuales sistemas de información en diferentes dominios de la sociedad determina que, si bien es importante asegurar que el software se desarrolla de acuerdo a las necesidades del usuario, también lo es garantizar que el mismo sea seguro (Castellaro, Romaniz, Ramos, & Pessolani, 2009).

Por lo anterior, es relevante comprender el concepto de software seguro, para esto Carrasco (2013) plantea el triángulo de la seguridad (ver Ilustración 9), donde cada vértice señala un aspecto fundamental del servicio a implantar: la funcionalidad, la seguridad y la experiencia del usuario. Este triángulo es controvertido para los usuarios y a la vez, es la causa de un gran esfuerzo, tanto en recursos como en tiempo, para todos los técnicos relacionados con la implantación de un nuevo servicio. Aunque lo ideal es que este triángulo sea equilátero, la realidad demuestra que cada vez que se hace hincapié en uno de los vértices, se aleja produciéndose una fuerte disminución de los otros dos (Carrasco, 2013).

Ilustración 1-9 Triángulo de la seguridad (Carrasco, 2013).



Con el propósito de entender mejor la importancia y papel de la seguridad en el triángulo se enuncian los siguientes aspectos como sus propiedades fundamentales (Castellaro et al., 2009):

Ilustración 1-10 Propiedades de la seguridad (Castellaro et al., 2009).

- **Confidencialidad**: se refiere a que el software debe asegurar que cualquiera de sus características, incluidas sus relaciones con el ambiente de ejecución y sus usuarios, los activos que administra y/o su contenido son accesibles solo para las entidades autorizadas e inaccesibles para el resto.

- **Integridad**: el software y los activos que administran son resistentes y flexibles a la subversión (modificaciones no autorizadas del código, los activos administrados, la configuración o el comportamiento del software por parte de entidades autorizadas). Esta propiedad se debe preservar durante el desarrollo del software y su ejecución.

- **Disponibilidad**: el software debe estar operativo y accesible a sus usuarios autorizados (humanos o procesos) siempre que se lo requiera; y desempeñarse adecuadamente para que los usuarios puedan realizar sus tareas en forma correcta y dar cumplimiento a los objetivos de la organización que lo utiliza.

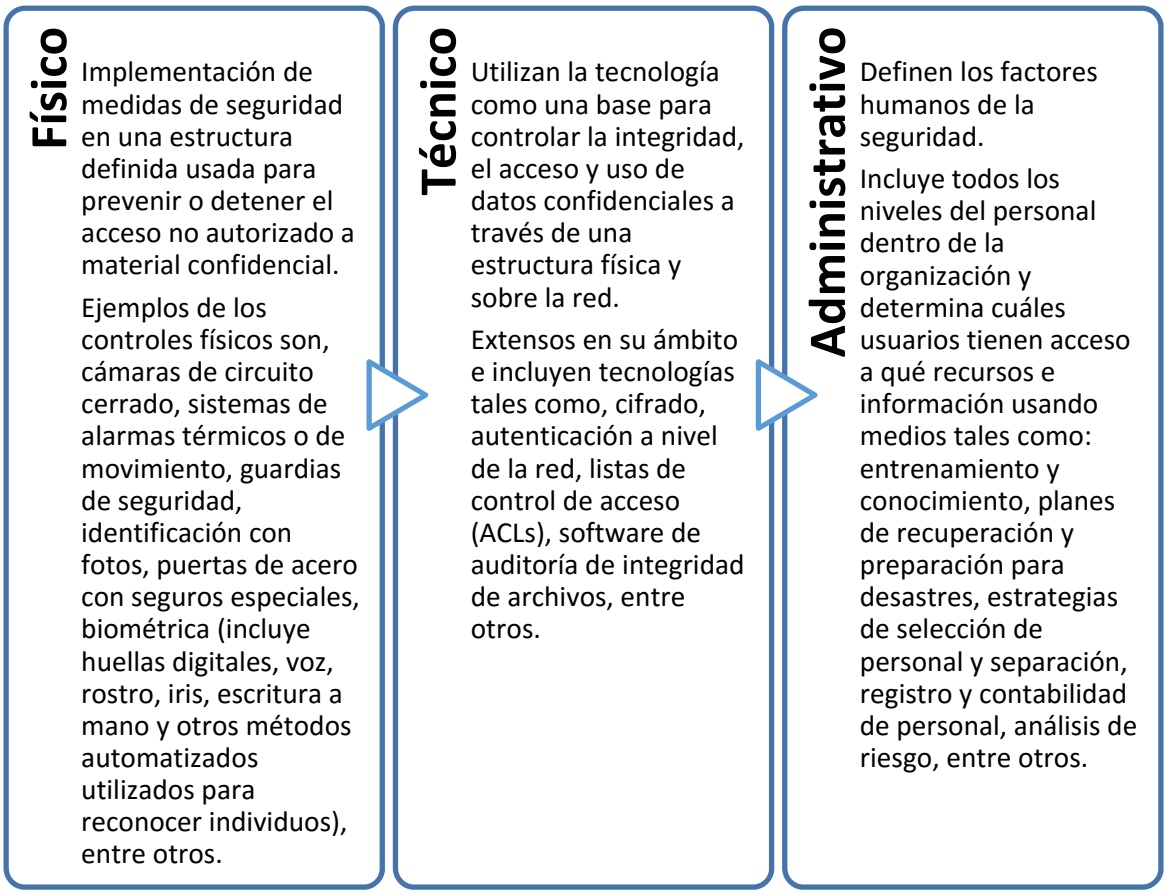
- **Trazabilidad:** todas las acciones relevantes relacionadas con la seguridad de una entidad que actúa como usuario se deben registrar y trazar a fin de poder establecer responsabilidades. La trazabilidad debe ser posible tanto durante la ocurrencia de las acciones registradas como a posteriori.

- **No Repudio**: La habilidad de prevenir que una entidad que actúa como usuario desmienta o niegue la responsabilidad de acciones que han sido ejecutadas.

Las consecuencias de vulnerar la seguridad del software se pueden describir en términos de los efectos sobre estas propiedades fundamentales, es por esto que la gestión de la seguridad en el desarrollo de software es un factor cada vez más determinante en la competitividad de las organizaciones en el entorno global (Franco & Guerrero, 2013).

Ya que la seguridad computacional, término general que cubre una gran área de computación y procesamiento de la información, a menudo se divide en tres categorías fundamentales llamadas controles (Ver Ilustración 11), con los cuales define los objetivos principales de una implementación de seguridad apropiada (Red Hat Inc. / MIT, 2005).

Ilustración 1-11 Categorías principales de controles de seguridad (Red Hat Inc. / MIT, 2005)



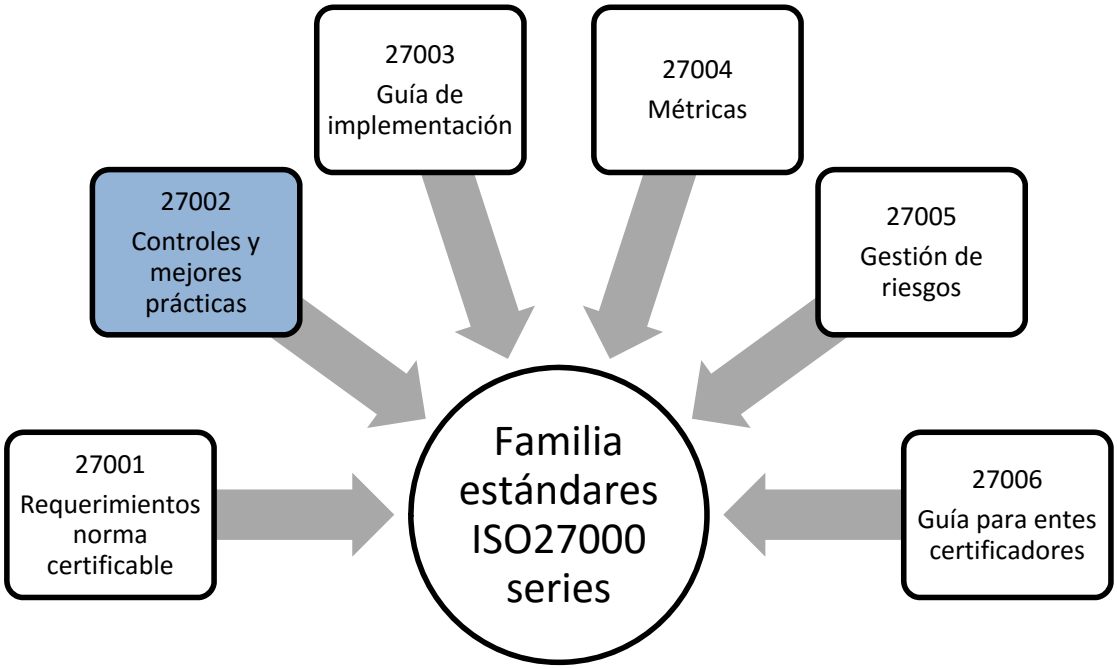
La seguridad en computación y la implementación de los controles que ésta plantea, se ha convertido en un gasto cuantificable y justificable para todos los presupuestos de tecnología debido a que las industrias dependen de sistemas computarizados y redes para ejecutar sus operaciones y transacciones de negocios diarias, por esto consideran sus datos como una parte importante de sus activos generales. En efecto muchos términos y medidas se han incorporado al vocabulario diario en los negocios, tales como costo total de propiedad (total cost of ownership, TCO) y calidad de servicios (QoS). Con estas medidas, las industrias calculan aspectos tales como integridad de los datos y alta disponibilidad como parte de los costos de planificación y administración de procesos. En algunas industrias, como el comercio electrónico, la disponibilidad y confianza de los datos pueden hacer la diferencia entre el éxito y el fracaso (Red Hat Inc. / MIT, 2005).

Desafortunadamente, la seguridad de sistemas y redes pueden ser una proposición difícil, requiriendo conocimiento intrincado de cómo una organización confía, utiliza, manipula y transmite su información. Entender la forma en que la organización lleva el negocio (y la gente que hace la organización) es primordial para implementar un plan de seguridad adecuado (Red Hat Inc. / MIT, 2005).

En consecuencia, la gestión del riesgo y el aseguramiento de la información han conducido a trabajar en la evolución de la mejora de procesos de software también conocida por sus siglas en inglés SPI (*Software Process Improvement*) en donde se han definido modelos estandarizados para buenas prácticas de seguridad como: la norma ISO 27000 series (*ISO, por sus siglas en inglés de International Organization for Standardization*), el estandar COBIT (*Control Objectives for Information and related Technology*), el ISO/IEC 15408-1:2009 Modelo de seguridad para aplicaciones, entre otros (Mesquida & Mas, 2015).

Por su parte, la serie ISO 27000 aglomera todas las normativas en materia de seguridad de la información y está compuesta por los siguientes estándares (Pesántez Verdezoto, 2015):

Ilustración 1-12 Familia de estándares ISO27000 (Pesántez Verdezoto, 2015).



Según el estándar ISO 27001, se define (Pesántez Verdezoto, 2015) la **integridad** como la propiedad de salvaguardar la exactitud de los activos; la **confidencialidad**, propiedad que hace que la información esté disponible y que sea divulgada a personas, entidades o procesos autorizados y, por último, la **disponibilidad** como propiedad de que la información esté disponible y utilizable cuando lo requiera una entidad autorizada. Así mismo, la norma no certificable ISO 27002 establece el lineamiento guía y principios generales para iniciar, implementar, mantener y mejorar la gestión de la Seguridad de la Información dentro de la organización, es decir, que esta norma contiene las mejores prácticas de los objetivos de control y los controles en las siguientes áreas de la gestión de la seguridad de la información para salvaguardar la propiedades de integridad, confidencialidad y disponibilidad (Pesántez Verdezoto, 2015). La última edición de 2013 este estándar ha sido actualizada a un total de 14 Dominios, 35 Objetivos de Control y 114 Controles publicándose inicialmente en inglés y en francés tras su acuerdo de publicación el 25 de Septiembre de 2013 (ISO/IEC 27002, 2013):

- Política de Seguridad
- Organización de la Seguridad de la Información
- Gestión de Activos
- Seguridad de Recursos Humanos

- Seguridad Física y Ambiental
- Gestión de Comunicaciones y Operaciones
- Control de Acceso
- **Adquisición, Desarrollo y Mantenimiento de Sistemas de Información**
- Gestión de Incidentes de Seguridad de la Información
- Gestión de la Continuidad Comercial
- Conformidad

Concretamente en el contexto regional y nacional existen diferentes entidades y organizaciones que optaron por utilizar la serie ISO 27000 como norma referente para garantizar la seguridad de la información, haciendo énfasis en el uso de los controles que plantea el anexo del estándar ISO 27002 (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015), como ejemplo tenemos:

La Superintendencia Financiera de Colombia, entidad encargada de preservar la confianza pública y la estabilidad del sistema financiero; mantener la integridad, la eficiencia y la transparencia del mercado de valores y demás activos financieros; y velar por el respeto a los derechos de los consumidores financieros y la debida prestación del servicio, establece en el Boletín Jurídico No 33 referente a Seguridad y calidad de información, proveedores financieros, acceso a información, el uso de la norma ISO 27000 para propender por la seguridad y calidad para la realización de operaciones, establecer una serie de medidas encaminadas a fortalecer la seguridad (confidencialidad, integridad y disponibilidad - ISO 27000) en el manejo de la información de los clientes y usuarios de las entidades vigiladas por la Superintendencia Financiera de Colombia (SFC/Superintendencia Financiera de Colombia, 2015).

Así mismo, el Ministerio de Tecnologías de la Información y las Comunicaciones a través del Plan Nacional de TIC Colombia 2010-2019, también contempla la relevancia del sector de la seguridad informática, cuyo objetivo es "Establecer lineamientos generales y prácticos en los temas de seguridad de la información desde la perspectiva del ciudadano; de la experiencia técnica y administrativa de las organizaciones, los estándares y las buenas prácticas; y de la protección de infraestructura crítica de la nación" (Franco & Guerrero, 2013) y para esto en el Modelo de Seguridad y Privacidad de la Información en la fase de planificación SE cuenta con el anexo de controles del estándar ISO 27002 (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).

Igualmente, el programa Agenda de Conectividad Estrategia de Gobierno en Línea presenta el modelo de seguridad de la información sustentado en la norma ISO 27001 (Ministerio de Tecnologías de la Información y las Comunicaciones, 2011).

Con lo anteriormente expuesto se tomará ISO 27002 como guía de buenas prácticas que describe los objetivos de control y controles recomendables en cuanto a seguridad de la información, en particular la categoría de control número 14. Adquisición, Desarrollo y Mantenimiento de Sistemas y dentro de ella los numerales técnicos identificables en cumplimiento durante el proceso del ciclo de vida del desarrollo de software asociados a los atributos de seguridad.

1.4 Modelos existentes para el desarrollo seguro en metodologías ágiles

Actualmente se tienen los siguientes modelos propuestos, en los cuales se plantea la integración de la seguridad en el desarrollo de software mediante metodologías ágiles (Ver Tabla 7):

Tabla 1-7 Modelos actuales propuestos para el desarrollo seguro en metodologías ágiles. (*Elaboración propia*).

Título	Publicación	Descripción
Integration Analysis of Security Activities from the Perspective of Agility	Agile India 2012, Article number 6170016, Pages 40-47.	Propone un enfoque que proporciona medida cuantitativa de la agilidad de las actividades de seguridad en términos de grado de agilidad real (RAD). Se determina el grado de compatibilidad de una actividad de seguridad de un proceso ágil. También presenta un análisis comparativo de las actividades de seguridad entre sí en el contexto de RAD y el factor de eficiencia de remoción de riesgo (RREF). RREF es una evaluación para saber qué tan efectiva es una actividad de seguridad para eliminar el riesgo. Esta comparación ayudará a un desarrollador durante el desarrollo de software para decidir qué actividad de seguridad es más beneficiosa que otra para la integración (Singhal, 2012).

Título	Publicación	Descripción
Security backlog in SCRUM security practices	Malaysian Conference in Software Engineering 2011, Article number 6140708, Pages 414-417.	Este trabajo propone un modelo para la integración de los principios de seguridad en el desarrollo incremental utilizando Scrum y sugiere los elementos a tener en cuenta en el "*backlog*" de seguridad, el cual puede ser usado para el análisis e implementación de características de seguridad en las fases de Scrum (Azham et al., 2011).
Integrating Software Security into Agile-SCRUM Method	KSII Transactions on Internet and Information Systems, Volume 8, Issue 2, 27 February 2014, Pages 646-663.	Este trabajo representa la continuación de la investigación anterior (*Security backlog* in Scrum *security practices*), con un enfoque en la evaluación en caso de estudio basado en la industria. Los resultados destacan una agilidad mejorada en SCRUM después de la integración de los SB. Además, el software de seguridad se puede desarrollar rápidamente, incluso en situaciones que implican cambios en los requisitos de software. Con base a los resultados experimentales, se identificó que, al integrar SB, es bastante factible desarrollar software seguro utilizando un modelo Scrum (Ghani, Azham, & Jeong, 2014).
Using Assurance Cases to Develop Iteratively Security Features Using SCRUM	9th International Conference on Availability, Reliability and Security, ARES 2014, Article number 6980323, Pages 490-497.	Propone el uso de casos de aseguramiento para mantener una visión global de las exigencias de seguridad en la medida en que este atributo se está desarrollando de forma iterativa y de un proceso que permita el desarrollo incremental de los elementos de seguridad al tiempo que garantiza los requisitos de seguridad de la función se cumplen (Othmane, Angin, & Bhargava, 2014).
An encyclopedic approach for realization of security activities with agile methodologies	5th International Conference on Confluence 2014: The Next Generation Information Technology Summit, 6 November 2014, Article number 6949242, Pages 767-772.	Presenta un enfoque en el cual las actividades de seguridad se pueden combinar con actividades ágiles mediante el cálculo del valor medio de la agilidad de ambas actividades, es decir, manteniendo aspectos tales como costo, tiempo, repetición, los beneficios que afectan a la agilidad de la actividad. Al aceptar tabla de compatibilidad valor difuso (FVCT), reporta la relación tanto de las actividades de seguridad como de la metodología ágil mediante valores difusos (Othmane et al., 2014).
Security and privacy behavior definition for behavior driven development	Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) Volume 8892, 2014, Pages 306-309	Método para la definición de los criterios de aceptación (*BehaveSafe*) mediante la creación de un gráfico de amenazas y contramedidas llamado *T & C gráfico*. Se realizar un ejercicio para estimar la eficiencia del método con un sistema basado en la web (Okubo et al., 2014).

Título	Publicación	Descripción
Architecture-Centric Testing for Security: An Agile Perspective	Book Agile Software Architecture: Aligning Agile Processes and Software Architectures, 2013, pages: 245-267	Modelo para determinar cómo la evaluación y el análisis centrado en la arquitectura pueden ayudar a asegurar sistemas desarrollados utilizando un ciclo de desarrollo ágil. Se aplica este enfoque en un caso de estudio para evaluar la seguridad de las arquitecturas de gestión de identidad. Se analiza la eficacia de este enfoque en la detección de comportamientos vulnerables y el costo-efectividad del perfeccionamiento de la arquitectura antes de vulnerabilidades se construyan en el sistema (Al-Azzani, Al-Natour, & Bahsoon, 2013).
Towards agile security risk management in RE and beyond	Workshop on Empirical Requirements Engineering (EmpiRE 2011), pages:33-36	Modelo que introduce la idea de un *peso ligero* en las metodologías ágiles de desarrollo ágil de software y de la filosofía de gestión de proyectos, para lograr la gestión de riesgos de seguridad integrados en el ciclo de vida de desarrollo (Franqueira, Bakalova, Tun, & Daneva, 2011).
Integrating Security and Software Engineering	Integrating Security and Software Engineering: Advances and Future Visions, 2007, pages: 143-159.	Explica cómo las funciones de seguridad se pueden integrar en un método ágil llamado FDD *Feature Driven Development* (Desarrollo guiado por características) (Siponen, Baskerville, & Kuivalainen, 2007).
Secure Feature Driven Development (SFDD) Model for Secure Software Development	Procedia - Social and Behavioral Sciences Volumen 129, 2014, pages: 546-553.	Versión mejorada del modelo de desarrollo FDD *Feature Driven Development* (Desarrollo guiado por características) impulsada por el desarrollo de software seguro. De hecho, el modelo mejorado se basa en nuestro en un estudio anterior y sus conclusiones, en las cuales se indicaba que el FDD existente plantea limitaciones para desarrollar software seguro. Por lo tanto, se propone una versión de FDD mejorada que apoya el desarrollo de software seguro, representado en un caso de estudio para comparar el nivel de seguridad en los estudiantes de pregrado y postgrado de nivel. El documento ilustra que la agilidad de FDD no se ve afectada de manera significativa, incluso después de la adición de nuevas fases.
Supporting evolving security models for an agile security evaluation	Information and Software Technology, Volume 57, Issue 1, 2015, Pages 217-247.	Propone un método de evaluación de seguridad ágil para la norma *Common Criteria*. Este método se complementa con la implementación de un análisis de detección de cambios en los requisitos de seguridad. Este sistema facilita el proceso de evaluación de seguridad ágil en un alto grado (Raschke et al., 2014).

Título	Publicación	Descripción
Agile Non Functional Requiremnents (NFR) Traceability Metamodel	8th Malaysian Software Engineering Conference (MySEC), 2014, pages: 228-233.	Este modelo se enfoca en el problema relacionado con la trazabilidad de los requerimientos no funcionales en proyectos ágiles, y a su vez, expone la trazabilidad y los efectos que pueden generar cambios en los requerimientos funcionales a los requerimientos no funcionales, en especial los referentes a seguridad y performance. Propone la combinación de los modelos de trazabilidad ágil y de trazabilidad de requerimientos no funcionales con el fin de rastrear requerimientos no funcionales en proyectos de desarrollo ágiles (Firdaus, Arbain, Ghani, Mohd, & Wan, 2014).
The NORMAP Methodology: Lightweight Engineering of Non-functional Requirements for Agile Process	19th Asia-Pacific Software Engineering Conference, 2012, pages: 322-325.	Modelo que presenta el estudio y desarrollo de un marco de trabajo para el de modelamiento de requisitos no funcionales que están adaptados para el proceso ágil en donde se busca combinar requisitos funcionales con requisitos no funcionales tratando estos últimos como artefactos de primera clase y vinculándolos con los primeros (Farid, 2012).
S-Scrum: a Secure Methodology for Agile Development of Web Services	The World of Computer Science and Information Technology Journal (WSCIT)	Este trabajo propone una versión mejorada de Scrum que incluye la seguridad, incorpora el análisis de la seguridad y las actividades de diseño de la misma en los procesos de Scrum. La metodología propuesta ha modificado los procesos de Scrum para dar cabida a las actividades de documentación que reflejan los aspectos de seguridad del servicio web de destino. La metodología propuesta se preocupa por la seguridad y también las necesidades cambiantes durante las etapas. La metodología propuesta se aplica parcialmente modelada y en un escenario típico para el desarrollo de un servicio de transferencia de dinero para demostrar la validez del enfoque (Mougouei, Fazlida, Sani & Almasi, 2013).
Secure Scrum: Development of Secure Software with Scrum	MuSe - Munich IT Security Research Group Munich University of Applied Sciences	Secure Scrum es una variación del marco de Scrum con especial énfasis en el desarrollo de software seguro durante todo el proceso de desarrollo de software. Pone énfasis en la aplicación de los aspectos y controles relacionados con la seguridad sin la necesidad de cambiar el proceso Scrum subyacente o influir en la dinámica del equipo. Secure Scrum permite que incluso expertos que no sean de seguridad puedan detectar problemas de seguridad, implementar características de seguridad, y/o verificar las implementaciones (Mougouei et al., 2013).

De los artículos analizados se encuentran aspectos que indican que durante el proceso de desarrollo de software ágil, gestionado en un marco de gobierno organizacional, se deba cumplir con los requisitos de seguridad empresarial. El cumplimiento de los requisitos debe ser resultado de la efectividad (i.e., la medida del impacto de la gestión), tanto en el logro de los resultados planificados como en el manejo de los recursos utilizados y disponibles en la organización. Esta efectividad debe ser la combinación de la eficiencia y la eficacia, entendiendo la eficiencia como la relación entre los recursos utilizados en un proyecto y los resultados conseguidos con el mismo, i.e., se entiende que la eficiencia se da cuando se utilizan menos recursos para lograr un mismo objetivo o cuando se logran más objetivos con los mismos o menos recursos. En complemento, la eficacia representa el grado en que se hacen las actividades planificadas y se alcanzan los objetivos planificados, orientada a minimizar los recursos invertidos (Atehortúa, 2005).

Igualmente, la creciente dependencia de tareas críticas respecto del software conlleva a que el valor del mismo no resida en la aptitud que posee de mejorar la productividad de las organizaciones, sino que posea la capacidad de continuar operando de manera confiable aún cuando deba enfrentar eventos que amenazan su utilización (Castellaro et al., 2009).

También, estos artículos al revisar la relación entre seguridad y las metodologías ágiles, plantean situaciones iniciales como:

- Métodos ágiles con pocas características explícitas para el manejo de la seguridad.
- Algunos métodos de seguridad discretos (como listas de verificación y estándares de gestión) pueden complementar en cierta forma las metodologías ágiles de desarrollo pero pueden quedar zonas de seguridad pendientes por resolver.
- Cuando una organización elige desarrollar el software bajo la metodología de desarrollo de ágil debe garantizar que se cumplan las políticas y estándares que defina el gobierno de procesos de tecnología, regulaciones, normativas propias del negocio, la gestión del riesgo y arquitectura empresarial y de seguridad de la organización; por lo tanto, debe existir un proceso que los integre, permita su medición y gestión para garantizar la seguridad de la información.

Con base en estas situaciones, las organizaciones acuden a la adaptación y conjunción de marcos para la gestión y gobierno de las tecnologías de información y mantenimiento de software, para lograr la

seguridad, resultando en una "mezcla" de recomendaciones, prácticas, estándares, lineamientos, entre otros aspectos, que en muchos casos puede resultar costosa y no efectiva si no se tiene conocimiento y experiencia en la aplicación de dichos marcos (qué hacer y cómo hacerlo) (Real, 2009).

Finalmente, luego de revisar cada una de las propuestas identificadas para resolver la necesidad de incorporar seguridad en el desarrollo ágil, se identifica que ninguna de ellas plantea la solución desde un punto de vista en donde se enfoque en generar acciones concretas para el cumplimiento de estos objetivos de control y controles recomendables en cuanto a seguridad de la información planteadas por el estándar ISO27002

Esto conlleva a la necesidad de plantear un modelo para garantizar la entrega de software seguro que integre prácticas ágiles, propendiendo que este proceso se desarrolle dentro de la arquitectura de seguridad y modelo de gobierno de las tecnologías de seguridad de la información.

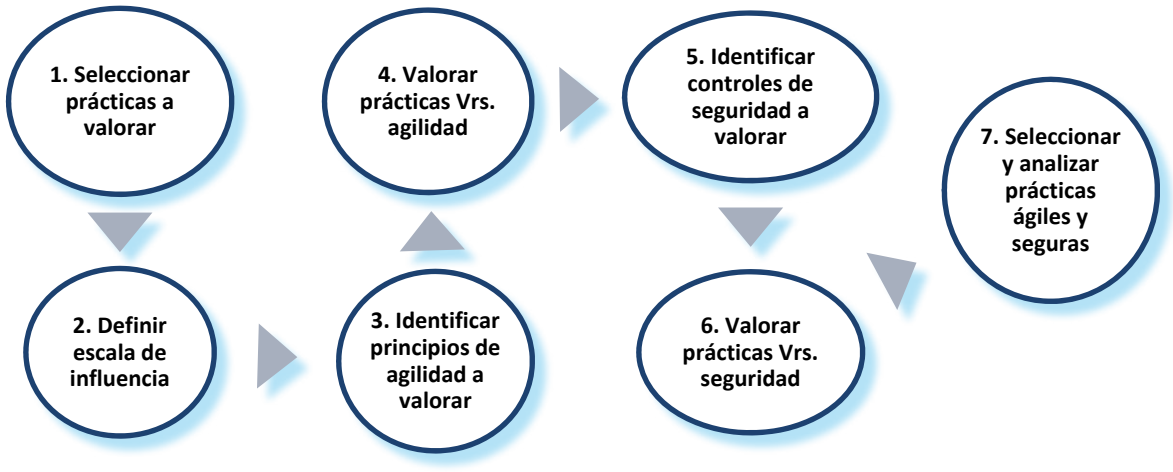
2. Análisis de cumplimiento de agilidad y seguridad de las prácticas actuales

Con el propósito de obtener una evaluación cuantitativa de la seguridad y agilidad en las prácticas para desarrollo de software, de tal manera que éstas puedan ser utilizadas dentro de proyectos con un enfoque ágil y que como resultado se pueda determinar cuáles son las prácticas que mejor se ajustan a estos conceptos, se presentará en los siguientes apartes el método utilizado y los pasos correspondientes para el análisis y selección de las mismas.

Ilustración 2-1 Macro actividades para la selección de las mejores prácticas ágiles y seguras. (*Elaboración propia*).

A continuación se exponen los pasos definidos que darán como resultado las prácticas que mejor cumplen con los criterios de seguridad y agilidad seleccionados:

Ilustración 2-2 Método y pasos para determinar prácticas ágiles y seguras (*Elaboración propia*).



A continuación el detalle de la ejecución y resultado de los pasos anteriormente descritos.

Paso 1: Seleccionar prácticas a valorar.

Los grupos de prácticas en los cuales se va a centrar este estudio, son los dirigidos a la programación ágil. Esta selección corresponde a 20 prácticas relacionadas con las áreas de diseño, desarrollo (DevOps) y pruebas. Este tipo de prácticas detallan a más bajo nivel técnicas de desarrollo de software de manera ágil, tal como se explica en la sección 1.2 Prácticas ágiles usadas en la industria para el desarrollo de software, más atrás página. Las prácticas seleccionadas se exponen en la Tabla 6 Prácticas para el desarrollo de software ágil. (Agile Alliance, 2015a).

Paso 2: Definir escala de influencia.

Esta actividad consiste en definir una escala de valoración cuantitativa propia para determinar si la práctica evaluada influye en el cumplimiento del control de seguridad y en el logro del principio ágil. Con dicha escala serán evaluadas cada una las 20 prácticas seleccionadas.

Tabla 2-1 Escala de valoración de la agilidad y seguridad en las prácticas (*Elaboración propia*).

Valoración	Peso	Descripción
Influye	1	Se dará un peso de 1 cuando la práctica analizada favorece el logro del principio ágil a la luz del análisis cuantitativo definido y para el caso de la seguridad este permite satisfacer parcial o totalmente el control establecido con base en la norma ISO 27002.
No Influye	0	Se dará un peso de 0 cuando la práctica analizada no influye en el logro del principio ágil a la luz del análisis cuantitativo definido y para el caso de la seguridad este no influye de manera directa en el cumplimiento del control establecido con base en la norma ISO 27002.

Paso 3: Identificar principios de agilidad a valorar.

El objetivo es identificar dentro del total de los 12 principios de agilidad aquellos que serán objeto del estudio, para esto se define un indicador cuantitativo propuesto para cada uno de los principios, es decir se establece un criterio propio con el que se pueda dar medición concreta para al evaluar si la práctica influye en que ese indicador pueda calcularse. Así mismo, se establece que no se tomarán aquellos principios que corresponden a competencias humanas y de motivación no medibles directamente en el desarrollo de software.

Al analizar una práctica el objetivo es identificar que dicha práctica contribuye con el cumplimiento del principio de agilidad a la luz de poder alcanzar el indicador cuantitativo definido en cada principio, la Tabla 9 presenta los principios de agilidad a evaluar. La definición del indicador cuantitativo propuesto facilitará la tarea de evaluación, debido a que se trata de poder eliminar subjetividad al poder llevar la valoración a una cifra que ayude a sumarizar en el indicador.

Tabla 2-2 Análisis cuantitativo de los principios ágiles (*Elaboración propia*).

Principios de agilidad (Beck et al., 2001)	**Indicador cuantitativo propuesto**
AG1 Satisfacción del cliente. La principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.	Número de liberaciones en tiempo determinado definido con el cliente.
AG2 Adaptación a los cambios. Los requisitos cambiantes serán bienvenidos, incluso incorporados tarde al desarrollo. Los procesos ágiles se doblegan al cambio como ventaja competitiva para el cliente.	Número de requisitos que salen a producción incorporados en la última iteración. Tiempo entre la solicitud de un requisito y su salida a producción.
AG3 Entregas de software. Entregar con frecuencia software que funcione, en periodos de un par de semanas a un par de meses, con preferencia de periodos de tiempo cortos.	Longitud de la iteración.
AG4 Trabajo en equipo. Las personas de negocios y los desarrolladores deben trabajar juntos diariamente a través del proyecto.	Número de reuniones donde asistió una persona de negocio y al menos un desarrollador. Número de días en 1 semana en los cuales hubo al menos una persona de desarrollo y área de negocio.
AG5. Motivación en el trabajo. Elaborar proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realizar su trabajo.	**No se tomará** para la valoración, corresponde a competencias humanas y de motivación no medibles en el desarrollo de software.
AG6. Diálogo. La forma más eficiente y efectiva de comunicar información al equipo de desarrollo y entre el propio equipo es mediante la conversación cara a cara	Número de reuniones presenciales donde asistió una persona de negocio y al menos un desarrollador.
AG7. Software funcional. El software que funciona es la principal métrica de progreso.	Cuántas de las liberaciones realizadas van a producción y son usadas por el cliente.
AG8. Desarrollo sostenible. Los procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.	En un periodo de tiempo definido revisar cuántas iteraciones se hicieron con respecto a las posibles de acuerdo a la política definida, longitud de la iteración definida.
AG 9. Atención continua. La atención continua a la excelencia técnica enaltece la agilidad.	Porcentaje de adherencia con relación a estándares al lenguaje utilizado.
AG 10. Simplicidad. La simplicidad como arte de maximizar la cantidad de trabajo no acabado es esencial.	Número de características liberadas usadas en un periodo de tiempo medible.

Principios de agilidad (Beck et al., 2001)	**Indicador cuantitativo propuesto**
AG11. Organización. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto-organizan	**No se tomará** para la valoración, corresponde a competencias humanas y de motivación, no medibles en el desarrollo de software.
AG12. Efectividad. En intervalos regulares, el equipo reflexiona sobre la forma de ser efectivo y ajusta su conducta en consecuencia	Número de ajustes implementados provenientes de reuniones tipo retro (lecciones aprendidas).

Paso 4: Valorar prácticas vrs. Agilidad.

Esta actividad consiste en revisar si la práctica evaluada influye en el logro del principio ágil, es decir, si dicha práctica contribuye con el cumplimiento del principio de agilidad a la luz de poder alcanzar el indicador cuantitativo propuesto. Esta revisión dará como resultado la Tabla 10, Matriz descriptiva de logro principio ágil por práctica.

A continuación, se presenta la matriz con el análisis descriptivo para cada práctica determinando si esta ayuda en el logro del indicador cuantitativo definido para cada uno de los principios ágiles seleccionados.

Tabla 2-3 Matriz descriptiva de logro principio ágil por práctica. (*Elaboración propia*).

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P1 Prb. usabilidad	Favorece la entrega con valor al cliente/usuario teniendo en cuenta una meta a alcanzar durante la prueba, permite identificar si lo que se va a entregar es usable.	Favorece la identificación temprana de requisitos cambiantes identificados por el usuario.	No influye directamente sobre la duración de la iteración.	Favorece el trabajo conjunto entre el desarrollador y el usuario o persona de negocio.	Favorece el trabajo conjunto entre el desarrollador y el usuario o persona de negocio.	Favorece la identificación de usabilidad de la versión probada por el cliente/ usuario.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	No influye debido a que esta prueba no tiene un enfoque directo de verificación técnica.	Favorece la identificación de características usadas en la versión probada por el cliente/usuario.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	0	1	1	1	0	0	1	0	6
P2 Prb. Exploratorias	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos y proporciona autonomía de enfoque al mismo tiempo.	Favorece la identificación temprana de requisitos cambiantes identificados por el equipo en la exploración.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba y el grado de autonomía que proporciona al equipo ejecutor.	No influye en el relacionamiento entre los desarrolladores y el usuario o persona de negocio.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio.	Favorece la identificación de software funcional acorde a las metas definidas.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba y el grado de autonomía que proporciona al equipo ejecutor.	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	1	0	0	1	1	0	0	0	5
P3 Prb. unidad	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos.	Favorece la identificación temprana de requisitos cambiantes identificados por el equipo.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba y el grado de autonomía que proporciona al equipo ejecutor.	No influye en el relacionamiento entre los desarrolladores y el usuario o persona de negocio.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio.	Favorece la identificación de software funcional acorde a las metas definidas.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba y el grado de autonomía que proporciona al equipo ejecutor.	Favorece debido a que esta prueba tiene enfoque funcional o técnico que permite revisar adherencia con relación a estándares al lenguaje utilizado.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	1	0	0	1	1	1	0	0	6

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P4 TDD	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos.	Favorece la identificación temprana e implementación de requisitos cambiantes identificados por el equipo debido al trabajo conjunto de las etapas de desarrollo y pruebas.	Favorece que con alta frecuencia haya una versión de software que funciona debido al trabajo conjunto de las etapas de desarrollo y pruebas.	No influye en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores y probadores.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio debido a que tienen con mayor frecuencia versiones de software que funcionan y por ende pueden mostrar o probar con el usuario.	Favorece la identificación de software funcional acorde con la frecuencia de liberación de versiones que funcionan.	Favorece el desarrollo sostenible debido al trabajo conjunto de las etapas de desarrollo y pruebas, esto hace que tengan software funcionando con mayor frecuencia para revisar con el usuario/ cliente.	Favorece debido a que tiene enfoque funcional o técnico que permite revisar adherencia con relación a estándares al lenguaje utilizado.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	1	0	1	1	1	1	0	0	7
P5 Objetos simulados	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos a verificar durante la prueba.	No influye la incorporación de cambios ya que corresponde a una técnica de simulación de funcionamiento del software acorde a los requisitos, si hay cambios requieren de más tiempo para el ajuste.	Favorece la duración de la iteración debido a la definición simulada de los escenarios de respuesta de acuerdo a lo estipulado en los requisitos, además fomenta la reutilización.	No influye en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores y probadores.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio debido a que tienen con mayor frecuencia versiones de software que funcionan de acuerdo a lo simulado y por ende pueden mostrar o probar con el usuario.	Favorece la identificación de software funcional con base en lo simulado.	Favorece la duración de la iteración debido a la definición simulada de objetivos a lograr en la prueba y el grado de autonomía que proporciona al equipo ejecutor.	Favorece debido a que esta prueba tiene un enfoque directo de verificación técnica simulada.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	0	1	0	1	1	1	1	0	0	6

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P6 Prb. aceptación	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos de aceptación acordados con el usuario.	No influye debido a la formalidad del alcance de aceptación de la prueba para la iteración en curso, es decir por su carácter binario de pasa o falla, cualquier aspecto minoritario puede hacer que no pase la aceptación y puesta en producción.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba.	Favorece el relacionamiento entre los desarrolladores y el usuario o persona de negocio al definir conjuntamente los criterios de aceptación.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio durante la iteración.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	Favorece la identificación de características usadas en la versión probada por el cliente/usuario.	Favorece sobre la reflexión de trabajo en equipo mediante la identificación y entendimiento de los criterios de aceptación.	
Valoración	1	0	1	1	0	1	0	0	1	1	6
P7 ATDD	Favorece la entrega con valor al cliente y de manera temprana ya que el experto dueño del producto, cliente o dominio es capaz de especificar nuevas funcionalidades escribiendo nuevas pruebas de aceptación o de casos de prueba, sin necesidad de consultar a los desarrolladores.	Favorece la identificación temprana e implementación de requisitos cambiantes identificados por el equipo debido al trabajo conjunto de las etapas de desarrollo, pruebas y usuario.	Favorece que con alta frecuencia haya una versión de software que funciona debido al trabajo conjunto de las etapas de desarrollo y pruebas con el trabajo colaborativo del usuario.	Favorece en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores, probadores y usuarios.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario.	Favorece la duración de la iteración o número de iteraciones meta debido a la interacción continua de desarrolladores, probadores y usuarios.	Favorece debido a que tiene enfoque funcional o técnico que permite revisar adherencia con relación a estándares al lenguaje utilizado.	No influye la simplicidad debido a la cantidad de herramientas y entrenamiento que requiere para su implementación por tanto esto puede impactar el número de características liberadas.	Favorece sobre la reflexión de trabajo en equipo mediante la identificación y entendimiento de los criterios de aceptación, mejoramiento en los métodos de trabajo y entrenamiento en las herramientas usadas durante todo el proceso.	
Valoración	1	1	1	1	1	1	1	1	0	1	9

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P8 BDD	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos de aceptación acordados y validados con el usuario.	Favorece la identificación temprana e implementación de requisitos cambiantes identificados por el equipo debido al trabajo conjunto de las etapas de desarrollo, pruebas y usuario.	Favorece que con alta frecuencia haya una versión de software que funciona debido al trabajo conjunto de las etapas de desarrollo y pruebas con el trabajo colaborativo del usuario.	Favorece en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores, probadores y usuarios.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario.	Favorece la duración de la iteración o número de iteraciones meta debido a la interacción continua de desarrolladores, probadores y usuarios.	Favorece debido a que tiene enfoque funcional o técnico que permite revisar adherencia con relación a estándares al lenguaje utilizado.	No influye la simplicidad debido a la cantidad de herramientas y entrenamiento que requiere para su implementación por tanto esto puede impactar el número de características liberadas.	Favorece sobre la reflexión de trabajo en equipo mediante la identificación y entendimiento de los criterios de aceptación, mejoramiento en los métodos de trabajo y entrenamiento en las herramientas usadas durante todo el proceso.	
Valoración	1	1	1	1	1	1	1	1	0	1	9
P9 Plantilla D-C-E	Favorece la entrega con valor al cliente y de manera temprana ya que permite definir unos objetivos concretos de aceptación acordados con el usuario.	No influye debido a la formalidad del alcance de aceptación de la prueba, es decir por su carácter binario de pasa o falla, cualquier aspecto minoritario puede hacer no pase la aceptación y puesta en producción.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la prueba.	Favorece el relacionamiento entre los desarrolladores y el usuario o persona de negocio al definir conjuntamente los criterios de aceptación.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio durante la iteración.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	Favorece la identificación de características usadas en la versión probada por el cliente/usuario.	Favorece sobre la reflexión de trabajo en equipo mediante la identificación y entendimiento de los criterios de aceptación.	
Valoración	1	0	1	1	0	1	0	0	1	1	6

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P10 Plantilla R-C-R	Favorece la entrega con valor al cliente debido a la claridad para el equipo con respecto al contenido de las historias de usuario/ requisitos.	No influye en el número de cambios incorporados y liberaciones realizadas.	Favorece la duración de la iteración debido a la definición de objetivos claros a lograr en la historia de usuario.	Favorece el relacionamiento entre los desarrolladores y el usuario o persona de negocio al definir conjuntamente las historias de usuario.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio durante la iteración.	No influye en el número de liberaciones realizadas.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	Favorece la identificación de características usadas en la versión probada por el cliente/usuario.	Favorece la reflexión de trabajo en equipo en el desarrollo de las historias de usuario.	
Valoración	1	0	1	1	0	0	0	0	1	1	5
P11 Control versión	Favorece la entrega con valor al cliente debido al control de versiones que permite ejercer sobre el software funcional.	Favorece la incorporación de cambios y el control sobre las versiones funcionales.	Favorece debido a que facilita el control de las versiones funcionales durante la iteración.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control técnico sobre el software.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio durante la iteración.	Favorece la identificación de versiones software funcional acorde a las metas definidas con el usuario.	Favorece la duración de la iteración o número de iteraciones meta debido al control que se puede ejercer sobre las versiones del software funcional.	Favorece debido a que permite controlar las versiones estables y funcionales. Además, permite establecer mecanismos técnicos para determinar los cambios registrados en el software.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	Favorece la reflexión de trabajo en equipo porque a través de los mecanismos técnicos se pueden determinar los cambios registrados en el software.	
Valoración	1	1	1	0	0	1	1	1	0	1	7
P12 Const. Auto	Favorece la entrega con valor al cliente debido a la automatización en la construcción de código, fomenta la reutilización, por ende lograr más en menos tiempo.	Favorece la incorporación de cambios debido la automatización en la construcción de código, fomenta la reutilización, por ende lograr más en menos tiempo.	Favorece la longitud de la iteración debido la automatización en la construcción de código, fomenta la reutilización, por ende lograr más en menos tiempo.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control técnico sobre el software.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio durante la iteración.	No influye para determinar el número de liberaciones realizadas y usadas por el cliente.	Favorece el desarrollo sostenible debido al trabajo conjunto para la construcción automática del software.	Favorece debido a que tiene enfoque técnico que permite revisar adherencia con relación a estándares al lenguaje utilizados.	No influye la simplicidad debido a la cantidad de herramientas y entrenamiento que requiere para su implementación por tanto esto puede impactar el número de características liberadas.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	1	0	0	0	1	1	0	0	5

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P13 Int. Continua	Favorece la entrega con valor al cliente debido a la integración continua de versiones , pruebas y software funcional.	Favorece la identificación temprana e implementación de requisitos cambiantes identificados por el equipo debido al trabajo conjunto de las etapas de desarrollo, pruebas y usuario mediante la integración continua.	Favorece que con alta frecuencia haya una versión de software que funciona debido al trabajo conjunto de las etapas de desarrollo y pruebas con el trabajo colaborativo del usuario.	Favorece en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores, probadores y usuarios.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario debido a las pruebas y entregas continuas.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario. El control de versiones y las pruebas continuas.	Favorece el desarrollo sostenible debido al trabajo conjunto en la integración continua.	Favorece debido a que tiene enfoque técnico que permite revisar adherencia con relación a estándares al lenguaje utilizados.	No influye la simplicidad debido a la cantidad de herramientas y entrenamiento que requiere para su implementación por tanto esto puede impactar el número de características liberadas.	Favorece la reflexión de trabajo en equipo porque a través de los mecanismos técnicos se pueden determinar los cambios registrados en el software y probar los mismos.	
Valoración	1	1	1	1	1	1	1	1	0	1	9
P14 Desp. Continuo	Favorece la entrega con valor al cliente debido a la integración continua de versiones, pruebas y software funcional.	Favorece la identificación temprana e implementación de requisitos cambiantes identificados por el equipo debido al trabajo conjunto de las etapas de desarrollo, pruebas y usuario mediante la integración continua.	Favorece que con alta frecuencia haya una versión de software que funciona debido al trabajo conjunto de las etapas de desarrollo y pruebas con el trabajo colaborativo del usuario.	Favorece en el relacionamiento entre los desarrolladores y el usuario o persona de negocio. La implementación de esta práctica se presenta a nivel de trabajo conjunto entre desarrolladores, probadores y usuarios.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario debido a las pruebas y entregas continuas.	Favorece la identificación de software funcional acorde a las metas definidas con el usuario. El control de versiones y las pruebas continuas.	Favorece el desarrollo sostenible debido al trabajo conjunto en la integración continua.	Favorece debido a que tiene enfoque técnico que permite revisar adherencia con relación a estándares al lenguaje utilizados.	No influye la simplicidad debido a la cantidad de herramientas y entrenamiento que requiere para su implementación por tanto esto puede impactar el número de características liberadas.	Favorece la reflexión de trabajo en equipo porque a través de los mecanismos técnicos se pueden determinar los cambios registrados en el software y probar los mismos.	
Valoración	1	1	1	1	1	1	1	1	0	1	9

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P15 Tarjetas CRC	Favorece la entrega con valor al cliente debido a la claridad para el equipo con respecto al contenido de las funcionalidades en la iteración.	No influye en el número de cambios incorporados y liberaciones realizadas.	Favorece la duración de la iteración debido a la definición de contenidos claros para el desarrollo de funcionalidades.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	No influye en el número de liberaciones realizadas.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	Favorece la reflexión de trabajo en equipo en el desarrollo de las tarjetas.	
Valoración	1	0	1	0	0	0	0	0	0	1	3
P16 Sesión rápida diseño	Favorece la entrega con valor al cliente debido a la claridad para el equipo con respecto al contenido de las funcionalidades en la iteración	No influye en el número de cambios incorporados y liberaciones realizadas	Favorece la duración de la iteración debido a la definición de contenidos claros para el desarrollo de funcionalidades.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	No influye en el número de liberaciones realizadas	No influye directamente sobre la duración de la iteración o número de iteraciones meta	No influye debido a que esta prueba no tiene necesariamente un enfoque directo de verificación técnica.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	Favorece la reflexión de trabajo en equipo en el desarrollo de las tarjetas.	
Valoración	1	0	1	0	0	0	0	0	0	1	3
P17 Reglas simplicidad	Favorece la entrega con valor al cliente debido a la exigencia de automatización en las pruebas para validar que sea simple.	No influye en el número de cambios incorporados y liberaciones realizadas.	Favorece la duración de la iteración debido a la definición de contenidos simples para el desarrollo de funcionalidades.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio. Es un control de los desarrolladores sobre lo que se construirá.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario lo cual contribuye a priorizar el desarrollo de lo que será usado en producción.	Favorece el desarrollo sostenible debido a la implementación de estrategias que permiten simplificar el alcance y maximizar el valor.	Favorece debido al uso de las pruebas automatizadas, la orientación a la simplicidad de clases, métodos y alcance, por tanto facilita la revisión de adherencia con relación a estándares del lenguaje usado.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario lo cual contribuye a priorizar el desarrollo de lo que será puesto en producción	Favorece la identificación de opciones de mejora debido al enfoque de búsqueda de la simplicidad, las cuales pueden ser luego socializadas con todo el equipo.	
Valoración	1	0	1	0	0	1	1	1	1	1	7

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P18 Reconstrucción	No influye en la entrega con valor que percibe el cliente ya que al aplicar esta práctica no se cambia la funcionalidad externa del producto.	No influye en el número de cambios incorporados y liberaciones realizadas.	No influye directamente sobre la duración de la iteración.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio. Es un control técnico de los desarrolladores sobre lo que se construirá.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio. Es un control técnico de los desarrolladores sobre lo que se construirá.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario orientándolo a la parametrización mediante intervención estructural al código, garantiza el funcionamiento del software.	Favorece el desarrollo sostenible debido a la implementación de estrategias que permiten simplificar la implementación técnica necesaria.	Favorece debido a que tiene enfoque técnico que permite revisar adherencia con relación a estándares al lenguaje utilizados.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario orientándolo a la parametrización mediante intervención estructural al código.	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	0	0	0	0	0	1	1	1	1	0	4
P19 Diseño Simple	Favorece la entrega con valor al cliente debido a su fundamento en las reglas de simplicidad	Favorece la incorporación de cambios debido al fomento en el uso de patrones de diseño y la filosofía de diseño continuo.	Favorece la duración de la iteración debido al uso de reglas simples para el desarrollo de funcionalidades.	Favorece en el relacionamiento entre los desarrolladores y el usuario o persona de negocio debido al fomento de la interacción continua en la validación del diseño.	Favorece en el relacionamiento cara a cara entre los desarrolladores y el usuario debido al fomento de la interacción continua en la validación del diseño.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario lo cual contribuye a priorizar el desarrollo de lo que será usado en producción	No influye directamente sobre la duración de la iteración o número de iteraciones meta	Favorece debido al uso de las pruebas automatizadas, arquitecturas y patrones de diseño con base en las reglas simples, por tanto facilita la revisión de adherencia con relación a estándares del lenguaje usado.	Favorece debido a la búsqueda de simplicidad en la implementación de lo solicitado por el usuario lo cual contribuye a priorizar el desarrollo de lo que será puesto en producción	No influye directamente sobre la reflexión de trabajo en equipo.	
Valoración	1	1	1	1	1	1	0	1	1	0	8

Principio/ Práctica	AG1 Satisfacción del cliente	AG2 Adaptación a los cambios	AG3 Entregas de software	AG4 Trabajo en equipo	AG6 Diálogo	AG7 Software funcional	AG8 Desarrollo sostenible	AG9 Atención continua	AG10 Simplicidad	AG12 Efectividad	Σ
P20 Lenguaje ubicuo	No influye en la entrega con valor que percibe el cliente ya que al aplicar esta práctica no se cambia la funcionalidad externa del producto.	No influye en el número de cambios incorporados y liberaciones realizadas.	Favorece la duración de la iteración debido al uso de un lenguaje común entre el equipo el cual permite mayor entendimiento entre lo técnico y la funcional a desarrollar.	No influye en el relacionamiento diario entre los desarrolladores y el usuario o persona de negocio.	No influye en el relacionamiento cara a cara entre los desarrolladores y el usuario o persona de negocio.	No influye en el número de liberaciones realizadas.	No influye directamente sobre la duración de la iteración o número de iteraciones meta.	Favorece debido a que tiene enfoque técnico que permite revisar adherencia con relación a estándares al lenguaje utilizados.	No influye con la identificación de características usar en la versión liberada al cliente/usuario.	Favorece la reflexión de trabajo en equipo debido al uso de un lenguaje común.	
Valoración	0	0	1	0	0	0	0	1	0	1	3

Paso 5: Identificar controles de seguridad a valorar.

El objetivo en este paso es identificar los controles técnicos de seguridad objeto del estudio con base en la norma ISO 27002 específicamente dentro del dominio 14. Adquisición, Desarrollo y Mantenimiento de Sistemas, esto dará como resultado la lista de controles de seguridad seleccionados para que al analizar cada una de las 20 prácticas se debe identificar si al aplicarla en una metodología ágil esta contribuye con el cumplimiento del control. Para la clasificación y selección de los controles aplicables a valorar con base en la norma, se utilizan las definiciones sobre controles técnicos, físicos y administrativos, presentadas en la ilustración 11, de acuerdo a lo presentado en la segunda columna de la Tabla 11.

Tabla 2-4. Análisis de aplicabilidad de controles 14.
Adquisición, Desarrollo y Mantenimiento de Sistemas ISO 27002 (Elaboración propia).

ISO 27002 - Categoría de control 14.ADQUISICIÓN, DESARROLLO Y MANTENIMIENTO DE LOS SISTEMAS DE INFORMACIÓN (ISO/IEC 27002, 2013)**.**	**CLASIFICACIÓN Y ANÁLISIS DE SELECCIÓN**
14.1.1 Análisis y especificación de los requisitos de seguridad.	**Controles técnicos** que permitan identificar los requisitos de seguridad de la información para que sean incluidos en la definición de nuevos sistemas de información o para mejoras a los sistemas de información existentes (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.1.2 Seguridad de las comunicaciones en servicios accesibles por redes públicas.	**Controles técnicos** para incorporar en el desarrollo de aplicaciones controles de confidencialidad.
14.1.3 Protección de las transacciones por redes telemáticas.	**Controles técnicos** para incorporar en el desarrollo de aplicaciones controles de integridad
14.2.1 Política de desarrollo seguro de software.	**Controles administrativos** para mantener una política organizacional de desarrollo seguro.
14.2.2 Procedimientos de control de cambios en los sistemas.	**Controles técnicos** para garantizar que los cambios a los sistemas dentro del ciclo de vida de desarrollo se documenten, registren y controlen mediante el uso de procedimientos formales de control de cambios (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.2.3 Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo.	**Controles técnicos** para revisar las aplicaciones críticas del negocio, y ponerlas a prueba para asegurar que no haya impacto adverso en las operaciones o seguridad de la organización cuando se cambian las plataformas de operación (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).

ISO 27002 - Categoría de control 14.ADQUISICIÓN, DESARROLLO Y MANTENIMIENTO DE LOS SISTEMAS DE INFORMACIÓN (ISO/IEC 27002, 2013).	CLASIFICACIÓN Y ANÁLISIS DE SELECCIÓN
14.2.4 Restricciones a los cambios en los paquetes de software.	**Controles técnicos** para desestimular el cambio y si hay cambio hacerlo formalmente revisando el impacto del cambio en la aplicación y el entorno del sistema.
14.2.5 Uso de principios de ingeniería en protección de sistemas.	**Controles administrativos** para establecer, documentar y mantener principios para la construcción de sistemas seguros, y aplicarlos a cualquier actividad de implementación de sistemas de información (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.2.6 Seguridad en entornos de desarrollo.	**Controles técnicos** para establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas que comprendan todo el ciclo de vida de desarrollo de sistemas (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.2.7 Externalización del desarrollo de software.	**Controles administrativos** para supervisar y hacer seguimiento de la actividad de desarrollo de sistemas contratados externamente (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.2.8 Pruebas de seguridad de sistemas.	**Controles técnicos** para establecer en el desarrollo pruebas de funcionalidad de la seguridad (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.2.9 Pruebas de aceptación.	**Controles técnicos** para establecer programas de prueba para aceptación y criterios de aceptación relacionados con los sistemas de información nuevos, actualizaciones y nuevas versiones (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).
14.3.1 Protección de los datos utilizados en pruebas.	**Controles técnicos** para garantizar la protección de los datos usados para pruebas (MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones, 2015).

Como objeto de análisis de este trabajo se seleccionan los controles clasificados como técnicos, debido a que son aquellos que requieren implementaciones durante el proceso de desarrollo de software, ya que utilizan la tecnología como una base para controlar la integridad, el acceso y uso de datos confidenciales a través de una estructura física y sobre la red e incluyen tecnologías tales como, cifrado, autenticación a nivel de la red, listas de control de acceso (ACLs), software de auditoría de integridad de archivos, entre otros. Este análisis da como resultado 10 controles técnicos los cuales serán utilizados para la valoración de las prácticas.

Este tipo de controles contribuyen con el desarrollo de software seguro, tal como se explica Fundamentos de Seguridad, en donde se plantea que la criticidad y tendencia de uso de los actuales sistemas de información en diferentes dominios de la sociedad determina que, si bien es importante asegurar que el software se desarrolla de acuerdo a las necesidades del usuario, también lo es garantizar que el mismo sea seguro (Castellaro et al., 2009).

Paso 6: Valorar prácticas Vrs. Seguridad.

Esta actividad consiste en valorar si la práctica evaluada influye en el cumplimiento del control de seguridad. Para esto se utiliza la escala de valoración definida en el paso 2 del proceso. Este paso dará como resultado la Matriz descriptiva de cumplimiento de seguridad para cada práctica.

Con base en lo anterior se toman los 10 controles para el desarrollo de la valoración de cumplimiento en cada práctica. A continuación, se presenta la matriz que contiene el análisis descriptivo para cada práctica determinando si ésta ayuda en el cumplimiento del control de seguridad seleccionado. La matriz presenta el análisis para las 10 primeras prácticas como muestra de lo realizado.

Tabla 2-5 Matriz descriptiva ejemplo análisis del cumplimiento de seguridad para 10 prácticas (*Elaboración propia*).

Dominio Control / Práctica	14.1.1 Análisis y especificación de los requisitos de seguridad.	14.1.2 Seguridad de las comunicaciones en servicios accesibles por redes públicas.	14.1.3 Protección de las transacciones por redes telemáticas	14.2.2 Procedimientos de control de cambios en los sistemas	14.2.3 Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo	14.2.4 Restricciones en los cambios a los paquetes de software	14.2.6 Seguridad en entornos de desarrollo	14.2.8 Pruebas de seguridad de sistemas	14.2.9 Pruebas de aceptación	14.3.1 Protección de los datos utilizados en pruebas
P6 Prb. aceptación	Favorece porque permite validar de manera formal que los requisitos de seguridad fueron incluidos dentro de la iteración.	Favorece porque permite al usuario validar de manera formal que los requisitos de seguridad asociados a los servicios accesibles por redes públicas fueron incluidos dentro de la iteración .	Favorece debido a que las pruebas de aceptación incluyen validaciones sobre la protección de las transacciones al ser estas requisitos básicos de funcionamiento en el software.	Favorece debido a que las pruebas de aceptación permiten definir una descripción formal de la conducta de un producto de software que será aceptada por el usuario, por tanto cualquier solicitud de cambio en requisitos o funcionalidades son un insumo para al proceso formal de control de cambios en el sistema.	Favorece debido a que la prueba de aceptación permite de manera formal identificar los cambios las plataformas de operación, para revisar las aplicaciones críticas del negocio, y ponerlas a prueba para asegurar que no haya impacto adverso en las operaciones o seguridad de la organización.	Favorece debido a que en su misión podría ayudar a comprobar la fidelidad del producto a ser puesto en producción con respecto al paquete de software original.	No influye con establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas.	Favorece debido a que la prueba de aceptación permite de manera formal revisar la funcionalidad de la seguridad en el software	Favorece permite formalmente establecer programas de prueba para aceptación y criterios de aceptación relacionados.	No influye debido a que no estipula características o alcances sobre los datos que se deben establecer para la prueba, ni proporciona herramientas para proteger dichos datos.
Valoración	1	1	1	1	1	1	0	1	1	0
P7 ATDD	Favorece porque permite validar de manera formal que los requisitos de seguridad fueron incluidos dentro de la iteración.	Favorece porque permite al usuario validar de manera formal que los requisitos de seguridad asociados a los servicios accesibles por redes públicas fueron incluidos dentro de la iteración .	Favorece debido a que las pruebas de aceptación incluyen validaciones sobre la protección de las transacciones al ser estas requisitos básicos de funcionamiento en el software.	Favorece debido a que las pruebas de aceptación automatizadas permiten al usuario revisar la funcionalidad esperada del software y con base en ella establecer solicitudes de cambio en requisitos o funcionalidades como insumo para al proceso formal de control de cambios en el sistema.	Favorece debido a que la prueba de aceptación permite de manera formal identificar los cambios las plataformas de operación, para revisar las aplicaciones críticas del negocio, y ponerlas a prueba para asegurar que no haya impacto adverso en las operaciones o seguridad de la organización.	Favorece debido a que en su misión podría ayudar a comprobar la fidelidad del producto a ser puesto en producción con respecto al paquete de software original.	No influye con establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas.	Favorece debido a que la prueba de aceptación permite de manera formal revisar la funcionalidad de la seguridad en el software	Favorece porque permite formalmente establecer programas de prueba para aceptación automatizadas y criterios de aceptación relacionados debido a que el experto dueño del producto, cliente o dominio es capaz de especificar nuevas funcionalidades escribiendo nuevas pruebas de aceptación o de casos de prueba, sin necesidad de consultar a los desarrolladores.	No influye debido a que no estipula características o alcances sobre los datos que se deben establecer para la prueba, ni proporciona herramientas para proteger dichos datos.

Dominio Control / Práctica	14.1.1 Análisis y especificación de los requisitos de seguridad.	14.1.2 Seguridad de las comunicaciones en servicios accesibles por redes públicas.	14.1.3 Protección de las transacciones por redes telemáticas	14.2.2 Procedimientos de control de cambios en los sistemas	14.2.3 Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo	14.2.4 Restricciones en los cambios a los paquetes de software	14.2.6 Seguridad en entornos de desarrollo	14.2.8 Pruebas de seguridad de sistemas	14.2.9 Pruebas de aceptación	14.3.1 Protección de los datos utilizados en pruebas
Valoración	1	1	1	1	1	1	0	1	1	0
P8 BDD	Favorece porque permite validar de manera formal que los requisitos de seguridad fueron incluidos dentro de la iteración.	Favorece porque permite al usuario validar de manera formal que los requisitos de seguridad fueron incluidos dentro de la iteración asociados a los servicios accesibles por redes públicas.	Favorece debido a que las pruebas de aceptación incluyen validaciones sobre la protección de las transacciones al ser estas requisitos básicos de funcionamiento en el software.	Favorece debido a que las pruebas de aceptación automatizadas permiten al usuario revisar la funcionalidad esperada del software y con base en ella establecer solicitudes de cambio en requisitos o funcionalidades como insumo para al proceso formal de control de cambios en el sistema.	Favorece debido a que la prueba de aceptación permite de manera formal identificar los cambios las plataformas de operación, para revisar las aplicaciones críticas del negocio, y ponerlas a prueba para asegurar que no haya impacto adverso en las operaciones o seguridad de la organización.	Favorece debido a que en su misión podría ayudar a comprobar la fidelidad del producto a ser puesto en producción con respecto al paquete de software original.	No influye con establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas.	Favorece debido a que la prueba de aceptación permite de manera formal revisar la funcionalidad de la seguridad en el software	Favorece porque permite formalmente establecer programas de prueba para aceptación automatizadas y criterios de aceptación relacionados debido a que el experto dueño del producto, cliente o dominio es capaz de especificar nuevas funcionalidades escribiendo nuevas pruebas de aceptación o de casos de prueba, sin necesidad de consultar a los desarrolladores.	No influye debido a que no estipula características o alcances sobre los datos que se deben establecer para la prueba, ni proporciona herramientas para proteger dichos datos.
Valoración	1	1	1	1	1	1	0	1	1	0

Dominio Control / Práctica	14.1.1 Análisis y especificación de los requisitos de seguridad.	14.1.2 Seguridad de las comunicaciones en servicios accesibles por redes públicas.	14.1.3 Protección de las transacciones por redes telemáticas	14.2.2 Procedimientos de control de cambios en los sistemas	14.2.3 Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo	14.2.4 Restricciones en los cambios a los paquetes de software	14.2.6 Seguridad en entornos de desarrollo	14.2.8 Pruebas de seguridad de sistemas	14.2.9 Pruebas de aceptación	14.3.1 Protección de los datos utilizados en pruebas
P9 Plantilla D-C-E	Favorece porque permite especificar contextos de aceptación para validar de manera formal que los requisitos de seguridad que fueron incluidos dentro de la iteración.	Favorece porque permite especificar contextos de aceptación para validar de manera formal que los requisitos de seguridad asociados a los servicios accesibles por redes públicas fueron incluidos dentro de la iteración .	Favorece porque permite especificar contextos de aceptación para validar la protección de las transacciones por redes públicas.	No influye con la incorporación de cambios ya que corresponde a una práctica que guía la identificación de escenarios de prueba con base en la información dada para la iteración.	Favorece porque permite especificar contextos de aceptación para validar los cambios en las plataformas de operación, para revisar las aplicaciones críticas del negocio, y ponerlas a prueba asegurando que no haya impacto adverso en las operaciones o seguridad de la organización.	Favorece porque permite especificar contextos de aceptación para comprobar la fidelidad del producto a ser puesto en producción con respecto al paquete de software original.	No influye con establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas.	Favorece porque permite especificar contextos de aceptación para revisar la funcionalidad de la seguridad en el software	Favorece porque permite formalmente establecer contextos, acciones y un conjunto particular de consecuencias observables a obtener con respecto al funcionamiento del software.	No influye debido a que no estipula características o alcances sobre los datos que se deben establecer para la prueba, ni proporciona herramientas para proteger dichos datos.
Valoración	1	1	1	0	1	1	0	1	1	0
P10 Plantilla R-C-R	Favorece porque permite identificar y especificar requisitos de seguridad a incluir dentro de la iteración	Favorece porque permite definir historias de usuario para proteger los sistemas de actividades fraudulentas, disputas contractuales y divulgación y modificación no autorizadas.	No influye con desarrollar controles de integridad para la protección de las transacciones por las redes porque las historias de usuario no determina controles sobre la información involucrada en las transacciones de los servicios de las aplicaciones.	Favorece la incorporación de cambios ya que corresponde a una práctica que guía la identificación de escenarios para ser dentro de las características del producto en la iteración.	No influye porque no establece parámetros para validar los cambios en las plataformas de operación, para revisar las aplicaciones críticas del negocio, y ponerlas a prueba asegurando que no haya impacto adverso en las operaciones o seguridad de la organización.	Favorece porque permite describir roles, características, razones asociadas a las solicitudes (historias de usuario), entre ellas las que corresponden a cambios.	No influye con establecer y proteger los ambientes de desarrollo seguros para las tareas de desarrollo e integración de sistemas.	No influye, por si sola no es suficiente para incluir las pruebas de requisitos de seguridad de la información y la adherencia a prácticas de desarrollo de sistemas seguros.	No influye, por si sola no es suficiente para incluir las pruebas de requisitos de seguridad de la información y la adherencia a prácticas de desarrollo de sistemas seguros.	No influye debido a que no estipula características o alcances sobre los datos que se deben establecer para la prueba, ni proporciona herramientas para proteger dichos datos.
Valoración	1	1	0	1	0	1	0	0	0	0

Paso 7: Seleccionar y analizar prácticas ágiles y seguras.

Este paso consiste en seleccionar las prácticas que mejor se ajustan al cumplimiento de agilidad y seguridad con base en la sumatoria de las valoraciones individuales de cada práctica con respecto al cumplimiento de seguridad y agilidad realizados en los pasos anteriores. Así mismo, analizar los resultados obtenidos con base en las áreas de desarrollo con mayor representación acorde a las prácticas más ágiles y seguras encontradas.

A continuación, se presenta en la Tabla 13 la matriz PCP, en la cual se presentan los valores numéricos acorde a la valoración individual y totalizada de las Prácticas, Controles y Principios. Esta matriz permitirá identificar las prácticas que mejor se ajustan a los criterios definidos para el cumplimiento de agilidad y seguridad. En la siguiente tabla, se presenta el resultado de sumatoria (∑) con respecto a escala de valoración de la agilidad y seguridad en las prácticas.

Tabla 2-6 Matriz PCP evaluación de prácticas, controles y principios (*Elaboración propia*).

	Controles ISO 27002											Principios de agilidad										
Prácticas ágiles	14.1.1	14.1.2	14.1.3	14.2.2	14.2.3	14.2.4	14.2.6	14.2.8	14.2.9	14.3.1	∑	AG1	AG2	AG3	AG4	AG6	AG7	AG8	AG9	AG10	AG12	∑
P1	0	0	0	0	1	0	0	0	0	0	1	1	1	0	1	1	1	0	0	1	0	6
P2	1	1	1	1	1	1	0	1	0	0	7	1	1	1	0	0	1	1	0	0	0	5
P3	1	1	1	1	1	0	0	1	0	0	6	1	1	1	0	0	1	1	1	0	0	6
P4	1	1	1	0	1	0	0	1	0	1	6	1	1	1	0	1	1	1	1	0	0	7
P5	1	1	1	0	1	0	0	1	0	1	6	1	0	1	0	1	1	1	1	0	0	6
P6	1	1	1	1	1	1	0	1	1	0	8	1	0	1	1	0	1	0	0	1	1	6
P7	1	1	1	1	1	1	0	1	1	0	8	1	1	1	1	1	1	1	1	0	1	9
P8	1	1	1	1	1	1	0	1	1	0	8	1	1	1	1	1	1	1	1	0	1	9
P9	1	1	1	0	1	1	0	1	1	0	7	1	0	1	1	0	1	0	0	1	1	6
P10	1	1	0	1	0	1	0	0	0	0	4	1	0	1	1	0	0	0	0	1	1	5
P11	0	0	0	1	1	1	0	0	0	0	3	1	1	1	0	0	1	1	1	0	1	7
P12	0	0	0	0	0	1	0	0	0	0	1	1	1	1	0	0	0	1	1	0	0	5
P13	1	0	0	1	1	1	0	1	1	1	7	1	1	1	1	1	1	1	1	0	1	9
P14	1	0	0	1	1	1	0	1	1	1	7	1	1	1	1	1	1	1	1	0	1	9
P15	1	0	0	0	0	0	0	0	1	0	2	1	0	1	0	0	0	0	0	0	1	3
P16	1	0	0	0	0	0	0	0	1	0	2	1	0	1	0	0	0	0	0	0	1	3
P17	0	0	0	0	0	0	1	0	0	0	1	1	0	1	0	0	1	1	1	1	1	7
P18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	4
P19	1	0	0	1	0	0	1	0	1	0	4	1	1	1	1	1	1	0	1	1	0	8
P20	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	1	0	1	3

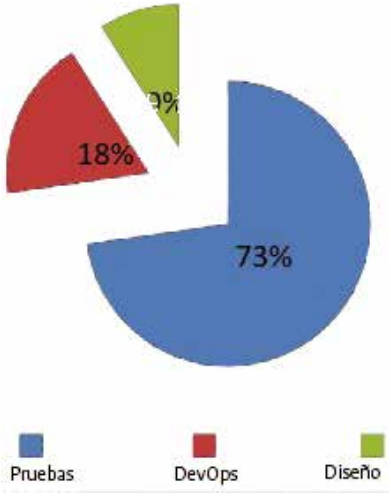
El criterio de selección de las prácticas más ágiles y seguras consiste en aquellas que tengan un porcentaje de cumplimiento total de seguridad y agilidad mayor o igual al 60%. Tal como se listan en la Tabla 14, existen 11 de las 20 prácticas que cumplen con este criterio.

Tabla 2-7 Lista de prácticas ágiles y seguras (*Elaboración propia*).

Prácticas ágiles	∑ Controles ISO 27002 - CC. 14	∑ Principios de agilidad	Agilidad + Seguridad por práctica	% de cumplimiento A+S por práctica	Área de práctica
P7 ATDD	8	9	17	85%	Pruebas
P8 BDD	8	9	17	85%	Pruebas
P13 Integración continua	7	9	16	80%	DevOps
P14 Despliegue continuo	7	9	16	80%	DevOps
P6 Pruebas de aceptación	8	6	14	70%	Pruebas
P4 TDD	6	7	13	65%	Pruebas
P9 Plantilla D-C-E	7	6	13	65%	Pruebas
P2 Pruebas exploratorias	7	5	12	60%	Pruebas
P3 Pruebas de unidad	6	6	12	60%	Pruebas
P5 Objetos simulados	6	6	12	60%	Pruebas
P19 Diseño simple	4	8	12	60%	Diseño
P11 Control de versión	3	7	10	50%	DevOps
P10 Plantilla R-C-R	4	5	9	45%	Pruebas
P17 Reglas de simplicidad	1	7	8	40%	Diseño
P1 Prueba de usabilidad	1	6	7	35%	Pruebas
P12 Construcción automática	1	5	6	30%	DevOps
P15 Tarjetas CRC	2	3	5	25%	DevOps
P16 Sesión rápida diseño	2	3	5	25%	Diseño
P18 Reconstrucción	0	4	4	20%	Diseño
P20 Lenguaje ubicuo	1	3	4	20%	Diseño

Al realizar una clasificación de los resultados teniendo en cuenta a las prácticas por área de desarrollo: diseño, devOps, pruebas, se obtiene que un 73% de las prácticas más ágiles y seguras se encuentran dentro del área de pruebas, lo que refuerza el concepto que la calidad es del equipo y de pensar en pruebas antes que escribir código.

Ilustración 2-3 Distribución de prácticas ágiles y seguras por área de desarrollo (*Elaboración propia*).



Finalmente, en la Tabla 15 se presenta el resumen los pasos anteriormente descritos.

Tabla 2-8 Resumen de pasos para selección de las prácticas más ágiles y seguras (*Elaboración propia*).

ID	Actividad	Descripción
1	**Seleccionar prácticas a valorar**	Los grupos de prácticas en los cuales se va a centrar este estudio, son los dirigidos a la programación ágil. Esta selección corresponde a 20 prácticas relacionadas con las áreas de diseño, desarrollo (DevOps) y pruebas. Este tipo de prácticas detallan a más bajo nivel técnicas de desarrollo de software de manera ágil, tal como se explica en la sección 1.2 Prácticas ágiles usadas en la industria para el desarrollo de software, más atrás página .
2	**Definir escala de influencia**	Consiste en definir una escala de valoración propia para determinar si la práctica evaluada influye en el cumplimiento del control de seguridad y en el logro del principio ágil.
3	**Identificar principios de agilidad a valorar**	El objetivo es identificar dentro del total de los 12 principios de agilidad aquellos que serán objeto del estudio, para esto se define un indicador cuantitativo propuesto para cada uno de los principios, es decir se establece un criterio propio con el que se pueda dar medición concreta para al evaluar si la práctica evaluada influye en que ese indicador puede calcularse. Así mismo, se establece que no se tomarán aquellos principios que corresponden a competencias humanas y de motivación no medibles en el desarrollo de software.

ID	Actividad	Descripción
4	**Valorar prácticas Vrs. agilidad**	Esta actividad consiste en revisar si la práctica evaluada influye en el logro del principio ágil, es decir, si dicha práctica contribuye con el cumplimiento del principio de agilidad a la luz de poder alcanzar el indicador cuantitativo propuesto.
5	**Identificar controles de seguridad a valorar**	El objetivo es identificar los controles técnicos de seguridad objeto del estudio con base en la norma ISO 27002 específicamente dentro del dominio 14. Adquisición, Desarrollo y Mantenimiento de Sistemas. Para la clasificación y selección de los controles aplicables a valorar, se utilizan las definiciones sobre controles técnicos, físicos y administrativos, presentadas en la Ilustración 11 Categorías principales de controles de seguridad (Red Hat Inc. / MIT, 2005).
6	**Valorar prácticas Vrs. seguridad**	Esta actividad consiste en valorar si la práctica evaluada influye en el cumplimiento del control de seguridad. Se utiliza la pregunta: ¿De acuerdo a la experiencia la práctica P influye en el cumplimiento del control de seguridad Y?
7	**Seleccionar y analizar prácticas ágiles y seguras**	Consiste en listar las prácticas que mejor se ajustan al cumplimiento de agilidad y seguridad con base en la sumatoria de las valoraciones individuales de cada práctica con respecto al cumplimiento de seguridad y agilidad. Criterio de selección de las prácticas finales es de un % de cumplimiento de seguridad y agilidad mayor o igual al 60%. Finalmente, con base en los hallazgos se presenta el análisis de resultados.

Este capítulo entrega como resultado principal las 11 prácticas de desarrollo más ágiles y seguras dentro de las 20 analizadas a la luz del cumplimiento de los principios de agilidad y seguridad acorde con ISO27002, las cuales servirán de insumo el análisis y formulación de escenarios en el siguiente capítulo, el cual tiene por objeto formular los posibles eventos que hacen referencia a los componentes claves de las variables estratégicas en el entorno de las metodologías ágiles y seguridad en el ciclo de vida de desarrollo de software, cuya combinación será la causante del campo de los posibles escenarios y la identificación del escenario apuesta que incluye las prácticas ágiles y seguras que conducen al estado deseado en el desarrollo de proyectos de software.

3. Análisis de escenarios para el logro de agilidad y seguridad en las prácticas.

3.1 Conceptos básicos de la prospectiva estratégica

La dirección estratégica ha recorrido un proceso dinámico dentro del cual se han construido diferentes enfoques que van desde modelos de intervención sobre la realidad, con esquemas de planificación deterministas y estructurados, hasta enfoques complejos de interacción, relacionados con la construcción y reconocimiento de estrategias vistas como un tejido de acciones congruentes, cada vez más urgentes y necesarias, que comprometen la supervivencia de la organización (Montoya & Montoya, 2003).

La prospectiva se ha desarrollado como una disciplina transversal cuyo propósito es la construcción, desde el presente, de posibles escenarios futuros, en el marco de una visión global, sistémica, dinámica y abierta, teniendo como objeto la anticipación de la ocurrencia de eventos con alto grado de probabilidad en un horizonte de tiempo determinado o como consecuencia lógica de eventos actuales y resultado de la extrapolación de tendencias (Montoya, 2015a). Esto lo hace a través de diferentes técnicas de priorización, jerarquización y proyección, con base en los datos disponibles, pero además, teniendo en cuenta las evoluciones futuras de las variables y los comportamientos de los actores implicados (Montoya, 2015b). Una de sus grandes virtudes es la de servir como instrumento para producir acuerdos en grupos heterogéneos.

De acuerdo con Michel Godet (1987), la prospectiva es “la anticipación al servicio de la acción”, con el reconocimiento de que el futuro aún no existe y que solamente depende del hombre o un colectivo de construirlo con base en la toma de decisiones adecuadas para lograrlo (Godet, 1987).

Existen dos escuelas científicas que dominan el campo de la prospectiva a nivel mundial. La primera fundada en Francia en la década de los años 60 por Bertrand de Jouvenel y Michel Godet, la cual está

fundada en el humanismo para proponer que el futuro puede ser creado y modificado por las acciones de los actores sociales, ya sea individuales u organizados, y propone estudios que caractericen la sociedad futura en sus diversos enfoques: social, económico y cultural. La segunda escuela, es la inglesa, con aportes centrales en las Universidades de Sussex y Manchester. Esta corriente considera a la tecnología como el principal motor del cambio en la sociedad, y desde el análisis del cambio tecnológico se proyecta hacia la construcción de escenarios futuros, por lo que considera que la acción de los actores sociales no es tan importante como para marcar el rumbo del futuro (Montoya, 2012).

Las inquietudes sobre los estudios prospectivos tienen origen en la crisis del petróleo en los años 1970s. Al respecto, vale la pena mencionar que la petrolera Royal Shell fue una de las primeras corporaciones multinacionales que utilizó estudios prospectivos para realizar actividades de planeación en 1968, lo que le permitió anticipar escenarios que preveían la crisis de 1973 (Van der Heijden,1998).

Como producto concreto de un ejercicio de prospectiva se encuentran las variables estratégicas a partir de los factores de cambio, el diseño de los escenarios posibles y las estrategias para alcanzar el escenario apuesta. Por tanto, la construcción de escenarios se ha convertido en un concepto importante y en una metodología determinante para la investigación de futuros que permite canalizar la necesidad colectiva. Un escenario es una descripción que supone la intervención de un conjunto de hipótesis, que para que sean creíbles y útiles deben garantizar pertinencia, coherencia, verosimilitud, importancia y transparencia. En efecto un escenario no es una realidad futura, sino un medio para que lo represente, con el objetivo de aclarar la acción presente a la luz de los futuros posibles y deseables (Godet, 1993).

El desarrollo de este ejercicio tiene por objeto formular los posibles eventos que hacen referencia a los componentes claves de las variables estratégicas en el entorno de las metodologías ágiles y seguridad en el ciclo de vida de desarrollo de software, cuya combinación será la causante del campo de los posibles escenarios y la identificación del escenario apuesta que incluye las prácticas ágiles y seguras que conducen al estado deseado en el desarrollo de proyectos de software.

Este ejercicio se desarrolló utilizando el Método de Sistema de Matrices de Impactos Cruzados (SMIC), el cual permite evaluar las probabilidades de un conjunto de eventos que se suceden unos a otros, pues integra las interacciones existentes entre los diferentes eventos (Godet, 1993).

La realización de una hipótesis en un horizonte de tiempo determinado constituye un evento y el conjunto de las hipótesis un marco referencial en el que hay tantos estados posibles, es decir, imágenes finales

como combinaciones de juegos de hipótesis. Es así que de las imágenes posibles, se establecen a partir de la información obtenida de los actores, aquellas que merecen ser estudiadas considerando la probabilidad de realización (Godet, 1993).

En este documento se presentan los resultados obtenidos en las etapas generales que enmarcan el ejercicio de análisis de escenarios. En una primera etapa, se identificaron los actores correspondientes a varios expertos en metodologías ágiles, seguridad de la información y prospectiva estratégica. En la segunda etapa, se identificaron las hipótesis o eventos de futuro. En la tercera etapa, se probabilizaron las hipótesis por los expertos del sistema a través de la encuesta SMIC. En la cuarta etapa, se probabilizaron los escenarios que fueron analizados y agrupados en tres diferentes categorías según sus características de probabilidad; el primer grupo lo conforman los escenarios alternos o de núcleo tendencial, el segundo grupo los escenarios improbables y finalmente el tercer grupo lo componen los escenarios imposibles. En una quinta etapa se realizó el análisis de sensibilidad, el cual muestra qué tanto fluctúan las hipótesis con respecto a la variación de la probabilidad de ocurrencia de cada una de las mismas, y se determinaron las más influyentes y dependientes.

Como última etapa se seleccionaron y describieron los escenarios o imágenes del futuro en los cuales las prácticas ágiles y seguras se podrían encontrar en una proyección de tiempo de 3 años dentro de los proyectos de desarrollo de software.

3.2 Sistema de matrices de impactos cruzados

El Método SMIC PROB – EXPERT corresponde a uno de los más favorables de aquellos en donde se analizan los impactos cruzados, debido a su puesta en práctica sencilla y a que permite fácilmente la interpretación de los resultado. Este método tiene como punto de partida el establecimiento de unas hipótesis fundamentales y complementarias, que son posteriormente valoradas en cuanto a sus probabilidades simples y condicionales de SI ocurrencia y de NO ocurrencia (Godet, 1993). Esto finalmente llevará a identificar aquellas imágenes más probables de ocurrir en los proyectos con metodologías ágiles y que tienen como propósito generar software seguro en un horizonte de tiempo de 3 años. A pesar de su componente matemático, esta herramienta no deja de ser subjetiva y depende de las opiniones de los expertos consultados.

3.2.1 Participación de los actores

Para el desarrollo del ejercicio de análisis de escenarios se contó con la participación de actores correspondientes a: un experto en el área de tecnología con experiencia en gestión de proyectos mediante metodologías basadas en planes y ágiles, especialista en seguridad informática y gestión empresarial, profesional ceritficado internacionalmente en fundamentos de testing y auditoria en sistemas de información, así mismo se contó con la asesoría de experto en seguridad informática certificado internacionalmente y un experto en prospectiva estratégica, esto con la intención de aprovechar la multiplicidad de los mismos y lograr que las visiones individuales convergan en una serie de escenarios o imágenes de futuro, y que finalmente se seleccione uno como apuesta para la propuesta estratégica.

3.2.2 Identificación de hipótesis

A largo del proceso prospectivo, se puede verificar la existencia de un hilo conductor que se origina en las variables consideradas como estratégicas, las cuales se convierten en eventos con los cuales se obtiene el escenario apuesta (Mojica, 2005). Con base en el análisis estructural y la participación de los actores se identificaron las hipótesis asociadas a las variables estratégicas, que se pueden observar en la Tabla 16.

Tabla 3-1 Hipótesis asociadas a las variables estratégicas (*Elaboración propia*).

Variables	Evento	Horizonte	Hipótesis de futuro
Valores	val	Qué tan probable es que en los proyectos que se desarrollen a tres años ...	Estén completamente orientados a la generación de valor del proyecto con base y cumplimiento de los 4 valores del agilismo acorde con Ilustración 1 Valores del agilismo (Beck et al., 2001)..
Principios	ppios		Cumplan con todas las 12 características que favorecen un proceso ágil de uno tradicional como se describe en Ilustración 2 Principios del Agilismo (Beck et al., 2001).
Objetivos de control	oc		Existan los controles de mejores prácticas para implementar, operar y mantener un sistema de gestión de la seguridad con base en la norma ISO 27000 (Tabla 11). Análisis de aplicabilidad de controles 14. Adquisición, Desarrollo y Mantenimiento de Sistemas ISO 27002.
Prácticas ágiles	pract		Garanticen que se desarrollan actividades y procedimientos que logran apoyar la ejecución y el mejoramiento de los procesos con base en las prácticas ágiles actuales acorde con la Tabla 6 Prácticas para el desarrollo de software ágil. Pág. 1 (Agile Alliance, 2015a).
Metodologías ágiles	meth		Garanticen que se pueda satisfacer al cliente mediante la entrega temprana y continua de software con valor mediante el uso de las prácticas ágiles existentes en la Tabla 5 Evolución de las metodologías ágiles (Izaquita, 2011. Pág. 16).

De esta manera, se obtuvieron seis (5) hipótesis de futuro asociadas a las variables estratégicas, a las cuales se les asigna un valor de probabilidad simple y compuesta, considerando la ocurrencia y la no ocurrencia de las mismas, generando un conjunto de escenarios que representan las imágenes de futuro.

3.2.3 Probabilización de las hipótesis

Los actores atribuyen a cada hipótesis una probabilidad simple y condicionada de ocurrencia. Para calificar el nivel de probabilidad de ocurrencia de cada hipótesis se utilizó una escala que va desde muy improbable hasta muy probable, pasando por los estados de improbable, tanto probable como improbable y probable de suceder, las cuales venían acompañadas de una escala numérica de 0.1, 0.3, 0.5, 0.7 y 0.9 respectivamente.

En la Tabla 17 se ilustra, para cada hipótesis, la probabilidad simple asignada por los actores. De esta manera, se puede concluir que la tendencia de la valoración asignada por los actores es optimista frente a la posible realización de las hipótesis; sin embargo, este análisis no permite observar la interrelación entre las mismas; aspecto que se desarrolla en el siguiente apartado, en el cual se encuentran las probabilidades de cada hipótesis condicionada con la ocurrencia o no de las demás. Los valores de la tabla están comprendidos entre 0 y 1.

Tabla 3-2 Probabilidad simple de las hipótesis. (*Elaboración propia*).

	Probabilidades
1 : val	0.7
2 : ppios	0.5
3 : oc	0.5
4 : pract	0.3
5 : meth	0.7

Definidas las probabilidades simples, la tarea siguiente es evaluar las probabilidades condicionadas entre los eventos. En primer lugar se presentan las probabilidades condicionadas positivas (la probabilidad de que el evento i suceda si ocurre el evento j), y posteriormente se analizaron las probabilidades condicionadas negativas (la probabilidad de que suceda un evento i si no ocurre un evento j). Esta valoración se realizó con la misma escala de las probabilidades simples (0.1, 0.3, 0.5, 0.7, 0.9) según correspondiera. Los valores de la tabla están comprendidos entre 0 y 1.

Tabla 3-3 Probabilidades condicionadas positivas (*Elaboración propia*).

	val	ppios	oc	pract	meth
1 : val	0.7	0.9	0.7	0.9	0.9
2 : ppios	0.7	0.5	0.5	0.7	0.7
3 : oc	0.5	0.5	0.5	0.9	0.5
4 : pract	0.5	0.7	0.7	0.3	0.7
5 : meth	0.9	0.9	0.7	0.9	0.7

Tabla 3-4 Probabilidades condicionadas negativas (*Elaboración propia*).

	val	ppios	oc	pract	meth
1 : val	0	0.3	0.7	0.7	0.7
2 : ppios	0.3	0	0.7	0.3	0.5
3 : oc	0.7	0.7	0	0.5	0.5
4 : pract	0.5	0.5	0.3	0	0.5
5 : meth	0.3	0.3	0.7	0.5	0

3.2.4 Escenarios estratégicos

La aplicación de esta técnica permitió evaluar las probabilidades de ocurrencia del conjunto de eventos teniendo en cuenta las interacciones existentes entre los mismos, bajo lo cual se identificaron las imágenes de futuro, que para este caso con 5 hipótesis, corresponde a 32 escenarios.

2^n= número de imágenes o escenarios de futuro n = número de hipótesis.

2^5 = 32 imágenes o escenarios de futuro.

La metodología permite agrupar las imágenes o escenarios de futuro en tres grupos: Escenarios alternos o núcleo tendencial, menos probables y los imposibles de acuerdo a las probabilidades correspondientes a cada uno de los mismos, los cuales se describen a continuación ver Ilustración 17.

En los escenarios alternos se encuentra que los valores están frecuentemente presentes en las diferentes imágenes de futuro. Así mismo, el cumplimiento de los principios y la metodología, lo cual podría llevar a concluir que al hacer cumplir dichas variables a cabalidad se garantiza el cumplimiento de la prácticas y la seguridad mediante el uso inherente de controles.

Ilustración 3-1 Probabilización de los escenarios (*Elaboración propia*).

Escenarios alternos

Son aquellos que presentan la mayor probabilidad de ocurrencia de todos los escenarios posibles por lo que seguramente se encontrará la respuesta en alguna de las imágenes de futuro que ellos describen.

En la Ilustración 16 se presentan 8 escenarios en orden descendente de acuerdo a la probabilidad simple de ocurrencia, superior o igual al 50%. Es así, que los escenarios que el escenarios clasificados

Escenarios menos probables

Estos escenarios corresponden a aquellos que presentan la menor probabilidad de ocurrencia de todos los escenarios posibles, por lo que difcilmente se encontrará en alguna de estas imágenes de futuro la propuesta.

En la Ilustración 16 se presentan 17 escenarios en orden descendente de acuerdo a la probabilidad simple de ocurrencia menor al 50%.

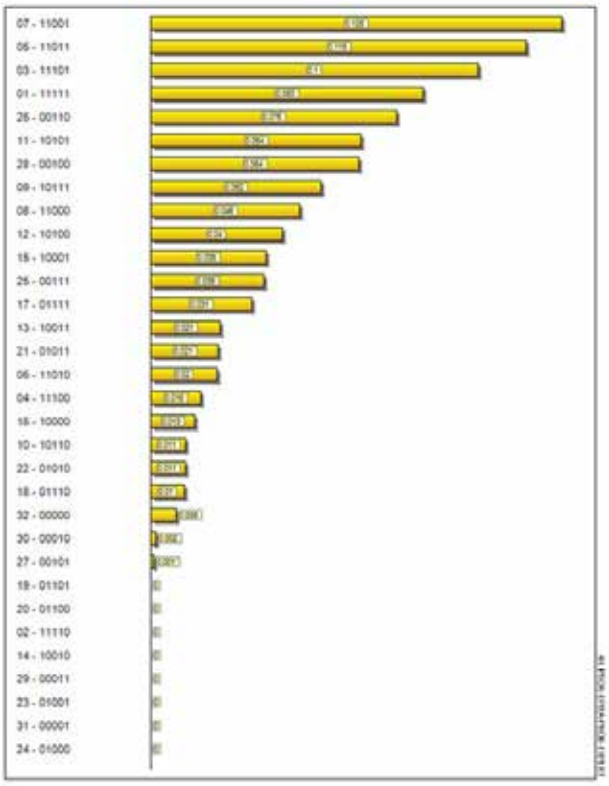
Improbables

Aquellos que no presentan probabilidad de ocurrencia, son los ultimos 7.

Ilustración 3-2

Histograma de probabilidad de escenarios

(*Elaboración propia*).



En la Tabla 20 se presenta el nombre asignado a los primeros 5 escenarios con mayor probabilidad de ocurrencia.

Tabla 3-5 Nombramiento de escenarios con mayor probabilidad de ocurrencia (*Elaboración propia*).

Val	Ppios	OC	Pract	Meth	Nombre escenario
1	1	0	0	1	**Agilismo básico**
1	1	0	1	1	**En camino al agilismo seguro**
1	1	1	0	1	**Agilismo para el cumplimiento.**
1	1	1	1	1	**Ágil 100%**
0	0	1	1	0	**Agilismo emergente**

A continuación, en la Tabla 21 se presenta una descripción y análisis estratégico para los principales escenarios anteriormente nombrados:

Tabla 3-6 Escenarios estratégicos (*Elaboración propia*).

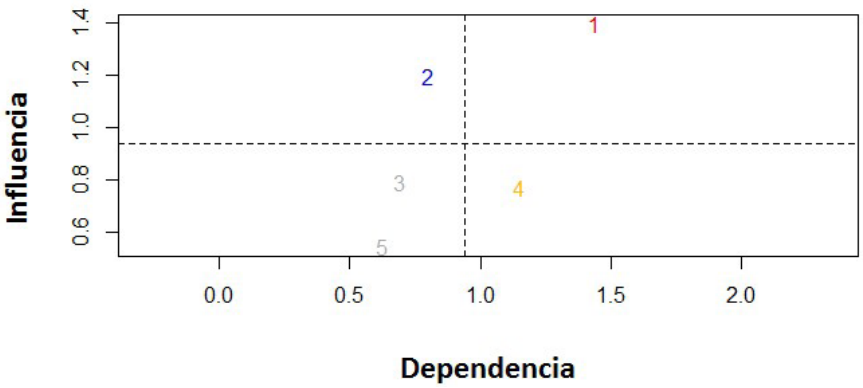
Escenario	Descripción	Estrategia recomendada para lograr agilismo con prácticas seguras
Agilismo básico	Este escenario consiste en aquellas organizaciones que tienen implementadas las variables estratégicas: Valores, Principios y Metodologías ágiles, sin embargo, no cuentan con un concepto interiorizado de controles de seguridad y prácticas ágiles que soporten el ciclo de vida de desarrollo del software.	Estas organizaciones se debe enfocar en implementar acciones que permitan: ■ Definir, declarar y desplegar las políticas para la alineación estratégica de todos los equipos internos y externos con el agilismo, la seguridad y las prácticas ágiles para el desarrollo de software en los proyectos dentro de la organización. ■ Generar espacios para fomentar el trabajo colaborativo. ■ Revisar las competencias del personal para formar profesionales altamente calificados que trabajen en la gestión y aplicación de las tecnologías de la información con base en la filosofía del agilismo. ■ Intervenir el entorno físico para generar un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc (Letelier & Penadés, 2006). ■ Realizar una identificación de las prácticas que existen en la organización para apoyar el ciclo de vida de desarrollo de software y determinar un plan de trabajo para implementar las prácticas más ágiles y seguras planteadas en este trabajo. ■ Identificar cuáles son los controles de seguridad necesarios y deseables a implementar con base en las características del sector o regulaciones que deba cumplir la organización. ■ Implementar estrategia para la gestión del conocimiento. ■ Fortalecer relacionamiento con proveedores y aliados estratégicos.

Escenario	Descripción	Estrategia recomendada para lograr agilismo con prácticas seguras
En camino al agilismo seguro	Este escenario consiste en aquellas organizaciones que tienen implementadas las variables estratégicas: Valores, Principios, Prácticas y Metodologías ágiles, sin embargo no cuentan con un concepto interiorizado sobre la seguridad y los objetivos de control acorde con la norma. ISO 27002. Aún no cuentan con un alcance sobre los controles que permite la implementación de software seguro alineado con las mejores prácticas de seguridad.	Estas organizaciones se debe enfocar en implementar acciones que permitan: ■ Identificar cuáles son los controles de seguridad necesarios y deseables a implementar con base en las características del sector o regulaciones que deba cumplir la organización. ■ Alinear el ciclo de vida de desarrollo de software con el Sistema de gestión de seguridad de la información con el cual cuente la compañía. ■ Determinar las mejores prácticas de seguridad con las cuales se identifique la organización. ■ Definir, declarar y desplegar las políticas para la alineación estratégica de todos los equipos internos y externos con el agilismo, la seguridad y las prácticas ágiles para el desarrollo de software en los proyectos dentro de la organización. ■ Implementar estrategia para la gestión del conocimiento. ■ Fortalecer relacionamiento con proveedores y aliados estratégicos.
Agilismo para el cumplimiento.	Este escenario consiste en aquellas organizaciones que tienen implementadas las variables estratégicas: Valores, Principios, Objetivos de control y Metodologías ágiles, sin embargo no cuentan con un diseño e implementación de prácticas ágiles formales. En este escenario posiblemente lo más importante es el cumplimiento del objetivo o regulación, más no el establecimiento de prácticas formales y repetitivas para el logro del mismo.	Estas organizaciones se debe enfocar en implementar acciones que permitan: ■ Realizar una identificación de las prácticas que existen en la organización para apoyar el ciclo de vida de desarrollo de software y determinar un plan de trabajo para implementar las prácticas más ágiles y seguras planteadas en este trabajo. ■ Definir, declarar y desplegar las políticas para la alineación estratégica de todos los equipos internos y externos con el agilismo, la seguridad y las prácticas ágiles para el desarrollo de software en los proyectos dentro de la organización. ■ Generar espacios para fomentar el trabajo colaborativo alrededor del uso formal de las prácticas. ■ Revisar las competencias del personal para formar profesionales altamente calificados que trabajen en la gestión y aplicación de las prácticas. ■ Implementar estrategias para la gestión del conocimiento.
Ágil 100%!	Este escenario consiste en aquellas organizaciones que tienen implementadas todas las variables estratégicas: Valores, Principios, Objetivos de control, Prácticas y Metodologías ágiles. Escenario en el cual la organización ha crecido y madurado en el establecimiento de la filosofía ágil.	Se debe enfocar en implementar acciones de mejora continua en cada una de las variables estratégicas las cuales incluyen aspectos como: ■ Estrategias para la motivación, valoración y retención del personal. ■ Gestión del conocimiento. ■ Fortalecer relacionamiento y alineación ágil con proveedores y aliados estratégicos. ■ Mejorar la eficiencia y productividad. ■ Realizar negociaciones de tecnología a gran escala. ■ Redefinir esquemas de facturación de servicios y proveedores. ■ Actualizar las tecnologías y capacitar al personal. ■ Emprender acciones que permitan continuar desarrollando el trabajo colaborativo. ■ Definir métricas e indicadores para cuantificar el valor generado.

Escenario	Descripción	Estrategia recomendada para lograr agilismo con prácticas seguras
Agilismo emergente	Este escenario consiste en aquellas organizaciones que tienen implementadas las variables estratégicas: Objetivos de control y prácticas ágiles. Sin embargo, no cuenta con la interiorización de la filosofía ágil en donde se encuentran las variables asociadas a los Valores, Principios, Metodologías ágiles.	Estas organizaciones se debe enfocar en implementar acciones que permitan: ■ Definir, declarar y desplegar las políticas para la alineación estratégica de todos los equipos internos y externos con el agilismo, la seguridad y las prácticas ágiles para el desarrollo de software en los proyectos dentro de la organización. ■ Generar espacios para fomentar el trabajo colaborativo. ■ Determinar las prácticas ágiles que mejor se ajustan a sus necesidades y visión estratégica. ■ Revisar las competencias del personal para formar profesionales altamente calificados que trabajen en la gestión y aplicación de las tecnologías de la información con base en la filosofía del agilismo. ■ Intervenir el entorno físico para generar un ambiente que permita la comunicación y colaboración entre todos los miembros del equipo durante todo el tiempo, cualquier resistencia del cliente o del equipo de desarrollo hacia las prácticas y principios puede llevar al proceso al fracaso (el clima de trabajo, la colaboración y la relación contractual son claves), el uso de tecnologías que no tengan un ciclo rápido de realimentación o que no soporten fácilmente el cambio, etc. ■ Realizar una identificación de las prácticas que existen en la organización para apoyar el ciclo de vida de desarrollo de software y determinar un plan de trabajo para implementar las prácticas más ágiles y seguras planteadas en este trabajo. ■ Identificar cuáles son los controles de seguridad necesarios y deseables a implementar con base en las características del sector o regulaciones que deba cumplir la organización. ■ Definir las estrategias para la motivación, valoración y retención del personal. ■ Implementar estrategia para la gestión del conocimiento. ■ Fortalecer relacionamiento y alineación ágil con proveedores y aliados estratégicos. ■ Mejorar la eficiencia y productividad. ■ Realizar negociaciones de tecnología a gran escala. ■ Redefinir esquemas de facturación de servicios y proveedores. ■ Actualizar las tecnologías y capacitar al personal. ■ Emprender acciones que permitan continuar desarrollando el trabajo colaborativo. ■ Definir métricas e indicadores para cuantificar el valor generado.

Igualmente al realizar el análisis de sensibilidad de las influencias y las dependencias entre las hipótesis se encuentra lo siguiente, éste permite identificar las hipótesis a favorecer o impedir para que el sistema evolucione en un sentido deseado:

Ilustración 3-3 Influencia y dependencia de variables (*Elaboración propia*).



1. Val: Variable altamente influyente y dependiente.
2. Ppios: Variable muy influyente con nivel medio de dependencia.
3. OC: Variable poco influyente y poco dependiente.
4. Pract: variable dependiente y poco influyente.
5. Meth: Variable sin influencia y poco dependiente.

De acuerdo a la Ilustración 18 se observa que las variables Val (1) y Ppios (2) son muy influyentes y dependientes, se interpreta que su cumplimiento contribuyen notablemente en la consecución de los objetivos. De aquí se puede concluir que el lograr implementar los valores y principios del agilismo son por esencia los elementos primordiales para el entorno, independientemente de la metodología Meth (5) usada, variable que por su parte es la menos influyente y dependiente. Así mismo, las prácticas y los controles son poco influyentes pero si tienen alta dependencia con respecto a las demás variables, es decir se ejerce mayor control sobre las mismas.

Para culminar la priorización de las acciones se deben enfocar en seleccionar aquellas asociadas al logro de los valores y principios ágiles al considerarse su alto nivel importancia para el cumplimiento de las metas. Es importante considerar que las labores inmediatas resultan favorables en el corto plazo, y mientras que las otras variables, pueden ser ejecutadas en el mediano o largo plazo considerando que primero se deben generar las condiciones necesarias para llevar a cabo el cumplimiento de valores y principios del agilismo.

4. Conclusiones

- Al revisar estado actual de uso y tendencia de las metodologías para el desarrollo de software se encuentra que los problemas detectados en los modelos tradicionales o de tipo cascada se fundamentan, por un lado, en el entorno altamente cambiante propio de la industria y, por el otro, en el proceso mismo de desarrollo donde el resultado depende de la actividad cognitiva de las personas más que de las prácticas y controles empleados, por tanto indica que las metodologías en cascada resultan "pesadas" y prohibitivas para responder satisfactoriamente a los cambios de negocio en los proyectos de software, es así como en promedio de los proyectos revisados en los últimos 5 años, el 71% de ellos no entregaron valor satisfactorio al usuario final. Por consiguiente las metodologías ágiles buscan proporcionar efectividad en proyectos con requisitos que cambian frecuentemente durante el desarrollo del proyecto y reducir los tiempos del desarrollo con respecto a las metodologías tradicionales, sin dejar de cumplir con la calidad requerida por la organización en su producto y/o servicio, lo que ha generado una gran acogida en el entorno global.

- Finalmente, luego de revisar cada una de las propuestas identificadas para resolver la necesidad de incorporar seguridad en el desarrollo ágil, se identifica que ninguna de ellas plantea la solución a dicha necesidad desde un punto de vista en donde se enfoque en generar acciones concretas para el cumplimiento de estos objetivos de control recomendables en cuanto a seguridad de la información planteadas por la norma ISO 27002. Esto representa una oportunidad para plantear una propuesta que permita garantizar la entrega de software seguro mediante la integración de prácticas ágiles, propendiendo que este proceso se desarrolle con base en las mejores prácticas para la implementación de un sistema de gestión de seguridad de la información.

- En conclusión, no existe una estrategia universal para hacer frente con éxito a cualquier proyecto de desarrollo de software en cuanto a la incorporación de prácticas ágiles y seguras. No obstante, una

premisa de la prospectiva moderna es que no hay un solo futuro predeterminado, sino muchos futuros posibles. El conocimiento de dichas posibilidades es lo que mejor nos ayudará a prestar atención a lo que el futuro nos puede traer y a encontrar la senda hacia un futuro más favorable, es por ello que mediante un ejercicio prospectivo se logra plantear la propuesta estratégica de prácticas seguras para el desarrollo de software con metodologías ágiles con base en escenarios posibles en los cuales se analizan las variables estratégicas asociadas al entorno como lo son: Valores, Principios, Objetivos de control, Prácticas y Metodologías ágiles. Este análisis permite concluir que así una organización se encuentre en el mejor escenario estratégico de implementación del agilismo, debe seguir trabajando en alcanzar la mejora continua para lograr entregar valor a usuarios finales y clientes. Asimismo, la identificación y caracterización de los escenarios más probables en su ocurrencia permiten dar norte a una organización sobre acciones concretas a emprender en cuanto a la implementación de las prácticas más ágiles y seguras, con base en su estado actual de agilismo y seguridad y el estado objetivo al cual quiera llegar.

Como trabajo futuro se recomienda desarrollar herramientas de consultoría para apoyar a las organizaciones en determinar el nivel de madurez en la implementación de las variables estratégicas y las prácticas más ágiles y seguras planteadas en este estudio: **Valores, Principios, Objetivos de control, Prácticas y Metodologías ágiles**. Así mismo estas herramientas deberían permitir cuantificar la generación de valor que proporciona la implementación de agilismo con la que cuenta la organización.

5. Bibliografía

Agile Alliance. (2015a). Glosario ágil. Retrieved November 19, 2015, from https://www.agilealliance.org/agile101/guide-to-agile/agile-glossary/

Agile Alliance. (2015b). Guía de prácticas ágiles. Retrieved November 19, 2015, from http://guide.agilealliance.org/

Alaimo, M. (2013). Introducción a la Agilidad y Scrum, 32.

Al-Azzani, S., Al-Natour, A., & Bahsoon, R. (2013). Architecture-Centric Testing for Security: An Agile Perspective. In *Agile Software Architecture: Aligning Agile Processes and Software Architectures* (pp. 245–267). Elsevier Inc.

Azham, Z., Ghani, I., & Ithnin, N. (2011). Security backlog in Scrum security practices. *2011 Malaysian Conference in Software Engineering*, (September), 414–417. http://doi.org/10.1109/MySEC.2011.6140708

Beck, K., Beedle, M., Bennekum, A. van, Cockburn, A., Cunningham, W., Fowler, M., … Thomas, D. (2001). Manifiesto por el desarrollo ágil de software. Retrieved April 6, 2016, from http://agilemanifesto.org/iso/es/

Cañadillas, D. (2010). Implantación de arquitectura de desarrollo ágil del software. Retrieved from http://orff.uc3m.es/bitstream/handle/10016/10757/PFC-ARQUITECTURA_AGIL_DESARROLLO.pdf

Carrasco, A. (2013). Conceptos de seguridad informática y su reflejo en la Cámara de Cuentas de Andalucía. Sevilla: Cámara de Cuentas de Andalucía. Retrieved from http://www.auditoriapublica.com/hemeroteca/Pag 111-117 N 61.pdf

Castellaro, M., Romaniz, S., Ramos, J., & Pessolani, P. (2009). Hacia la Ingeniería de Software Seguro. *Facultad Regional Santa Fe - Universidad Tecnológica Nacional*, *610*, 10.

Farid, W. M. (2012). The Normap methodology: Lightweight engineering of non-functional requirements for agile processes. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC*, *1*, 322–325. http://doi.org/10.1109/APSEC.2012.23

Firdaus, A., Arbain, B., Ghani, I., Mohd, W., & Wan, N. (2014). Agile Non Functional Requiremnents (NFR) Traceability Metamodel. *2014 8th Malaysian Software Engineering Conference (MySEC) Agile*, 228–233.

Franco, D. C., & Guerrero, C. D. (2013). Sistema de Administración de Controles de Seguridad Informática basado en ISO / IEC 27002, 1–10.

Franqueira, V. N. L., Bakalova, Z., Tun, T. T., & Daneva, M. (2011). Towards agile security risk management in RE and beyond. In Workshop on Empirical Requirements Engineering (EmpiRE 2011) (pp. 33–36). IEEE. http://doi.org/10.1109/EmpiRE.2011.6046253

Ghani, I., Azham, Z., & Jeong, S. R. (2014). Integrating software security into agile-Scrum method. KSII Transactions on Internet and Information Systems, 8(2), 646–663. http://doi.org/10.3837/tiis.2014.02.0019

Godet, M. (1993).De la anticipación a la acción, manual de prospectiva estratégica. Alfaomega, Barcelona, España: (1 ed.).

Hastie, S., & Wojewoda, S. (2015). Standish Group 2015 Chaos Report. Retrieved April 6, 2016, from http://www.infoq.com/articles/standish-chaos-2015

ISO/IEC 25000. (2015). Calidad del producto de software. Retrieved June 20, 2007, from http://iso25000.com/index.php/normas-iso-25000/iso-25010

ISO/IEC 27002. (2013). Information technology — Security techniques — Code of practice for information security controls COPYRIGHT PROTECTED DOCUMENT. Iec, 27002(27002). Retrieved from www.iso.org

Izaquita, M. E. (2011). Agilizando Lo Ágil: Un Framework Para La Desarrollo De Software Bajo El Modelo Cmmi En Compañías Que Usan Metodologías Ágiles De Desarrollo De Software Usando El Modelo Acelerado De Implementación (Aim) Maria.

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). Retrieved from http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

Martinez, F. I. (2012). Método para la educación de problemas y objetivos en el coaching ágil. Universidad Nacional de Colombia.

Menzinsky, A., & Palacio, J. (2015). Ficha sinóptica Scrum. Retrieved April 10, 2015, from https://www.google.com.co /url?sa=i&rct=j&q=&esrc =s&source=imgres&cd= &cad=rja&uact=8&ved=0ahUKEwjA7dOj4ITMAhVCWh4KHWnvAb8QjRwIBw&url=http://scrum.menzinsky.com/2015/02/como-se-realizan-las-hojas-de-ruta-del.html&psig=AFQjCNH5DelV06TddULBawuOZO6

Mesquida, A. L., & Mas, A. (2015). Implementing information security best practices on software lifecycle processes: The ISO/IEC 15504 Security Extension. Computers & Security, 48, 19–34. http://doi.org/10.1016/j.cose.2014.09.003

Ministerio de Tecnologías de la Información y las Comunicaciones. (2011). Modelo de seguridad de la información para la estrategia de gobierno en línea 2.0.

MINTIC / Ministerio de Tecnologías de la Información y las Comunicaciones. Guía : Controles de Seguridad y Privacidad de la Información Guía Técnica (2015). Colombia: Ministerio de Tecnologías de la Información y las Comunicaciones. Retrieved from http://www.mintic.gov.co/gestionti/615/articles-5482_Controles.pdf

Mojica, F.J. (2005). La construcción del futuro: Concepto y modelo de la prospectiva estratégica, territorial y tecnológica. (1ed.). Bogotá, Colombia: Universidad del Externado.

Montoya, I. A. (2012). Tópicos avanzados en gestión tecnológica 2 Prospectiva estratégica. Medellin.

Montoya, I. A. (2015a). Modulo 3: Planeación y dirección estratégica. Retrieved June 20, 2005, from http://slideplayer.es/slide/3455878/

Montoya, I. A. (2015b). Módulo 3: Planeación y dirección estratégica - Balance ScoreCard (BSC), Cuadro de Mando Integral. Retrieved from http://168.176.60.11/cursos/eLearning/dnp/3/html/contenido-1.2.4-prospectiva.html

Montoya, I. A., & Montoya, L. A. (2003). El direccionamiento estratégico y su aplicación en los sistemas complejos y en la gerencia ambiental, (July 2016).

Moreno, S., Gonzalez, C., & Echartea, C. (2008). Evaluación de la Calidad en Uso de Sitios Web Asistida por Software : SW AQUA Software Aided Quality in Use Assessment. Avances En Sistemas E Informatica, 5(1), 147–154.

Mougouei, D., Fazlida, N., Sani, M., & Almasi, M. M. (2013). S-Scrum: A secure methodology for agile development of web services. The World of Computer Science and Information Technology Journal (WSCIT), 3(1), 15–19.

Navarro, A., Fernández, J. D., & Morales, J. (2013). Revisión de metodologías ágiles para el desarrollo de software A review of agile methodologies for software development. Prospect, 11(2), 30–39.

Okubo, T., Kakizaki, Y., Kobashi, T., Washizaki, H., Ogata, S., Kaiya, H., & Yoshioka, N. (2014). Security and privacy behavior definition for behavior driven development. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8892, 306–309. Retrieved from http://www.scopus.com/inward/record.url?eid=2-s2.0-84911896868&partnerID=tZOtx3y1

Othmane, L. Ben, Angin, P., & Bhargava, B. (2014). Using Assurance Cases to Develop Iteratively Security Features Using Scrum. In 2014 Ninth International Conference on Availability, Reliability and Security (pp. 490–497). IEEE. http://doi.org/10.1109/ARES.2014.73

Pesántez Verdezoto, J. R. (2015). ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL Facultad de Ingeniería en Electricidad y Computación Previo a la obtención del Título de : Presentada por : JOFFRE RAMIRO PESÁNTEZ VERDEZOTO GUAYAQUIL – ECUADOR AÑO : 2015.

Raschke, W., Zilli, M., Baumgartner, P., Loinig, J., Steger, C., & Kreiner, C. (2014). Supporting evolving security models for an agile security evaluation. In 2014 IEEE 1st International Workshop on Evolving Security and Privacy Requirements Engineering (ESPRE) (pp. 31–36). IEEE. http://doi.org/10.1109/ESPRE.2014.6890525

Red Hat Inc. / MIT. (2005). Generalidades sobre la Seguridad Red Hat Enterprise Linux 4: Manual de seguridad. Retrieved November 10, 2015, from http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-sg-es-4/s1-sgs-ov-controls.html

ScrumAlliance. (2015). The 2015 State of Scrum Report. Retrieved from https://www.scrumalliance.org/why-scrum/state-of-scrum-report/2015-state-of-scrum

SFC/Superintendencia Financiera de Colombia. (2015). Misión. Retrieved November 10, 2015, from https://www.superfinanciera.gov.co/jsp/loader.jsf?lServicio=Publicaciones &lTipo= publicaciones &lFuncion=loadContenidoPublicacion&id=20483

Singhal, A. (2012). Integration Analysis of Security Activities from the Perspective of Agility. In 2012 Agile India (pp. 40–47). IEEE. http://doi.org/10.1109/AgileIndia.2012.9

Siponen, M., Baskerville, R., & Kuivalainen, T. (2007). Integrating Security and Software Engineering. (H. Mouratidis & P. Giorgini, Eds.)Integrating Security and Software Engineering: Advances and Future Visions. IGI Global. http://doi.org/10.4018/978-1-59904-147-6

Tassey, G. (2002). The economic impacts of inadequate infrastructure for software testing. Quality, 309. http://doi.org/10.1080/10438590500197315