Módulo 3: Metodologías de Desarrollo de Software

Metodología de Desarrollo de Software:

- Definición.

  La metodología es un frameworks o marco de trabajo que permite la estructuración del proyecto generando así resultados óptimos según parámetros establecidos

- Aportes de aplicar una metodología.

  Ordena, contiene, permite definir límites.

  Aporta una garantia de calidad

  Aporta una forma de estimar y controlar costes

  Evita una perdida de los esfuerzos en rectificar fallos.

  Al ser un proceso estructurado tambien organiza la forma en la que el proyecto va ser realizado

Metodologías tradicionales y ágiles

- Tradicionales
  Prototipo Permite que todo el sistema, o algunos de sus partes, se construyan rápidamente para comprender con facilidad y aclarar ciertos aspectos en los que se aseguren que el desarrollador.
  Metodologia en cascada
  Es unproceso de desarrollo secuencial de proyectos que suele utilizarse en eldesarrollo de software.
  Metodologia en V
  Es un procedimiento uniforme para desarrollar productos de tecnologías de la información y la comunicación (TIC)
  Metodologia evolutiva
  Es el desarrollo de una implantación del sistema inicial, se la expone a los comentarios del usuario, se refina en N versiones hasta que se desarrolle el sistema adecuado
  Metodologia incremental
  El modelo incremental combina elementos del modelo líneal secuencial (aplicados repetidamente) con la filosofía interactiva de construcción de prototipos.
- Agiles
  Trabajar en equipo entre cliente y proveedor, donde sus integrantes colaboran con el fin de avanzas gradualmente y lograr la entrega del producto de calidad en tiempos y costos planeados
  Scrum
  Es un marco de trabajo o un framework que se utiliza dentro de equipos que manejan proyectos complejos, Es agil y flexible para gestionar

Etapas del proceso de desarrollo de software

- En qué consiste la etapa de planeación

  Corresponde al momento inicial del proyecto, es donde se habla con los clientes, se identifica todo lo que se necesita para levantar el proyecto, es todo lo que se hace antes de la codificación.

- En qué consiste el Análisis de requerimientos

  Extraer los requisitos de un producto de software, es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere la habilidad y la experiencia en la ingeniería de software para reconocer requisitos incompletos.

- Técnicas que se usa para levantar requerimientos

  Analisis de documentación
  Observacion
  Entrevistas
  Encuestas o cuestionarios
  Mesas de trabajo
  Tormenta de ideas
  Historia de usuario

- En qué consiste el Diseño de Sistemas: (UML, diagramas de: Casos de Uso, Diagramas de Clases, etc. a nivel general)

  UML (Lenguaje Unificado de Modelado) Es un lenguaje grafico para visualizar, especificar, construir y documentar un sistema.

  Casos de Uso **(Diagrama de clases)** para cumplir las funciones que realizara el sistema, se transforman las entidades definidas en el análisis de requisitos en clases de diseño obteniendo un modelo cercano a la programación orientada a objetos

  Diagrama de clases **(Casos de Uso)** se refiere a determinar como funcionara de manera general sin entrar en detalles. Consiste en incorporar consideraciones de la implementación tecnológica como el hardware, la red, etc.

- En qué consiste la Etapa de codificación

  Reducir un diseño a código puede ser la parte mas obvia del trabajo de ingeniería de software, pero no es necesariamente la porción mas larga. La complejidad y la duración de esta etapa depende de los lenguajes utilizados

- En qué consiste la Etapa de Pruebas: (Funcionales y no funcionales)

  Consiste en comprobar que el software realice correctamente las tareas indicadas en la especificación. Una técnica de prueba es probar por separado cada modulo, luego de manera integral para asi llegar al objetivo
  Existen os formas o enfoques el primero con personas inexpertas que comprueben que las tareas se realicen correctamente, el segundo enfoque esta conformado por un grupo de programadores con experiencia que buscaran el fallo del producto

- En qué consiste la Implementación

  Todo lo concerniente a la documentación del propio desarrollo del software y la gestión del proyecto, pasando por modelaciones UML, diagramas, pruebas,manuales etc. Todo esto

con el propósito de posibles correcciones, usabilidad, mantenimiento y ampliaciones del sistema

- En qué consiste la etapa de Mantenimiento

  Mantener y mejorar el software para enfrentar errors descubiertos y nuevos requisites. Una, esto puede llevar mas tiempo incluso mas que el desarrolllo inicial del software, Alrededor de 2/3 partes de software tiene que ver con dar mantenimiento, pequeña parte consiste en arreglar errores el otro porcentaje tiene que ver comn ampliaciones.

Seguridad de Software

- Introducción, definición de Seguridad de Software
  Las **vulnerabilidades de alta severidad** dan lugar a que un atacante pueda explotarlas rápidamente y hacerse con el **control total del sistema**.
  Se sigue confiando que la implantación de tecnologías y dispositivos de seguridad de red.
  Se hace necesario por tanto el disponer de **software seguro que funcione en un entorno agresivo y malicioso**.
  Confianza y garantía de funcionamiento conforme a su especificación y diseño
  Para conseguir minimizar al máximo los ataques en la capa de aplicación y por tanto en número de vulnerabilidades explotables, **es necesario el incluir la seguridad desde principio en el ciclo de vida de desarrollo del software (SDLC)**

- Principales causas de aparición de las vulnerabilidades
  Tamaño excesivo y complejidad de las aplicaciones.
  Mezcla de código proveniente de varios orígenes
  Integración de los componentes del software defectuosa
  Debilidades y fallos en la especificación de requisitos y diseño
  Uso entornos de ejecución con componentes que contienen vulnerabilidades o código malicioso embebido.
  No realización de pruebas seguridad basadas en riesgo.
  Falta de la herramientas y un entorno de pruebas adecuados.
  Cambios de requisitos del proyecto durante la etapa de codificación.

- Propiedades complementarias de un software seguro
  Fiabilidad.
  Autenticación.
  Trazabilidad.
  Robustez.
  Resiliencia.
  Tolerancia

- Principios de diseño de seguridad de software
  Defensa en profundidad. Introducir múltiples capas de seguridad para reducir la probabilidad de compromiso del Sistema

  Simplicidad del diseño. Reducir la complejidad del diseño para minimizar el número de vulnerabilidades y debilidades en el sistema explotables por el atacante

  Mínimo privilegio. Principio según el cual se concede a cada entidad (usuario, proceso o dispositivo) el conjunto más restrictivo de privilegios necesarios para el desempeño de sus tareas autorizadas

  Separación de privilegios. Asignación a las diferentes entidades de un rol que implique el acceso a un subconjunto de funciones o tareas y a los datos necesarios

  Separación de dominios. Minimizar la probabilidad de que actores maliciosos obtengan

fácilmente acceso a las ubicaciones de memoria u objetos de datos del sistema.

Separación código, ejecutables, datos configuración y programa. Reducir la probabilidad de que un ciberatacante que haya accedido a los datos del programa fácilmente pueda localizar y acceder a los archivos ejecutables y datos de configuración del programa.

Entorno de producción o ejecución inseguro. Evitar vulnerabilidades aplicando una serie de principios de validación de las entradas.

Registro de eventos de seguridad. Generar eventos (logs) de seguridad, para garantizar las acciones realizadas por un ciberatacante se observan y registran

Fallar de forma segura. Reducir la probabilidad de que un fallo en el software, pueda saltarse los mecanismos de seguridad de la aplicación, dejándolo en un modo de fallo inseguro vulnerable a los ataques.

Diseño de software resistente. Reducir al mínimo la cantidad de tiempo que el componente de un software defectuoso o fallido sigue siendo incapaz de protegerse de los ataques

La seguridad por oscuridad es un error. Concienciarse de que la seguridad por oscuridad es un mecanismo de defensa que puede proporcionar a un atacante información para comprometer la seguridad de una aplicación

Seguridad por defecto. Reducir la superficie de ataque de una aplicación o Sistema

- Amenazas a la seguridad de software
  Es mucho más fácil encontrar vulnerabilidades en el software, que desarrollarlo seguro.
  La perspectiva del ciberatacante → buscar vulnerabilidades en el sistema bajo la visión de afuera hacia adentro.
  Requiere comprender la tipología y motivaciones que hay detrás de los ataques a fin de entender realmente los riesgos asociados y analizar el software de la manera que lo harían para atacarlo.

- Elementos claves de un proceso de SDLC
  Pruebas de seguridad.
  Despliegue y distribución segura.
  Sostenimiento seguro.
  Herramientas de apoyo al desarrollo.
  Gestión de configuración de sistemas y procesos.
  Conocimientos de seguridad de los desarrolladores.
  Gestión segura del proyecto y compromiso de la alta

Metodologías de Desarrollo de Software Seguro

- Introducción a la seguridad de S-SDLC
  Aumento de los ataques al software vulnerable, capa aplicación.
  Insuficiencia de las protecciones a nivel de infraestructura.
  Minimizar al máximo los ataques en la capa de aplicación y número de vulnerabilidades explotables.
  El desarrollo de software seguro y confiable requiere la adopción de un proceso sistemático o disciplina.
  Abordar la seguridad en cada fases del ciclo de vida:
  Principios de diseño seguro(mínimo privilegio, etc.)
  Buenas prácticas de seguridad (especificación requisitos seguridad, casos de abuso, análisis de riesgo, análisis de código, pruebas de penetración dinámicas,etc.).

A este nuevo ciclo de vida con prácticas de seguridad incluidas lo llamaremos S-SDLC.
Una de las principales ventajas de la adopción de un S-SDLC es el descubrimiento de vulnerabilidades y debilidades en sus etapas tempranas y su importante ahorro de costes.
Requiere hacer algunos cambios en la forma de construir software en la mayoría de las organizaciones

- Seguridad en la Fases de S-SDLC
  La Seguridad del Software es algo más que la eliminación de las vulnerabilidades y realización de pruebas de penetración, es una disciplina.
  Adopción de un enfoque sistémico para incorporar buenas prácticas de Seguridad del Software "touchpoint" y unos hitos de control.

- Casos de abuso
  Es la inversa de un caso de uso es decir una función que el sistema no debe permitir o una secuencia completa de acciones

- Ingeniería de requisitos de seguridad
  (Fundamentales, Generales, Operacionales)
  Completos
  Coherentes
  Precisos
  Trazables
  Verificables
  (Politica, Normativa, Casos, Analisis, Implicaciones, Avances, Reuniones, Vulnerabilidades, Requisitos)

- Análisis de riesgo arquitectónico
  Mitigar gran parte de los problemas mediante análisis y autoataques

- Patrones de diseño
  Esta destinado a contribuir con el diseño menos vulnerable mas resistente y tolerante a los ataques

- Pruebas de seguridad basados en riesgo
  Pruebas funcionales sirven para determinar el comportamiento del sistema ante un ataque (Verificar la operación, la fiabilidad, la falta de defectos, la capacidad de supervivencia)

- Modelado de ataques
  Un método sistematico para caracterizar la seguridad de un sistema basado en la combinación y dependencias de las vulnerabilidades del mismo que un atacante puede aprovechar.

- Revisión de código
  Debe ser la actividad as importante en seguridad ya que se han de realizar en el curso del desarrollo

- Pruebas de penetración
  Incluye los peores escenarios en los que puede haber ataques
  (Politicas, amenazas, riesgos, secuencias)

- Configuraciones seguras

- Operaciones de seguridad
  Reducir al minimo las posinilidades de acceso y manipulación del software
  (Distribucion, despliegue, operaciones)

- Revisión externa
  Revision de personal externo ajeno al desarrollo