El Costo del software de baja calidad en EE.UU.: Un informe de 2022

Del Problema a las Soluciones

HERB KRASNER
MIEMBRO, CONSEJO ASESOR
CONSORCIO PARA LA INFORMACIÓN Y CALIDAD DE SOFTWARE (CISQ)
HKRASNER@UTEXAS.EDU
FECHA: 22 DE DICIEMBRE, 2022

Indice

Lista de Figuras y Tablas

Aviso de derechos de autor

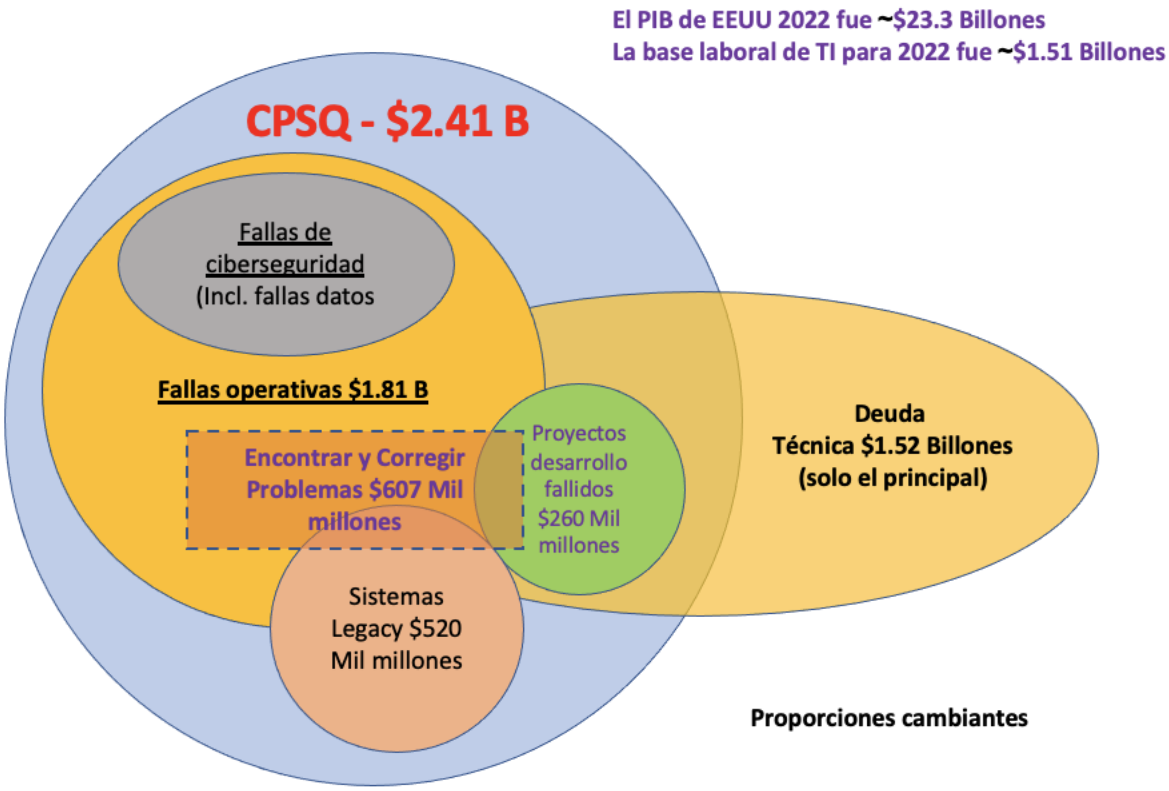
1. RESUMEN EJECUTIVO PARA 2022

Las condiciones económicas clave de los Estados Unidos que enmarcan el contexto de este informe bienal son:

- Un PIB proyectado para el 2022 de $ 23.35 billones, con un aumento de aproximadamente el 2% desde 2020.
- Una tasa de inflación del 15% durante el período de 2 años.
- Un pequeño crecimiento del 4% en la base laboral de TI durante esos 2 años a $ 1.51 billones, y
- El número de puestos de trabajo de TI sin cubrir se sitúa en ~ 300,000 a finales de agosto.

En este informe de actualización del 2022, estimamos que el costo de la baja calidad del software en los Estados Unidos ha crecido en al menos $ 2.41 billones, pero no en proporciones similares a las experimentadas en 2020. La deuda técnica acumulada de software (DT) ha crecido a ~ $ 1.52 billones[1].

Figura–1-0 costo del software de baja calidad en EE.UU en 2022



Las 3 principales áreas problemáticas en las que nos centraremos este año son:

1. Las pérdidas por delitos cibernéticos debido a las vulnerabilidades de software existentes aumentaron de forma importante
   - Las pérdidas aumentaron a un 64% de 2020 a 2021. Esas pérdidas aún no se han determinado para el 2022.
   - Varios ataques a infraestructuras críticas ocasionaron una cantidad inconmensurable de dolor y sufrimiento durante los últimos 2 años (por ejemplo, Colonial Pipeline).
2. Los problemas de la cadena de suministro de software con componentes subyacentes de terceros (especialmente *Open Source* software) han aumentado significativamente.

   - En el 2021, el 77% de las organizaciones reportaron un aumento en el uso de *Open Source* Software.
   - Una aplicación de tamaño mediano (menos de 1 millón de líneas de código) integra de 200 a 300 componentes de terceros en promedio.
   - El número de fallas causadas por debilidades en las partes de *Open Source* de una cadena de suministro de software aumentó en un 650% entre 2020 y 2021.

3. El creciente impacto de la Deuda Técnica (DT) se ha convertido en el mayor obstáculo para realizar cambios en las bases de código existentes.

   - El principal de DT aumentó a ~ $ 1.52 billones (porque las deficiencias no se están solucionando).
   - A pesar de una tasa proyectada de crecimiento del 15% en puestos de informática / TI que se crearán durante la próxima década, el número de puestos de trabajo de TI sin cubrir en EE.UU. se situó en alrededor de 300.000 a finales de agosto.
   - A finales de 2019 se predijo que para 2025, el 40 % de los presupuestos de TI se gastarán en mantenimiento de DT, siendo una de las principales razones por las que muchos proyectos de modernización fracasan.
   - El número de horas semanales que un desarrollador promedio en una empresa dedica a abordar "DT" es de 13.5 de 41.1, o sea 33% de su tiempo.

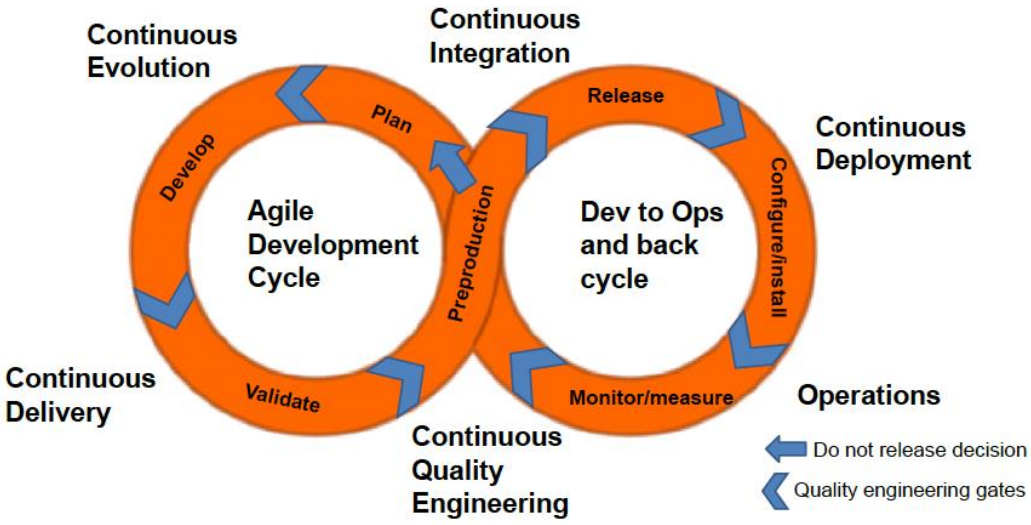
En este informe de 2022 dirigimos nuestra atención a los desarrollos recientes y las soluciones emergentes para ayudar a mejorar la mala situación de calidad del software tal como existe ahora, y estabilizar / reducir la tasa de crecimiento del costo de software de baja calidad en el futuro cercano.

Las tres áreas principales de soluciones en las que nos centraremos involucran las tendencias emergentes en herramientas modernas para ayudar a encontrar y corregir deficiencias de software, y estándares y herramientas que pueden ayudar a identificar oportunidades para reducir la creciente DT. Vemos éstas como las formas más probables de comenzar a controlar el problema de la baja calidad del software. Nos centraremos en la aparición de:

- Estándares de calidad/taxonomías de problemas de software.
- Herramientas para comprender, encontrar y corregir deficiencias/DT.
- Herramientas de IA/aprendizaje automático (ML) para ingeniería de software

Todas estas áreas de soluciones emergentes se pueden centrar en el soporte del modelo DevQualOps que introdujimos en el 2020. Dado que la seguridad del software es una subcategoría de la calidad del software, DevSecOps se ve como un submodelo de DevQualOps.

Figura 1-1 Modelo DevQualOps



Aunque el costo del software de baja calidad y la DT han aumentado significativamente durante la serie de nuestros tres informes (el problema), también lo han hecho los desarrollos en la tecnología / prácticas para remediar esos problemas (soluciones).

ES POSIBLE QUE LA TENDENCIA EN EL COSTO DEL SOFTWARE DE BAJA CALIDAD GENERAL SE NIVELE EN LA PRÓXIMA DÉCADA SI LAS ORGANIZACIONES ADOPTAN LAS RECOMENDACIONES QUE HEMOS PRESENTADO EN ESTA SERIE DE INFORMES. Esperamos que las soluciones sugeridas en este documento se adopten más ampliamente en la corriente principal de la concepción, desarrollo, producción y evolución del software.

Además de las recomendaciones generales de nuestros informes anteriores, agregamos las siguientes recomendaciones más específicas para el desarrollo de software y las organizaciones de TI:

- Utilizar los estándares de calidad del software, las mediciones relacionadas y las herramientas que están surgiendo.
- Analice y evalúe la calidad de todos los componentes de terceros / *Open Source* que se incluirán en cualquier sistema. Monitoree de cerca el funcionamiento. Aplique los parches de manera oportuna.
- Evite los modelos de DevOps e integración continua/implementación continua que no incluyan las mejores prácticas continuas de ingeniería de calidad y herramientas.
- Integre la corrección continua de DT en su SDLC.
- Invierta en el profesionalismo y el conocimiento de sus ingenieros de software.
- Considere la posibilidad de certificar a sus desarrolladores para el conocimiento del código crítico y las debilidades de arquitectura en ISO / IEC 5055 cuando OMG logre que la prueba de certificación "Desarrollador confiable" esté disponible a fines de 2023 o 2024.

Nuestro próximo informe está planeado para 2024, cuando esperamos que algunas de las soluciones identificadas en este informe se pongan al día con los problemas y se muestren en un cambio positivo en la tendencia de costo del software de baja calidad.

Nota 1 – Véase en el apéndice B la metodología detallada de estimación de costes utilizada

Nota 1 del traductor: *Billions* se traduce a miles de millones, *Trillions* se traduce a billones

2. RESUMEN DE LOS INFORMES ANTERIORES DEL COSTO DEL SOFTWARE DE BAJA CALIDAD

Este es el tercer informe de nuestra serie bianual sobre el costo del software de baja calidad. En nuestros dos primeros informes sobre este tema nos enfocamos en identificar las áreas problemáticas en la calidad del software que podrían verse a través de la lente de los costos adicionales debido a la baja calidad. Al hacerlo, esos informes ayudaron a difundir la noticia de que se trataba de un problema importante que merecía soluciones que podrían aplicarse. Sin embargo, en este informe comenzamos a esbozar soluciones más específicas a los problemas subyacentes identificados en nuestros dos primeros informes.

Informe 2018

En nuestro informe de 2018 nos centramos en definir la calidad del software y las categorías de baja calidad del software que nos permitirían identificar mejor las áreas problemáticas y los síntomas del problema de la baja calidad.

El objetivo fue crear una estimación de primer orden de COSTO DEL SOFTWARE DE BAJA CALIDAD en los EE.UU. mediante el examen de la información conocida en varias categorías informadas que se identificaron a partir de una amplia búsqueda de referencias. Estas categorías principales identificadas fueron: 1) problemas del sistema l*egacy*, 2) pérdidas por fallas de software, 3) proyectos problemáticos / cancelados, 4) encontrar y corregir defectos, y 5) DT de software. En ese informe se establecieron conceptos y definiciones básicos a los que nos referimos en las actualizaciones. Esas definiciones y conceptos eran:

- Abreviaturas comunes utilizadas: IT, US, LOC, CoSQ, COSTO DEL SOFTWARE DE BAJA CALIDAD (pág. 3)
- Qué es el software (pág. 6)
- Cuánto se gastaba en TI / software en ese momento (pág. 7)
- El modelo iceberg de los costos ocultos de calidad del software (pág. 10)
- ¿Cuáles son los costos de mantenimiento del software l*egacy* (pág. 12)?
- Resumen de las principales historias de fallas de software en las noticias (pág. 16)
- Qué es la DT de software (pág. 19)
- El impacto del talento disponible en la calidad del software y sus costos (pág. 21)
- Qué es la calidad del software (pág. 28)
- La definición del costo del modelo de calidad del software (pág. 30)
- Nuestras conclusiones sobre el costo total de la baja calidad de software en las categorías analizadas (pág. 36)

Estas definiciones y conceptos siguen siendo válidos, excepto que en los cuatro años transcurridos la definición de calidad del software se ha vuelto más estandarizada y, por lo tanto, más medible.

Informe 2020

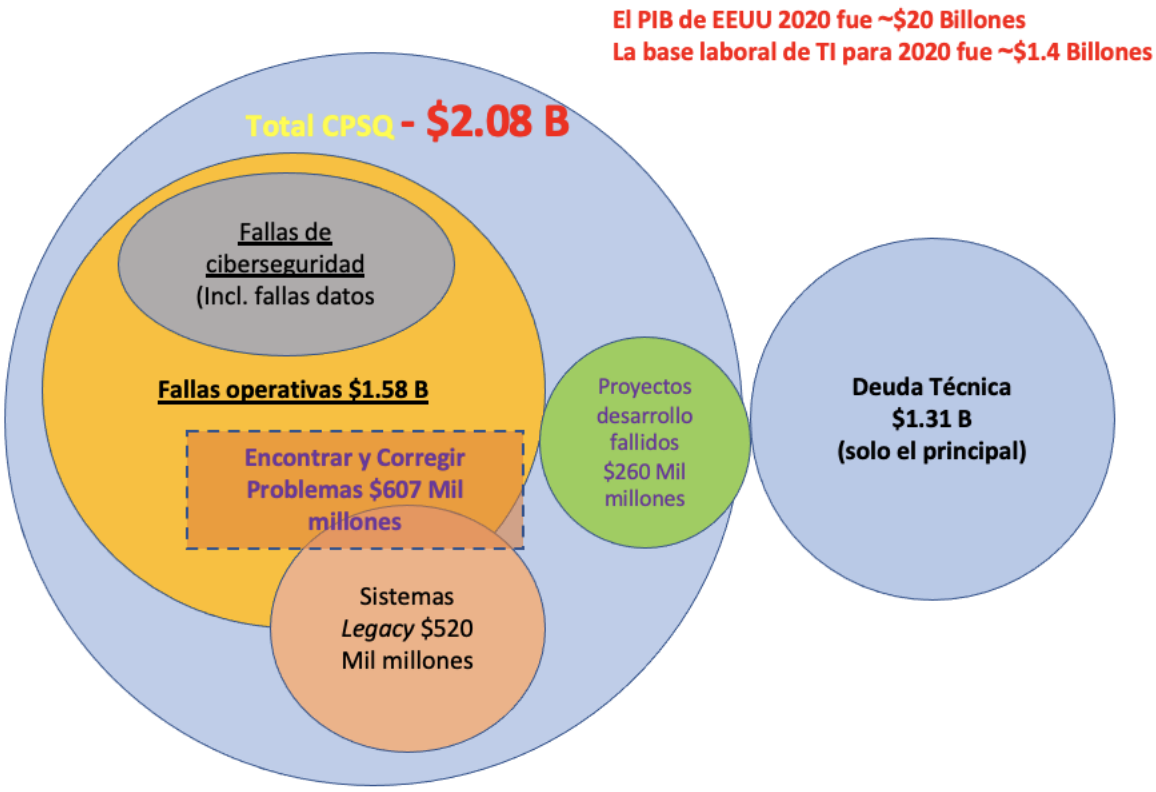
En nuestro informe de 2020, que generó mucha más atención, elaboramos muchos de los informes de fallas conocidos públicamente para enfatizar la magnitud del problema de la baja calidad del software. Presentamos la mayoría de las estrategias, tácticas, modelos y mejores procesos y prácticas que podrían usarse para abordar el problema a través de un enfoque coherente que las organizaciones podrían usar. Concluimos ese informe describiendo el modelo DevQualOps que podría ser implementado por organizaciones y proyectos para los cuales la alta calidad era un objetivo. Esta estrategia y modelo

también se utilizaron para presentar nuestras recomendaciones para todos los niveles de una organización, comenzando con la *C-suite* hasta los ingenieros de software y disciplinas relacionadas.

En el 2020 identificamos qué acciones específicas se pueden tomar a nivel de: 1) profesional individual de sofware, 2) líder de equipo / proyecto y 3) nivel gerencial / ejecutivo de una organización. También revelamos un importante (pero poco conocido) estudio de IBM que explica la diferencia en las prácticas entre las organizaciones de software de alto rendimiento y bajo rendimiento. Ese estudio reveló una diferencia de 5-10X en el rendimiento entre el 10% superior y el 10% inferior de las organizaciones en la muestra. Cuando se profundiza en los datos, la razón es claramente la adopción de las mejores prácticas de gestión de calidad.

En nuestro informe del 2020, que generó mucha más atención, estimamos que el costo de la baja calidad del software en los Estados Unidos ese año fue de $ 2.08 billones, desglosado como se ve en la siguiente figura:

Figura 2-1 costo del software de baja calidad en 2020



Elaboramos muchas de las fallas conocidas públicamente para enfatizar la magnitud del problema de la baja calidad del software. Presentamos la mayoría de las estrategias, tácticas, modelos y mejores

procesos / prácticas que podrían usarse para abordar el problema a través de un enfoque coherente que las organizaciones podrían usar. Concluimos ese informe describiendo el modelo DevQualOps que podría ser implementado por organizaciones y proyectos para los cuales la alta calidad era un objetivo. Esta estrategia y modelo se utilizaron para presentar nuestras recomendaciones para todos los niveles de una organización, comenzando con *la C-suite* hasta los ingenieros de software y disciplinas relacionadas.

En el 2020 recomendamos que la captura de los datos esenciales necesarios para determinar su propio costo de baja calidad de software era posible utilizando las herramientas actuales. De hecho, las organizaciones orientadas a la calidad están capturando los datos de esfuerzo interno necesarios para determinar su costo organizacional de encontrar y corregir deficiencias en los sistemas de informes de problemas y seguimiento de errores que ahora utilizan. En algunos casos en los que tales herramientas no están disponibles, una simple hoja de cálculo es suficiente. Este formulario de ejemplo se puede completar en 1 minuto al final de cada día. Lo que a menudo falta en las organizaciones de menor madurez es la voluntad de hacerlo.

Figura 2-2 Hoja de cálculo de esfuerzo de costo del software de baja calidad del ingeniero de software

La cantidad de esfuerzo para encontrar y corregir errores en el paquete de software XYZ (registro personal para ingenieros de SW) - por período de tiempo TBD										
ID de error	**Encontrar error**	**replicar error**	**Crear prueba**	**Análisis de causa raíz**	**Crear corrección**	**Solución a la falla**	**Demuestre que funciona**	**¿Liberar arreglo/no?**	**Registro de la corrección**	**Distribuir corrección**
Elementos de esfuerzo de costo del software de baja calidad varios para el paquete de software XYZ (por ejemplo)										
			Problema 1	Problema 2	Problema 3					
Problemas de atención al cliente/soporte										
Otros desechos, chatarra, reelaboración										
Fallos y control de daños relacionados										
Pago de la DT										

Véase el Apéndice A para un resumen más detallado del Informe 2020.

El resto de este informe profundiza en los problemas de calidad de software más apremiantes que están surgiendo en la práctica y las tecnologías relacionadas, y en los desarrollos recientes y las soluciones emergentes para ayudar a mejorar la mala situación de calidad del software tal como existe ahora. Comenzamos mirando una muestra de las mayores historias de fallas de software operativo de los últimos 2 años.

3. HISTORIAS DE FALLAS DE SOFTWARE Y MODELOS DE DEFICIENCIA EMERGENTES

Desde nuestro informe de 2020, el estado de la calidad del software en los Estados Unidos no ha mejorado y parece estar empeorando. Esto se puede ver en el siguiente subconjunto seleccionado de historias de fallas de software en los últimos dos años. Sin duda, existen muchos más casos que no han recibido este nivel de atención pública.

El año 2021 comenzó con una noticia cuando se conoció el alcance del hackeo de SolarWinds y luego durante todo el año se revelaron los costos e impactos reales de ese hack. A mediados del período de 2 años, el ataque Colonial Pipeline nos mostró las vulnerabilidades en nuestra infraestructura crítica, y luego el hack Log4j nos mostró que la tecnología *Open Source* ampliamente utilizada era vulnerable. A medida que nos acercamos al final de 2022, han surgido varias historias de fallas nuevas (por ejemplo, Wintermute, GIT, BBN Chain) para recordarnos la naturaleza expansiva y el impacto de estas fallas de software.

Tabla 3-1 - Mayores fallas de software de 2021-2022

Noticia	Explicación/Impacto	Descripción
SolarWinds Orion: 2020-21 (también conocido como "hack de rayos solares") En los primeros nueve meses de 2021, la violación de Orion le costó a SolarWinds $ 40 millones, y pronto aumentó a $ 90 millones, que luego incluyeron respuesta a incidentes y servicios forenses para compañías que se vieron afectadas por este incidente y tienen cobertura de seguro cibernético. Para el verano de 2021, nos enteramos de que el ataque de SolarWinds le costó a las empresas afectadas un promedio de $ 12 millones. Las empresas en los Estados Unidos reportaron un promedio de un impacto del 14%	SolarWinds Orion es un paquete de software de administración de redes empresariales que incluye monitoreo de rendimiento y aplicaciones y administración de configuración de red junto con varios tipos diferentes de herramientas de análisis. Es común que los administradores de red configuren SolarWinds Orion con privilegios generalizados, lo que lo convierte en un objetivo vulnerable. Según los informes, el malware introducido (presumiblemente por el Servicio de Inteligencia Exterior de Rusia) afectó a muchas empresas y organizaciones. Incluso los departamentos gubernamentales como Seguridad Nacional, Estado, Comercio y Tesoro se vieron afectados, ya que había evidencia de que faltaban correos electrónicos en sus sistemas. En el centro de la vulnerabilidad se encontró que se había comprometido la cadena de suministro de una biblioteca de vínculos dinámicos (DLL). SolarWinds informó que poco más de 18,000 de sus clientes descargaron una	El 13 de diciembre de 2020, CISA lanzó la Directiva de emergencia 21-01: Mitigar el compromiso del código de orion de SolarWinds, Para obtener más detalles, consulte este documento informativo. Los perpetradores luego procedieron a agregar malware a uno de los servicios de software más utilizados de la compañía, Orion. El incidente de piratería fue sigiloso y no destructivo, lo que le permitió pasar desapercibido por SolarWinds y permanecer allí durante meses. El código se extendió a otros clientes al enganchar una de las actualizaciones de software regulares que SolarWinds envía a sus clientes. Ahí, el malware estableció una puerta trasera para los hackers informáticos, permitiéndoles instalar malware aún más invasivo y espiar a sus objetivos y filtrar cualquier información que consideraran importante.

Noticia	*Explicación/Impacto*	*Descripción*
en sus ingresos anuales. Las acciones de SolarWinds se desplomaron desde $ 25 / acción a fines de 2020 a un precio actual de aproximadamente $ 9 / acción.	versión afectada, aunque no todos fueron pirateados activamente. Aproximadamente 100 empresas conocidas se vieron afectadas.	
La violación de datos de T-Mobile afecta a 50 millones de clientes El 18 de marzo de 2021, un hacker accedió ilegalmente y adquirió datos personales de más de 50 millones de clientes. T-Mobile se enteró de la violación masiva de datos el 17 de agosto de 2021.	El tipo de información personal que fue comprometida variaba según el individuo, pero incluía nombres, fechas de nacimiento, direcciones, números de teléfono, licencias de conducir, números de identificación del gobierno, números de seguro social y PIN prepagados de T-Mobile. Para proteger a sus clientes, el operador de telefonía móvil les informó de la violación de seguridad y los alentó a tomar medidas proactivas regularmente para mantener sus datos seguros. Aunque el CEO de T-Mobile se disculpó por la violación de la seguridad de los datos y prometió reforzar las defensas, muchos clientes afectados por la violación de datos han decidido emprender acciones legales. Las demandas alegan que los deficientes protocolos de seguridad de T-Mobile son los culpables y permitieron a los hackers informáticos obtener acceso a los servicios de la compañía y extraer la información personal de millones de personas. El costo total para T-Mobile y sus clientes aún no se ha determinado.	Se utilizó un enrutador vulnerable para obtener acceso a los servidores de T-Mobile. La seguridad de T-Mobile fue descrita por el hacker como pésima.
La falla de TikTok restablece los seguidores a cero El 3 de mayo de 2021, cuando los usuarios de TikTok iniciaron sesión en la aplicación, lo último que esperaban	TikTok experimentó una falla que mostró el recuento incorrecto de seguidores / seguidos. Algunos usuarios incluso tuvieron problemas para acceder a la aplicación, y la aplicación bloqueó sus cuentas. Los usuarios llevaron su frustración a las redes sociales. Pronto #TikTokDown	Una empresa con una base de usuarios tan grande no puede permitir que se filtren errores de software. Fallas como esta se pueden prevenir fácilmente con mejores pruebas de software.

Noticia	*Explicación/Impacto*	*Descripción*
ver era que todos sus usuarios se habían ido.	fue tendencia. Cada vez más usuarios se presentaron pidiéndole al gigante de las redes sociales que solucionara la falla y restaurara sus cuentas y seguidores. TikTok confirmó la falla, informando a sus usuarios que estaban trabajando para reparar el problema. La falla se resolvió de la noche a la mañana,	
El costoso ataque de ransomware de Colonial Pipeline El ataque a Colonial Pipeline es uno de los peores ataques cibernéticos Esto ocurrió en 2021-22. Este ataque interrumpió casi la mitad del suministro de combustible en la costa este de los Estados Unidos. Causó escasez de gasolina en el sureste y un aumento en los precios del combustible.	Colonial Pipeline es el oleoducto de productos refinados más grande de los Estados Unidos, un sistema de 5,500 millas (8,851 km) involucrado en el transporte de más de 100 millones de galones desde la ciudad de Houston en Texas hasta el puerto de Nueva York. Transporta el 45% del combustible consumido en la costa este de los Estados Unidos. El 29 de abril, los hackers informáticos obtuvieron acceso a la red de Colonial Pipeline a través de una cuenta de red (VPN), que permitía a los empleados acceder de forma remota a la red de la empresa. Los hackers obtuvieron credenciales válidas que les permitieron violar la red de Colonial Pipeline, porque la cuenta VPN no usaba autenticación multifactor. Después de una semana, el 7 de mayo, Colonial Pipeline recibió una nota de rescate exigiendo que se pagara un rescate de criptomonedas. Poco después, el oleoducto fue cerrado. Con aproximadamente 2.5 millones de barriles de combustible en el sureste de los Estados Unidos diariamente, la interrupción paralizó la entrega de combustible. Resultó en largas filas en las estaciones de servicio, algunas de las cuales sufrieron el desabasto, y precios más altos para el combustible. Los hackers robaron casi 100 gigabytes de datos y amenazaron con filtrarlos si no pagaban el rescate. Colonial Pipeline pagó un rescate de 75 Bitcoins ($ 5 millones) a los hackers informáticos, que se creía que eran el	El ataque de ransomware en Colonial Pipeline muestra el extenso daño que pueden causar las medidas de seguridad insuficientes y las vulnerabilidades del sistema. El ataque empezó cuando un grupo hacker identificado como DarkSide accedió la red de Colonial Pipeline. Los atacantes se robaron 100 gigabites de datos en una ventana de dos horas. Posterior al robo de datos, los atacantes infectaron la red de TI de Colonial Pipeline con ransomware que afectó muchos sistemas de las computadoras, incluyendo facturación y contabilidad. Colonial Pipeline se vio forzada a apagar los sistemas para prevenir que se esparciera el ransomware. Una vez pagaron a los hackers de DarkSide para obtener la llave de desencriptación, pudieron reiniciar los sistemas. La causa raíz parece ser el robo de una clave de acceso a la VPN de Colonial.

Noticia	*Explicación/Impacto*	*Descripción*
	grupo de cibercrimen conocido como DarkSide. El costo total de las vulnerabilidades de seguridad subyacentes es incalculable.	
Twitter Una vulnerabilidad en el software de Twitter expuso a un número indeterminado de propietarios de cuentas anónimas a un posible compromiso de identidad durante el último año. Se informó que los datos sobre 5,4 millones de usuarios fueron ofrecidos en venta en línea.	Los datos obtenidos de la explotación de esa vulnerabilidad se vendían en un popular foro de *hacking* por $ 30,000. La vulnerabilidad permitió a alguien determinar durante el inicio de sesión si un número de teléfono o dirección de correo electrónico en particular estaba vinculado a una cuenta de Twitter existente, revelando así a los propietarios de la cuenta. Un investigador de seguridad descubrió la falla en enero, informó a Twitter y recibió una recompensa de $ 5,000. El error se introdujo en una actualización de software de junio de 2021 y fue corregido.	La revelación de la violación se produjo mientras Twitter estaba en una batalla legal con el CEO de Tesla, Elon Musk, por su intento de retirarse de su oferta anterior de comprar Twitter, con sede en San Francisco, por $ 44 mil millones. La violación es especialmente preocupante porque muchos propietarios de cuentas de Twitter, incluidos activistas de derechos humanos no revelan sus identidades en sus perfiles por razones de seguridad que incluyen el temor a la persecución por parte de autoridades represivas.
Error de clasificación de Facebook Un grupo de ingenieros de Facebook identificó una "falla masiva de clasificación" que expuso hasta la mitad de todas las vistas de *News Feed* a posibles “riesgos de integridad”	Los ingenieros notaron por primera vez el problema en octubre de 2021, cuando una repentina oleada de desinformación comenzó a fluir a través del *News Feed*. El sistema Downranking de Facebook no degradó adecuadamente la probable desnudez, la violencia e incluso los medios estatales rusos. El problema fue internamente designado como SEV de nivel uno, una etiqueta reservada para crisis técnicas de alta prioridad. Incapaces de encontrar la causa raíz, los ingenieros de software vieron cómo el aumento disminuía unas semanas más tarde y luego estallaba repetidamente hasta que finalmente se solucionó el problema de clasificación el 11 de marzo de 2022.	La vulnerabilidad técnica aparentemente se introdujo en 2019, pero no creó un impacto notable hasta octubre de 2021. En un sistema grande y complejo como este, los errores son inevitables y comprensibles. ¿Qué sucede cuando una poderosa plataforma red social tiene una de estas fallas? ¿Cómo lo sabrían los usuarios?

Noticia	*Explicación/Impacto*	*Descripción*
	El daño a la reputación de Facebook por moderar contenido controvertido fue incalculable.	
Tesla retira casi 12.000 vehículos En noviembre de 2021, Tesla retiró cerca de 12,000 vehículos después de descubrir una falla en su software beta de conducción autónoma completa.	Tras su actualización más reciente el 23 de octubre, Tesla comenzó a recibir informes de clientes que informaban que sus vehículos habían identificado falsamente amenazas de colisión frontal que causaban que el sistema de frenado automático de emergencia (AEB) se activara y detuviera repentinamente el vehículo, lo que aumentaba el riesgo de una colisión trasera y lesiones a aquellos dentro del vehículo. Tesla descubrió un error de comunicación en el software beta 10.3 Full-Self Driving (FSD). Es decir, el error de software indicó una falsa colisión hacia adelante.	Para mitigar los posibles riesgos de seguridad, Tesla pidió a su equipo de control de calidad que investigara e identificara la causa del error de software. El fabricante de automóviles publicó rápidamente un Informe de retiro de seguridad para retirar los vehículos afectados: ciertos vehículos Model S, Model X y Model 3 fabricados 2017-2021, y ciertos modelos Model Y fabricados 2020-2021. Tesla lanzó una actualización separada para abordar el problema de software y notificó a los propietarios de vehículos sobre el problema y la resolución. Afortunadamente, no hubo accidentes o lesiones como resultado del error de software.
Grand Theft Auto Lo que prometía ser un remaster de alta calidad de los clásicos de Grand Theft Auto, GTA III, Vice City y San Andreas, resultó ser un juego de baja calidad lleno de errores, fallas y malas decisiones de diseño.	Cuando se lanzó el juego en noviembre de 2021, la recepción de los fanáticos estuvo lejos de ser excelente, y Rockstar Games recibió muchas reacciones violentas. Algunos usuarios incluso llegaron a pedir un reembolso. ¿Por qué? Porque la calidad era mala, muy, muy mala. Los gráficos de NPC eran terribles, los modelos de personajes tenían manchas, la velocidad de fotogramas caía constantemente, los efectos de lluvia dificultaban la visión, las misiones y los minijuegos no funcionaban según lo previsto y la calidad del audio era espantosa. Todos estos problemas juntos hicieron que el juego fuera casi imposible de jugar. El costo para la reputación de la empresa no se ha calculado.	Desde entonces, el editor de videojuegos ha descubierto, y aparentemente solucionado, la larga lista de errores de software. Sin embargo, el daño estaba hecho, y tomará mucho tiempo para que el editor se recupere de este error.
El error del software Log4j deja vulnerables a millones de servidores web	Lo que hace que este error sea tan aterrador es el hecho de que Log4j, una bitácora *Open Source* es utilizada por muchas empresas en todo el mundo, incluidas organizaciones de alto perfil	Se están haciendo esfuerzos para solucionar el problema (llamado log4shell). Los equipos de todo el mundo están trabajando arduamente para

Noticia	Explicación/Impacto	Descripción
	como Apple, Amazon, Cisco, IBM, Microsoft y muchas más. Muchas partes relacionadas — empresas, los clientes y los usuarios por igual - están muy preocupados. El software Lob4j se utiliza para dejar en bitácora todas las actividades que suceden en un rango extenso de sistemas, tales como errores y operaciones de rutina del sistema, y presenta mensajes de diagnóstico a los administradores de sistema y usuarios. El ejemplo más común de Log4j aplicado es el mensaje de error 404 que todos conocemos. Los hackers pueden aprovecharse de estos diagnósticos para escanear sistemas vulnerables e instalar malware, robar credenciales y obtener información confidencial. Debido a lo extenso del daño que podría causar, muchos creen que Log4j es el error de software con la vulnerabilidad más alta en años. Los hackers han utilizado esta falla para forzar a las víctimas a minar sumas pequeñas de criptomonedas y para piratear servidores privados de Minecraft. Es una combinación de una nueva vulnerabilidad que simultáneamente se puede esparcir y de la cual se puede sacar provecho fácilmente. El Netherlands National Cyber Security Centre ha identificado cientos de aplicaciones comunes de software que son vulnerables a la falla si no se actualiza, y un número importante que podría no tener un parche disponible aún. En un blog, Microsoft dijo que ha estado observando como China, Irán, Korea del Norte y Turquía han aprovechado esta ventaja.	parchear los sistemas afectados antes de que los hackers puedan explotarlos, mientras se insta a las organizaciones a instalar las últimas actualizaciones de seguridad para contrarrestar la amenaza lo antes posible. Si bien barrer sus redes y aplicar un parche podría ser una solución por ahora, muchas empresas aún quedan vulnerables y esta solución aún puede no ser suficiente. Solo el tiempo lo dirá.

Noticia	*Explicación/Impacto*	*Descripción*
Wintermute, septiembre de 2022 Los hackers robaron activos digitales por valor de alrededor de $ 160 millones de la firma de comercio de criptomonedas Wintermute. El hack involucró una serie de transacciones no autorizadas que transfirieron USD Coin, Binance USD, Tether USD, Wrapped ETH, y otras 66 criptomonedas a la billetera del atacante.	Según un informe de mayo de 2022 de Bishop, los incidentes de seguridad de Fox que golpearon las plataformas DeFi resultaron en pérdidas por una suma de **$ 1.8 mil millones** solo en 2021, y los servicios experimentaron un promedio de cinco hackeos por mes. Para empeorar las cosas, los hackers han robado $ 1.3 mil millones en criptomonedas solo en los primeros tres meses de 2022, en comparación con los $ 3.2 mil millones que se robaron durante todo 2021, lo que indica un "aumento meteórico" en los delitos criptográficos.	El ataque se dirigió al espacio de finanzas descentralizadas (DeFi). El paquete *Open Source* implicado es Profanity, una herramienta de generación de direcciones personalizadas de Ethereum, donde recientemente se reveló que se podría abusar de una vulnerabilidad para volver a calcular las claves de billetera privada de las direcciones creadas con este utilitario.
IDE de Spyder Python, septiembre de 2022 (Open Source) Originalmente divulgado en agosto de 2007, el error tiene que ver con cómo se puede aprovechar un archivo de destino especialmente diseñado para sobrescribir archivos arbitrarios en una máquina de destino simplemente al abrir el archivo.	Se cree que hasta 350,000 proyectos de *Open Source* son vulnerables a esta debilidad como resultado de una falla de seguridad en un módulo de Python que ha permanecido sin parches durante 15 años. Los repositorios *Open Source* abarcan una serie de verticales de la industria, como el desarrollo de software, inteligencia artificial / aprendizaje automático, desarrollo web, medios, seguridad y gestión de TI.	Ahora rastreado como CVE-2007-4559 (puntuación CVSS: 6.8), está arraigado en el módulo tarfile, cuya explotación exitosa podría conducir a la ejecución de código a partir de una escritura arbitraria de archivos.
BNB Chain, agosto de 2022 BNB Chain, es una Blockchain vinculada al intercambio de criptomonedas Binance. BNB, que significa 'Build and Build' (anteriormente llamado Binance Coin), es el token de gas blockchain que 'alimenta' las transacciones en BNB Chain, como se señaló a principios de este año.	En agosto se informó que se había robado una criptomoneda estimada en $ 2 mil millones en 13 ataques *cross-chain bridge*, lo que representa el 69% del total de fondos criptográficos robados en 2022 hasta ahora.	Este es el último de una serie de incidentes importantes dirigidos a *cross-chain bridges*, que facilitan la transferencia de activos entre blockchains, este año, después de los de Axie Infinity, Harmony Horizon Bridge y Nomad Bridge.
GIT, agosto de 2022	En agosto, una vulnerabilidad en la herramienta de desarrollo *Open Source*	La tecnología *Open Source* siempre ha tenido el potencial de fallas, ya que se basa en el

Noticia	*Explicación/Impacto*	*Descripción*
Git es un sistema de control de versiones *Open Source* muy popular, que cuenta con más de 80 millones de usuarios activos.	Git que, si no se aborda, permite a los hackers las llaves del reino. En total, se encontraron 332,000 sitios web como potencialmente vulnerables, incluidos 2,500 que residen en el dominio .gov. Los usuarios de GitHub están siendo atacados con copias maliciosas de repositorios legítimos. Si bien la mayoría de los cambios en el código malicioso se realizaron en los últimos meses, se descubrió que algunos datan de hace siete años. Según lo informado por GitHub, un actor de amenazas logró robar datos de "docenas de víctimas".	código de acceso público. Este tipo de vulnerabilidad, en una plataforma tan popular, puede tener "graves consecuencias" para las empresas afectadas. Más recientemente, se descubrió el error RepoJacking que podría permitir a un atacante tomar el control de un repositorio de GitHub y potencialmente infectar todas las aplicaciones y otro código que depende de él con malware.
OpenLightSpeed, 22 de noviembre 2022	OpenLiteSpeed es una edición *Open Source* del servidor de red LiteSpeed el sexto servidor web más popular, que cuenta con 1.9 millones de servidores alrededor del mundo La unidad 42 de Palo Alto Networks indicó que se han descubierto múltiples fallas de alta severidad en el código *Open Source* que se pueden utilizar para alcanzar código de ejecución remota.	Las vulnerabilidades descubiertas incluyen: 1. Código de ejecución remota (CVE-2022- 0073) de alta severidad (CVSS 8.8) 2. Escalación de privilegios (CVE-2022- 0074) de alta severidad (CVSS 8.8) 3. Directory Traversal (CVE-2022- 0072) calificada de severidad media(CVSS 5.8)
FTX Crypto Hack, 11 de noviembre de 2022 **FTX, una compañía de 32.000 millones de dólares, se vaporizó de la noche a la mañana.**	El 11 de noviembre de 2022, el CEO de la firma de intercambio de criptomonedas FTX renunció y dijo que se estaban declarando en bancarrota. Aparentemente, $ 473 millones en activos criptográficos fueron robados. FTX estaba "investigando anomalías" con respecto a los movimientos en las billeteras criptográficas "relacionadas con la consolidación de los saldos de FTX en los intercambios". Las monedas estables y otros tokens faltantes se estaban convirtiendo rápidamente a Ether, la segunda criptomoneda más grande después de Bitcoin, en intercambios descentralizados, que es	Los hechos subyacentes aún no están claros en este momento.

Noticia	*Explicación/Impacto*	*Descripción*
	una técnica común utilizada por los hackers informáticos para evitar que sus fondos sean incautados.	
CommonSpirit Health, octubre-noviembre. 2022	Un ataque de ransomware paralizante en el segundo sistema de salud sin fines de lucro más grande de Estados Unidos muestra lo que sucede cuando la infraestructura crítica de atención médica se cae. El ataque a CommonSpirit Health, que tiene 142 hospitales en 21 estados, dejó la TI bloqueada, retrasó las cirugías y causó interrupciones generalizadas en la atención al paciente. También dejó a millones de pacientes esperando al menos dos semanas para saber si su información personal estaba comprometida.	Los hechos subyacentes aún no están claros en este momento.

A medida que vemos las noticias, aparecen aún más: por ejemplo,

Ataques de ransomware para el cuidado de la salud, septiembre de 2022

• https://thehackernews.com/2022/10/cisa-warns-of-daixin-team-hackers.html?_m=3n%2e009a%2e2869%2eye0ao43m8z%2e1u6n
Text4Shell – Octubre de 2022

• https://thehackernews.com/2022/10/hackers-started-exploiting-critical.html?_m=3n%2e009a%2e2868%2eye0ao43m8z%2e1u63
vulnerabilidades de alta gravedad de OpenSSL: el parche para 2 ya está disponible - 1 de noviembre de 2022

• https://thehackernews.com/2022/11/just-in-openssl-releases-patch-for-2.html
vulnerabilidades críticas en 3 software de sistemas de control industrial - 8 de noviembre de 2022

• https://www.cisa.gov/uscert/ncas/current-activity/2022/11/03/cisa-releases-three-industrial-control-systems-advisories

Las principales debilidades de software identificadas de 2021-2022

Todas las historias de fallas de software presentadas en la tabla anterior se deben a debilidades encontradas en el software de esos sistemas. Las debilidades del software son: fallas, errores, vulnerabilidades u otros tipos de deficiencias encontradas en el código de soluciones de software, arquitectura, implementación o diseño. Exponen potencialmente los sistemas que contienen ese software a fallas y /o ciberataques.

MITRE recientemente compartió sus 25 debilidades más comunes y peligrosas que afectan al software en los dos años calendario anteriores. Estos se consideran los más peligrosos porque generalmente son fáciles de descubrir, tienen un alto impacto y prevalecen en el software lanzado durante los últimos dos años.

Para crear esta lista, MITRE calificó cada debilidad conocida de su base de datos CWE (Common Weaknesses Enumeration) en función de su prevalencia y gravedad después de analizar los datos de 37,899 CVE (Common Vulnerabilities Enumeration) de la Base de Datos Nacional de Vulnerabilidades (NVD) del NIST y el Catálogo de Vulnerabilidades Explotadas Conocidas (KEV) de CISA.

Aunque estas debilidades se han aplicado principalmente al aspecto de calidad de la seguridad, afirmamos que se aplican de manera más general a la calidad del software. La siguiente tabla proporciona información sobre las 25 debilidades más críticas y actuales.

Tabla 3-2 Top 25 CWEs

Rango	ID	Nombre	Puntuación	KEV Conteo (CVE)	Cambio de Rango vs. 2021
1	CWE-787	Escritura fuera de límites	64.20	62	0
2	CWE-79	Neutralización incorrecta de la entrada durante la generación de páginas web ('cross-site scripting')	45.97	2	0
3	CWE-89	Neutralización incorrecta de elementos especiales utilizados en un comando SQL ('inyección SQL')	22.11	7	+3 ▲
4	CWE-20	Validación de entrada incorrecta	20.63	20	0
5	CWE-125	Lectura fuera de los límites	17.67	1	-2 ▼
6	CWE-78	Neutralización incorrecta de elementos especiales utilizados en un comando del sistema operativo ('Inyección de comandos del sistema operativo')	17.53	32	-1 ▼
7	CWE-416	Úsalo después de gratis	15.50	28	0
8	CWE-22	Limitación incorrecta de un nombre de ruta a un directorio restringido ('recorrido de ruta')	14.08	19	0
9	CWE-352	Falsificación de solicitudes entre sitios (CSRF)	11.53	1	0
10	CWE-434	Carga sin restricciones de archivos con tipo peligroso	9.56	6	0
11	CWE-476	Desreferencia de puntero NULL	7.56	0	+4 ▲
12	CWE-502	Deserialización de datos que no son de confianza	6.68	7	+1 ▲
13	CWE-190	Desbordamiento de enteros o envolvente	6.53	2	-1 ▼
14	CWE-287	Autenticación incorrecta	6.35	4	0

15	CWE-798	Uso de credenciales codificadas de forma rígida	5.66	0	+1 ▲
16	CWE-862	Falta autorización	5.53	1	+2 ▲
17	CWE-77	Neutralización inadecuada de elementos especiales utilizados en un comando	5.42	5	+8 ▲
18	CW -306	Falta autenticación para la función crítica	5.15	6	-7 ▼
19	CWE-119	Restricción inapropiada de operaciones dentro de los límites de un buffer de memoria	4.85	6	-2 ▼
20	CWE-276	Permisos predeterminados incorrectos	4.84	0	-1 ▼
21	CWE-918	Falsificación de solicitudes del lado del servidor (SSRF)	4.27	8	+3 ▲
22	CWE-362	Ejecución simultánea mediante recursos compartidos con recursos inadecuados 'Condición de carrera'	3.57	6	+11 ▲
23	CWE-400	Consumo de recursos no controlado	3.56	2	+4 ▲
24	CWE-611	Restricción incorrecta de XML Referencia de entidad externa	3.38	0	-1 ▼
25	CWE-94	Control incorrecto de la generación de código ('inyección de código')	3.32	4	+3 ▲

Muchos profesionales que se ocupan del software encontrarán en el CWE Top 25 un recurso práctico y conveniente para ayudarlos a mitigar el riesgo de calidad de su software.

En abril de 2022, en asociación con el FBI y la NSA, las autoridades de ciberseguridad de todo el mundo publicaron su lista de las 15 fallas de seguridad a las que se les saca más provecho, con enlaces a las entradas de la Base de Datos Nacional de Vulnerabilidades y el malware asociado.

Tabla 3-3 Top 15 CVEs

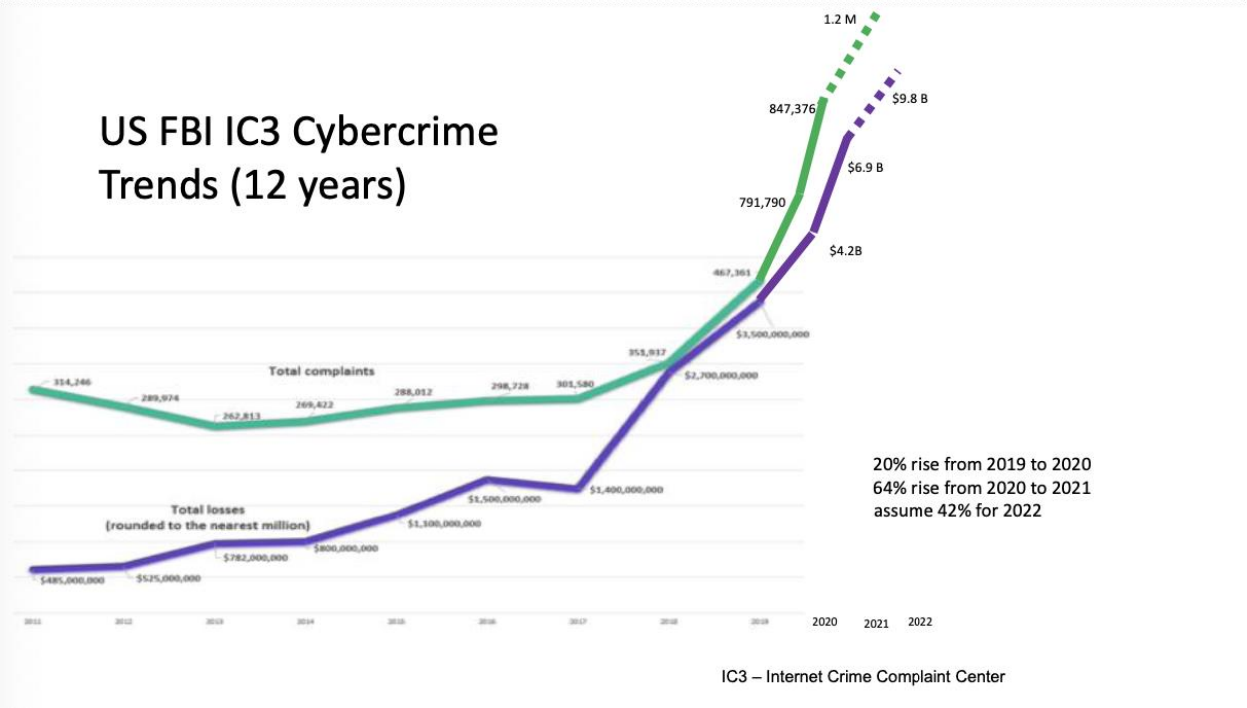
CVE	Vulnerabilidad	Empresa y producto donde se encontró	Tipo
CVE-2021-44228	Log4Shell	Apache Log4j	Ejecución de Código Remoto (RCE)
CVE-2021-40539		Zoho ManageEngine AD SelfService Plus	RCE
CVE-2021-34523	ProixyShell	Microsoft Exchange Server	Mejora de privilegio
CVE-2021-34473	ProxyShell	Microsoft Exchange Server	RCE
CVE-2021-31207	ProxyShell	Microsoft Exchange Server	Violación de dispositivo de seguridad
CVE-2021-27065	ProxyLogon	Microsoft Exchange Server	RCE
CVE-2021-26858	ProxyLogon	Microsoft Exchange Server	RCE
CVE-2021-26857	ProxyLogon	Microsoft Exchange Server	RCE
CVE-2021-26855	ProxyLogon	Microsoft Exchange Server	RCE
CVE-2021-26084		Atlassian Confuence Server aand Data Center	Ejecución de Código arbitrario
CVE-2021-21972		Cliente de VMWare vSphere	RCE
CVE-2020-1472	ZeroLogon	Microsoft Netlogon Remote Protocol (MS-NRPC)	Mejora de privilegio
CVE-2020-0688		Microsoft Exchange Server	RCE
CVE-2019-11510		Pulse Secure Pulse Connect Secure	Lectura de archivos arbitraria
CVE-2018-13379		Fortinet FortiOS and FortiProxy	Traspaso de rutas

Se publicaron las medidas de mitigación relacionadas que deberían ayudar a disminuir el riesgo asociado con estos. Cuando las bases de datos anteriores están vinculadas a los estándares emergentes para la calidad del software y sus patrones subyacentes, entonces la industria tiene una base sólida para medir y controlar la calidad del software.

4. EL CRECIENTE COSTO DE LA CIBERDELINCUENCIA

Se prevé que el cibercrimen le cueste al mundo $ 7 billones de dólares en 2022, según Cybersecurity Ventures. Si se midiera como un país, entonces el cibercrimen sería la tercera economía más grande del mundo después de Estados Unidos y China. El número de incidentes de delitos cibernéticos y sus costos relacionados han ido en aumento durante más de una década, como se ve en el gráfico a continuación. Estos datos se basan solo en aquellos incidentes que se han reportado al FBI, que por supuesto, es mucho menor que la imagen total de todos los delitos cibernéticos de los Estados Unidos.

Figura 4-1 Tendencias de la ciberdelincuencia en Estados Unidos: últimos 12 años



Los costos totales asociados con los ataques cibernéticos (demandas, aumentos de tarifas de seguros, investigaciones criminales y mala publicidad) pueden llevar rápidamente a una empresa a la quiebra.

Titulares de la industria de la ciberseguridad

Muchas noticias de ciberseguridad surgieron en 2021-22. Los hackers y los ciberdelincuentes atacaron despiadadamente a empresas, gobiernos e individuos. Aquí hay un vistazo a algunos de los principales titulares de la industria:

- Según el informe "The State of Incident Response 2021" de VMware, al 82% de las organizaciones encuestadas les preocupa que su empresa sea vulnerable a un ataque cibernético. El informe encontró que el 49% de las organizaciones carecen de la experiencia y las herramientas para una respuesta adecuada a incidentes.
- La lista de los más buscados del FBI presenta a más de 70 individuos y grupos que han conspirado para cometer los crímenes más dañinos contra los Estados Unidos. Estos delitos incluyen intrusiones informáticas, fraude electrónico, robo de identidad, espionaje, robo de secretos comerciales y muchos otros delitos.

- Las VPN son especialmente vulnerables, ya que seis empresas chinas poseen el 30% de las VPN, y 97 de las principales VPN están administradas por 23 empresas matrices, muchas de las cuales tienen su sede en países con leyes de privacidad laxas.

- Las organizaciones están llevando a cabo más análisis de pruebas de seguridad de aplicaciones que nunca, según el informe "State of Software Security v12" de Veracode. En 2021, la mayoría de las empresas escaneaban aplicaciones aproximadamente tres veces por semana, frente a tres veces al año en 2010.
- Los ataques de seguridad aumentaron un 31% de 2020 a 2021, según el informe "State of Cybersecurity Resilience 2021" de Accenture. El número de ataques por empresa aumentó de 206 a 270 año tras año.
- Según Debricked, en promedio se necesitan *más de 800 días* para descubrir una falla de seguridad en *Open Source*. Por ejemplo, la vulnerabilidad Log4shell (CVE-2021-44228) estuvo oculta por 2649 días.

  También se han publicado informes similares:

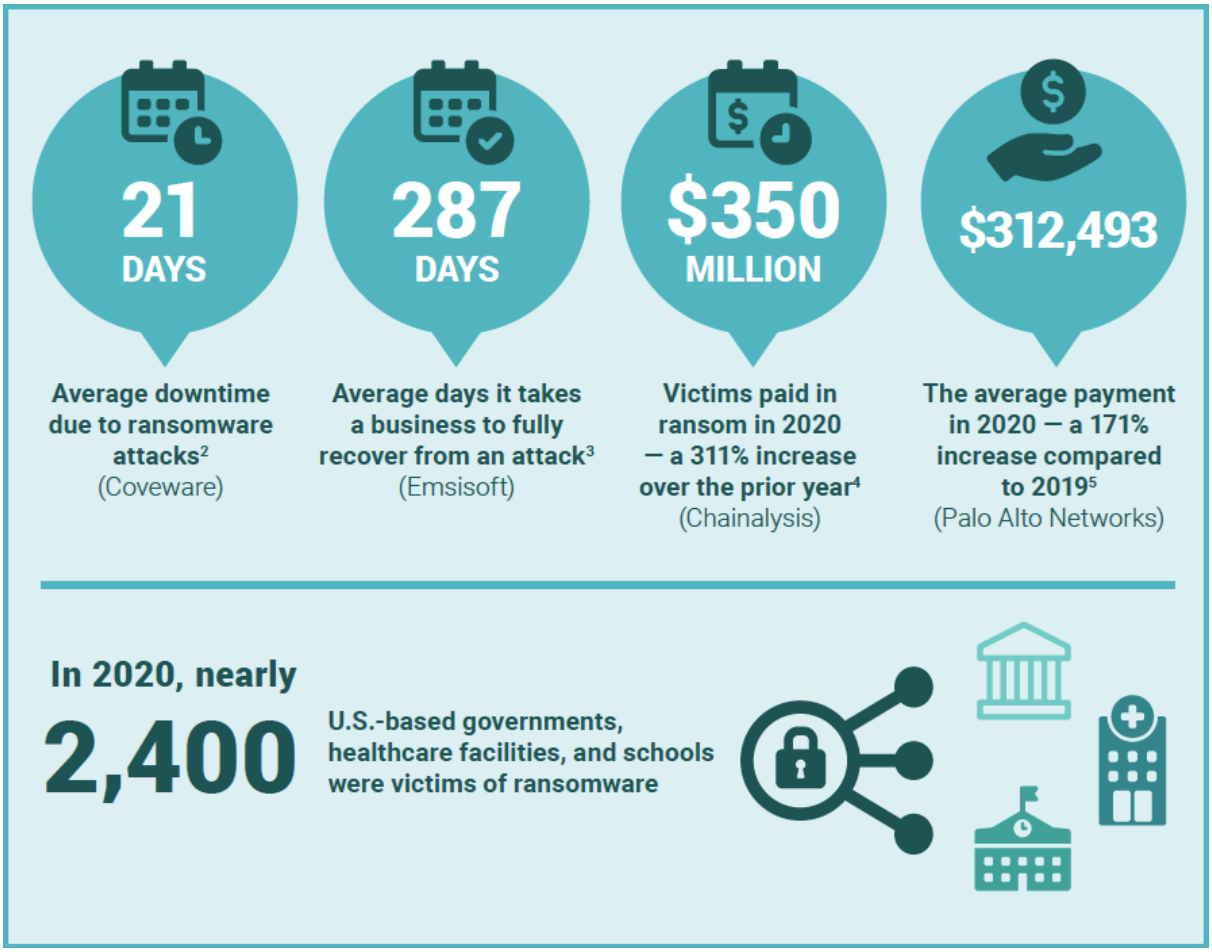
  1. Los ciberdelincuentes pueden penetrar en el 93 por ciento de las redes de las empresas (betanews.com)
  2. Los ataques a la cadena de suministro de software afectaron a tres de cada cinco empresas en 2021 | CSO en línea
  3. El 82% de los CIO cree que sus cadenas de suministro de software son vulnerables (betanews.com)
  4. Las empresas sufrieron un 50% más de intentos de ciberataques por semana en 2021 (darkreading.com)
  5. Los ataques de ransomware y los pagos de rescate son rampantes entre las organizaciones de infraestructura crítica – Help Net Security
  6. Tendencias, estadísticas y hechos del ransomware en 2022 (techtarget.com)

  Las tendencias más candentes de ciberdelincuencia en 21-22 fueron:

  1. Ransomware
  2. Cryptojacking
  3. Deepfakes
  4. Ataques de videoconferencia
  5. Ataques de IoT y OT
  6. Ataques a la cadena de suministro/*Open Source*
  7. Soluciones extendidas de detección y respuesta (también conocidas como XDR)
  8. Ataques a infraestructuras críticas

El impacto financiero de Ransomware se ve mejor en el siguiente cuadro resumen del Informe del Grupo de Trabajo de IST Ransomware de abril de 2021.

Figura 4-2 – Impacto del ransomware en 2020



Más recientemente, la Agencia de Seguridad y Ciberseguridad e Infraestructura informó en febrero de 2022 que tiene conocimiento de incidentes de ransomware contra 14 de los 16 sectores de infraestructura crítica de EE. UU.

Tal vez ninguna tendencia de ciberseguridad fue mayor en 2021-22 que el flagelo de los ataques de ransomware en la cadena de suministro.

- Entre los mayores ataques se encuentra el ataque de ransomware Colonial Pipeline, que afectó a la costa este de los Estados Unidos en mayo de 2021.
- Hubo problemas continuos relacionados con la seguridad de la cadena de suministro derivados de una violación al proveedor de software de administración SolarWinds.

A partir de 2022, el costo promedio de una violación de datos en los Estados Unidos ascendió a $ 9.44 millones, frente a $ 9.05 millones en el año anterior.

Otra buena manera de ver qué tan rápido el cibercrimen se ha convertido en un problema importante es por la cantidad de dinero que varias organizaciones invertirán en esa área. Por ejemplo:

- El año pasado, Google comprometió $ 10 mil millones durante 5 años para financiar un programa para fortalecer la ciberseguridad, incluida la expansión de los programas de confianza cero, ayudar a asegurar la cadena de suministro de software y mejorar la seguridad *Open Source*.

- La administración Biden solicitó $ 2.1 mil millones en el presupuesto discrecional de 2022 para la Agencia de Seguridad y Ciberseguridad e Infraestructura (CISA). Eso es un aumento de $ 110 millones desde el nivel de 2021 y se basa en los $ 650 millones proporcionados para CISA en el Plan de Rescate Estadounidense. El dinero se destinaría a:
    1. Mejorar sus herramientas de ciberseguridad
    2. Contratación de expertos
    3. Obtener servicios de apoyo para proteger y defender los sistemas tecnológicos federales
    4. Creación de un Fondo de Respuesta y Recuperación Cibernética

5. CADENAS DE SUMINISTRO DE SOFTWARE (SSC) CON *OPEN SOURCE*

Una cadena de suministro de software está constituida por los componentes, bibliotecas, herramientas, datos y procesos utilizados para desarrollar, construir, publicar y evolucionar un sistema de software. Los creadores de software a menudo crean sus productos en gran parte ensamblando componentes de software *Open Source*, de terceros y comerciales. Esta forma de crear software permite el desarrollo rápido de características y la reutilización masiva del código existente, pero abre las puertas a las vulnerabilidades de la cadena de suministro. El software *Open Source* (OSS) desempeña un papel fundamental en el ecosistema de TI actual. La gran mayoría de las bases de código modernas contienen componentes *Open Source*, y el *Open Source* a menudo comprende el 70% o más del código general. Según CAST Software, una aplicación de tamaño mediano (menos de 1 millón de líneas de código) incluye de 200 a 300 componentes de terceros en promedio.

Las 4 razones principales citadas para usar *Open Source* son:

- Acceso a innovaciones y últimas tecnologías
- Sin costo de licencia, reducción general de costos
- Para modernizar los *stack* de tecnología
- Funcionalidad para mejorar la velocidad de desarrollo

En el 2021, según Perforce, el 77% de las organizaciones informaron un aumento en el uso de software *Open Source*, y el 36% indicó un aumento significativo. Solo el 1,6% de los más de dos mil encuestados indicaron que redujeron el uso de software *Open Source*. Sin embargo, solo al 13% le preocupa que su *Open Source* no sea seguro o no esté probado, mientras que el 27,5% no tiene reservas en el uso de *Open Source*. Parecería haber una desconexión entre los riesgos que corren esas organizaciones en relación con los riesgos reales de utilizar determinados componentes de *Open Source*.

Según un informe reciente de la encuesta de IDC, el 86% de los encuestados dijeron que a veces o siempre intentan encontrar opciones *Open Source* sobre otros tipos de software. Sin embargo, la mayoría de las organizaciones no son conscientes de hasta qué punto ya utilizan *Open Source* y subestiman su dependencia de él, una dependencia que conlleva algunos riesgos. La investigación de IDC encontró que el 68% de las organizaciones que usan cualquier tipo de software *Open Source* reconocieron que se habían visto afectadas por una vulnerabilidad o compromiso asociado con una tecnología *Open Source* en los últimos dos años. Las amenazas de las vulnerabilidades por *Open Source* afectan a las nuevas aplicaciones que las empresas están construyendo, las aplicaciones *legacy* críticas y el software ofrecido por los proveedores.

La reutilización masiva de componentes y bibliotecas *Open Source* ha acelerado drásticamente el ciclo de desarrollo y la capacidad de ofrecer funcionalidad de acuerdo con las expectativas del cliente. Pero la contraparte de esta ganancia ha sido una pérdida de control sobre el origen del código que entra en un sistema de producción. Esta cadena de dependencias expone a las organizaciones y sus clientes a vulnerabilidades introducidas por cambios que están fuera de su control directo.

El número de ataques que utilizan el ecosistema *Open Source* como vector de propagación que llegan a las cadenas de suministro de software aumentó en un 650% entre 2020 y 2021. La Agencia Europea de Ciberseguridad (ENISA) predijo que los ataques a la cadena de suministro se cuadruplicarán para 2022. Otros expertos han pronosticado un incremento mayor.

Estas vulnerabilidades preexistentes pueden permitir un ataque dirigido a los componentes menos confiables de la cadena de suministro de un sistema. Esto puede desencadenar una falla que crea una reacción en cadena que se propaga a una red de proveedores y luego se propaga a través de Internet a

muchos otros sistemas interconectados. Estos ataques están dirigidos al código fuente de los componentes de estos sistemas de software.

El mejor ejemplo en los últimos dos años fue el ataque de SolarWinds, donde una actualización de software defectuosa, ocultando un virus devastador, llegó a hasta 18,000 clientes, incluidas compañías de alto perfil e instituciones gubernamentales en todo el mundo. En el caso de la seguridad del software, ha habido un aumento del 430% en los ataques SSC. En el reciente ataque de SolarWinds, una simple entrega de actualización de software del cliente incluyó un virus devastador. Esta actualización infectada llegó a 425 de las compañías Fortune 500, que incluían compañías de telecomunicaciones, firmas de contabilidad, gobierno e instituciones académicas.

Esto fue seguido pronto por Log4Shell, que explotó una vulnerabilidad dentro del utilitario de bitácoras Apache Log4j. El gran impacto en este caso radica en el uso generalizado de esta biblioteca Java y la posibilidad de que más atacantes tengan código remoto cargado y ejecutado por el *logger*. Hay un enorme daño potencial (es decir, costos), como sucedió en Log4j, SolarWinds, Mimecast, Ledger, Kaseya, Ethereum y SITA.

Estudios relevantes muestran el alcance de este problema

La base de datos Synopsys Black Duck Audit representa la actividad *Open Source* de más de 20.000 fuentes en todo el mundo. Su informe de 2020 (el quinto de su serie) describió su estudio de 2019 de los hallazgos de auditoría de 1,253 bases de código comerciales en 17 industrias. Por base de código se refieren al código fuente y las bibliotecas que subyacen a una aplicación, servicio o biblioteca. Sus hallazgos de 2020 incluyeron lo siguiente:

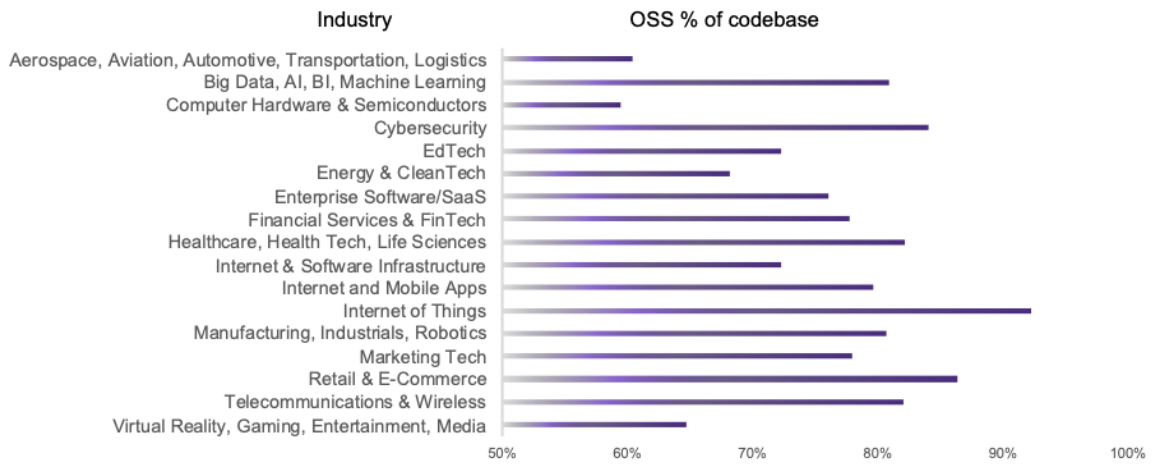
- El 82% de los componentes *Open Source* encontrados estaban desactualizados (es decir, sin parches o no tenían el soporte adecuado)
- El 99% de las bases de código auditadas contenían componentes *Open Source*
- *Open Source* constituyó el 70% de las bases de código auditadas (se duplicó en 5 años)
- El 75% de las bases de código contenían vulnerabilidades (frente al 60% en 2018) y el 49% contenía alto riesgo de vulnerabilidades (por ejemplo, Heartbleed)
- Se identificó un promedio de 82 vulnerabilidades por base de código
- Los lenguajes más frecuentes encontrados fueron: JavaScript (74%), C++ (57%), shell (54%), C (50%), Python (46%), Java (40%), TypeScript (36%), C# (36%), Perl (30%), Ruby (25%)
- Los 10 principales componentes *Open Source* encontrados (en orden de ocurrencia) fueron: jQuery, Bootstrap, Font Awesome, Lodash, jQuery UI, Underscore-stay, Inherits, isArray, Visionmedia y Minimatch

Existen datos más recientes sobre el alcance del problema *Open Source* en su estudio de 2021. Su informe de 2022 (el sexto de su serie) describió su estudio de 2021 sobre los hallazgos de auditoría de 2.409 bases de código comerciales en 17 industrias. Demostraron que el *Open Source* sigue siendo predominante y omnipresente.

Figura 5-1 Resultados de la encuesta de componentes OSS de Synopsys

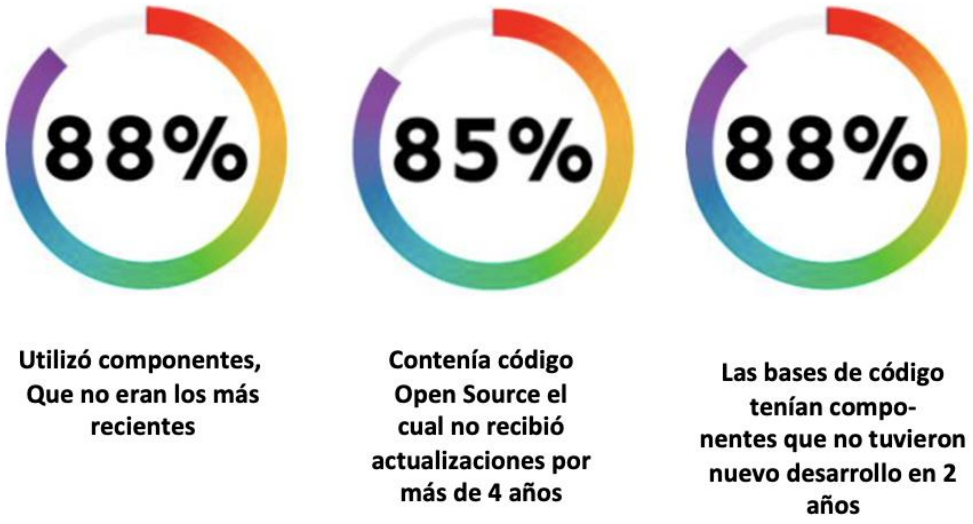
Cuando examinaron el porcentaje de las bases de código (eje X a continuación) que era *Open Source* por industria, pudieron mostrar lo siguiente:

Figura 5-2 Resultados de la encuesta de componentes Open Source de Synopsys – por industria



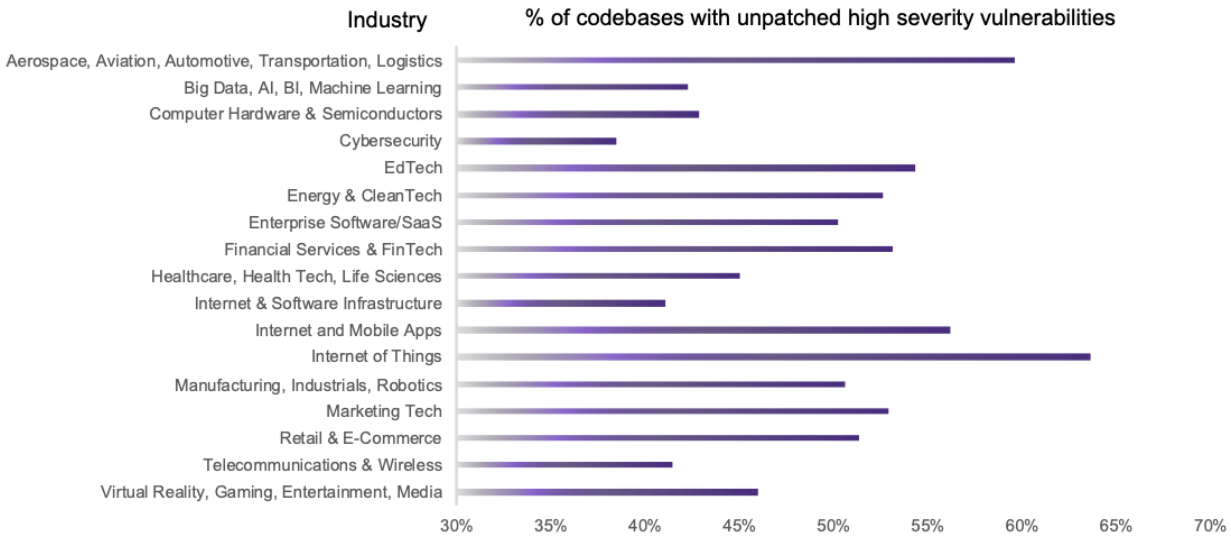
Pudieron mostrar hasta qué punto las organizaciones todavía están luchando para rastrear y administrar su *Open Source*.

Figura 5-3 Resultados de la encuesta de componentes Open Source de Synopsys – sin mantenimiento



Pudieron mostrar qué porcentaje de las bases de código contenían vulnerabilidades de alta gravedad y sin parches (por industria)

FIGURA 5-4 RESULTADOS DE LA ENCUESTA DE COMPONENTES OPEN SOURCE DE SYNOPSYS: ERRORES DE ALTA GRAVEDAD SIN PARCHEAR



Otro informe de la encuesta de Synopsys exploró las estrategias que las organizaciones de todo el mundo están utilizando para abordar la gestión de vulnerabilidades de usar *Open Source*, así como el creciente problema de los componentes *Open Source* obsoletos o abandonados en el código comercial.

Un informe basado en escaneos de clientes de Snyk del 1 de enero al 30 de septiembre de este año (sesgado a favor de los ecosistemas Java) encontró que las 10 vulnerabilidades críticas y altas más frecuentes en *Open Source* fueron:

1. Denegación de servicio (DoS)
2. Ejecución remota de código (RCE)
3. Deserialización de datos que no son de confianza
4. Inyección SQL
5. prototipo de contaminación
6. Archivo temporal inseguro
7. Directorio/ruta transversa
8. Escalada de privilegios
9. Denegación de servicio de expresión regular (ReDoS)
10. Desreferencia de puntero NULL

Otra fuente útil de información es la Base de Datos Nacional de Vulnerabilidades (NVD) del NIST, que enumera las vulnerabilidades conocidas de los principales proveedores de software comercial, tales como: Oracle, Microsoft, IBM y Adobe. Esos cuatro representan casi el 17% del total de vulnerabilidades, para todos los productos y versiones combinados.

En 2019 se realizó un análisis del NVD y se publicaron los siguientes resultados de vulnerabilidades por proveedor y por tipos de debilidad (gráficos proporcionados por CAST).

Figura 5-5 Resultados de la encuesta de vulnerabilidad del NIST – por proveedor

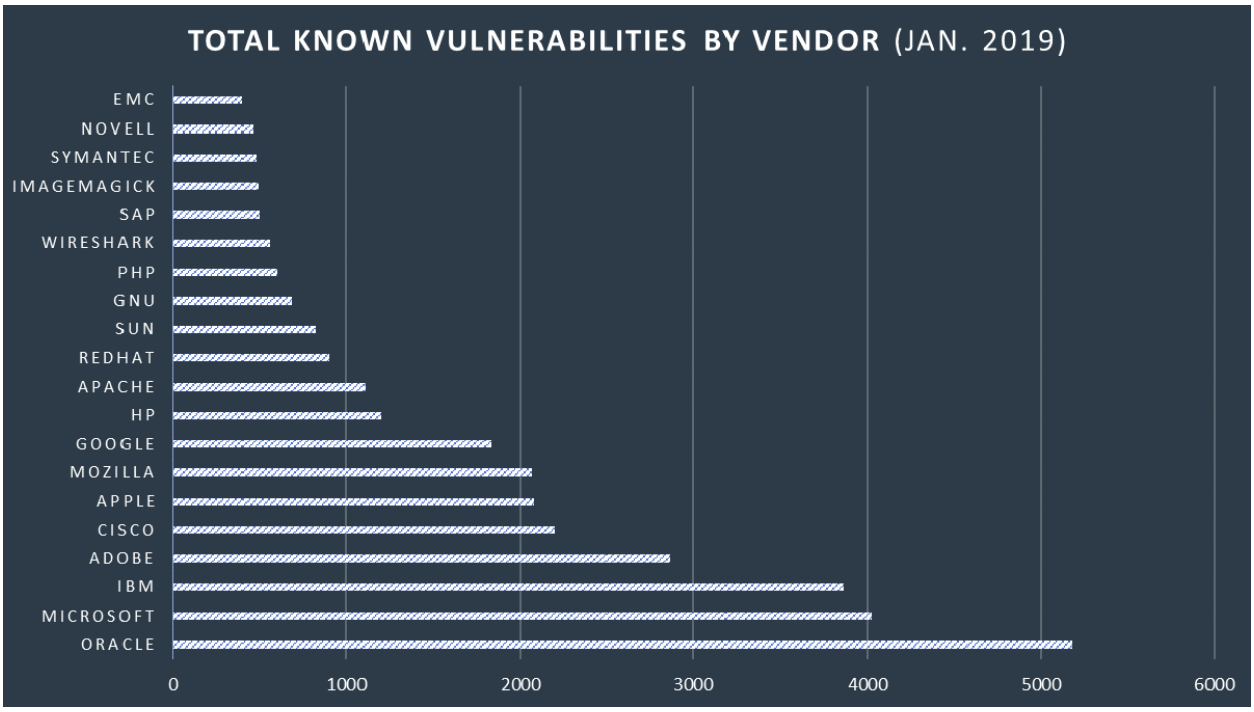
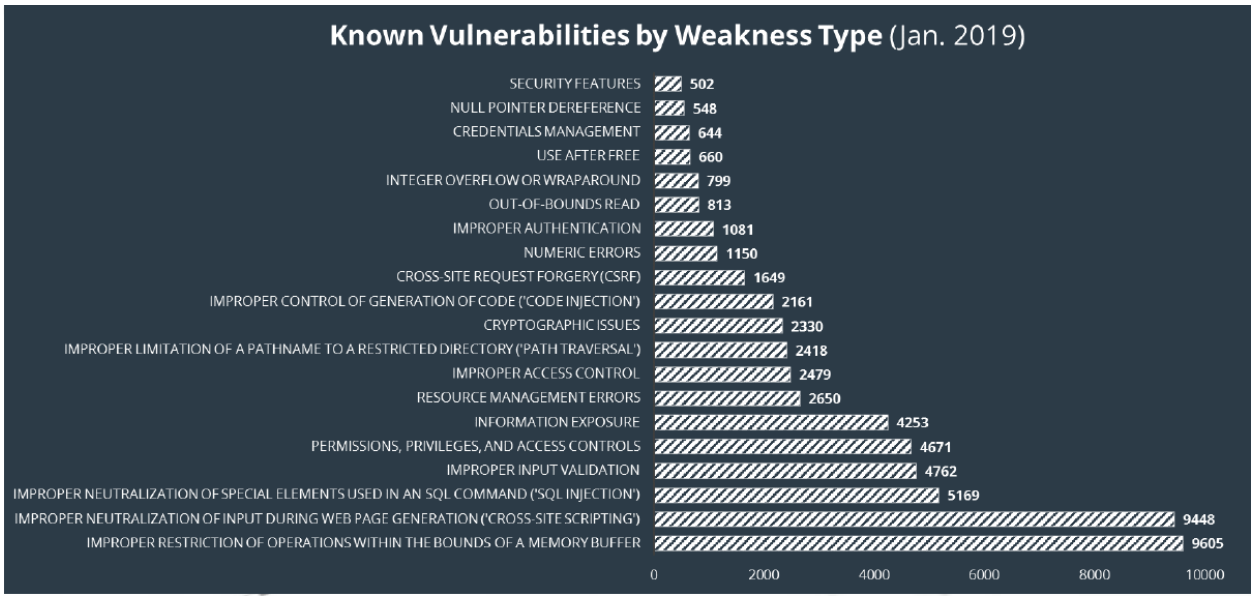


Figura 5-6 Resultados de la encuesta de vulnerabilidad del NIST – por debilidad



Un informe similar más reciente no se ha publicado desde 2019, pero sería más valioso si se hiciera hoy.

Las mejores prácticas recomendadas para *Open Source*

Las plataformas de software *Open Source* populares en la actualidad incluyen GitHub, Fat Free CRM, InfluxDB, D3.js, R, TensorFlow, Keras, Serverless, Apache Airflow, Activiti, PrestaShop y OpenCart. La calidad de estos componentes de software es empíricamente desconocida y, por lo tanto, podría introducir fallas que comprometan el éxito.

Cada lenguaje de programación tiene fallas estructurales únicas que pueden llevar a un desarrollador a crear una falla. Por ejemplo, los lenguajes de bajo nivel como Assembly, C o C ++ son vulnerables al desbordamiento de búfer que los hackers informáticos pueden explotar para escribir malware en la memoria adyacente una vez que la capacidad del búfer está llena. Otra vulnerabilidad común que se encuentra en lenguajes como SQL, JavaScript y PHP es la inyección de código, donde los hackers informáticos se aprovechan de fallas en el procesamiento de datos que hacen que la entrada del usuario se interprete como comandos del sistema o incluya scripts maliciosos en los archivos cargados. Un buen recurso para identificar problemas específicos del lenguaje es el informe del marco de errores de software, que incluye una jerarquía taxonómica de debilidades que se aplica a todos los idiomas. Contiene sugerencias específicas para evitar debilidades que surgen de construcciones que están especificadas de manera incompleta, exhiben un comportamiento indefinido, dependen de la implementación o son difíciles de usar correctamente.

Algunos consejos prácticos para las organizaciones incluyen:

- Gestione la cadena de suministro de software como gestiona cualquier otro riesgo corporativo crítico.

- La próxima vulnerabilidad similar a Log4J es inevitable. Usted debe estar listo. Organice un equipo de respuesta a incidentes de software (SIRT) para proteger la cadena de suministro de software.
- No te olvides de COTS. Muchos proveedores de software independientes (ISV) utilizan software *Open Source* libremente. Deben escanear los archivos binarios y las dependencias de compilación de software.
- Ayuda a saber dónde se encuentra su mayor exposición. Haga un inventario de todo para saber lo que tiene, y entender la composición de sus aplicaciones y la omnipresencia de los componentes *Open Source* en toda su cartera de aplicaciones. Establezca y mantenga un inventario de software o una lista de materiales de software (también conocida como SBOM).
- Analice continuamente la cadena de suministro de software integrando escaneos de código fuente en sus procesos de integración continua/implementación continua de DevQualOps. Mitigue rápidamente las vulnerabilidades conocidas para reducir el tiempo de exposición. Supervise continuamente los componentes de software contra bases de datos de vulnerabilidades conocidas.
- Cree tanto aseguramiento para el código incluido (es decir, software *Open Source*, bibliotecas y paquetes) como para el código que desarrolla de forma nativa.

6. DEUDA TECNICA (DT)

La DT se acumula cuando los tomadores de decisiones buscan una solución a corto plazo para un problema de desarrollo de software, en lugar de una solución más exhaustiva a largo plazo, y esto viene con costos sustanciales, inicialmente ocultos, que las organizaciones deben pagar más tarde.

La DT también es una medida del peso de una empresa que se deriva del envejecimiento y los sistemas de TI inflexibles. En una encuesta de McKinsey, el 87% de los CIO globales dijeron que la complejidad de sus sistemas existentes les impide invertir en la próxima generación de servicios. Los CIO globales dicen que su DT total está entre el 20% y el 40% del valor total de su "patrimonio tecnológico" antes de la depreciación. Software AG dice que el 78% de las organizaciones encuestadas han acumulado más DT en el último año que en años anteriores, pero solo el 42% de las empresas sienten que tienen la capacidad de evaluar toda su DT.

Las señales de que una organización está sobrecargada con DT incluyen:

- Una acumulación de solicitudes de proyectos de las unidades de negocio.
- Se está dando trabajo a contratistas y consultores para reparar o mantener los sistemas existentes.
- Un aumento en los casos de soporte sobre la funcionalidad principal que se ve afectada.
- El departamento de TI ha decidido no actualizar el software que la empresa sigue utilizando.
- Cuando una solicitud relativamente simple de modificación se convierte en un proyecto importante.
- La cantidad de deuda que debe ser atendida limita las opciones en el uso del presupuesto de TI
- La cantidad de trabajo no planificado crece notablemente.

Hay 2 partes para la DT:

- *El Principal* se refiere al costo de refactorizar / modificar artefactos de software para que alcancen un nivel deseado de mantenibilidad y capacidad de evolución.
- *El interés* es el esfuerzo adicional que los desarrolladores gastarán al realizar esos cambios debido a la existencia de DT, que se acumula con el tiempo a medida que el software se vuelve más frágil. Cada minuto gastado en un código no del todo correcto agrega intereses sobre la deuda.

La DT es el resultado de una construcción subóptima que es conveniente a corto plazo, pero establece un contexto técnico que puede hacer que un cambio futuro sea más costoso o incluso imposible. Gran parte de la deuda actual fue creada por técnicas de desarrollo "rápidas y sucias" (por ejemplo, ágil sin disciplina de ingeniería de software). En cualquier caso, la totalidad o parte de esta deuda puede necesitar reembolso. Cuando pagar o si se debe pagar del todo implica compensaciones difíciles.

Hay muchos tipos de DT, como requisitos, arquitectura, código, pruebas y operativo. La DT se puede inyectar en cualquier etapa del desarrollo de software, extendiéndose a través de otras fases y partes del sistema y causando varios problemas. Y tal como hemos visto en el costo del software de baja calidad, prevenir la DT o eliminarla temprano, es la estrategia más rentable a largo plazo.

La DT (como el costo del software de baja calidad) es importante porque ayuda a facilitar el discurso entre los ingenieros y la gerencia sobre cómo invertir mejor los recursos limitados en mantenimiento correctivo y mejora del código en lugar de agregar nuevas características y funcionalidades. Se alcanza el punto de inflexión, por ejemplo, cuando el costo de las nuevas características, correcciones de errores

y mantenimiento exceden el presupuesto del proyecto, lo que hace que se alcance un estado de bancarrota técnica.

El costo aproximado de la DT en el 2022

Según Stripe, el número de horas que un desarrollador promedio en una empresa dedica a abordar "DT" es de 13.5 de 41.1, o 33%. Un estudio escandinavo de 2019 reveló que los desarrolladores malgastan, en promedio, el 23% de su tiempo debido a la DT.

En nuestro informe de 2020, estimamos que el principio de DT en los Estados Unidos era de ~ $ 1.3 billones, que aumentaría a **$ 1.52 billones en 2022** debido únicamente a la inflación. Esta cifra es aproximadamente igual al total de dólares gastados en toda la base laboral de TI de EE. UU. en 2022. Todavía no tenemos buenas estimaciones sobre el interés acumulado. Sin embargo, reconocemos que la DT es un gran problema, que empeorará mucho si no hacemos nada en absoluto. El potencial de administrar la DT se ve en la investigación de Stepsize que reveló que las organizaciones que administran activamente la DT obtendrán al menos un 50% más de velocidad en la entrega.

Progreso en la medición de DT de SOTWARE

Uno de los principales problemas en el tratamiento de la DT ha sido la falta de una forma de medir esa deuda. Para ayudar a superar ese problema, CISQ / OMG lideró el desarrollo de un estándar de medición de la DT automatizado (ATD), que actualmente se está actualizando con una nueva versión prevista para 2023.

El estándar ATD estima el esfuerzo para corregir todas las instancias de las debilidades de software incluidas en el estándar ISO / IEC 5055: 2021 Medidas automatizadas de calidad del código fuente que permanecen en el código de una aplicación de software en el lanzamiento. Esta estimación se puede utilizar para predecir los costos futuros de mantenimiento correctivo. Esta medida se calcula mediante herramientas de análisis estático.

El costo de solucionar problemas estructurales de calidad constituye el principal de la deuda, mientras que las ineficiencias que causan, como un mayor esfuerzo de mantenimiento o recursos informáticos excesivos, representan intereses sobre la deuda. La medida expresa el costo de la calidad del software en términos que una empresa puede entender al estimar los costos futuros de mantenimiento correctivo para remediar defectos estructurales en el código.

La CISQ encuestó a desarrolladores en varias organizaciones para estimar cuánto tiempo tomaría corregir cada una de las debilidades en el código bien construido. Las estimaciones proporcionaron valores predeterminados para el esfuerzo para corregir cada debilidad. Para calcular la DT, ajustamos el valor predeterminado para cada aparición de una debilidad específica por factores que afectan a la dificultad de corregirla como la complejidad del componente, su exposición al resto del sistema, etc. Los esfuerzos ajustados para cada ocurrencia se suman para producir un esfuerzo total de remediación para esa debilidad. El esfuerzo total de remediación para las debilidades en una característica de calidad se suma para crear un esfuerzo de remediación para esa característica. Finalmente, los esfuerzos de remediación para las cuatro características de calidad (Confiabilidad, Seguridad, Eficiencia del Rendimiento y Mantenibilidad) se suman para producir la medida de DT.

¿Dónde más buscar soluciones automatizadas para encontrar y arreglar DT?

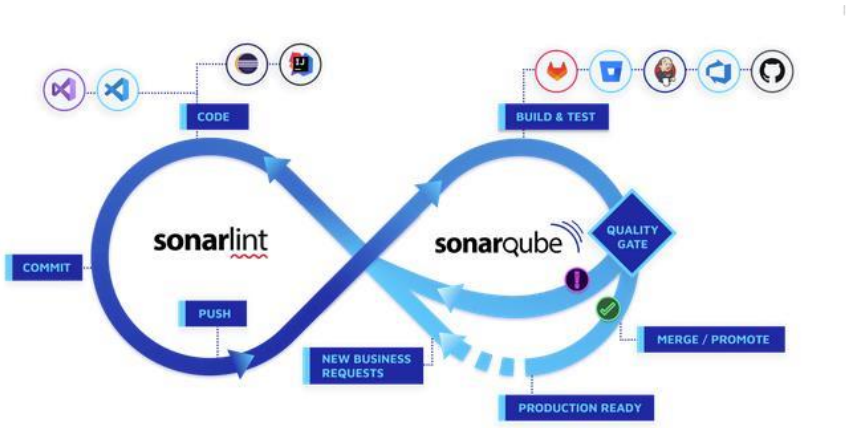
En la última década, la DT de R&D avanzó de un concepto hacia prácticas y herramientas de ingeniería específicas para identificar, monitorear y remediar problemas de DT. Creemos que pasarán varios años más antes de que el manejo de DT crezca desde la adolescencia hasta la edad adulta.

No obstante, algunas herramientas útiles han surgido como analizadores de deuda de código estático. Por ejemplo: SonarQube, CAST, Synopsys y NDepend. Faltan estándares en estas herramientas para: métricas de DT, índices, modelos de calidad, reglas de análisis estático, modelos de remediación de DT y definiciones de los diversos conceptos de DT. La medición de DT será necesaria para estimar el capital y el interés para priorizar la gestión de DT en la práctica. Tecnologías como CAST MRI detectan no solo debilidades a nivel de unidad de código, sino también debilidades en la arquitectura, como llamadas de salto de capa.

Herramientas como SonarQube / SonarLint, han desarrollado complementos para estimar un principal de la DT basado en un modelo de "código olor" y violación de reglas. La desventaja de estas herramientas es que crean la impresión de que la DT consiste en deficiencias de código de bajo nivel y nada más, en lugar de la arquitectura mucho más costosa y las características de calidad dispersas.

Muchas herramientas modernas de análisis estático admiten nuestro modelo DevQualOps, y también admiten muchos idiomas y las herramientas de administración de configuración más populares.

Figura 6-1 Modelo de DevOps de Sonar



En los próximos años, podemos esperar ver más trabajos investigando el impacto que la DT tiene en las cualidades internas, como fallas, confiabilidad y mantenimiento del código, tal vez con un mayor soporte para lenguajes no orientados a objetos. Al presentar el argumento para pagar la DT, el interés debe incluir no solo la mantenibilidad sino también otras formas, como gastos operativos, costos de oportunidad, seguridad, problemas de experiencia del usuario y valor del producto.

El consejo actual para los ingenieros de software cuando se trata de DT es:

- prestar especial atención a cómo se crean las dependencias internas, ya que existe un delicado equilibrio entre la variabilidad y el número de dependencias por módulo: si son demasiadas, y se enredan, haciendo que el sistema sea difícil de modificar y muy poco, y el sistema es difícil de Modificar porque se reutilizan menos módulos "tal cual" (un gráfico de

dependencias en forma de árbol), lo que resulta en múltiples módulos que implementan una funcionalidad similar (y aplicar el mismo cambio a todos ellos es duplicativo).

- Equilibre cuidadosamente cómo se crean estas dependencias diseñando reglas arquitectónicas claras que eviten la creación de dependencias no deseadas que terminen generando malos olores. (por ejemplo, Arcan)
- mantenerse al tanto de las nuevas herramientas de análisis de DT y adoptar y utilizar en consecuencia para el análisis de refactorización.
- Refactorizar continuamente, es decir, realizar los cambios en la estructura interna de su software para que sea más fácil de entender y más barato de modificar en el futuro sin cambiar su comportamiento observable.

En An Empirical Study of Refactoring Challenges and Benefits at Microsoft, los desarrolladores informaron de las siguientes ganancias de refactorización:

- Mantenimiento mejorado (30%)
- Legibilidad mejorada (43%)
- Menos errores (27%)
- Rendimiento mejorado (12%)
- Reducción del tamaño del código (12%)
- Reducción del código duplicado (18%)
- Capacidad de prueba mejorada (12%)
- Extensibilidad mejorada y más fácil de agregar nuevas características (27%)
- Modularidad mejorada (19%)
- Reducción del tiempo de comercialización (5%)

7. ESTÁNDARES DE CALIDAD DEL SOFTWARE

En nuestro informe de 2018 proporcionamos la siguiente discusión sobre la definición de calidad del software.

"Calidad" puede significar cosas diferentes para diferentes personas. El concepto y el vocabulario de la calidad son difíciles de alcanzar. El significado difiere dependiendo de las circunstancias y percepciones. La definición del diccionario de Calidad (en general)

1. el estándar de algo medido contra otras cosas de un tipo similar; el grado de excelencia de algo.
2. Un atributo distintivo o característica poseída por algo.

La norma ISO 8402 define la calidad como "la totalidad de las características y características de un producto o servicio que tiene que ver con su capacidad para satisfacer necesidades declaradas o implícitas [ahora]". La calidad es un concepto diferente cuando se centra en un producto de software tangible frente a la percepción de un servicio de calidad habilitado por el software. Por ejemplo, ISO/IEC 25010 define un modelo de las características de calidad de un producto de software o sistema, mientras que ISO/IEC 25019 define las características de calidad en uso experimentadas al usar dichos productos. El significado de calidad es, por lo tanto, basado en el tiempo o situacional.

Los consumidores ahora ven la calidad como una medida fundamental de su percepción / experiencia total con un producto o servicio, así como de la empresa, la red de entrega y mantenimiento que lo proporciona y respalda, una especie de métrica unificada de "calidad-valor".

Si bien lo anterior puede ser suficiente para discusiones generales, existe la necesidad de que cada proyecto tenga su propia definición más específica. Por lo tanto, la calidad del software se describe con mayor precisión como una combinación de los siguientes aspectos:

1. Conformidad con los requisitos

- Los requisitos están claramente establecidos y el producto debe ajustarse a ellos
- Cualquier desviación de los requisitos se considera un defecto
- Un producto de buena calidad contiene menos defectos

2. Aptitud para el uso/propósito

- Ajuste a las expectativas del usuario: satisfaga las necesidades del usuario
- Un producto de buena calidad proporciona una mejor satisfacción del usuario

3. Cumplimiento de estándares
   - En muchas industrias y organizaciones, ciertas normas externas e internas se deben cumplir.
   - Un producto de buena calidad cumple con los estándares de calidad requeridos (ISO / IEC 25010 e ISO / IEC 5055) y el proceso utilizado para desarrollarlo (CMMI, SPICE).

4. Aspectos subyacentes, que incluyen

- Calidad estructural (por ejemplo, complejidad)
- Calidad estética (por ejemplo, apariencia, facilidad de uso, etc.) •

Cada aplicación o dominio empresarial se enfrenta a un conjunto específico de problemas de calidad de software, y la calidad del software debe definirse en consecuencia. Se debe crear una definición formada a partir de los aspectos anteriores y / o estándares aplicables para su organización y para cada proyecto.

En 2018 discutimos la diferencia entre buena y baja calidad de software y concluimos que ***si hubiera una medida simple para el software "bueno", todos lo estaríamos usando y todos lo exigirían.***

Históricamente, varias métricas se han utilizado a menudo como indicadores, generalmente en combinación. Por ejemplo:

- La tendencia de defectos a lo largo del tiempo se usa a menudo para diferenciar: bueno es una curva decreciente, pobre es una curva creciente.
- La cobertura del código de prueba se ha utilizado como sustituto, pero no habla de la calidad de las pruebas en si mismas.
- Complejidad ciclomática, profundidad de herencia, grado de acoplamiento de clase, complejidad estructural, y algunas otras métricas, son indicadores de código por debajo de la media.
- La cantidad de esfuerzo que se necesita para entender lo que hace un fragmento de código es otro indicador.

Lo que ha cambiado desde 2018 es la aparición de estándares ampliamente reconocidos para ayudarnos a lidiar con el espinoso problema de medir la calidad del software.

El surgimiento de estándares para definir la calidad del software

El marco más útil y común que está disponible para ayudar a cada proyecto a definir con mayor precisión sus objetivos de calidad de software es ahora la serie ISO / IEC 25000.

La serie de normas ISO/IEC 25000, conocida como SQuaRE (System and Software Quality Requirements and Evaluation), contiene un marco para evaluar la calidad del producto de software. ISO / IEC 25010 define un conjunto de ocho características de calidad de software, o "-ilidades" del sistema, es decir, seguridad, confiabilidad y mantenibilidad. ISO/IEC 25023 describe cómo aplicar las características de calidad para medir la calidad del producto de software.

Sin embargo, las medidas definidas en 25023 evalúan en gran medida la calidad a nivel de comportamiento en lugar de a nivel de problemas de calidad específicos encontrados en el código fuente.

Para llenar ese vacío, CISQ lideró la creación de ISO 5055 que definió las medidas de nivel de código fuente para cuatro de las 8 características de calidad: confiabilidad, eficiencia de rendimiento, seguridad y mantenibilidad . Estos ahora se están automatizando en el desarrollo de nuevas herramientas de análisis de calidad de código. Las medidas para estas cuatro características de calidad ahora se han adoptado como estándares internacionales en ISO / IEC 5055: 2021.

Cada medida de calidad del código ISO 5055 para confiabilidad, eficiencia del rendimiento, seguridad y mantenibilidad se compone de un conjunto seleccionado de debilidades (CWE) de la taxonomía Common Weakness Enumeration (CWE). El CWE es un buen punto de referencia para desarrolladores y herramientas Y codifica más de **800** debilidades conocidas del software. Cada CWE es un patrón de código conocido que es un potencial punto de falla encontrado en muchos sistemas existentes.

Los equipos de desarrollo pueden usar los estándares de calidad de código anteriores para evaluar la calidad estructural del software antes de cada lanzamiento, evitando así que se entreguen fallas peligrosas en entornos operativos, donde serán órdenes de magnitud más costosos de encontrar y corregir.

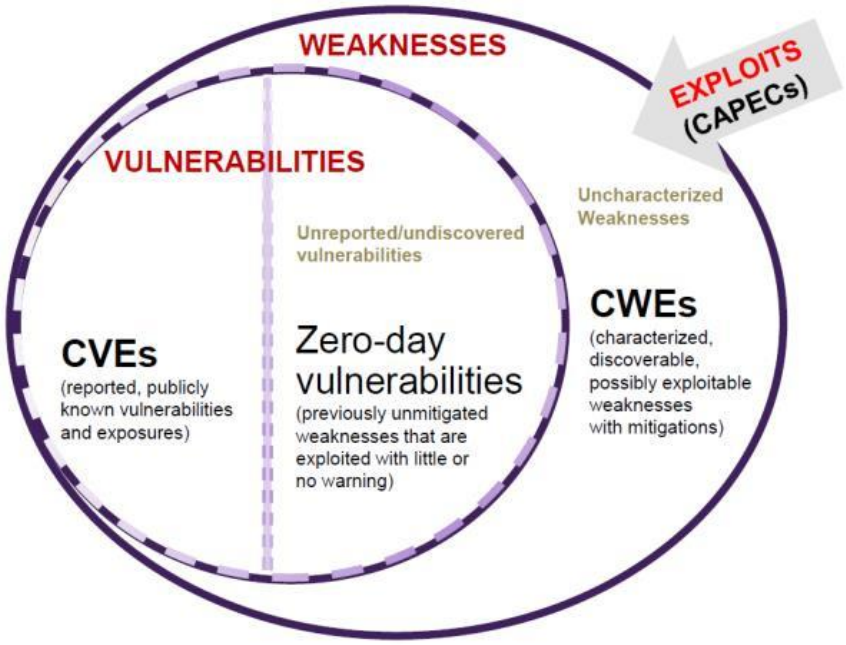
Aunque esta no es la respuesta completa para garantizar una producción de software de alta calidad, es un gran paso adelante desde nuestro informe original.

A menudo se considera "vulnerabilidades" y "debilidades" en el software como palabras intercambiables. Si bien están relacionados, son diferentes. Estos términos están definidos en las normas internacionales. Las definiciones normalizadas de debilidades y vulnerabilidades forman parte de la serie UIT-T CYBEX 1500 (CVE UIT-T X.1520, CWE ITU-T X.1524, CAPEC ITU-T X.1544) como se describe a continuación.

- **Debilidad:** Error o condición de falla en la arquitectura, el diseño, el código o el proceso que, si no se aborda, podría contribuir a que una capacidad cibernética sea vulnerable a la explotación; representa vectores de origen potenciales para explotaciones de día cero.
- **Vulnerabilidad: error en el** software que puede ser utilizado directamente por un hacker para obtener acceso a un sistema o red, o **exposición:** problema de configuración o un error en la lógica que permite el acceso no autorizado o la explotación.
- **Explotar:** Acción que aprovecha la(s) debilidad(es) para lograr un impacto técnico negativo.

La existencia de una activación diseñada para aprovechar una debilidad (o múltiples debilidades) y lograr un impacto técnico negativo es lo que hace que una debilidad sea una vulnerabilidad. Las debilidades se enumeran en el repositorio de enumeración de debilidades comunes (CWE) (cwe.mitre.org). Las vulnerabilidades (CVE) se publican tanto en el diccionario de vulnerabilidades y exposiciones comunes (CVE.mitre.org) como en la base de datos nacional de vulnerabilidades (nvd.nist.gov). Los métodos que utilizan los hackers para sacar provecho de las debilidades y vulnerabilidades se detallan en la enumeración y clasificación de patrones de ataque comunes (capec.mitre.org). La siguiente figura resume las relaciones entre estos conceptos.

Figura 7-1: Vulnerabilidades, debilidades y activaciones



Debido a que el número, el tamaño y la complejidad de los sistemas de software aumentan cada día, también lo hace el número de debilidades y vulnerabilidades. Los atacantes cibernéticos utilizan vulnerabilidades y debilidades conocidas para sacar provecho de los sistemas. La eliminación de las vulnerabilidades conocidas (CVE) y las debilidades más atroces (CWE) reduciría sustancialmente el impacto de los ataques cibernéticos y las fugas de datos. Vea la última versión de CWE, los 25 CWE principales y 138 CWE en los estándares ISO/IEC 5055 Automated Source Code Quality Measure desarrollados por CISQ. Si todo el software nuevo (111 mil millones de LOC por año en todo el mundo) se creara sin estas vulnerabilidades conocidas y debilidades a las que se les saca ventaja, el costo del software de baja calidad se desplomaría.

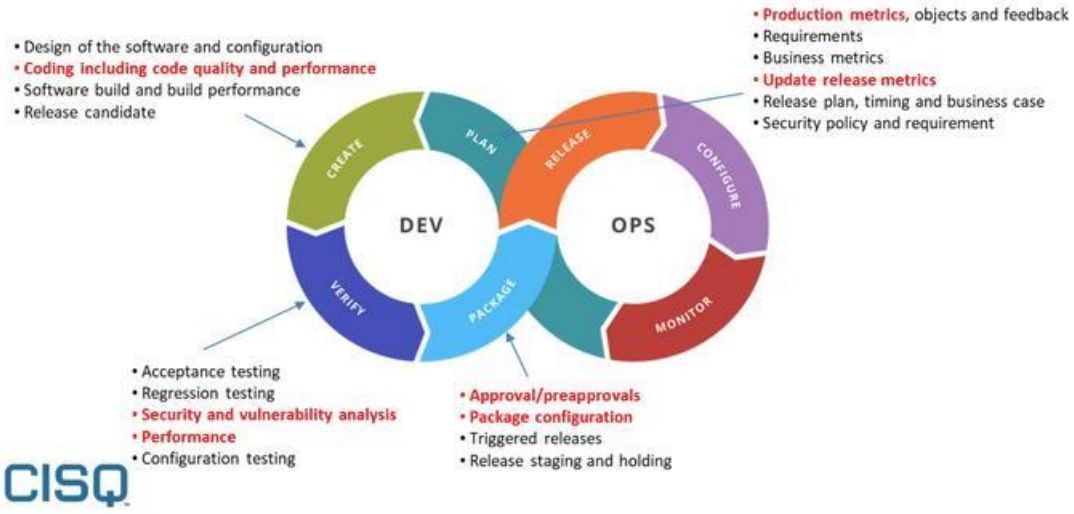
Como lo indica el crecimiento de las violaciones de datos, la protección de datos y la privacidad están en lo más alto de prioridades de las organizaciones. Muchas organizaciones se someterán a evaluaciones de procesos asociadas con regulaciones para proteger los datos, incluido el Reglamento General de Protección de Datos (GDPR), la Ley de Privacidad del Consumidor de California (CCPA), la Ley de Portabilidad y Responsabilidad del Seguro Médico (HIPAA) y la Certificación del Modelo de Madurez de Ciberseguridad (CMMC).

El escaneo del código que se ejecutará o se está ejecutando en activos conectados a la red empresarial que procesan o transmiten datos determinaría si los sistemas o dispositivos permiten la fuga de datos o carecen de protecciones adecuadas para mitigar el acceso no autorizado para leer o modificar datos. Si es así, dicho análisis revelaría si los controles de protección de datos / privacidad asociados con la evaluación del proceso se implementaron inadecuadamente.

Para abordar esto, CISQ desarrolló una Medida Automatizada de Protección de Datos de Código Fuente (ASCDPM) que se puede utilizar en pruebas de seguridad de aplicaciones y desarrollo de software para proporcionar verificación independiente de procesos que revelan vectores de origen para fuga de datos o corrupción de datos; proporcionando indicadores para el incumplimiento de las respectivas directrices de protección de datos y privacidad. Según el CWE, los elementos de medida (debilidades que violan las reglas de calidad del software) que componen el ASCDPM CISQ contienen 36 debilidades principales y 53 debilidades contribuyentes.

CISQ está abordando directamente el modelo DevQualOps mediante su trabajo en curso, como se muestra en la figura a continuación.

Figura 7-2 Modelo CISQ DevQualOps



8. COMPRENSIÓN, DETECCIÓN Y CORRECCIÓN DE DEFICIENCIAS

Como pudimos mostrar en nuestro informe de 2020, todas las categorías principales de costos de baja calidad de software están respaldadas por el costo de encontrar y corregir las deficiencias que existen o se inyectan en los sistemas de software.

Idealmente, una fábrica de desarrollo de software debería reducir los errores tanto como sea posible antes del envío, pero en la mayoría de las situaciones, se convierte en una compensación. Para ser competitiva, una organización puede querer entregar características o productos a los clientes más rápidamente a un costo mínimo. El problema siempre ha sido que la calidad sufre en esta compensación porque ha sido mucho más difícil de medir que el tiempo y el costo.

Lo más importante es que sabemos que los errores no se pueden prevenir por completo: por ejemplo, no puede probar cada escenario de usuario o todas las rutas de ejecución en el código.

La evidencia empírica sugiere que las organizaciones que incorporan análisis de calidad automatizados y prácticas de DevQualOps observarán una mejor calidad a través del descubrimiento mejorado de deficiencias mediante la integración de herramientas de análisis y monitoreo en sus entornos de desarrollo e implementación.

Prácticamente, un porcentaje significativo del costo de un proyecto de software hoy en día no se gasta en la actividad creativa de la construcción de software, sino más bien en la actividad correctiva de depuración y corrección de errores. Sin embargo, la tarea de depuración es inherentemente complicada. La mayoría de los sistemas carecen de especificaciones formales que describan el comportamiento previsto del programa. Sin una documentación formal o sistemática del comportamiento correcto, la definición de un "error" o "error" a menudo reside en la mente del ingeniero de software o en las expectativas a veces nebulosas del usuario sobre el comportamiento del programa.

Los errores de desarrollo de software de todo tipo, en código fuente, configuraciones, pruebas u otros artefactos, son un problema amplio y costoso. Los desarrolladores consumen una proporción significativa de tiempo y esfuerzo de ingeniería para comprender, encontrar y corregir errores en su código, las empresas pierden cuota de mercado cuando las vulnerabilidades en el software que venden afectan a los clientes, y la productividad general se ve afectada por el software que no funciona según lo previsto o es propenso a vulnerabilidades.

El costo de encontrar y corregir deficiencias es el elemento de gasto más grande en el ciclo de vida de desarrollo de software. Durante una esperanza de vida de 25 años de un gran sistema de software, casi cincuenta centavos de cada dólar se destinarán a encontrar y corregir errores. Los sistemas grandes tienen potenciales de deficiencia mucho más altos que son más difíciles de eliminar que para los sistemas pequeños debido a su tamaño y complejidad. Cuanto antes se encuentren deficiencias en el ciclo de vida del desarrollo, más económica será la entrega general. Un buen recurso es el libro, La economía de la calidad del software.

El informe CAST Crash 2020 nos dio una idea de dónde buscar posibles mejoras en COSTO DEL SOFTWARE DE BAJA CALIDAD. Su último punto de referencia sobre la calidad estructural de las aplicaciones de TI se desarrolló a partir de su base de datos de 2.505 aplicaciones que consta de 1.549 BLOC (miles de millones de líneas de código), distribuidas en 533 organizaciones y 26 países. Las cinco características de calidad de software analizadas en su informe son robustez, seguridad, eficiencia de rendimiento (también conocido como rendimiento), capacidad de cambio y transferibilidad. Estos son cuatro de los ocho de las principales características encontradas en ISO/EIC 25010 modelo estándar

para calidad de producto de software (mutabilidad y transferibilidad son sub-características bajo Mantenibilidad en ISO/IEC 25010).

Los hallazgos de CAST se basaron en el cálculo de las densidades de debilidades críticas en las aplicaciones y revelaron que:

- El tamaño de una aplicación tenía una relación insignificante o nula con su calidad estructural.
- Las densidades de debilidades críticas para la seguridad fueron más altas que las de robustez y Mutabilidad.
- Las densidades más bajas se observaron para la transferibilidad (equivalente a la comprensibilidad).
- El segmento de la industria es de menor importancia que otros factores, pero esto solo se aplica a Java-EE aplicaciones, para las cuales las telecomunicaciones, los ISV de software y la consultoría de TI tenían las densidades más altas de Debilidades de robustez, seguridad y mutabilidad.
- La mayoría de las industrias mostraron una amplia variabilidad en las densidades de debilidad críticas y numerosos valores atípicos extremos de puntuaciones.
- La seguridad fue el área donde las densidades medias de debilidades críticas y la variabilidad de las puntuaciones fueron los más altos-
- Los factores que afectan más los atributos de calidad como la robustez, seguridad o mutabilidad parecen ser más específicos a la aplicación, el equipo de desarrollo y las condiciones específicas en el ambiente de desarrollo.

Algunas recomendaciones del informe fueron:

- Se debe prestar mayor atención a las prácticas de codificación seguras, ya que muchas aplicaciones tenían densidades de debilidades críticas de seguridad que eran inaceptablemente altas. Las puntuaciones de seguridad mostraron una variación más amplia que los de cualquier otra característica de calidad.
- Analizar el código fuente de forma regular antes de su lanzamiento para detectar violaciones de las reglas de calidad que ponen operaciones o costos en riesgo. Las violaciones a nivel del sistema son las más críticas, ya que cuestan mucho más solucionarlas y puede tomar varios ciclos de liberación para eliminarlas por completo.
- Tratar la mejora estructural de la calidad como un proceso iterativo perseguido a lo largo de numerosas versiones para alcanzar los umbrales de calidad óptimos.

Si bien la adopción de estas recomendaciones basadas en la evidencia no puede garantizar una alta calidad estructural, se ha demostrado empíricamente que están asociadas con aplicaciones de menor riesgo.

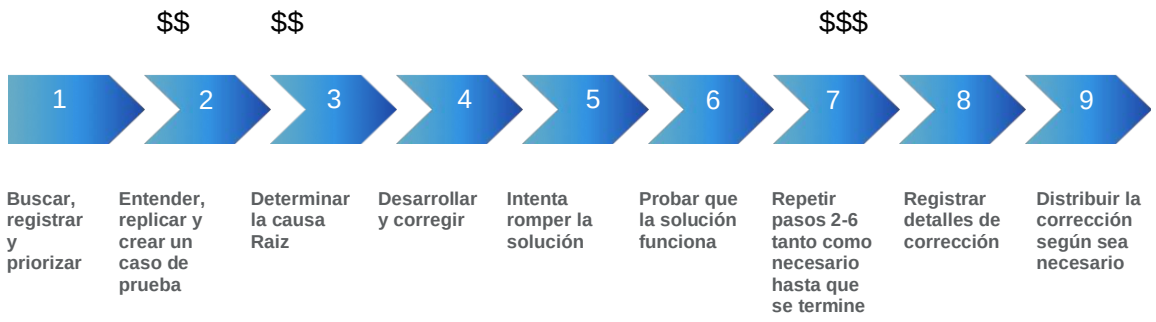
Para comprender mejor los costos involucrados aquí, debemos mirar los procesos involucrados. Primero, dado que existe una diferencia de costo de orden de magnitud entre las deficiencias internas y externas (implementadas), necesitábamos comprender esas categorías:

- Categoría 1 - Deficiencia interna: los costos son costos asociados con deficiencias de software descubiertas antes de que el sistema abandone la organización de desarrollo y se implemente en el entorno operativo. Estas deficiencias ocurren cuando un sistema no cumple con un cierto requisito (o regla crítica de codificación / arquitectura), lo que resulta en desperdicio, desecho y / o retrabajo. Las deficiencias podrían estar en los productos de trabajo del desarrollo, el proceso de desarrollo y / o los componentes si no cumplen con los estándares y requisitos de calidad. Desafortunadamente, muy pocas organizaciones rastrean esta categoría antes del comienzo de las pruebas del sistema.
- Categoría 2 - Deficiencia externa de costos: son costos que ocurren cuando la falla del software para alcanzar los estándares de calidad no se detecta hasta después de que se transfiere a la

operación o al cliente. Se incurre en costos externos de falla / deficiencia durante el uso del cliente. La categoría más grande de costo es el esfuerzo profesional para replicar, encontrar y corregir todas las deficiencias y reevaluaciones de campo para verificar las correcciones.

En segundo lugar, es instructivo observar el proceso de ingeniería de software para comprender, encontrar y corregir deficiencias para que podamos observar dónde se gasta el esfuerzo real. Un modelo simple se muestra en la siguiente figura. Los signos del dólares indican dónde se concentra la mayor parte del esfuerzo / costo.

Figura 8-1: El proceso de comprensión, búsqueda y corrección de deficiencias de software



Este proceso solo crece en importancia a medida que el software evoluciona y se implementa continuamente y a medida que la sociedad se vuelve cada vez más dependiente de los sistemas de software en todos los aspectos de la vida moderna. Una versión de alta madurez de este proceso busca otros lugares en el código donde se podría haber cometido el mismo error para corregir cualquier otra ocurrencia, y luego analiza la causa raíz y la elimina.

Ayuda a saber dónde buscar deficiencias, vulnerabilidades, debilidades, errores, oportunidades de refactorización, etc. Aquí hay algunos consejos prácticos:

1. A nivel de código, ciertos lenguajes son más propensos a errores que otros (por ejemplo, JavaScript: es fácil de aprender, pero también es fácil inyectar errores con él).
2. A nivel de diseño / arquitectura, cuanto más complejos sean los componentes y las interdependencias, más probable es que tenga errores complicados y confusos.
3. Las interfaces de usuario altamente interactivas tienen más probabilidades de tener errores.
4. Las bibliotecas de terceros tienen más probabilidades de tener errores. Si es *Open Source*, al menos puedes ir y cavar en el código. Si es de fuente cerrada, a menudo está a merced del vendedor.
5. Se garantiza que la primera versión de algo nuevo (beta) tendrá muchos errores
6. Errores que se han escapado a la naturaleza que representan situaciones inusuales / imprevistas que Depende del contexto de uso (por ejemplo, datos de usuario dañados) son algunos de los más difíciles de encontrar

Nuevas herramientas para ayudar

Una de las nuevas generaciones de herramientas de *debugging* que ha surgido para ayudar se llama *time-travel debugging*. Los *time-travel debuggers* permiten a los ingenieros de software:

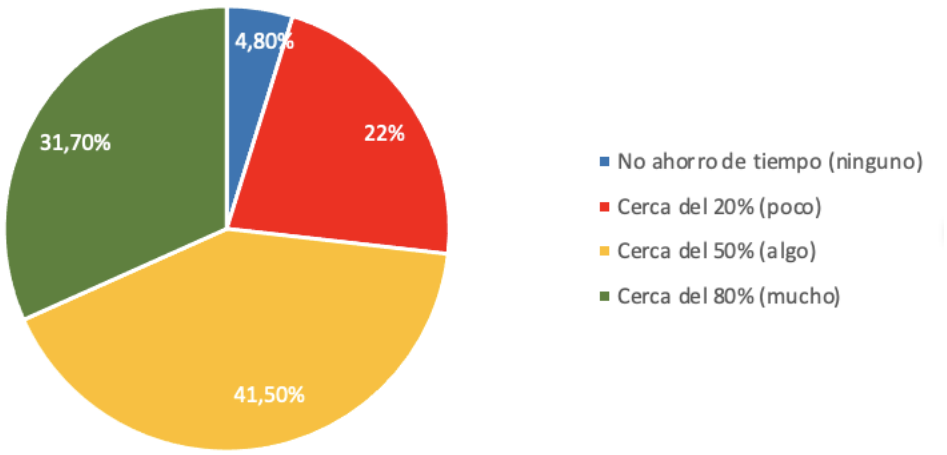
1. Realizar un seguimiento del historial de ejecución de una aplicación e inspeccionar el estado completo de la aplicación en ese momento
2. Ejecute la navegación de rutas en orden directo e inverso, escalonando, ejecutando y utilizando puntos de interrupción, puntos de observación, puntos de captura, etc.
3. Utilice las capacidades de navegación basadas en el contexto
4. Habilitar el registro dinámico y la revisión
5. Reproducir desde un estado fallido
6. Utilice las capacidades de colaboración basadas en anotaciones

Esta capacidad para encontrar y corregir deficiencias más rápidamente se demuestra mejor en los siguientes datos del estudio. En noviembre de 2022, Undo realizó una encuesta preliminar a sus clientes. Las respuestas recibidas fueron predominantemente de las industrias de gestión de datos, redes y automatización de diseño electrónico.

Figura 8-2: Resultados del estudio de Undo sobre time-travel debugging

Cuando utiliza time-travel debugging, cuánto tiempo ahorra en promedio?

41 respuestas



Están surgiendo nuevos programas de cazadores de errores

Mientras tanto, hemos observado la aparición de programas de cazadores de errores que se están utilizando para identificar errores en la naturaleza que necesitan ser corregidos, como se ve en los siguientes dos ejemplos centrados en el problema *Open Source*.

1. Google ha lanzado un nuevo programa que pagará recompensas por los errores encontrados en sus proyectos *Open Source*. Puede ganar hasta $ 31,337 por encontrar un error en el *software Open Source* de Google. Google dice que el Programa de Recompensas de Vulnerabilidad de Software Open Source (OSS VRP) cubre varios códigos de Chrome y Android en las operaciones más amplias de la compañía, lo que ha resultado en más de $ 38 millones pagados a más de 13,000 contribuciones, de un total de 84

países. Además, Google se ha comprometido a invertir $ 10 mil millones para mejorar la ciberseguridad entre sus propios usuarios y consumidores de software de Open Source. Google dice que el *Open Source* VRP se centra en "todas las versiones actualizadas" *Open Source* almacenadas en los espacios de organización GitHub propiedad de Google, como GoogleAPIs y GoogleCloudPlatform, aunque los "premios principales" están reservados para los proyectos más sensibles, que Google establece que son Bazel, Angular, Golang, Protocol buffers y Fuchsia; Una lista que se espera que se expanda después del lanzamiento inicial del programa.

2. La iniciativa SOS.dev 'Secure Open Source Rewards' ayudará a prevenir ataques a la cadena de suministro de software al incentivar a los investigadores a ofrecer actualizaciones de seguridad a proyectos esenciales. Esta nueva iniciativa tiene como objetivo recompensar a los desarrolladores y expertos en seguridad que mejoran la infraestructura crucial utilizando software *Open Source*. Según quienes lo apoyan, la iniciativa de recompensas, que es 'Secure Open Source', cubrirá más terreno que los esquemas de recompensas de errores en este momento. Al alentar a los académicos y desarrolladores a realizar cambios de seguridad, el programa "fortalecería los proyectos vitales Open Source" y ayudaría a proteger contra las amenazas de la cadena de suministro de aplicaciones y software. Hasta $ 10,000 están disponibles por cada error encontrado.

La vanguardia para la reparación automatizada de programas

Las técnicas automatizadas para la detección, mitigación o prevención de errores tienen una larga historia en la investigación de la informática. Los lenguajes de programación y sus sistemas de tipos y compiladores pueden advertir a los programadores cuando cometen ciertos tipos de errores o eliminarlos por completo por diseño. Los análisis estáticos, a veces integrados en entornos de desarrollo integrados o ejecutados en tiempo de confirmación, pueden marcar patrones arquitectónicos o de codificación problemáticos o incluso, cada vez más, encontrar errores semánticos profundos. Las técnicas dinámicas de auto reparación pueden imponer políticas de seguridad u otras políticas de corrección al imponer la integridad del flujo de control, evitar la inyección de código o desinfectar automáticamente las entradas. Por lo tanto, estas técnicas pueden detectar y recuperarse de errores en tiempo de ejecución, sin la intervención del usuario o del desarrollador.

Por el contrario, las técnicas para la reparación automática de software generalmente tienen como objetivo producir cambios (parches) en el código fuente del programa para abordar el error por completo (en lugar de encontrar errores, ayudar a los programadores a evitar errores o ayudar a los sistemas a recuperarse dinámicamente de ellos).

A veces estos objetivos pueden ir de la mano. Por ejemplo, algunas herramientas estáticas de búsqueda de errores proporcionan cada vez más a los desarrolladores punteros o sugerencias para ayudarles a comprender y solucionar el problema subyacente; De hecho, las sugerencias de solución más rápida de las herramientas de búsqueda de errores pueden conducir a una mayor adopción. Del mismo modo, los compiladores hacen cada vez más sugerencias para abordar los errores marcados, y se proponen técnicas de investigación para abordar errores semánticamente más complejos, marcados por técnicas estáticas. Por lo tanto, tales enfoques utilizan un enfoque estático de búsqueda de errores para encontrar un defecto y luego pueden usar la técnica estática para localizar automáticamente el error y validar que un parche propuesto lo aborda (es decir, determinando que el analizador estático ya no marca la deficiencia en cuestión).

Sin embargo, una mayor preponderancia de las técnicas actuales para la reparación automática de programas es de naturaleza dinámica. Es decir, estos métodos utilizan pruebas fallidas o bloqueos del programa para demostrar la existencia de un fallo; El objetivo del proceso de reparación de errores es modificar el código fuente del programa para que las pruebas pasen ahora o el programa ya no se bloquee. Otras pruebas de programas existentes se utilizan normalmente para ayudar al proceso de reparación de programas a evitar romper involuntariamente otro comportamiento deseable, de la misma manera que los conjuntos de pruebas de integración continua ayudan a los programadores humanos a

evitar hacer lo mismo al modificar manualmente sus sistemas. De hecho, algunas técnicas propuestas y actualmente implementadas están dirigidas exactamente a ese caso de uso: reparar un programa con respecto a una prueba de integración continua fallida.

Los éxitos de la reparación automatizada de programas, tal como está el campo hoy, han sido significativos. Las técnicas exitosas varían en términos de si abordan tipos de deficiencias particulares o si pretenden ser más generales para una variedad más amplia de propiedades del programa que se pueden capturar en una prueba de fallo. Ha habido un tremendo progreso en términos de mejorar la generalidad de las técnicas y la escalabilidad con respecto a los programas y los espacios de búsqueda. Las técnicas modernas de investigación de todo tipo han reportado resultados exitosos en programas de cientos de miles a millones de líneas de código. La escalabilidad a grandes espacios de búsqueda (más allá de simplemente a programas grandes) es importante para permitir la reparación de errores complejos de varias partes o programas que son significativamente incorrectos. Cada vez más, tales técnicas están comenzando a penetrar en las mejores prácticas de ingeniería (por ejemplo, Getafix, en Bloomberg).

9. INTELIGENCIA ARTIFICIAL (IA) Y APRENDIZAJE AUTOMÁTICO (MACHINE LEARNING -ML-) EN INGENIERÍA DE SOFTWARE

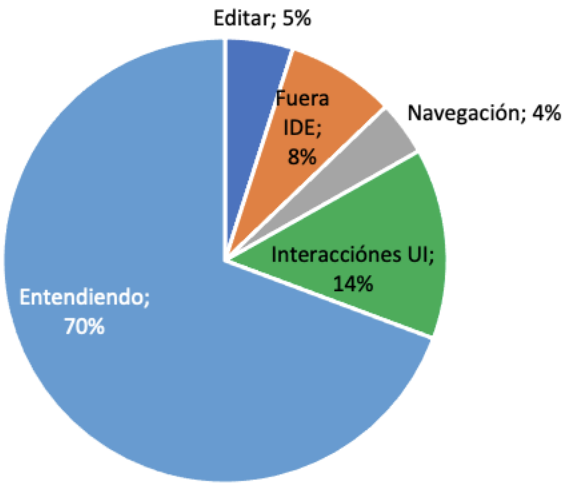
Todos podemos recordar el viejo dicho sobre cómo los hijos del zapatero no tenían zapatos, porque estaba demasiado ocupado satisfaciendo a sus clientes que pagaban. Tal era la situación en el desarrollo de software de IA hasta hace poco, cuando los desarrolladores de IA centraron su atención en ayudar a resolver los complejos problemas del desarrollo de software en sí.

ML, aprendizaje profundo y procesamiento del lenguaje natural (NLP) se consideran con frecuencia como tres técnicas dentro del dominio más amplio de la IA. Cuando se combinan, estas técnicas crean algunas posibilidades sin precedentes para transformar el proceso de desarrollo de software. Con un renovado interés en AI / ML y una uniformidad emergente de los procesos de desarrollo de software (repositorios comunes, así como integración continua/implementación continua), la industria está madura para absorber estas ideas en la corriente principal.

En la década de 1980, cuando se formó el proyecto de estudios empíricos de ingenieros de software del Consorcio de Microelectrónica y Tecnología Informática (MCC), nos centramos en el proceso de construcción de nuevos sistemas de software desde cero. Los métodos de desarrollo de software predominantes en la actualidad tienen más que ver con la combinación / integración de componentes existentes en sistemas aún más complejos. En este enfoque moderno, la dificultad ahora se ha desplazado a la comprensión de lo que hacen los componentes y cómo interactúan / dependen unos de otros. Y así, trabajar con bases de código preexistentes que el equipo de desarrollo probablemente no escribió ellos mismos es la nueva normalidad.

Por lo tanto, no es de extrañar, como R. Minelli, A. Mochi y M. Lanza informaron recientemente, que los desarrolladores de software ahora gasten aproximadamente el 70% de su tiempo relacionado con la codificación en comprender el código, mientras que la escritura de código solo representa alrededor del 5%.

Figura 9-1 Tiempo dedicado a comprender el código existente



Aquí hay algunas áreas de ingeniería de software que creemos que las herramientas de IA / ML tendrán el mayor impacto en el futuro cercano. Sin embargo, estos solo pueden ser realizables con un buen entorno de desarrollo sólido centrado en ML con las herramientas adecuadas.

Estimaciones precisas

Los desarrolladores de software son conocidos por rara vez ser capaces de proporcionar buenas estimaciones sobre plazos y costos. La IA entrenada en datos de proyectos anteriores puede ayudar a proporcionar estimaciones más precisas para que los equipos puedan predecir mejor el tiempo, el esfuerzo y el presupuesto requeridos. Ahora, si podemos lograr que la C-Suite respete esas estimaciones, la presión para entregar demasiado pronto podría disminuir y los proyectos fallidos minimizados. Este tipo de información puede ayudar a una organización a decidir qué proyectos hacer y cuáles no. Cuando puede informar con precisión a los clientes sobre los plazos de entrega del software, aumenta la satisfacción del cliente, la retención y la reputación comercial.

Establecimiento de objetivos de calidad

Cuando se combina con estándares de calidad de software mejor definidos (por ejemplo.ISO 25000), serán posibles nuevas herramientas de medición de calidad mejoradas por IA. Esto ayudará a las organizaciones a utilizar los rendimientos anteriores en la calidad del software para aprender y luego establecer mejores objetivos de calidad del proyecto para nuevos proyectos.

Gestión de errores

Cuando proporciona datos pasados y análisis de software a un asistente de programación impulsado por IA, puede aprender de la experiencia e identificar errores comunes. Si luego se marcan en el proceso de desarrollo, se reduciría la necesidad de volver a trabajar. El aprendizaje automático también puede ser utilizado por los equipos de operaciones en la fase posterior a la implementación para marcar errores de forma proactiva y descubrir anomalías mediante el análisis de los registros del sistema.

La administración de errores es responsable de la mayor parte del tiempo de inactividad en el desarrollo de software, especialmente si ofrece software como servicio (SaaS) o plataforma como servicio (PaaS). Con los clientes que utilizan sus servicios durante todo el día, cada minuto de tiempo de inactividad cuesta dinero y afecta negativamente la reputación de una organización.

Comprensión, búsqueda y corrección de errores

Cuando se detecta un error (o deficiencia) en el software, un desarrollador tiene que entenderlo, encontrarlo, corregirlo y asegurarse de que se solucione. Se trata de un proceso que requiere mucho tiempo, como se indica en la sección 8 del presente informe. Con la inteligencia artificial, puede detectar y diagnosticar errores en el software de forma semiautomática sin mucha participación humana. Este proceso es más eficiente y rentable, y conduce a un software de mayor calidad.

Además, con los últimos algoritmos y avances en inteligencia artificial y aprendizaje automático, los desarrolladores y evaluadores pueden predecir y prevenir errores automáticamente buscando patrones conocidos o aprendidos en las bases de datos.

Buscar patrones en el código

La IA puede llevar el análisis de código estático al siguiente nivel, utilizando millones de líneas de código para aprender patrones de programación correctos e incorrectos y luego encontrar esos patrones en otro código. Esto es especialmente importante cuando se integra código de terceros en un sistema.

Esto se vuelve mucho más efectivo cuando se combina con bases de conocimiento emergentes de patrones de codificación deficientes como se ve en los repositorios MITRE CWE, CVE.

Pruebas de software automatizadas asistidas por IA

Las pruebas de software son una fase crucial en el desarrollo de software, lo que ayuda a garantizar la calidad del producto. Ciertas pruebas de software deben repetirse cada vez que se cambia el código fuente y repetir esas mismas pruebas puede llevar mucho tiempo y ser costoso. Capturar y aprender de ese proceso es una fortaleza de la IA.

Existe una amplia gama de herramientas que emplean IA para crear casos de prueba y realizar pruebas de regresión. Estas herramientas de IA pueden automatizar las pruebas y garantizar aún más la repetición de las pruebas sin errores. Appvance, Functionize y Testim.io son algunos ejemplos de plataformas de prueba basadas en IA y aprendizaje automático.

Eggplant y Test Sigma son dos herramientas populares de prueba de software asistidas por IA que ayudan a los probadores de software a escribir y ejecutar pruebas automatizadas para mitigar errores y mejorar la eficiencia del código de software.

Seguridad del software

Las organizaciones de todo el mundo están utilizando la IA para capturar datos de seguridad y el aprendizaje automático para distinguir el comportamiento anómalo del comportamiento típico. Los sistemas de IA se pueden utilizar para detectar malware para cybersecurity, ejecutar reconocimiento de patrones, y observar los comportamientos del malware antes de que ingrese al sistema.

Asistentes de desarrollo de IA/ML

Un informe reciente encontró que el desarrollo de software mejorado por IA aumentó la productividad de un desarrollador en 10 veces, lo que elevó el rendimiento de nivel principiante al del experto. Dentro del modelo DevQualOps, el aprendizaje automático puede acortar varios procesos, especialmente las pruebas de software. La IA puede ejecutar pruebas automáticamente, en lugar de que los analistas de control de calidad las ejecuten manualmente. Esto no solo ahorra tiempo, sino que garantiza que se prueben más escenarios. De hecho, la IA es fundamental para el proceso de garantía de calidad, ya que la garantía de calidad manual tiene una alta probabilidad de error. La IA permite que una computadora realice pruebas rápidas y precisas que reducen la tasa de fallas y acortan el proceso de desarrollo.

Estas nuevas herramientas basadas en IA llevan las cosas un paso más allá, analizando y entendiendo todos esos millones de líneas de código indocumentado y encontrando fragmentos útiles a medida que los necesita, sin tener que buscarlos.

Una mejor comprensión del comportamiento del usuario

Los algoritmos de aprendizaje automático pueden ayudar a comprender el comportamiento del usuario y luego entregar contenido variable ajustando el tamaño de la pantalla, el tamaño de fuente, los botones y varios otros elementos en la página.

Estas respuestas personalizadas y dinámicas pueden mejorar la experiencia del usuario y permiten a los desarrolladores realizar los cambios apropiados en el código mediante la observación de los datos de interacción del usuario en tiempo real.

AI y ML se implementan en portales de Online Marketplace, donde pueden mejorar la funcionalidad del software, capturar los comentarios de los usuarios, reducir los puntos de fricción, evitar carritos abandonados y aumentar las tasas de conversión.

Algunas nuevas herramientas emergentes recientemente

El entorno de desarrollo *Open Source* original Eclipse ahora tiene la capacidad que proporciona un conjunto básico de componentes para crear aplicaciones que incorporan IA.

Copiloto de GitHub

GitHub alberga millones de proyectos, que, juntos, suman miles de millones de líneas de código. GitHub, trabajando con el modelo de aprendizaje automático Codex de OpenAI (un modelo de lenguaje centrado en el código como el conocido GPT-3) ha creado una herramienta para crear y entrenar un servicio que funciona con su editor de código para sugerir los próximos pasos a medida que trabaja. Llamándolo Copilot, GitHub lo describe como un "programador de pares de IA". Por lo tanto, es una herramienta colaborativa más que prescriptiva.

Copilot ha sido entrenado en los millones de líneas de código en repositorios públicos. Instalado como una extensión de Visual Studio Code, Copilot funciona dentro del contexto de la ventana del editor actual, proporcionando sugerencias basadas en su tipo y proporcionando detalles en lo que usa. Su código privado no se utiliza para entrenar el servicio con nuevos ejemplos de código. Las únicas señales son el código que se está usando.

No debe esperar que el código que produce Copilot sea correcto. Por un lado, todavía es temprano para este tipo de aplicación, con poca capacitación más allá del conjunto de datos inicial. A medida que más y más personas usan Copilot, y se basa en cómo usan sus sugerencias para el aprendizaje automático de refuerzo, sus sugerencias deberían mejorar. Sin embargo, aún tendrá que tomar decisiones sobre los fragmentos que usa y cómo los usa. Debe tener cuidado con el código que genera Copilot por razones de seguridad. Es imposible para GitHub auditar todo el código que está usando para entrenar a Copilot. Incluso con herramientas como Dependabot y el escáner de seguridad CodeQL, hay una gran cantidad de código de baja calidad que exhibe patrones malos y errores comunes.

Hay algunas ideas interesantes en Copilot: cómo toma sus comentarios y los convierte en código, o cómo sugiere las pruebas que se pueden usar como parte de un proceso de integración continua / implementación continua (CI / CD). Incorporar IA en las partes de desarrollo y prueba de un modelo de DevOps de integración continua / implementación continua tiene mucho sentido, ya que puede ayudar a reducir la carga de los desarrolladores, permitiéndoles centrarse en el código. Pero, nuevamente, aún debe asegurarse de que esas pruebas sean apropiadas y que proporcionen el nivel correcto de cobertura de código. No está limitado a una solución a la vez, ya que puede hojear los resultados en su editor, viendo qué funciona mejor para usted antes de aceptarlo.

Aquí algunas otras cosas interesantes que Copilot puede hacer.

Microsoft BugLab

Si bien hay docenas de herramientas disponibles para el análisis estático de código en varios lenguajes para encontrar fallas de seguridad, los investigadores han estado explorando técnicas que utilizan el aprendizaje automático para mejorar la capacidad de detectar fallas y corregirlas. Esto se debe a que encontrar y corregir errores en el código puede ser difícil y costoso, incluso cuando se usa IA para encontrarlos.

Investigadores de Microsoft Research Cambridge, Reino Unido, han detallado recientemente su trabajo en BugLab, una implementación de Python de "un enfoque para el aprendizaje autosupervisado de detección y reparación de errores". Es "autosupervisado" en el sentido de que los dos modelos detrás de BugLab fueron entrenados sin datos etiquetados. Esta ambición de no entrenamiento fue impulsada por la falta de errores anotados del mundo real para entrenar modelos de aprendizaje profundo de búsqueda de errores. Si bien hay una gran cantidad de código fuente disponible para dicha capacitación, en gran medida no está anotado.

BugLab tiene como objetivo encontrar errores difíciles de detectar frente a errores críticos que ya se pueden encontrar a través de análisis de programas tradicionales. Su enfoque promete evitar el costoso proceso de codificar manualmente un modelo para encontrar estos errores.

El grupo afirma haber encontrado 19 errores previamente desconocidos en paquetes Python *Open Source* de PyPI como se detalla en el documento, *Auto-Supervised Bug Detection and Repair*, presentado en la conferencia Neural Information Processing Systems (NeurIPS) 2021.

Más allá de razonar sobre la estructura de una pieza de código, creen que los errores se pueden encontrar "al comprender también las pistas ambiguas de lenguaje natural que los desarrolladores de software dejan en los comentarios de código, nombres de variables y más".

Su enfoque en BugLab, que utiliza dos modelos en competencia, se basa en los esfuerzos de aprendizaje autosupervisados existentes en el campo que utilizan el aprendizaje profundo, la visión por computadora y el procesamiento del lenguaje natural (NLP). Se parece o está "inspirado por" GAN o redes generativas antagónicas, las redes neuronales que a veces se usan para crear falsificaciones profundas.

Los dos modelos de BugLab incluyen un selector de errores y un detector de errores: "Dado algún código existente, que se presume que es correcto, un modelo selector de errores decide si debe introducir un error, dónde introducirlo y su forma exacta (por ejemplo, reemplazar un "+" específico con un "-"). Dada la elección del selector, el código se edita para introducir el error. Luego, otro modelo, el detector de errores, intenta determinar si se introdujo un error en el código y, de ser así, localizarlo y corregirlo".

A partir del conjunto de datos de prueba de los investigadores de 2.374 errores de paquetes de Python de la vida real, mostraron que el 26% de los errores se pueden encontrar y corregir automáticamente.

Sin embargo, su técnica marcó demasiados falsos positivos o errores que en realidad no eran errores. Por ejemplo, aunque detectó algunos errores conocidos, solo 19 de las 1,000 advertencias reportadas por BugLab eran en realidad errores de la vida real.

En cuanto a las 19 fallas de día cero que encontraron, informaron 11 de ellas en GitHub, de las cuales seis se han fusionado y cinco están pendientes de aprobación. Algunas de las 19 fallas eran demasiado menores para molestarse en informar.

Facebook’s Getafix

Durante los últimos años, Facebook ha estado utilizando una herramienta desarrollada internamente llamada Getafix, que ellos indican contribuye a la estabilidad de las aplicaciones que usan miles de millones de personas.

Afirman que de todas las advertencias corregidas por los ingenieros de Facebook desde que se implementó el servicio Getafix, el 42% se corrigió aceptando la sugerencia de corrección y, en el 9% de los casos, los ingenieros escribieron una solución semánticamente idéntica. Han comenzado con éxito a automatizar el descubrimiento y la aplicación de reglas de "lint". Los cambios realizados en respuesta a la revisión del código a menudo son correcciones a los antipatrones comunes que fueron señalados por un revisor, y encontrar y corregir estos antipatrones se puede incluir en una regla de pelusa.

En su opinión, las oportunidades más prometedoras, pero relativamente sin explotar para usar ML pertinentes a aspectos del equipo y los estados de producción son.

- *Codereview*: aunque ampliamente considerado como esencial para mantener la calidad del software, la revisión manual del código es un compromiso de tiempo significativo para los ingenieros de software. Las técnicas de ML pueden ayudar a automatizar las revisiones rutinarias de código (como el formato y las mejores prácticas de codificación). Más ambiciosamente, tal vez ML pueda resolver automáticamente un comentario de revisión de código de rutina.
- *Evaluación del riesgo de un cambio de código*: En principio, cualquier cambio de código aumenta el riesgo de una aplicación. Podría decirse que toda la tubería de pruebas y verificación existe esencialmente para reducir este riesgo. ¿Se pueden diseñar técnicas basadas en ML que proporcionen una evaluación cuantitativa del riesgo de un cambio de código, complementando la canalización habitual de pruebas y verificación? Los avances aquí afectarán tanto a las pruebas (al priorizar las pruebas relacionadas con cambios más riesgosos) como a la gestión de versiones (al llevar a cabo un control de calidad adicional para las versiones de código más riesgosas). En comparación, las técnicas para evaluar el impacto de un cambio adoptan una visión binaria de la afectación y, debido a las limitaciones del análisis estático, a menudo serían demasiado pesimistas en su evaluación.

- *Solución de problemas*: Para aplicaciones ampliamente implementadas, los clientes envían sus comentarios implícitamente (telemetría o bloqueos) y, a veces, explícitamente enviando comentarios. El volumen de esta retroalimentación puede ser enorme. Esta es otra área en la que ML puede ayudar de varias maneras: no solo en la clasificación de estos informes, sino también en su agrupación para identificar problemas comunes, encontrar pistas importantes de los registros de telemetría y cambios de código que podrían conectarse al problema en cuestión.

Kite

Kite, al sugerir código reutilizable sensible al contexto, puede ayudar a un desarrollador a disminuir las pulsaciones de teclas en un 47%. Fue entrenado en modelos que han pasado por más de 25 millones de archivos y, como resultado, puede ofrecer sugerencias de múltiples líneas. Kite es compatible con 12+ lenguajes que incluyen Java, PHP, HTML/CSS, Javascript, Typescript, Kotlin, Ruby y Python. **Codota es similar.**

Visual Studio IntelliCode

IntelliCode es de Microsoft y viene integrado con el IDE de Microsoft llamado Visual Studio. En Visual Studio, admite C# y XAML, mientras que es compatible con Java, Python, JavaScript y TypeScript en Visual Studio Code. Esta herramienta de finalización de código de IA recibió su capacitación de los códigos de medio millón de proyectos *Open Source* de GitHub. Por lo tanto, puede guiarlo con sugerencias más inteligentes teniendo en cuenta el código y el contexto actuales. Para ello, necesita ayuda de nombres y posiciones de variables, la lista IntelliSense, las bibliotecas que usa y las funciones en código cercano. Si bien esta herramienta le mostrará sugerencias en orden alfabético de forma predeterminada, siempre puede alternar entre las opciones. Su característica de finalización de código de línea completa, disponible en la versión 2022 de Visual Studio, indica el siguiente fragmento de código basado en la predicción en línea de texto gris.

Próximos pasos

Lo que falta son las herramientas de prevención de deficiencias basadas en IA que se entrenan en patrones conocidos (por ejemplo, CVE, CWE, etc.) para detectarlos en tiempo real a medida que los desarrolladores de software escriben el código que crea esas deficiencias / debilidades, evitando así que entren en el flujo de desarrollo de nuevo código. Cuando estas herramientas finalmente lleguen, el costo del software de baja calidad caerá en picada, y podremos centrar nuestra atención en remediar el creciente DT.

10. CONCLUSIONES, RECOMENDACIONES Y PRÓXIMOS PASOS

Conclusiones

Las condiciones económicas clave de Estados Unidos que enmarcan el contexto de este informe bienal son:

- Un PIB proyectado para el 2022 de $ 23.35 billones, un aumento de aproximadamente el 2% desde el 2020
- Una tasa de inflación acumulada del 15% durante el período de 2 años
- Un pequeño crecimiento del 4% en la base laboral de TI durante esos 2 años, y
- El número de puestos de trabajo de TI sin cubrir se situaba en alrededor de 300.000 a finales de agosto.

En este informe de actualización del 2022, estimamos que el costo de la baja calidad del software en los Estados Unidos ha crecido a al menos $ 2.41 billones[1], pero no en proporciones similares a las vistas en el 2020. La DT de software acumulado ha crecido a ~ $ 1.52 billones. Estos se deben principalmente a:

- El enorme aumento de los costos de cibercrimen a $ 1.44 billones en el 2022, lo que representa la mayor parte del aumento en costo del software de baja calidad, y
- La escasez de ingenieros de software calificados junto con el uso rezagado de las herramientas disponibles explica el aumento de DT, en gran parte porque las deficiencias no se están solucionando al mismo ritmo que en 2020.

Aunque el costo del software de baja calidad y la DT han aumentado significativamente a lo largo de la serie de nuestros tres informes (el problema), también lo han hecho los desarrollos en la tecnología / prácticas para remediar esos problemas (soluciones).

ES POSIBLE QUE LA TENDENCIA EN EL COSTO DEL SOFTWARE DE BAJA CALIDAD GENERAL SE NIVELE EN LA PRÓXIMA DÉCADA SI LAS ORGANIZACIONES ADOPTAN LAS RECOMENDACIONES QUE HEMOS PRESENTADO EN ESTA SERIE DE INFORMES. Esperamos que las soluciones sugeridas en este documento se adopten más ampliamente en la corriente principal de la concepción, desarrollo, producción y evolución del software.

Recomendaciones

Además de las recomendaciones generales de nuestros informes anteriores, agregamos las siguientes recomendaciones más específicas para el desarrollo de software y las organizaciones de TI:

- Utilizar los estándares de calidad del software, las mediciones relacionadas y las herramientas que están surgiendo.
- Analice y evalúe la calidad de todos los componentes de terceros/ *Open Source* que se incluirán en cualquier sistema. Monitorear de cerca en funcionamiento. Aplique los parches de manera oportuna.
- Evite los modelos de DevOps e integración continua/implementación continua que no incluyan las mejores prácticas y herramientas de la ingeniería de calidad continua. Adopte DevQualOps en su lugar.
- Integre la corrección continua de DT en su SDLC
- Invierta en el profesionalismo, el conocimiento y las herramientas de sus ingenieros de software, y

- Considere la posibilidad de certificar a sus desarrolladores para el conocimiento del código crítico y la debilidades de arquitectura en ISO / IEC 5055 (cuando OMG haga que su prueba de certificación "Desarrollador confiable" esté disponible a fines de 2023 o 2024).

Lidiar con la escasez del mercado laboral de TI

Según la Oficina de Estadísticas Laborales de los Estados Unidos (BLS), las posiciones de TI que se enumeran a continuación son ejemplos seleccionados que se proyectan para crecer y pagar muy por encima del promedio. Debido a la escasez actual, estos puestos cruciales serán más difíciles de cubrir, especialmente en las tres primeras categorías a continuación.

Desarrolladores de software, analistas de control de calidad y probadores

Los desarrolladores de software diseñan aplicaciones o programas informáticos. Los analistas y evaluadores de control de calidad de software identifican problemas con aplicaciones o programas e informan defectos.

Salario medio 2020: $ 110,140 por año
Educación típica de nivel de entrada: licenciatura
Número de empleos, 2020: 1,847,900
Crecimiento proyectado, 2020–2030: 22% (Mucho más rápido que el promedio)
Vacantes ocupacionales proyectadas, promedio anual 2020–2030: **189,200**

Analistas de Sistemas Computacionales

Los analistas de sistemas informáticos estudian los sistemas informáticos actuales de una organización y encuentran soluciones que son más eficientes y efectivas.
Salario medio 2020: $ 93,730 por año
Educación típica de nivel de entrada: licenciatura
Número de empleos, 2020: 607,800
Crecimiento proyectado, 2020–2030: 7% (aproximadamente tan rápido como el promedio)
Vacantes ocupacionales proyectadas, promedio anual 2020–2030: **47,500**

Analistas de Seguridad de la Información

Los analistas de seguridad de la información planifican y llevan a cabo medidas de seguridad para proteger las redes y sistemas informáticos de una organización.
Salario medio 2020: $ 103,590 por año
Educación típica de nivel de entrada: licenciatura
Número de empleos, 2020: 141,200
Crecimiento proyectado, 2020–2030: 33% (Mucho más rápido que el promedio)
Vacantes ocupacionales proyectadas, promedio anual 2020–2030: **16,300**

Científicos de investigación informática e información

Los científicos de investigación informática e información diseñan usos innovadores para la tecnología informática nueva y existente.
Salario medio 2020: $ 126,830 por año
Educación típica de nivel de entrada: maestría
Número de empleos, 2020: 33,000
Crecimiento proyectado, 2020–2030: 22% (Mucho más rápido que el promedio)
Vacantes ocupacionales proyectadas, promedio anual 2020–2030: 3,200

Desarrolladores Web y Diseñadores Digitales

Los desarrolladores web crean y mantienen sitios web. Los diseñadores digitales desarrollan, crean y prueban el diseño, las funciones y la navegación del sitio web o la interfaz para la usabilidad.
Salario medio 2020: $ 77,200 por año
Educación básica típica: Licenciatura
Número de empleos, 2020: 199,400
Crecimiento proyectado, 2020–2030: 13% (más rápido que el promedio)
Vacantes ocupacionales proyectadas, promedio anual 2020–2030: 17,900

Consulte el Manual de perspectivas ocupacionales de BLS para obtener más detalles y más detalles.

La mayor demanda de esos profesionales continuará durante la próxima década debido al aumento del teletrabajo y los acuerdos de trabajo híbridos, la expansión de los teleservicios y la mejora de las medidas de ciberseguridad para proteger la información.

En general, el mercado laboral para el talento tecnológico en 2022 sigue siendo fuerte. En agosto, la tasa de desempleo para las ocupaciones tecnológicas en los Estados Unidos se situó en el 2,3%, según la Asociación de la Industria de Tecnología Informática (CompTIA), significativamente más baja que la tasa de desempleo de los Estados Unidos del 3,7% ese mes, que en sí misma es baja según los estándares históricos. Se estima que hay 8.7 millones de trabajadores tecnológicos en los Estados Unidos, según las cifras de CompTIA publicadas a principios de este año.

En total, más de 118,000 personas han perdido sus empleos en tecnología este año, según Layoffs.fyi, un sitio que rastrea los recortes de empleos reportados públicamente en la industria. Pero todos los despedidos tienen excelentes perspectivas en empresas establecidas o en nuevas empresas.

Tampoco está claro si los despidos masivos a mediados de noviembre de 2022 en la industria de la tecnología de las redes sociales tendrán algún impacto en esta escasez de profesionales del software.

Próximos pasos

Nuestro próximo informe está previsto tentativamente para 2024, cuando esperamos que algunas de las soluciones identificadas en este informe se pongan al día con los problemas y se muestren en un cambio positivo en la tendencia de costo del software de baja calidad. Nuestro próximo informe probablemente se centrará en la confiabilidad de los sistemas de infraestructura crítica, en la atención médica, las elecciones y la distribución de energía.

Esperamos que esta área en particular sea de creciente interés en 2024.

Un enfoque en la infraestructura crítica de la tecnología de sistemas electorales

En el área de la tecnología de sistemas de votación, hemos recorrido un largo camino desde los días de los *hanging chads* (pestañas colgantes de papel en las boletas de votación). Durante dos décadas, el auge de la tecnología de votación ha ayudado a automatizar un proceso que estaba en extrema necesidad, permitiendo así que millones de votantes voten sin aumentar el esfuerzo humano de administrar ese proceso. Desafortunadamente, como efecto secundario, estos sistemas se han convertido en un foco de intenso debate político.

Debido a que nuestro sistema electoral es en realidad una colección de sistemas de votación independientes, el rango y la diversidad de los sistemas electorales van desde "totalmente de papel" hasta semiautomatizados. Por ejemplo, en Texas hay 254 sistemas diferentes a nivel de condado en uso. En gran parte del país, cuando alguien vota, simplemente llena una boleta de papel, que

generalmente se alimenta a través de un dispositivo de conteo llamado escáner óptico. En otros lugares, algunos votantes utilizan configuraciones totalmente digitales, llamadas sistemas electrónicos de grabación directa, que a veces usa la computadora para marcar y contar los votos. En otros lugares, se utilizan sistemas híbridos y de cosecha propia. Sabemos que al menos el 30% de los votos emitidos en las elecciones generales de 2020 fueron en algún tipo de máquina, a diferencia de una boleta marcada a mano. Esta diversidad de tecnología electoral se ve en la siguiente figura.

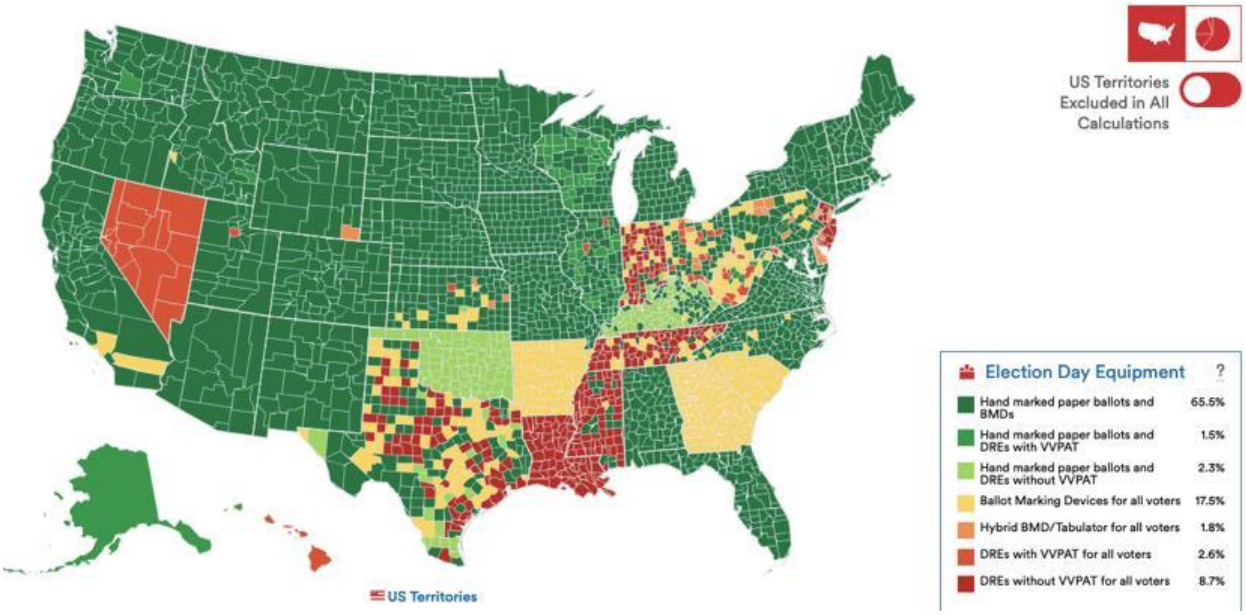


Figura 10-1 La diversidad de la tecnología electoral en los Estados Unidos: Elecciones generales de 2020

Hoy en día, el mercado de máquinas de votación está dominado por tres proveedores principales: Election Systems & Software, Dominion Voting Systems y Hart InterCivic. Según una estimación, toda la industria genera aproximadamente $ 300 millones en ingresos anuales.

La mayoría de sus productos contienen un dispositivo semiautomático de marcado de boletas (BMD). Si bien los diseños específicos varían, los BMD tienen una pantalla táctil de computadora para que los votantes hagan sus selecciones. Luego, la máquina imprime una boleta de papel que se puede introducir en un escáner. A diferencia de las boletas de papel marcadas a mano, las BMD tienen la capacidad de acomodar a cada votante utilizando una variedad de dispositivos de accesibilidad, incluidos los que no pueden ver, manejar papel o incluso tocar una pantalla. Las máquinas han proliferado desde 2002, cuando el Congreso aprobó la Ley Help America Vote.

Las máquinas de votación tienen otras ventajas sobre las boletas de papel: pueden ofrecer múltiples opciones de idioma, admitir jurisdicciones más grandes que necesitan miles de tipos de boletas diferentes y garantizar que los votantes no pierdan inadvertidamente una carrera o cometan un error que descalifique su boleta. No está claro qué tan grande podría ser el problema.

El costo de **seguridad del software**

Los defensores dicen que los sistemas de votación electrónica pueden ser relativamente seguros, mejorar la accesibilidad y simplificar la votación y el conteo de votos. Pero, los críticos han argumentado que son inseguros y deben usarse con la menor frecuencia posible. Hay hackers trabajando duro en ambos lados.

La votación semiautomática trae temores de que alguien pueda manipular las máquinas y manipular los resultados. Y, según algunos expertos, el comportamiento de las empresas proveedoras ha hecho poco para inspirar la confianza del público. Estos críticos dicen que el software en BMD es complejo, a menudo mal organizado y extremadamente largo, lo que facilita la inserción de código que no se detecta. Pero no hay datos empíricos para respaldar o negar esa afirmación.

Debido a que las carreras y los candidatos cambian cada elección, se debe cargar un nuevo diseño de boleta antes de cada contienda, lo que ofrece otra oportunidad para que se cuele código malicioso. Y debido a que la votación se realiza de forma anónima, es imposible vincular una boleta específica con la persona que la emitió después del hecho.

En respuesta a tales preocupaciones, las compañías de máquinas de votación han reconocido que sus equipos pueden tener vulnerabilidades. Pero, dicen, casi todas las máquinas dejan un rastro de papel que puede ser auditado, lo que permite detectar incidentes durante la certificación.

En medio de estas preocupaciones, un puñado de innovadores están tratando de crear una máquina de votación que sea fácil de usar, basada en software *Open Source* y significativamente más difícil de hackear que los modelos existentes. Pero lo que hemos visto recientemente en las vulnerabilidades en OSS es una afirmación por probar.

Un ejemplo de un prototipo de investigación de tal máquina se ve en uno de los artículos recientes publicados en las Comunicaciones de la ACM en noviembre de 2021. Al igual que con la mayoría de los prototipos de investigación, este modelo aún no ha sido sometido a rigurosas pruebas externas por parte de usuarios hostiles.

Esperemos que para 2024 tengamos algunos datos experimentales sobre la calidad del software en este tipo de máquinas de votación prototipo.

Nota 1 – Véase en el Apéndice B la metodología detallada de estimación de costes utilizada.

1. AGRADECIMIENTOS

Sobre el autor. Herb Krasner es profesor jubilado de Ingeniería de Software en la Universidad de Texas en Austin. Ha sido consultor de la industria durante cinco décadas; ayudando a las organizaciones a establecer una línea de base y mejorar sus capacidades de desarrollo de software. Es un miembro activo del Consejo Asesor de CISQ y es conocido por su investigación previa en los estudios empíricos de profesionales de software, el ROI de la mejora de procesos de software y el costo de la calidad del software. Puede ser contactado en hkrasner@utexas.edu.

Gracias a nuestro equipo de revisión técnica por sus comentarios y contribuciones de revisión más útiles a los borradores de este informe: Anita D'Amico, Marco Puebla, Greg Law y Laila Lotfi. Un agradecimiento especial a mi viejo amigo, colaborador y revisor jefe, Don Shafer, fundador del Grupo Atenas y miembro técnico, con quien he trabajado en muchas iniciativas importantes desde mediados de la década de 1980. Muchas gracias a nuestra administradora de proyectos, Katie Hart; y a mi amigo Bill "Tex" Curtis, Director Ejecutivo de CISQ, por sus cinco décadas de amistad, colaboración profesional y apoyo. Gracias a mi esposa Judy por su asistencia profesional de edición, además de 51 años de amor, apoyo y aguantar conmigo.

Acerca de CISQ. El Consortium for Information & Software QualityTM (CISQTM) desarrolla estándares internacionales para automatizar la medición de la calidad del software y promover el desarrollo y mantenimiento de software seguro, confiable y confiable. A través de su trabajo, se han desarrollado estándares respaldados por la industria para medir el tamaño del software, la calidad estructural y el TD a partir del código fuente. Estos estándares son utilizados por muchas organizaciones de TI, proveedores de servicios de TI y proveedores de software en la contratación, desarrollo, prueba, aceptación e implementación de aplicaciones de software.

Traducción al español. Shirley Solano, GBM y Marcial Saurez, GBM. Disponible en www.clubdeinvestigacion.com

Patrocinadores del Informe 2022

Synopsys

Synopsys es la compañía de automatización de diseño electrónico # 1 que se enfoca en el diseño y verificación de silicio, propiedad intelectual de silicio y seguridad y calidad del software. Su tecnología está presente en los automóviles autónomos, la inteligencia artificial y los productos de consumo de Internet de las cosas. Han invertido más de $ 1 mil millones en la construcción de la solución de seguridad de software definitiva. El Grupo de Integridad de Software de Synopsys se centra en la seguridad y calidad del software. Proporcionan análisis estático, análisis de composición de software y soluciones de análisis dinámico que permiten a los equipos encontrar y corregir rápidamente debilidades, vulnerabilidades y defectos en el código propietario, los componentes *Open Source* y el comportamiento de las aplicaciones. Clasificado como el líder en el Cuadrante Mágico de Gartner para Pruebas de Seguridad de Aplicaciones, con una combinación de herramientas, servicios y experiencia líderes en la industria, Synopsys SIG ayuda a las organizaciones a optimizar la seguridad y la calidad en DevSecOps y durante todo el ciclo de vida del desarrollo de software.

Undo

Undo es la compañía de depuración de viajes en el tiempo para Linux. Equipan a los desarrolladores con la tecnología para comprender el código complejo y corregir errores más rápido. Los desarrolladores pasan demasiado tiempo averiguando qué hace realmente el código, ya sea para comprender el código de otras personas o para encontrar y corregir errores. La depuración puede llevar mucho tiempo cuando no se pueden reproducir los errores de software. La depuración de viajes en el tiempo resuelve este

problema al hacer que los errores sean 100% reproducibles. Al llevar la depuración de viajes en el tiempo a CI y System Test, LiveRecorder de Undo permite a los desarrolladores ahorrar tiempo diagnosticando las causas raíz de nuevas regresiones, errores heredados y pruebas escamosas. Miles de desarrolladores de empresas tecnológicas líderes como SAP, Juniper Networks y Siemens utilizan LiveRecorder para mejorar la productividad de los desarrolladores, la velocidad de desarrollo y la calidad del software.

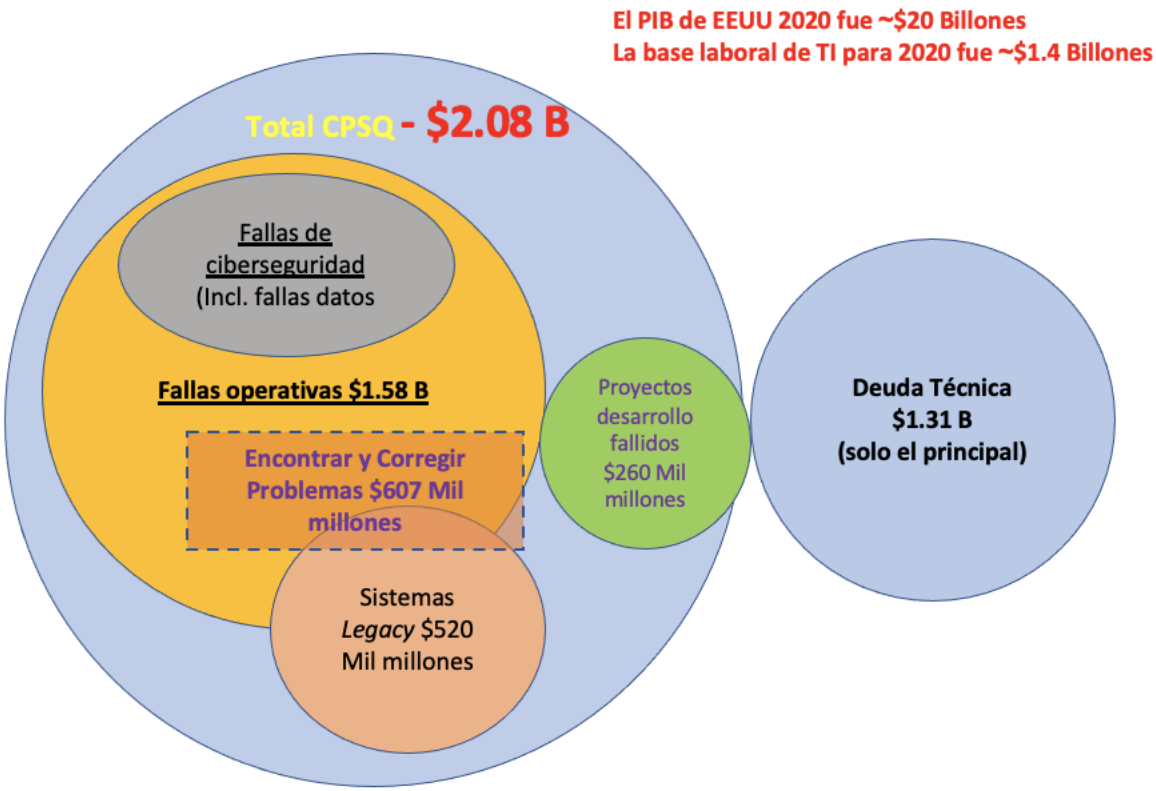
ANEXO A: RESUMEN DEL INFORME COSTO DEL SOFTWARE DE BAJA CALIDAD 2020

El propósito principal de nuestra serie de informes ha sido informar e inspirar a nuestros lectores a buscar el conocimiento de costo del software de baja calidad dentro de sus propias organizaciones, exponiendo primero el tamaño de este problema mayormente oculto. En nuestro informe de 2018 declaramos que los costos ocultos de la baja calidad del software eran de 6 a 50 veces los costos observables. Esto puede haber ayudado a que los lectores sean más conscientes de lo que podría hacerse observable.

El informe de 2020 se basó en algunos de los conceptos básicos de 2018 de qué es la calidad del software, cuál es el costo del modelo de calidad del software y la discusión de la calidad del software buena frente a la mala.

En nuestro informe de 2020, elaboramos muchos de los informes de fallas conocidos públicamente para enfatizar la magnitud del problema de la baja calidad del software. Presentamos la mayoría de las estrategias, tácticas, modelos y mejores procesos / prácticas que podrían usarse para abordar el problema a través de un enfoque coherente que las organizaciones podrían usar.

En el informe de 2020 mostramos un resumen de las estimaciones del costo de la baja calidad del software en los Estados Unidos para ese año, como se ve en la figura a continuación.



Pudimos construir este resultado utilizando un análisis, síntesis y extrapolación únicos de 88 fuentes existentes de información disponible en línea, mezcladas con conocimiento experto sobre software y su calidad. A continuación, se presenta un resumen de cada categoría principal del informe de 2020.

Fallas operativas: $ 1.56 billones

Además de citar ejemplos específicos, como Knight Capital y SolarWinds, identificamos que Tricentis Software Fail Watch, 5th Ed. reportó 606 fallas importantes de software desde 2017, causando una pérdida total de $ 1.7 billones en activos en 314 compañías. Esto promedió $ 2.8 mil millones por falla. Esto nos llevó a observar que el cibercrimen era un factor subyacente que estaba aumentando rápidamente.

Identificamos las tendencias que magnifican el impacto de las fallas de software, lo que aumenta los costos de falla:

- 100+ mil millones de nuevos LOC producidos en todo el mundo cada año -> 25 errores por cada 1000 LOC inyectados en promedio
- 96 zettabytes de datos digitales ahora almacenados (frente a 16 en 2016)
- Crecimiento de la ciberdelincuencia: el ransomware en EE. UU. cuesta $ 9 mil millones; $20B en todo el mundo en 2021
- Aumento de la transformación digital: difusión de los efectos de un mal funcionamiento del software en toda la cadena de valor

- Crecimiento de los sistemas de sistemas: expandiendo la complejidad exponencialmente y ocultando los desencadenantes para grandes fallas en una maraña de interacciones entre sistemas.
- Mayor competencia: especialmente en línea, ha priorizado la velocidad de negocios sobre el riesgo operativo y costos de mantenimiento correctivo
- Una gran apuesta para los sistemas que no están diseñados para esperar y gestionar fallas.

Nuestras recomendaciones generales para hacer frente a posibles fallas operativas en el futuro fueron:

- Evite que se creen y eliminen errores, fallas, debilidades y vulnerabilidades
- Encuentra y corrige errores lo más pronto posible
- Medir la calidad
- Adoptar prácticas de desarrollo de alta calidad
- Analizar componentes potencialmente defectuosos (por ejemplo, OSS)

Proyectos fallidos: $ 260 mil millones

Al examinar los resultados del proyecto de TI de la serie de informes CHAOS y extrapolar las tendencias a través de ellos, observamos que la industria se mantuvo estable en esta área con

- Aproximadamente el 20% de los proyectos fracasan rotundamente
- Aproximadamente el 35% de los proyectos tienen éxito
- Y el 45% restante en la categoría impugnada

Nuestras recomendaciones generales fueron:

- Evite proyectos grandes: Para proyectos de gran tamaño (10^4 FP) y superiores, los proyectos de baja calidad son 5-6X más probable que se cancelen que los proyectos de alta calidad.
- Definir qué significa calidad para un proyecto específico y luego centrarse en lograr resultados medibles frente a los objetivos de calidad declarados
- Utilice las mejores prácticas y herramientas conocidas para lograr una alta calidad
- No comprometa la calidad para la velocidad de operación

Problemas del sistema *legacy*: $ 520 mil millones

Después de décadas de operación, estos sistemas pueden haberse vuelto menos eficientes, menos seguros, más frágiles, incompatibles con las tecnologías y sistemas más nuevos, y más difíciles de dar soporte debido a la pérdida de conocimiento y / o una mayor complejidad o pérdida de soporte del proveedor. Notamos que estos sistemas típicamente:

- consumen el 70-75% del presupuesto total de TI
- representan el 80% del coste total de propiedad

El enfoque de modernización a utilizar depende de la prioridad de los problemas a resolver: funcionalidad, rendimiento, tecnología obsoleta, arquitectura inflexible, cargas de DT. Se identificaron varias estrategias (por ejemplo, contenedorización)

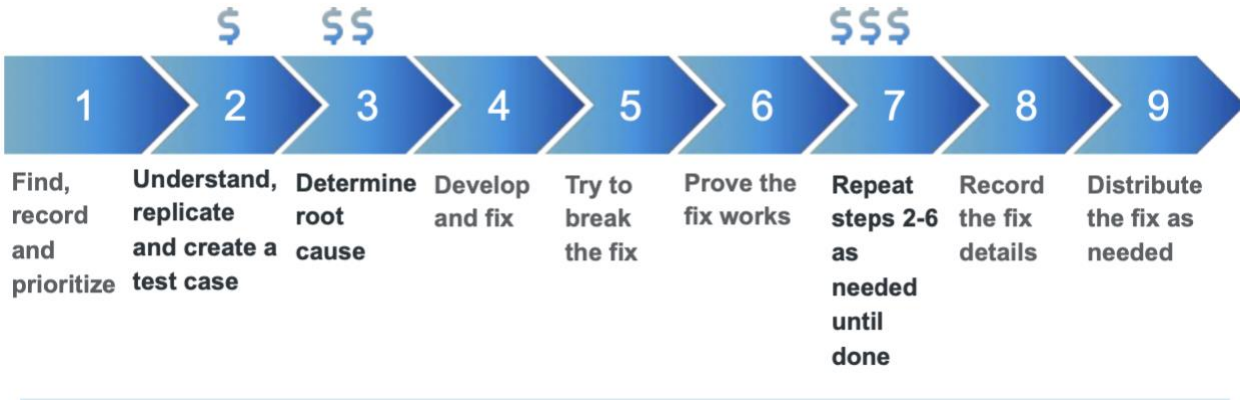
Nuestras recomendaciones generales fueron que:

- Todas estas estrategias son posibles superando la falta de comprensión y conocimiento de cómo funciona el sistema internamente.

- Cualquier herramienta que ayude a identificar debilidades, vulnerabilidades, síntomas de falla, defectos y Los objetivos de mejora son útiles
- La evaluación comparativa del estado de salud de un sistema *legacy* es un buen punto de partida.
- Los planos detallados de la conectividad del sistema son útiles para modernizar las arquitecturas que tienen degradado con el tiempo.

Encontrar y corregir errores: $ 607 B

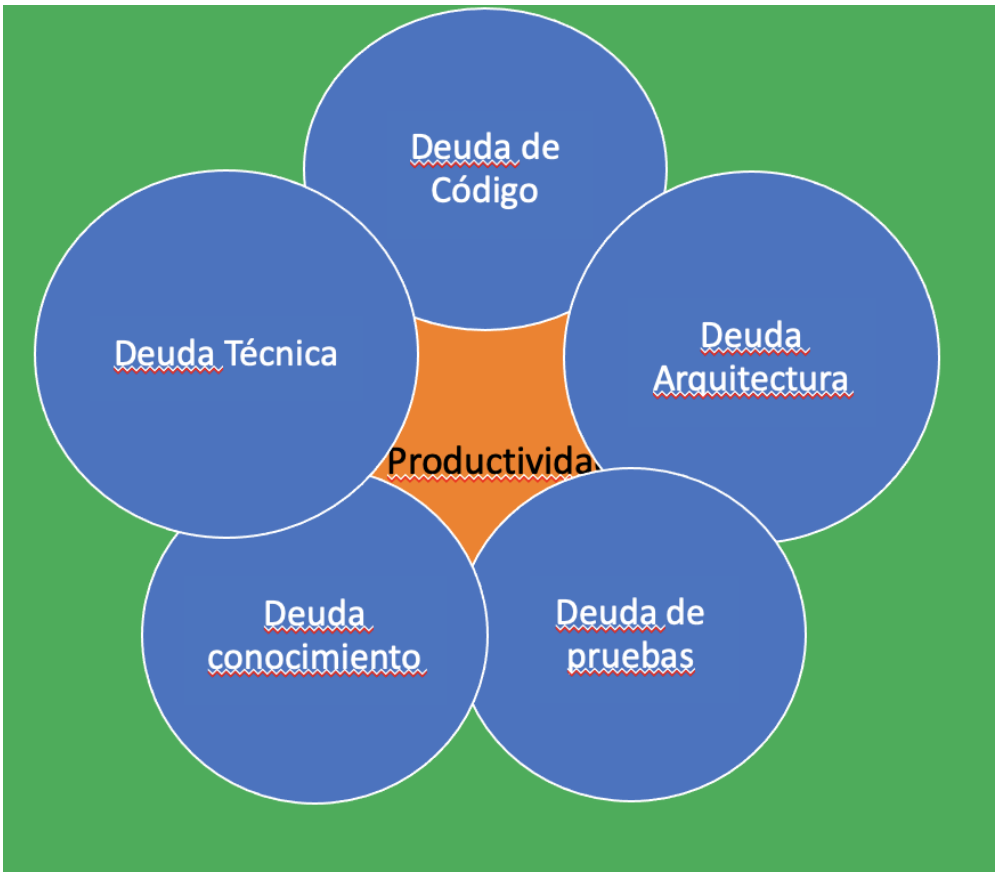
Esta área más importante para la ingeniería de calidad de software sustenta todas las otras áreas que hemos descrito anteriormente. Basándonos en nuestros estudios empíricos previos, pudimos describir dónde se gasta el esfuerzo relativo en este proceso, como se ve en el siguiente diagrama.



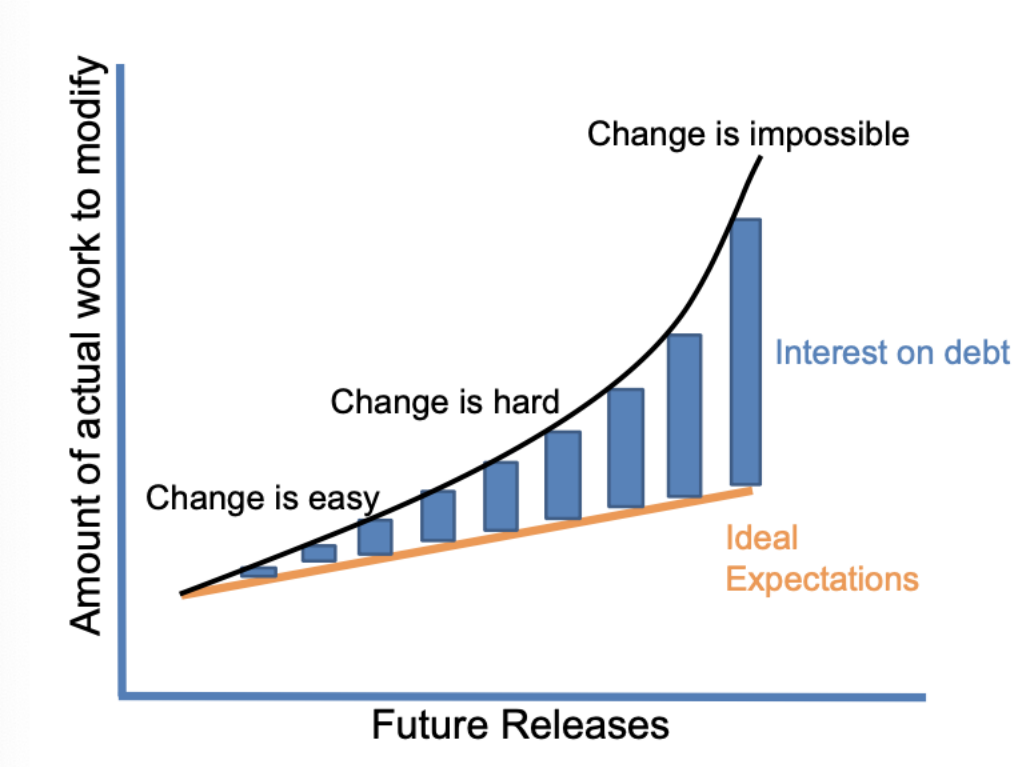
Nuestra recomendación general fue prevenir problemas primero y luego enfocarnos en dónde se gastan los $$$ en el modelo de proceso anterior para reducir costo del software de baja calidad y mejorar la calidad y productividad general. Dada la importancia de este proceso, profundizamos en este tema en este informe de 2022.

TD: $1.31 billones + interés

Describimos en 2020 que existen muchos tipos de software DT



Y cómo el impacto de la DT crece con el tiempo si no se trata.



Notamos que esta tendencia se ve exacerbada por el gran tamaño de la base de código y la rapidez con que está creciendo. Específicamente, que

- ~ 1.655 billones de LOC existen en todo el mundo y 513 mil millones en los Estados Unidos.
- El crecimiento del código es ahora ~ 100 mil millones de nuevos LOC por año, o ~ 7% de crecimiento por año.

La cuestión clave ahora es ¿cómo identificar y luego administrar el software DT?

Recomendaciones generales para mejorar la práctica

Informamos nuestras recomendaciones generales para aquellas organizaciones que querían mejorar su costo de perfil de calidad de software deficiente, lo que sugerimos que debería hacerse como un enfoque organizacional holístico que involucre los siguientes niveles de una organización.

Nivel Líderes/C-Suite

- Establecer la calidad como ciudadano de^{1a} clase -> seguridad+
- Haga mejores preguntas: externa e internamente
- Mida la calidad del software y costo del software de baja calidad en su organización

Equipos/proyectos

- Esfuércese por lograr un alto rendimiento
- Utilice las mejores prácticas y herramientas
- Definir y realizar un seguimiento de los objetivos de calidad
- Evite horarios o restricciones arbitrarios y poco realistas

Individuos

- Aprender y desarrollar un enfoque disciplinado
- No tengas miedo de las métricas de calidad
- Utilizar las fuentes de conocimiento existentes sobre patrones de errores y fallas estructurales de calidad

ANEXO B: MÉTODO DE ESTIMACIÓN DE COSTO DEL SOFTWARE DE BAJA CALIDAD

Como hemos dicho antes y continuamos afirmando aquí, la mayoría de las organizaciones aún no recopilan e informan su costo de números de calidad de software deficientes. Por ejemplo, en un estudio de 2017 de ejecutivos de TI, el 35% de los encuestados dijeron que no tenían idea de cuánto le costaban a su negocio las fallas del sistema de TI.

Los costos de fallas operativas son, con mucho, la categoría más grande de costo del software de baja calidad.

Es útil pensar en los costos de falla operativa de TI en términos de costos directos e indirectos.

Costos directos

Las pérdidas directas se dividirán en dos categorías generales:

- Pérdida de una aplicación o servicio. Esto puede ser más o menos grave, dependiendo de lo que haya fallado.

  Por ejemplo, una serie de interrupciones no planificadas puede matar a un negocio.

- Pérdida de datos. La pérdida de datos puede tener un impacto aún mayor en un negocio porque la pérdida de datos puede ser permanente. Esto también puede tener implicaciones financieras e incluso legales más allá de las pérdidas directas. El ransomware ha sido particularmente dañino en ese sentido.

  Existen algunos datos para ayudar a estimar los costos directos.

- Según un estudio realizado en 2008 por IBM Global Services, el costo promedio de la interrupción de sistemas de TI fue de $ 2.8 millones por hora. Debido a la inflación, ese número hoy sería de $ 3.88 millones por hora.
- Según una investigación de KPMG, el 48% de las empresas dice que más de 24 horas de tiempo de inactividad son inaceptables. Y por un 24% adicional, incluso una interrupción de 2 horas dañará su negocio.
- Una encuesta de Dunn & Bradstreet mostró que el 59% de las compañías de Fortune 500 experimentan 1.6 horas de tiempo de inactividad del sistema por semana o más.

Costos indirectos

Más allá de los costos directos de una falla del sistema de TI, existen costos adicionales, financieros y de otro tipo, que pueden afectar significativamente a un negocio. En algunos casos, estos pueden ser mucho más altos que los costos directos. Pero estos son mucho más difíciles de estimar.

Impacto en otros proyectos

- Según Harvard Business Review, el 27% de los proyectos de TI superan el presupuesto y el 70% no se completan a tiempo. A primera vista, esto podría parecer como cualquier otro costo hundido: el costo de hacer negocios. Pero no es tan simple. Cuando una falla de TI causa un

retraso en uno o más proyectos, el problema tiene un efecto compuesto en un negocio. El dinero que se gasta para completar el proyecto es dinero que no se puede gastar en otra cosa. Los empleados tienen que ser retirados de otros proyectos para satisfacer las necesidades de mano de obra no planificadas. Todos estos son costos adicionales que no se verán de inmediato, pero que se suman con el tiempo.

Daño a la reputación

- Las ventas perdidas no son solo un costo único. Para comprender realmente el impacto de una venta perdida, también debe considerar el valor de por vida de cualquier cliente perdido. En algunos casos, esto puede ser varias veces el costo directo inmediato.
- El daño a la reputación es un factor importante en las industrias de servicios financieros y de atención médica o en cualquier industria donde el acceso a la información personal de los clientes esté en riesgo. Si esta información se filtra debido a una violación de seguridad, la pérdida de confianza pública puede causar la pérdida de muchos negocios en el futuro.

Impacto regulatorio y de cumplimiento

- Si una falla de TI hace que una empresa no cumpla con la fecha límite de un cliente o no cumpla con las obligaciones contractuales, esos costos deberán reembolsarse a esos clientes. Peor aún, el incumplimiento de las condiciones de un SLA puede llevar a una empresa a problemas con los reguladores, lo que resulta en multas significativas.

Costos de remediación

- Más allá del costo directo de encontrar y solucionar el problema, después de una falla del sistema de TI a menudo es necesario un trabajo adicional para compensar esa falla.
- Estos costos pueden ser aún mayores si la reputación de la organización ha sido dañada, en cuyo caso podría ser necesaria una campaña de marketing completa solo para reparar el daño.

Impacto moral

- Más allá de la pérdida de ingresos y reputación, también está la cuestión de cómo una interrupción del sistema afecta a los empleados de una organización. Si el sistema se usa internamente, tiene un efecto directo en todos los involucrados. Para un gerente de TI en particular, incluso una breve interrupción puede sentirse como un fracaso personal. Los ejecutivos pueden sentirse presionados para encontrar a alguien a quien culpar, lo que puede erosionar aún más la moral.
- Una interrupción del sistema también desgasta a cualquiera que tenga que ayudar a reparar el daño. Las personas que necesitan trabajar horas extras están perdiendo tiempo con sus familias o renunciando a tiempo en sus pasatiempos. Si esto sucede con demasiada frecuencia, la fuerza laboral perderá la fe en su liderazgo y los mejores empleados comenzarán a solicitar trabajos con competidores.

Conclusión sobre los costos de falla del sistema operativo *: depende*.

Los factores más importantes para determinar los costos son el tipo de industria y la gravedad de la falla. Para el sector de los medios, la pérdida promedio por hora es de $ 90,000 por hora. Pero para las grandes corredurías financieras, las pérdidas ascienden a la asombrosa cifra de $ 6.48 millones por cada hora de tiempo de inactividad. La energía, los servicios financieros, la fabricación y las telecomunicaciones parecen ser las industrias con los mayores costos de tiempo de inactividad de TI. Los riesgos son particularmente altos para las empresas cuyo negocio requiere una alta tasa de capacidad de respuesta del cliente.

Suponiendo que nuestras estimaciones de costos en 2020 fueran cercanas, podemos proyectarlas hacia adelante en función de los cambios en las condiciones económicas. Debido a la inflación al 15% acumulativo, nuestros $ 1.56 billones estimados en 2020, se elevan a $ 1.8 billones en 2022. Sin embargo, creemos que el costo real es mucho más alto que nuestra estimación conservadora.

Solo en el área creciente del cibercrimen, si asumimos que los costos del cibercrimen son proporcionales a los de la economía mundial, entonces Estados Unidos tendría una participación del 24% basada en el tamaño relativo del PIB. El costo del cibercrimen en el mundo en 2022 se estimó en $ 6-7 billones. Por lo tanto, la participación de EE.UU. sería ~ $ 1.44 billones. Eso haría que el cibercrimen ~ 80% del costo total de todas las fallas operativas. Es dudoso que todos los demás tipos de fallas operativas representen solo el 20% de ese total.

En todas las demás áreas de COSTO DEL SOFTWARE DE BAJA CALIDAD simplemente asumimos que no hubo crecimiento, debido al desplazamiento de recursos escasos para hacer frente a fallas y deficiencias (a pesar de una tasa de inflación acumulada del 15%).