“Desarrollo Ágil de aplicaciones, sobre tecnologías de código abierto y los Ciber-Riesgos asociados”

¿Es seguro el desarrollo de nuevas tecnologías utilizando código abierto en metodologías ágiles?...

Comenzaré este nuevo artículo con la premisa que la mayoría de las organizaciones de hoy buscan la eficiencia máxima en el desarrollo de productos y servicios, con ciclos de vida cortos para cada producto y servicio, muy enfocadas en la adaptación a la necesidad propia del cliente objetivo, propio de la 4ta revolución industrial. Utilizando como base productiva la explotación en las tecnologías de la información, que les permiten aspirar a lograr el objetivo anteriormente descrito con el menor coste posible, aspirando a mantener un flujo financiero saludable. En el “***Estudio de la Agilidad en América Latina”***, realizado por IDC (International Data Corporation) a solicitud de la consultora tecnológica Everis en agosto de 2020, asegura que más de la mitad de las empresas líderes de la región latinoamericana, han logrado reducir costes gracias a la aplicación de métodos ágiles, dado que reducen los tiempos de trabajo y permiten el continuo lanzamiento de nuevos productos al mercado. Así mismo un informe de KPMG publicado en agosto 2020, sobre ***“Gestión estratégica del talento ante la nueva realidad”***, menciona que alrededor del 60% de las organizaciones indican que sus futuras prioridades son el reajuste de los espacios físicos y desarrollo de equipos y desarrollos de equipos ágiles debido al aumento del personal en modalidad de teletrabajo por COVID 19.

Las organizaciones, apuestan por transformar su desarrollo tecnológico y de aplicaciones, pasando de un desarrollo de software tradicional, en base a proyectos tipo cascada o secuencial, hacia uno lo más ágil y/o adaptable a las necesidades del cliente, posiblemente impulsados por el propio time to market de los productos, la presión del propio mercado en que les toca jugar, o simplemente por la estrategia de digitalización organizacional en que se encuentran, que apunta a la eficiencia digital. Según el estudio de Deloitte, publicado en el 2019 **“Peldaños hacia una empresa ágil”** existen 4 grandes etapas para el despliegue total de la transformación hacia una empresa ágil.

Cuatro etapas en el camino hacia una empresa ágil

	Número de trabajadores con mentalidad ágil	Entidades que son las más afectadas	Aceleradores para llegar más rápido a la siguiente etapa
1. TI tradicional	Pocos, seleccionados a mano	Departamento de TI	Establecer un modelo estructurado de operación. Tercerizar operaciones de TI a proveedores que usen metodologías ágiles.
2. TI bimodal	Algunos	Innovadores y laboratorios dentro del departamento de TI Desarrolladores de interfaces con el cliente y TI frontal.	Requiere que proveedores tercerizados usen metodologías ágiles. Implementar coaching de innovación o conformar un equipo de innovación para difundir el conocimiento acerca de maneras ágiles de trabajar.
3. TI ágil	La mayoría	Unidades restantes de desarrollo de producto dentro de TI	Implementar una transformación ágil escalada usando un enfoque de “cambio mínimo viable”
4. Empresa ágil	Todos	Funciones rezagadas	Proporcionar a otras funciones (no-TI) aprendizajes y orientación para apoyar maneras ágiles de trabajar.

Fuente: Análisis de Deloitte..

El desarrollo de una Metodología Agile o Ágil está orientada a la gestión de proyectos o iniciativas y puede utilizarse también en múltiples otros aspectos de la vida diaria, utiliza ciclos de desarrollo cortos, iterativos e incrementales, llamados Sprint para centrarse en la mejora continua del producto o servicio, más que centrarse en la gestión del proyecto mismo. Existen distintos marcos o metodologías ágiles utilizados, como Scrum o Kanban, por mencionar algunos. Independiente del marco o método ágil utilizado para gestión de proyectos de desarrollo de software, es necesario identificar e integrar a la agilidad todo el proceso productivo del mismo, como la gestión de cambios y estructura de testing, entre otros, para asegurar la agilidad e independencia. Así mismo, se debe considerar siempre una adecuada gestión de riesgos en cada ciclo. Como lo indica Javier Pardo en https://www.paradigmadigital.com/techbiz/arquitectura-software-catalizador-agilismo/ **"Independientemente de los matices en la definición de agilidad, lo que está claro es que, para ser ágiles, todos los engranajes y resortes de nuestro proceso o sistema deben serlo**".

Hoy existe un consenso bastante aceptado en que la gestión adecuada de los riesgos en todas las etapas o ciclos iterativos de los desarrollos ágiles resulta clave, ya que con los ciclos iterativos cortos minimizan cualquier impacto imprevisto en el desarrollo del producto, por ejemplo, la flexibilidad propia de los proyectos ágiles, reduce los riesgos relacionados con el negocio, el feedback periódico reduce los riesgos relativos a expectativas de los stakeholders, la modularidad o la arquitectura modularizada en el desarrollo ágil es otro factor muy clave para la detección y remediación oportuna de las posibles desviaciones o incidentes, dado que desarrollan pruebas totales en cada ciclo o entregable, **asegurando usabilidad o funcionalidad**, además de la mantenibilidad, adaptabilidad y escalabilidad. Por otra parte, al ser la propiedad del proyecto de todo el equipo de desarrollo y tener responsabilidad colectiva, reduce el riesgo general del fracaso, entre otros tantos buenos ejemplos. Sin embargo, según la organización y comunidad https://agileriskmanagement.org/about-us/, que documenta y difunde buenas prácticas de gestión de riesgos en proyectos ágiles, es posible que no aborden por completo los riesgos externos a los que está expuesto un proyecto. Estos pueden incluir riesgos ambientales, políticos y legales (por ejemplo, si los elementos del producto están sujetos a requisitos reglamentarios, políticas de seguridad o ciberamenzas en los componentes externos del desarrollo de software). Por lo que, es importante que el rol o personas encargadas de la gestión y liderazgo de los equipos de trabajo, utilicen espacios para analizar los posibles riesgos externos, que se pueden presentar y tener planes de acción para que se pueda adaptar al contexto.

Independiente de la etapa en que se encuentres las organizaciones, siempre se encontrarán con el inconveniente de hacer convivir ambos procesos en determinados momentos (desarrollo tradicional y el ágil), además de los costes o limitación de presupuestos que significa la transformación. Dicho lo anterior, es que muchas de estas organizaciones optan por el uso de herramientas de código abierto o del tipo open source, en que las barreras de entrada del punto de vista económico son mucho más bajas, además de ser muy flexibles y adaptables para el desarrollo ágil y no ágil, dotando al código abierto de una mayor flexibilidad y la creación de conocimiento interno importante.

Código abierto en los desarrollos ágiles

El termino Open Source, viene de los programas de código abierto desarrollados en comunidades de colaboración de forma descentralizada. Que el software sea Open Source, no significa que el software ejecutable se distribuya de forma gratuita, pero sí su código fuente por medio de las comunidades. Por lo general, se habla de "software libre" para destacar la libertad en los derechos de los usuarios finales, pero a veces puede confundirse con el significado de "gratuito".

Por nombrar algunas características, que se combinan muy bien con los desarrollos ágiles iterativos de software o aplicaciones comerciales, cualquiera puede leer el código, es totalmente transparente, adaptable y flexible, además de ser relativamente sencillo de auditar e incluso colaborar en el desarrollo de funciones. De hecho, el código abierto ha demostrado ser lo suficientemente estable y seguro, así como de ser capaz de detectar y solucionar vulnerabilidades en el menor tiempo posible.

La colaboración entre las empresas y las comunidades del código abierto ha derivado en la oportunidad de conocer los problemas de primera mano, que pudieran afectar a los usuarios, proporcionarles las metodologías y soluciones efectivas a vulnerabilidades comunes en foros o comunidades, lo que permite la colaboración en función a la seguridad propia del código y las aplicaciones liberando de forma continua parches de seguridad o actualizaciones.

No todo puede ser tan bueno en el Open Source

Es un hecho que la mayor parte de los programas desarrollados en la línea del Open Source, por comodidad, reutilizan librerías o componentes gratuitos disponibles en las comunidades de desarrollo abierto, dado que, si se tuvieran que desarrollar desde cero, llevaría mucho tiempo y trabajo, afectando a los tiempos y plazos donde la amenaza del Time to Market se hace presente. Es en este punto, se comienza a producir un punto de inflexión en la seguridad dado que la reutilización aumenta la probabilidad de materialización de riesgos de ciberseguridad, y las librerías o componentes muchas veces dejan de ser mantenidas por las comunidade. Esta situación es equivalente a la finalización del soporte técnico de un producto licenciado, por lo que se puede presentar un potencial backdoor producto de alguna vulnerabilidad no parcheada en dichos componentes reutilizados. Así lo demuestra un estudio llevado a cabo por la plataforma **Veracode,** que indica la existencia de una gran cantidad de programas de código abierto que reutilizan librerías también libres, y muchas de estas librerías, aunque funcionan, tienen graves problemas de seguridad. Adicionalmente, cada librería usa, a su vez, una gran cantidad de dependencias también de código abierto. **Veracode,** en su publicación **"State of Software Security (SOSS) V11"**, de un total de 132.465 aplicaciones escaneadas, desde abril 2019 a marzo 2020, encontró que 7 de cada 10 aplicaciones tienen un fallo de seguridad en una biblioteca de código abierto, perteneciente a un tercero en el escaneo inicial, existiendo más vulnerabilidades en bibliotecas de terceros, que el código base nativo. También, publica en el mismo informe, que el 47% de esas bibliotecas defectuosas en las aplicaciones son transitivas; en otras palabras, no son introducidas directamente por los desarrolladores, sino que son arrastradas por las bibliotecas ascendentes.

Así mismo, en el estudio se demuestra que no todas las bibliotecas tienen vulnerabilidades y exposiciones comunes (CVE); esto significa que los desarrolladores no pueden confiar solo en las CVE para conocer los defectos de la biblioteca. Por ejemplo, más del 61% de las bibliotecas defectuosas en JavaScript, contienen vulnerabilidades sin los CVE correspondientes.

Por otra parte, algunos ecosistemas de lenguajes tienden a atraer muchas más dependencias transitivas que otros. En más del 80% de las aplicaciones JavaScript, Ruby y PHP, la mayoría de las bibliotecas son dependencias transitivas. Entre los fallos principales de OWASP detectados en el mismo informe, las debilidades en torno al control de acceso son las más comunes y representan más del 25% de todos los fallos. Cross-Site Scripting, es la categoría de vulnerabilidad más común que se encuentra en las bibliotecas de código abierto, presente en el 30% de las bibliotecas, seguida de la deserialización insegura (23,5%) y el control de acceso roto (20,3%).

Otro ejemplo más reciente del riesgo del uso de las librerías de código abierto es la librería "**Libgcrypt**", con un conjunto de herramientas criptográficas de código abierto que se ofrece como parte del paquete de software GnuPG, para cifrar y firmar datos y comunicaciones. El 2 de febrero de 2021 el portal hispasec.com, informó que es vulnerable y debe ser parcheada lo antes posible por un desbordamiento de buffer que permite sobre escribir memoria más allá del mismo, además de **permitir redirigir el flujo de ejecución, sin tener que preocuparse por protecciones como el ASLR** (Address Space Layout Randomization), siendo posible, saltar a código cercano dentro de la librería afectada. Si no sabemos que nuestros desarrolladores utilizaron esta librería o en qué parte del código, tampoco sabremos del riesgo que corremos.

```
155   {
156     copylen = inlen;
157     if (copylen > blocksize - hd->count)
158       copylen = blocksize - hd->count;
159
160     if (copylen == 0)
161       break;
162
163     buf_cpy (&hd->buf[hd->count], inbuf, copylen);
164     hd->count += copylen;
165     inbuf += copylen;
166     inlen -= copylen;
167   }
```

El Software libre seguro no existe, lo construimos cada vez que lo programamos.

El software seguro, no es escribir aplicaciones a la perfección la primera vez, sino corregir las fallas de manera integral, oportuna y de forma ágil. Sabemos, que es más fácil encontrar y solucionar problemas en aplicaciones jóvenes o con menos rodaje de codificación, que utilizan lenguajes y marcos modernos, pero incluso con aplicaciones con más "rodaje" productivo o con tecnología más antigua, si los equipos de desarrollo utilizan prácticas de codificación seguras e iterativas, como desarrollar búsquedas frecuentes de fallas, la integración y automatización de los controles de políticas seguridad, agregando una mirada holística del estado de la aplicación, tienen más probabilidades de desarrollar un software seguro.

Existen métodos seguros ya conocidos para la inspección y análisis de código fuente, que permiten alcanzar el objetivo:

- Inspección estática de código fuente (SAST – Caja Blanca), que permite detectar desviaciones de seguridad en propio código fuente, en etapas tempranas del desarrollo de software, como reutilización de códigos inseguros, uso de protocolos no seguros, entre otros.
- Inspección dinámica de código fuente (DAST – Caja Negra), relacionada con el análisis de vulnerabilidades y permite a los desarrolladores detectar problemas durante la ejecución del código.
- Inspección interactiva de código fuente (IAST – Función de las anteriores o caja Gris), permite detectan vulnerabilidades, estas tienen un enfoque similar a un usuario del sistema.
- Análisis de Composición de software (SCA), una tecnología utilizada para identificar o descubrir componentes de fuente abierta y de terceros en uso en una aplicación y sus vulnerabilidades de seguridad conocidas, las herramientas de SCA examinan el software, para determinar los orígenes de todos los componentes y bibliotecas dentro del software.

Por otra parte, también existen otras herramientas de detección y protección de amenazas sobre las aplicaciones ya productivas, que podrían compensar de alguna forma el riesgo de la falta de revisiones preventivas en el desarrollo iterativo ágil, colaborando con la disminución de la probabilidad de la materialización de ciberataques o también, se podría decir que disminuyen la superficie de ataque sobre las aplicaciones, dentro de las que puedo comentar como WAF (Web Aplication Firewall), que previene de tráficos maliciosos hacia o sobre las aplicaciones y RASP (Runtime Application Self Protection), que añade sensores internos, que analiza la respuesta de la aplicación a cada petición de servicio en tiempo real y es compatible con cualquier entornos por ejemplo en la nube.

Cada uno de estos métodos o herramientas, tiene sus ventajas y desventajas. Por lo que, una combinación y correlación correcta de ellas es fundamental y se debe hacer dependiendo del valor asignado por el negocio, al propio del activo de software (aplicación) a proteger. De esta forma, se deben cubrir todas las etapas desde el proceso iterativo de desarrollo desde las etapas tempranas, hasta los entornos productivos de forma flexible, adaptable, integrado a la gestión del riesgo en todos los ciclos, permitiendo a los desarrollos ágiles, tener más información para evaluar los riesgos externos en cada ciclo iterativo de evaluación.

Conclusiones

Algunas de las posibles causas de los problemas del código abierto que afectan por un lado la seguridad y por otra afecta a la credibilidad de los proyectos ágiles

Normalmente, el código abierto lo mantiene una comunidad que se dedica a lanzar actualizaciones y parches, así como mantener la seguridad del proyecto. Sin embargo, en ocasiones ese proyecto se abandona. Podemos seguir utilizando el programa de código abierto, pero no recibimos actualizaciones. Esto, hace que puedan surgir vulnerabilidades que sean aprovechadas por los piratas informáticos, para poner en riesgo nuestra seguridad y privacidad.

También, hay que tener en cuenta que las vulnerabilidades que haya en este tipo de software, suelen hacerse públicas rápidamente. Si no tomamos las medidas adecuadas, ni un programa que utilizamos, no ha sido parcheado correctamente, nuestros datos pueden estar en peligro y puede invitar a que los piratas informáticos se aprovechen.

A veces, podemos hacer uso de programas de código abierto, que, a su vez utilizan diferentes complementos que también son software libre. El problema llega cuando alguno de esos complementos, deja de existir o no recibe actualizaciones

La conclusión en un comienzo no es tan buena como pensaba antes de escribir el artículo, ya que estamos hablando de un volumen importante de amenazas, que no siempre son observadas por los desarrolladores de software en el entorno ágil, dado que se puede confundir la agilidad y rapidez con la falta de revisiones periódicas y holísticas que permitan identificar los riesgos externos del propio desarrollo. Por otra parte, es muy importante, considerar siempre el descubrimiento y las actualizaciones de inventario de los componentes de arquitectura de software, que soportan el software del tipo Open Source, manteniendo las actualizaciones, que solucionan una gran parte de los problemas identificados.

Por tanto, puedo concluir una respuesta afirmativa a la pregunta ***¿Es seguro el desarrollo de nuevas tecnologías utilizando código abierto en metodologías ágiles*****?**, siempre y cuando los equipos de desarrollo y arquitectura utilicen prácticas de inspección y codificación seguras e iterativas, como desarrollar búsquedas frecuentes de fallas, la integración y automatización de aplicación de controles de políticas seguridad, agregando una mirada holística de gestión de riesgos externos. Dado que intrínsecamente en su definición el código abierto no es inseguro, depende mucho del uso que le demos, o para que lo utilicemos, además del contexto de la arquitectura sistémica en que se realice.

En las manos de los programadores y los equipos ágiles está el llevar a cabo medidas para prevenir o reducir los riesgos que pongan en peligro la seguridad o privacidad al usar software libre.

- Como primeras medidas, sería integrar la seguridad en el SDLC (System Development Life Cycle), desde las primeras fases hasta el final de ciclo y debe realizarse de forma continua e iterativa.
- Por supuesto, un punto importante, es mantener actualizado el software. Ya hemos mencionado, que a veces surgen vulnerabilidades que son corregidas mediante parches y actualizaciones. Es cierto, que no siempre están presentes, pero sí en la mayoría.
- Tener siempre las últimas versiones del software que utilicemos. Así, evitaremos esos problemas que puedan ser aprovechados por los piratas informáticos, para llevar a cabo sus ataques.
  Se debe estar al tanto de las vulnerabilidades externas. A veces, puede surgir quede que un complemento que usamos tenga algún fallo de seguridad y sea necesario desinstalarlo. Es importante, que tengamos esto en cuenta para la cadena de suministro de librerías y/o componentes utilizados como insumos para los desarrollos o mantenimientos.

El objetivo de la seguridad del desarrollo de software o aplicaciones, no es escribir aplicaciones a la perfección la primera vez, sino hacerlo lo mejor posible de forma eficiente y utilizando estándares de los ciclos de desarrollo seguro, detectando y corrigiendo las fallas de manera oportuna e integral.

"Aplicar foco en la detección y remediación de las fallas de seguridad en etapas tempranas del desarrollo iterativo libre, es claramente la base de un software seguro, además de ser más económico y eficiente"

Por Julio Francisco Naranjo Figueroa
Maestría en Ciberseguridad
Ingeniero Civil Informático.

Referencias:

Metodologías ágiles ganan popularidad en el sector financiero

https://blog.cobiscorp.com/agilidad-financiera-agile-0

Historia de Open Source

https://www.redhat.com/es/topics/open-source/what-is-open-source-software

"Peldaños hacia una empresa Ágil"

https://www2.deloitte.com/content/dam/Deloitte/pe/Documents/strategy/DI_DR25-Stepping-stones%20Pelda%C3%B1os%20hacia%20empresa%20%C3%A1gil.pdf

Gestión de Riesgos en Equipo Ágiles

https://agileriskmanagement.org/about-us/

https://wolfproject.es/scrum-riesgos-gestion/

http://giovannycifuentes.com/gestion-de-riesgos-en-equipos-agiles/

Software Security (SOSS) (Informe basado en escaneo de 130.000 Aplicaciones. Estado de la seguridad del software V11) Brinda una perspectiva única sobre las vulnerabilidades que afectan al código en la actualidad y las mejores prácticas que los desarrolladores y los profesionales de seguridad están empleando para prevenirlas y corregirlas

https://www.veracode.com/state-of-software-security-report

http://giovannycifuentes.com/gestion-de-riesgos-en-equipos-agiles/

https://wolfproject.es/scrum-riesgos-gestion/

Estudios de la Agilidad agosto 2020

https://www.everisnewhumanera.com/metodologias-agiles-latam?utm_source=everisPR&utm_medium=Release&utm_campaign=EstudioAgile

https://assets.kpmg/content/dam/kpmg/cr/pdf/KPMG_Gesti%C3%B3n_estrat%C3%A9gica_del_talento_humano_nueva_realidad.pdf

Vulnerabilidad de librería libgcrypt

https://unaaldia.hispasec.com/2021/02/vulnerabilidad-critica-en-la-libreria-libgcrypt-de-gpg.html

La arquitectura de software catalizador clave del agilismo

https://www.paradigmadigital.com/techbiz/arquitectura-software-catalizador-agilismo/