SCRUM

* Definición: marco de trabajo que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas a problemas complejos
* Características
* Cimientos
  + Empirismo: experiencia es la base
  + Autoorganización: de los grupos de trabajo
  + Colaboración: entre los que participen
  + Priorización: no focalizar en lo que no agregar valor
  + Time Boxing: técnica de planificación de proyectos
* Teoría de Scrum: se basa en el empirismo y el pensamiento Lean
* Pilares empíricos
  + Transparencia: proceso y trabajo visibles para todos
  + Inspección: artefactos y progreso hacia los objetivos deben inspeccionarse
  + Adaptación: si un aspecto del proceso se desvía fuera de los límites se debe ajustar
* Valores de Scrum: compromiso, foco, franqueza, respeto y coraje
* Scrum Team: 10 personas o menos
  + Scrum Master: establecer Scrum como se define en la guía, lograr la efectividad del Scrum Team. Le sirve al PO, al Scrum Team y a la organización
  + Product Owner: maximizar el valor del producto resultante del trabajo del Scrum Team. Gestión del Product Backlog
  + Developers: se comprometen a crear cualquier aspecto de un incremento utilizable en cada Sprint
* Eventos de Scrum: Sprint como contenedor de todos
  + Sprint Planning: iniciar el Sprint al establecer el trabajo que se realizará en este.
    - ¿Por qué es valioso este Sprint?
    - ¿Qué se puede hacer en este Sprint?
    - ¿Como se realizará el trabajo elegido?
  + Daily Scrum: inspecciona el progreso hacia el objetivo del Sprint y adapta el Sprint Backlog si es necesario. 15 minutos todos los días del Sprint. Participan los Developers
  + Sprint Review: inspecciona el resultado del Sprint y determina futuras adaptaciones. Se presentan resultados del trabajo. Participa el equipo más los invitados del PO. Se calcula la velocidad del equipo. Se mejora el producto
  + Sprint Retrospective: planifica formas de aumentar la efectividad y la calidad. Identifica cambios útiles. Concluye el Sprint. Se mejora el proceso.
* Artefactos de Scrum: representan trabajo o valor. Diseñados para maximizar la transparencia
  + Product Backlog: compromiso Objetivo del Producto (es a largo plazo). Lista emergente y ordenada de lo que se necesita para mejorar el producto. Ceremonia constante: refinamiento
  + Sprint Backlog: compromiso Objetivo del Sprint. Compuesto del objetivo del Sprint (por qué), el conjunto de elementos del Product Backlog (qué), y un plan de acción para entregar Increment (cómo). Se inspecciona su progreso en la Daily
  + Increment: compromiso Definition of Done. Es un peldaño concreto hacia el objetivo del producto. Debe ser utilizable. Pueden crearse múltiples Increments dentro de un sprint. Se presenta en la Sprint Review solo si cumple con la DoD
* Time Box
  + Sprint: 1 mes o menos
  + Sprint Planning: 8 horas máximo – S de 1 mes
  + Daily Meeting: 15 minutos
  + Sprint Review: 4 horas máximo – S de 1 mes
  + Sprint Retrospective: 3 horas máximo – S de 1 mes
  + Refinamiento del PB: 10% del tiempo del S
* Capacidad del equipo: métrica para determinar cuánta cantidad de trabajo puede comprometer un equipo para un determinado S. Se estima en:
  + Story Points: para equipos más maduros
  + Horas ideales: tiempo haciendo cosas realmente. Para equipos menos experimentados
  + Días disponibles + días para otras actividades Scrum + horas por día + horas de esfuerzo disponibles
* Herramientas de Scrum
  + Taskboard: tablero para realizar seguimiento del trabajo realizado. Forma de asignación de compromiso: Story, To Do, In Process, To Verify, Done. Usa tarjetas de tareas. El tablero usa Sistema PULL y una granularidad fina.
  + Gráficos del Backlog: brindan información sobre el progreso de un Sprint, Release o producto. Se compara la cantidad de trabajo que queda por hacer -en horas- y el tiempo -en días-
    - Sprint Burndown Chart: curva descendente de cuanto trabajo queda por terminar
    - Release BurnUp Chart: curva ascendente de cuando trabajo hemos hecho en cada Sprint.
* Niveles de planificación
  + Portfolio -1 año o más- = backlog de portfolio
  + Producto -varios meses- = visión de producto, roadmap y características de alto nivel
  + Release -3 o menos a 9 meses- = plan de release
  + Iteración -1 a 4 semanas- = objetivo del sprint y sprint backlog
  + Día -diaria- = inspección del progreso actual y adaptación

NEXUS

* Framework que consiste en roles, artefactos, técnicas y eventos que vinculan el trabajo de aproximadamente 3 a 9 equipos Scrum. Trabajan sobre un Product Backlog en común. Un solo PO. Un solo producto.
* Lidia con la complejidad de varios equipos trabajando sobre un PB. Dada por las dependencias
  + Requerimientos: alcance y forma en que se implementan
  + Conocimiento del dominio: debe mapearse a los equipos Scrum para minimizar interrupciones.
* Responsabilidades
  + Nexus Integration Team: aseguran que se produzca al menos un incremento integrado por Sprint
  + PO: maximiza el valor del producto y el trabajo realizado e integrado por los Scrum Teams ST en un Nexus. Gestiona el PBacklog
  + Scrum Master: asegura que el marco de trabajo Nexus se entienda y se promulgue
  + Uno o más miembros del NIT: miembros del ST que ayudan a los ST a adoptar herramientas y técnicas para mejorar su capacidad
* Eventos
  + Nexus Sprint: los ST producen un solo Integrated Increment.
  + Refinamiento entre equipos: reduce o elimina las dependencias entre equipos dentro de un Nexus. El PB debe descomponerse para esto. Es continuo
  + Nexus Sprint Planning: coordinar actividades de todos los ST dentro de un Nexus para un solo Sprint. Resultados…
    - Objetivo del Nexus
    - Un Nexus Sprint Backlog
    - Un Sprint Backlog
  + Nexus Daily Scrum: inspecciona el progreso hacia el objetivo del Sprint del Nexus. Asisten representantes de cada ST. La Daily Scrum de cada ST complementa la Nexus Daily Scrum
  + Nexus Sprint Review: reemplaza a las Sprint Reviews de cada ST
  + Nexus Sprint Retrospective: planifica formas de mejorar la calidad y eficacia en todo el Nexus. Las Retrospectives de los ST complementan la Nexus Sprint Retrospective
* Artefactos: representan trabajo o valor y estan diseñados para maximizar la transparencia
  + Product Backlog: hay uno solo que contiene la lista de todo lo que el Nexus y los ST necesitan para mejorar el producto. Compromiso es el objetivo del producto
  + Nexus Sprint Backlog: compuesto por el objetivo del Sprint del Nexus y elementos del PB de cada ST. Se va actualizando a medida que hay conocimiento. Compromiso es el objetivo del Sprint del Nexus, es la suma de todo el trabajo y los objetos de Sprint de los ST. Se crea en la Nexus Planning y se agrega al SB del Nexus.
  + Integrated Increment: es la suma de todo el trabajo integrado completado por un Nexus en relación con el objetivo del producto. Se inspecciona en la Nexus Sprint Review. El compromiso es la DoD.

FILOSOFÍA LEAN

* Surge en Japón con Toyota en los 40s. Hay menos reglas que en Agile.
* Filosofía de gestión que busca maximizar el valor generado al cliente, con el mínimo consumo de recursos y bajo la teoría de que todo esfuerzo que no produce valor se considera desperdicio.
* Optimiza el consumo de recursos, elimina o minimiza desperdicios y hace foco en el lugar donde se crea valor
* Sentencia: tomar lo que tengas, y a partir de ahí tratar de aplicar los principios y de mejorar lo que hoy ya tienes.
* 7 principios: tratan de optimizar lo que tiene que ver con la gestión del cambio. Esta gestión tiene que ver con temas individuales, no del proyecto.
  + Eliminar desperdicios: desperdicio como aquello que no agrega valor y que implica trabajo innecesario o retrabajo. Evitar: producir funcionalidades en exceso, retrabajo, que los artefactos se vuelvan obsoletos antes de terminarlos. 7 desperdicios:
    - Características extras
    - Trabajo a medias
    - Pasos extra en el proceso (como en los procesos definidos)
    - Búsqueda de información (debe ser de facil acceso)
    - Defectos (errores que se trasladan)
    - Esperas (cuando poder hacer mi trabajo depende del trabajo de otro)
    - Cambios de tareas (terminar tarea antes de pasar a la que sigue)
  + Amplificar aprendizaje: transparencia y transformación de conocimiento implícito en explicito. Aprendizaje colectivo. Intercambiar experiencias. Aprender de lo que se está haciendo. Conocimiento en forma de espiral
  + Tomar decisiones tardías: tomar decisiones en el último momento responsable. Postergar la toma para tener la mayor cantidad de información al hacerlo. Poder avanzar si tener todo definido
  + Entregar lo antes posible: entregarle software al cliente antes de que las necesidades cambien. Acotar los ciclos de desarrollo permite salir al mercado con un producto minimo que sea valioso. Entregar SW funcionando rápido.
  + Potenciar el equipo: equipos autoorganizados. Darles poder de decisión y libertad de acción. Delegación de decisiones y de responsabilidades
  + Crear la integridad: incluir todas las partes desde el principio, no focalizarse en unas y olvidarse de otras. Contemplar todas las partes de manera consistente. Incluye revisiones entre pares. También incluye calidad: no hacer retrabajo sino construir con calidad desde el principio.
    - Integridad percibida: el producto total tiene un balance que le gusta a la gente
    - Integridad conceptual: todos los componentes del sistema trabajan de forma coherente en conjunto
  + Visualizar todo el conjunto: visión holística que permite asociar y comprender todo. Los objetivos particulares de las partes están alineados con los objetivos globales.

KANBAN

* Una manera de aplicar Lean, es utilizar Kanban
* Framework para definir, gestionar y mejorar servicios que entregan trabajo del conocimiento.
* Enfoque usado para la gestión del cambio. Proceso para gestionar el cambio.
* Usa el principio “comenzar por donde estes”. No plantea introducir cambios revolucionarios, sino mejoras graduales a procesos de desarrollo ya existentes.
* Surge el concepto JUST IN TIME -como no tener stock de productos-
* Aplica administración de colas. Ejemplo de cajero y barista.
* Kanban en el desarrollo: buscamos que el proceso fluya de izquierda a derecha. Tratamos de que el WIP se limite para evitar embotellamiento
* Valores
  + Transparencia: compartir información mejora el flujo.
  + Equilibrio: diferentes aspectos, POV y capacidades deben ser equilibrados para conseguir efectividad
  + Colaboración: Kanban fue creado para mejorar la manera en la que las personas trabajan juntas.
  + Foco: en el cliente y en el valor que estos reciben.
  + Flujo: la realización del trabajo es el flujo de valor, tanto si es continuo o puntual.
  + Liderazgo: sin jerarquías. Es necesario en todos los niveles
  + Entendimiento: conocimiento de si mismo (individual y de la organización). Conocer el punto de inicio para ir mejorando
  + Acuerdo: compromiso de avanzar juntos hacia los objetivos.
  + Respeto: consideración por las demás personas.
* Principios
  + Gestión de cambios
    - Comenzar con lo que haces ahora: entender los procesos actuales tal y como estan siendo realizados, respetar los roles, las responsabilidades de cada persona y los puestos de trabajo
    - Acordar la búsqueda de la mejora a través del cambio evolutivo
    - Fomentar actos de liderazgo a todos los niveles
  + Entrega de servicios
    - Comprender y hacer foco en cumplir las expectativas del cliente y sus necesidades
    - Gestionar el trabajo: dejar que los equipos se autoorganicen
    - Revisar periódicamente la red de servicios y sus políticas para mejorar los resultados entregados.
* Practicas generales
  + Visualizar: usa tableros visuales para representar el flujo de trabajo, mostrando tareas pendientes, en progreso y terminadas. Brinda visión. El trabajo se divide en piezas como US, features, bugs, y se escribe en tarjetas señalizadoras. Cada etapa es representada por una columna (de producción o acumulación) por donde las Kan van pasando. Sistema de arrastre, sistema de colas.
  + Limitar el WIP: limita la cantidad de trabajo en curso en cualquier momento dado. Evita la sobrecarga del equipo. Limitamos para asegurar flujo y equilibrar la ocupación. Ej.: 2 Kan como máximo en WIP
  + Administrar el flujo: foco en lograr un flujo continuo y eficiente, se deben identificar y resolver cuellos de botellas y otros problemas. Ej.: autopista
  + Crear políticas explicitas: requiere de políticas y procedimientos claros y explícitos para todos los miembros del equipo. Definir criterios de finalización de tarea, prioridades o cualquier regla que afecte el proceso.
  + Establecer ciclos de feedback: implica establecer ciclos de retroalimentación con la cadencia adecuada, que favorezcan la colaboración, aprendizaje y las mejoras, con una base en los datos obtenidos en la retro. Definir frecuencia de reuniones.
  + Mejorar la colaboración: mejora continua a través de la colaboración y el aprendizaje. Cultura de mejora continua donde no hay cambios significativos en los procesos, sino que se identifican, visualizan y proponen mejoras graduales.
* Como aplicar Kanban
  + 1. Empezar con lo que se tiene: entender que se tiene, como trabaja el equipo y que hace, poder graficarlo en un tablero.
  + 2. Modelar el proceso -en el tablero-
    - Dividir el trabajo en piezas -definir unidad de trabajo: US, CU, tarea- se pueden usar clases de servicio
    - Visualizar el flujo de trabajo
  + 3. Limitar el WIP: limitar cada etapa, es decir máximo de tarjetas que puedo tener en progreso en cada una de las etapas. Vinculado directamente con la cantidad de recursos que tenga trabajando en esa parte del proceso.
  + 4. Definir políticas para cada clase de servicio identificada: especificar que criterios deben cumplirse para mover una tarea a otra columna, así como los limites de WIP para cada columna.
  + 5. Implementar Kanban empezando con columnas y tarjetas que reflejen el flujo actual
  + 6. Identificar cuellos de botella y resolverlos: se puede considerar mover recursos de una etapa a otra para equilibrar el flujo.
  + 7. Establecer cadencia para las actividades del proyecto
    - Team Kanban Meeting: diaria
    - Team Retrospective: quincenal o mensual
    - Team Replenishment Meeting: semanalmente o a demanda -seleccionar elementos de la lista de trabajo para realizar-

MÉTRICAS

* Definición: el monitoreo y control, esencial para detectar desvíos y corregirlos, se materializa a través de las métricas. Estas nunca tienen que usarse para medir personas: no reflejan adecuadamente el valor que una persona aporta al equipo.
* No sirve tener tantas métricas porque se pierde mucho tiempo en recolectarlas y despues no llegan a analizarse. El tiempo de demora debe ser minimo. No todas las métricas útiles en un proyecto lo son en otro. 🡪 me debo preguntar
  + ¿Nos da más información de la que tenemos ahora?
  + ¿Es esta información de beneficio practico?
  + ¿Nos dice lo que queremos saber?
* Recordar la triple restricción. Alcance tiempo y recursos.
* Ambientes tradicionales: basado en procesos definidos con un amplio conjunto de métricas a definir. La definición queda a cargo del líder de proyecto. 3 conjuntos denominados dominios de métricas: ámbito al que hacen referencia, en que sirven y son útiles.
  + Métricas de producto: aquellas que apuntan a defectos. Permiten observar que esta pasando con el producto que se está construyendo
  + Métricas de proyecto: tienen su aplicación en el proyecto donde se construye un producto. La utilidad termina cuando termina el proyecto. Utilidad refiere a lograr corregir algo que en el proyecto no esta funcionando bien.
    - Tamaño del producto: que tan grande y complejo o pequeño es. Métrica del producto. Se estima en función de los R, CU, US, features.
    - Esfuerzo: horas lineales. Métrica del proceso y proyecto. Depende del tamaño
    - Tiempo calendario: métrica del proceso y proyecto
    - Defectos: métrica del producto. Es una función del tamaño y el esfuerzo. Ej.: cantidad de defectos x punto de función. Proyectos mas grandes tienden a producto más defectos x línea de código.
      * Cobertura: mientras mas mejor. Nos da confianza para liberar el producto. Nos dice que tanto de nuestro SW está cubierto por los tests.
      * Defectos por severidad: ej.: % de defectos de severidad X
      * Densidad de defectos: se suele medir en cantidad de errores por línea de código.
  + Métricas de proceso: organizacionales o estratégicas. Las métricas de procesos son una despersonalización de las del proyecto. Busco ver si los proyectos terminan en tiempo y forma. Debo ajustar desvíos. Permiten saber donde aplicar mejoras, saber como hacer esto de la mejora continua.
* Ambientes agiles: acá se establecen pautas claras sobre que métricas son relevantes. Como la prioridad es satisfacer al cliente con entregas tempranas y continuas de software valioso, la principal medida es el software funcionando, por lo que se toma a las métricas como una salida, no como actividad. Medir solo lo necesario, lo que agregue valor al cliente.
  + RTF: mide la cantidad de features testeadas que estan funcionando, osea cuantas piezas de producto se terminaron y estan en ejecución.   
    Problema: no tiene en cuenta la complejidad.  
    Utilizada para obtener una visión general del avance del software en términos de features completas y testeadas.   
    Útil para evaluar estabilidad y evaluar la tendencia en el tiempo.
    - Running: funcionalidad entregada en un producto
    - Tested: funcionalidad desarrollada que paso las pruebas de aceptación
    - Features: dado que incluye aspectos determinados por el cliente.
  + Velocidad: complementa RTF. Capacidad del equipo para completar el trabajo por iteración. Cantidad de trabajo que un equipo puede completar. NO se estima, se calcula. Queda determinada por la cantidad de Story Points que el PO acepto, y se toma en la SReview.   
    Es la métrica mas importante del enfoque.   
    Suele representarse por un gráfico de barras para medir estabilidad. Gráficos son permanentes durante todo el proyecto y visibles para todos
  + Capacidad: la velocidad nos ayuda a determinarla. Estimación de cuanto trabajo puede completarse en un periodo de tiempo basado en la cantidad de tiempo ideal disponible del equipo. Se estima al principio del sprint. Se puede estimar en
    - Horas ideales
    - Story Points
* Ambientes Lean con Kanban
  + Cycle time: tiempo de ciclo. tiempo que sucede entre el inicio y el final del proceso, para un ítem dado. Se suele medir en días de trabajo o esfuerzo. Se lo conoce como ritmo de terminación.
  + Lead time: tiempo de entrega. Tiempo que sucede entre el momento en el cual se pide un ítem de trabajo y el momento en que se entrega (final del proceso). Se suele medir en días de trabajo. Se lo conoce como ritmo de entrega
  + Touch time: tiempo de tocado. Tiempo en el cual un ítem de trabajo fue realmente trabajado o tocado por el equipo. Cuantos días hábiles paso este ítem en columnas de WIP en oposición con columnas de cola o buffer.
  + Se busca obviamente que el Touch Time sea lo mas parecido posible al Cycle Time.
  + Eficacia del proceso se mide como

Texto

Descripción generada automáticamente

TESTING

* Asegurar la calidad vs Controlar la calidad: el control es una de las tareas dentro del aseguramiento. La calidad del producto depende de la calidad del proceso, por lo que debe ser controlado. Detectar errores en forma temprana ahorra esfuerzo, tiempo y recursos.
* Testing se enfoca en los riesgos, disminuyéndolos. Aumenta la confianza en el producto. Es una actividad destructiva, que trata de encontrar defectos. Es una de las tareas en el aseguramiento de calidad conjunto con el control de calidad de proceso y producto. El testing NO puede asegurar calidad. Asumimos que HAY defectos
* Aseguramiento de calidad
  + Control de calidad del proceso
  + Control de calidad del producto
  + Testing de software
* Principios
  + Un programador debería evitar probar su propio código.
  + Una unidad de programación no debería probar su propio desarrollo
  + Examinar el SW para probar que no hace lo que se supone que debería hacer es la mitad de la batalla, la otra es probar que hace lo que no se supone que debería hacer.
  + No planificar el esfuerzo del testing sobre la suposición de que no habrá errores.
  + El testing es una tarea extremadamente creativa e intelectualmente desafiante
  + Testing temprano. Cuando se hacen los R es el momento de empezar a probar.
  + Los test cases deben ser actualizados y rediseñados para evitar la paradoja del pesticida -al corregir errores se deja un residuo de errores sutiles-
  + El testing es dependiente del contexto
  + Falacia sobre la ausencia de errores. El testing es necesario porque la existencia de estos es inevitable.
* Mitos
  + El testing es el proceso para demostrar que los errores no estan presentes
  + El propósito es demostrar que un programa realiza las funciones previstas de forma correcta
  + El testing es el proceso de demostrar que un programa hace lo que se supone que debe hacer

Entonces no se prueba un sistema para demostrar que funciona, sino que se comienza con la suposición de que contiene defectos y se realiza el testing para encontrar la mayor cantidad posible.

* ¿Cuánto testing es suficiente?
  + El testing exhaustivo es imposible por la cantidad de tiempo que requiere
  + El momento de dejar de testear depende del nivel de riesgo o costo asociado al proyecto, y también del criterio de aceptación. Pueden ser definidos según:
    - Costos
    - % de test corridos sin fallas
    - Inexistencia de defectos de una determinada severidad
    - Pasa exitosamente el conjunto de pruebas diseñado y la cobertura estructural
    - Good enough: cierta cantidad de fallas no críticas es aceptable
    - Defectos detectados similar a la cantidad de defectos estimados.
* Error vs Defecto
  + La diferencia radica en el momento en que se advierte
  + Error: es una falla que se identifica en la misma etapa en que fue originada.
  + Defecto: es aquella falla que se identifica en una etapa posterior a aquella en que se originó, es decir un error que se traslado a una etapa posterior.
  + Nosotros encontramos defectos ya que justamente inspeccionamos cosas que se hicieron en la etapa de implementación.
* Defectos: Severidad y Prioridad
  + Severidad: define la gravedad del defecto y la determina el tester. Determina el impacto. Estas características son independientes entre sí y su severidad depende del tipo de negocio que se esté desarrollando:
    - Bloqueante: no permite continuar la ejecución del sistema
    - Critico: el sistema funciona, pero con un defecto critico
    - Mayor: la funcionalidad funciona, pero no de la manera correcta
    - Menor: la funcionalidad se ejecuta correctamente, pero con advertencias erróneas.
    - Cosmético: formatos
  + Prioridad: define la urgencia, desde el lado del negocio, con que debe ser corregido. Depende del contexto
    - Urgencia
    - Alta
    - Media
    - Baja
* Casos de prueba: es la unidad de la actividad de prueba. Es un conjunto de condiciones o variables bajo las cuales un tester puede determinar si el SW esta funcionando correctamente o no. La buena definición del CP nos ayuda a reproducir los defectos. Si no se pueden reproducir no son defectos.   
  Consiste en ejecutar una serie de pasos o acciones en una determinada funcionalidad, determinando por completo con que valores voy a hacer la ejecución para ver si el SW me da los resultados esperados o no. Partes:
  + 1. Objetivo: característica del sistema a comprobar
  + 2. Datos de entrada y de ambiente: valores o datos a introducir al sistema y que se encuentran en condiciones preestablecidas
  + 3. Comportamiento esperado: salida esperada en el sistema de acuerdo con los R del este.
* Estrategias de prueba: usadas para poder definir la menor cantidad de CP que cubran el sistema de la manera lo mas completa posible. Apuntan a probar los límites, aquellos lugares que son mas propensos a errores.
* Derivación de Casos de Prueba: de donde saco la información para armar los CP. Recordar que ninguna técnica es completa, cada una ataca distintos problemas y por ende es mejor combinarlas.
  + Documentos del cliente
  + Requerimientos: si estan definidos de manera muy general, hay que especificarlos.
  + Información de relevamiento
  + Especificaciones de programación
  + Código
* Condiciones de Prueba: representan las circunstancias bajo las cuales se llevarán a cabo las pruebas. Debe ser probada por al menos una prueba
* Ciclos de Prueba: un ciclo de prueba abarca la ejecución de la totalidad de los CP establecidos aplicados a una misma versión del sistema. Generalmente se hacen 2 ciclos, el ciclo 0 que siempre es manual, donde se configura todo, y luego el ciclo 1 donde se pueden automatizar las pruebas. Esto en loop hasta cumplir con el criterio de aceptación:
  + 1. Ejecutar todos los casos
  + 2. Ver cuales fallaron
  + 3. Corregir defectos (mandar a desarrollo)
  + 4. Comenzar otro ciclo
* Regresión: cuando concluye un ciclo de pruebas, y reemplazarse la versión del sistema sometida a estas, se debe realizar una verificación total de la nueva versión, a fin de prevenir la introducción de nuevos defectos al intentar solucionar los detectados.
* Niveles de prueba: determinan el foco o la granularidad de la prueba por ejecutar. se comienza por los componentes pequeños, al contrario del desarrollo. Es a donde pongo la mira a la hora de testear.
  + Testing unitario: se hacen sobre un componente de manera independiente. Se hacen teniendo acceso al código fuente, por lo que lo hace el mismo desarrollador. Herramientas: depuración, automatización. Foco en aspectos puntuales y aislados. Se encuentran errores, no defectos.
  + Testing de integración: es cuando se prueba o verifica el funcionamiento de dos o más componentes juntos que se relacionen. Cada componente debe haber sido probado unitariamente. Resultado: build. Las hace un tester. Se hace una integración de manera incremental, desde lo mas general donde se abarcan más componentes, hacia lo más específico (top-down), o desde lo más específico a lo más general (bottom-up)
  + Testing de sistema: es la prueba cuando el sistema está funcionando como un todo. Pruebo el sistema en toda su escala.
    - Determinar si el sistema en su globalidad opera satisfactoriamente
    - El entorno de prueba debe corresponder al entorno de producción (donde se ejecutará el SW de manera real) tanto como sea posible
    - Investigar RF y RNF
  + Testing de aceptación: las realiza el usuario para determinar si la aplicación se ajusta a sus necesidades. La idea es establecer confianza, no es el foco encontrar defectos. Se realizan en el despliegue. Comprende:
    - Pruebas alfa: ejecutadas por el usuario en ambiente de laboratorio
    - Pruebas beta: ejecutadas en ambientes de trabajo reales.
* Ambientes: refieren a los diferentes entornos utilizados en el ciclo de vida de desarrollo de SW. Cada ambiente cumple con un propósito especifico y se utiliza en diferentes etapas del desarrollo y despliegue.
  + Ambiente de Desarrollo: es donde los desarrolladores crean, prueban y depuran el SW. Entorno local o de red para escribir y probar antes de integrar el código con el resto del sistema, sin afectar entornos de producción. Se realizan pruebas unitarias
  + Ambiente de Prueba: suele ser una réplica o tener características del entorno de producción, pero sin afectar usuarios finales y sin TODAS las características. Se llevan a cabo pruebas exhaustivas del software desarrollado para comprobar que cumpla con los R establecidos. Se realizan pruebas de integración.
  + Ambiente de Preproducción: se realiza una prueba final de todo el SW antes de su lanzamiento. Se simulan las condiciones del entorno de producción. Pruebas de ultimo minuto como de RNF. Validar que este listo para ser implementado en producción. Se realizan pruebas de sistemas.
  + Ambiente de Producción: es cuando el SW está en funcionamiento y es accesible por usuarios finales. Es el entorno real. Es altamente controlado, se implementan medidas de respaldo y recuperación ante desastres para garantizar disponibilidad. Se realizan pruebas de aceptación.
* Proceso de Pruebas: testing es un proceso definido que se lleva a cabo durante toda la vida del producto, por ende, tiene etapas detalladas. Es un proceso estándar.
  + 1. Planificación: es la etapa de verificar que todos entienden las metas y los objetivos del cliente, los stakeholders, el proyecto y los riesgos de las pruebas que se quieren abordar. Determina como se incluirá el testing en el plan de proyecto, que entornos se emulará, quien hará cada cosa, cuando, etc. El resultado de la planificación es el plan de prueba, que contiene:
    - Riesgos y objetivos del testing
    - Recursos
    - Estrategia de testing
    - Criterio de aceptación

El aspecto de controlar implica:

* + - Revisar los resultados
    - Test coverage y criterio de aceptación
    - Tomar decisiones.
  + 2. Diseño (identificación y especificación de CP): teniendo en cuenta las bases de la planificación se identifican los datos necesarios, se diseñan y priorizan los CP (se define que entorno se va a emular, como se van a llevar a cabo, por quien). Se define si se usa regresión o no.
  + 3. Ejecución: cuando se ejecutan los CP, se registran y compraran los resultados, generando un reporte de defectos. Se trata de automatizar estas ejecuciones. Incluye: creación de los datos necesarios para las pruebas, automatización, implementar y verificar el ambiente, ejecutar los casos de prueba, registrar los resultados y comparar los resultados reales u obtenidos con los esperados.
  + 4. Analisis de fallas (evaluación y reporte): cuando hay defectos vuelven a ejecución. Se hace un seguimiento de la corrección de los defectos encontrados hasta que se cierren todos los CP, es decir hasta que se solucionen. Se evalúan los criterios de aceptación y se reportan los resultados a los interesados. Se confecciona el informe de reportes
  + 5. Fin de las pruebas: se define cuando es momento de dejar de probar, dependiendo de cada empresa puede ser cuando no hay defectos bloqueantes, críticos, mayores y una mínima cantidad de defectos menores y cosméticos. Se confecciona el informe final.
* ¿CUÁNTAS LINEAS DE CÓDIGO NECESITO PARA COMENZAR CON EL TESTING? 0
* Artefactos de testing
  + Plan de prueba: es análogo al plan de proyecto que se elabora en el enfoque tradicional -no forma parte de este-. Para su confección no es necesario tener código, sino los R en cualquier formato que se encuentren. Contiene
    - Recursos destinados a las pruebas
    - Herramientas para utilizar
    - Decisión de regresión o no
    - Datos necesarios para las pruebas
  + Casos de prueba: es el artefacto mas importante del testing. Consiste en la descripción de una secuencia de pasos a seguir para lograr cierto resultado esperado. Para esto se deben definir las precondiciones o condiciones de inicio y los datos que serán utilizados en ese escenario.
    - Se confeccionan con el objetivo de encontrar la mayor cantidad de defectos con el minimo esfuerzo y tiempo
    - Mientras no cambien los R, se pueden utilizar las veces que sea necesario
    - La característica mas importante de los CP es que deben ser reproducibles. Si un defecto encontrado no se puede reproducir no es un defecto.
  + Reporte de defectos o incidentes: luego de la ejecución de los CP, se elabora un reporte informando cuales fueron los CP que pasaron, y cuales los que fallaron, indicando en este caso cual es el problema y como se reproduce para su corrección.
    - Este artefacto permite a los desarrolladores corregir defectos para enviarlos nuevamente a testing
  + Informe final: es el único artefacto que es negociable en las empresas. Contiene
    - Cantidad de ciclos ejecutados
    - Cantidad de errores por ciclo
    - Tipos de prueba utilizados
    - Métricas e información final del incremento del producto.
    - Tiene utilidades estadísticas.
* Testing en el ciclo de vida del SW: si desarrollamos SW en ciclo de vida en cascada (secuencial) las pruebas se hacen al final, ya que es cuando nos entregan el producto, pero si desarrollamos con ágil (ciclo iterativo e incremental) realizamos pruebas por iteración. El problema es que se pueden incluir errores por la integración de los incrementos. Objetivos de testing temprano:
  + Dar visibilidad al equipo del producto y de como se va a probar
  + Disminuir los costos de correcciones de defectos en etapas posteriores
* Modelo en V: se utiliza para saber cuando se esta en condiciones de realizar determinadas actividades en función de la etapa de desarrollo en la que se encuentre el SW.
  + Las pruebas se realizan en orden de granularidad inverso al de desarrollo. En desarrollo se comienza con componentes de granularidad grueso, en testing de granularidad fina.

Surgen dos conceptos importantes

* + Verificación: para ver si estamos construyendo el software correctamente, de la manera correcta. Libre de defectos, apuntando a la perfección de su funcionalidad
  + Validación: para ver si estamos construyendo el sistema correcto, es decir, que el sistema cumpla efectivamente con los R del cliente.
* Métodos de prueba: se usan porque el tiempo y el presupuesto es limitado. Hay que pasar por la mayor cantidad de funcionalidades con la menor cantidad de pruebas.
* Estrategias de prueba: buscan, con un presupuesto y tiempo limitado, abarcar en las pruebas la mayor cantidad de SW posible: Caja negra y caja blanca
* CAJA NEGRA: se llama así porque nosotros especificamos las entradas y los resultados esperados, que es el comportamiento del sistema frente a esas entradas. Lo que nosotros desconocemos es como se llega a obtener esa salida. Si se obtiene la salida esperada, entonces la prueba pasa. No se dispone de la estructura interna de la implementación. Se realiza un testing exhaustivo de cada entrada, probando cada posible entrada conducida como CP, no necesariamente las validas. Dos métodos:
  + Métodos basados en especificaciones: son los que utilizan la documentación de especificaciones realizadas del producto.
    - Partición de equivalencia: consiste en identificar las clases de equivalencia, donde cada una define un subconjunto de datos que producen un resultado equivalente
    - Analisis de los valores limites: variante del anterior. Se utilizan los valores limites o bordes de las clases de equivalencia para la definición de los CP
  + Métodos basados en la experiencia: le experiencia y los conocimientos del tester son fundamentales para determinar las entradas y analizar los resultados.
    - Adivinanza de defectos: con la experiencia y la intuición identifican pruebas que probablemente expongan defectos, una lista de defectos posibles o situaciones propensas a error y realizando pruebas a partir de esa lista
    - Testing exploratorio: el tester mientras va probando el SW, va conociendo el sistema y junto con su experiencia y creatividad, genera nuevas pruebas a ejecutar.
* CAJA BLANCA: se basa en el análisis de la estructura interna del SW, o de un componente de SW. Se puede garantizar el testing coverage. En estos disponemos del código, pseudocódigo o un diagrama de flujo. Permiten diseñar CP maximizando la cantidad de defectos con la menor cantidad de CP posibles. Tiene dos fallas:
  + El numero de caminos lógicos únicos puede ser sumamente grande, tomando mucho tiempo en probar todo.
  + Las pruebas de camino exhaustivas no aseguran que el programa sea el correcto para las especificaciones. No detecta caminos faltantes ni determina errores de datos sensibles.

Hay distintas coberturas con niveles diferentes -formas de recorrer los distintos caminos que provee el código para desarrollar una funcionalidad-

* + Cobertura de enunciados o caminos básicos
  + Cobertura de sentencia
  + Cobertura de decisión
  + Cobertura de condición
  + Cobertura de condición/decisión
  + Cobertura múltiple
* Tipos de prueba
  + Smoke Test: es una primera corrida de los test de sistema para asegurarnos de que el SW funciona y que no se provocará una falla catastrófica. Es decir, verifica que las funcionalidades básicas de una aplicación o sistema funcionen correctamente antes de pasar a pruebas mas exhaustivas.
  + Sanity Test: se realiza despues de una ronda de pruebas exhaustivas para verificar que se han corregido los errores o defectos identificados, y como dice su objetivo, para verificar que no se hayan introducido nuevos problemas al corregir los anteriores, y que las funcionalidades claves del SW aun siguen funcionando.
  + Testing Funcional: las pruebas se basan en funciones y características (descriptas en documentos o entendidas por los testers) y su interoperabilidad con sistemas específicos. Responden al qué hace el sistema. Controla que el SW se comporte de la manera en que esta especificado en la documentación, cumpliendo con las funcionalidades definidas. Se basa en:
    - Basado en requerimientos: cuando se prueban R se apunta a probar una funcionalidad sola, utilizando la documentación, o acuerdos que contienen las pruebas de usuario y los criterios de aceptación de las US.
    - Basado en procesos de negocio: cuando se prueba todo un proceso de negocio completo, como, por ejemplo, una venta donde se busque el artículo, se lo seleccione y se lo facture.
  + Testing No Funcional: se basan en como trabaja el sistema haciendo foco en los RNF, responden al cómo lo hace el sistema. Acá se ve reflejada la necesidad de emular, de que los ambientes de prueba sean lo mas parecidos posible al entorno de producción. Algunos tipos de prueba son
    - Performance: se ve el tiempo de respuesta y la concurrencia. Si o si la deben pasar
    - Stress: se somete al sistema a condiciones no usuales, más allá de las normales, se busca que falle. Se ve el tiempo que hay que esperar para probar nuevamente el sistema y la robustez (tiempo de recuperación)
    - Carga: no solo mira la performance, mira el comportamiento de los dispositivos de hardware, como procesadores, discos, etc. Y de las comunicaciones.
    - Mantenimiento: para verificar si el producto es capaz de evolucionar, se controla que haya documentación, manual de configuración, etc. Se observa la facilidad para corregir las fallas o defectos.
    - Portabilidad: se prueba en los distintos entornos acordados con el cliente
    - Usabilidad: que sea cómodo y eficaz para el usuario
    - Fiabilidad: probamos que podamos depender del sistema. Resultados que se obtienen, seguridad física del SW
    - De interfaz de usuario: suelen ser mas complejas las GUI’s que las interfaces de comando
    - De configuración
  + TDD – Test Driven Development: el Desarrollo conducido por pruebas es una técnica avanzada que consiste en escribir las pruebas antes de empezar con el código. Involucra dos practicas:
    - Test First Development: en esta técnica se escriben primero las pruebas -generalmente unitarias- y luego en base a eso se escribe o genera el código que pase con éxito esa prueba.
    - Refactoring: tiene que ver con mejorar aspectos no funcionales del sistema. Consiste en reestructurar código ya existente, sin cambiar el comportamiento externo. El objetivo es lograr código de mayor calidad y claridad, y reducir su complejidad par que sea mas facil de entender y por ende de mantener. Entonces consiste en que una vez hecho el Test First Development, se hace refactoring, osea se limpia el código y se lo reduce, y se escribe un nuevo test. Realizando esto de forma cíclica hasta acabar.

ASEGURAMIENTO DE CALIDAD DE PROCESO Y PRODUCTO

* ASEGURAMIENTO DE CALIDAD: se centra en la implementación de medidas preventivas para garantizar que un producto alcance la máxima calidad posible. Esto implica tomar medidas anticipadas y realizar actividades antes de que el producto este terminado. En lugar de detectar y corregir errores una vez que ya existen, se pone el foco en prevenirlos desde el principio
* CALIDAD: son todos los aspectos y características de un producto o servicio que permiten que este cumpla con todas las necesidades, tanto las que se expresan de manera clara y directa, como las que estan implícitas o no se mencionan explícitamente. Calidad implica la capacidad de un producto o servicio de satisfacer y superar las expectativas y R del cliente. La visión de calidad que tenemos en un momento de tiempo puede cambiar en otro, ya que esta es relativa a las personas.
* ¿Qué cosas ocurren frecuentemente en los proyectos de desarrollo de SW?
  + Trabajo fuera de hora
  + Costos excedidos
  + Falta de cumplimiento de los compromisos
  + Atrasos en las entregas
  + No estan claros los requerimientos
  + ¿Dónde esta este componente?
  + El SW no hace lo que tiene que hacer
  + Fenómeno 90-90.

Estas situaciones evidencian la falta de calidad o la no calidad. Esta no solo tiene que ver con el producto sino también con el proyecto, los procesos y las personas que trabajan en él.

El aseguramiento de calidad de PROCESOS implica establecer y mantener estándares y procedimientos claros para llevar a cabo todas las actividades de manera consistente y eficiente. Definir las mejores prácticas, documentar los procesos, capacitar al personal y establecer controles para monitorear y medir el desempeño

El aseguramiento de calidad de PRODUCTOS implica garantizar que el producto cumple con los estándares de calidad establecidos. Esto implica inspeccionar, hacer pruebas y evaluaciones durante el proceso de fabricación, para identificar y poder corregir defectos o errores antes de que el producto final sea entregado. También se pueden implementar sistemas de gestión de calidad como ISO 9001.

La CALIDAD del SW muchas veces se ve reducida por:

* + - Falta de cumplimiento de compromiso
    - Costos excesivos
    - Demoras en tiempos de entregas
    - El SW no hace lo que se pide

En contraposición, un SW de calidad ofrece:

* + - Cumplimiento de las expectativas del cliente
    - Cumplimiento de las expectativas del usuario
    - Cumplimiento de las necesidades de la gerencia
    - Cumplimiento de las necesidades del equipo de desarrollo y mantenimiento
* Principios de aseguramiento de calidad
  + La calidad no se inyecta al final: no puede ser comprada ni integrada al final de la construcción, debe ser incorporada a medida que se va creando el producto, desde el momento 0
  + La calidad conlleva un esfuerzo de todos: asegurar la calidad no es trabajo de una persona ni un departamento, todos tienen que estar comprometidos
  + Las personas son clave para lograr la calidad: la capacitación, conocimiento y experiencia de las personas es fundamental.
  + Es necesario un sponsor a nivel gerencial: se necesita de alguien que de respaldo y lidere a nivel gerencial, que considere el desarrollo de un producto de calidad como algo fundamental y guie el trabajo hacia allí, y lo respalde con recursos y apoyo.
  + Se debe liderar con el ejemplo: los lideres deben ser ejemplo de compromiso y calidad, demostrando practicas y comportamientos que fomenten la cultura del aseguramiento de calidad en toda la organización.
  + La calidad se mide con el testing: no se puede controlar lo que no se mide. Las pruebas son fundamentales para medir y controlar la calidad del SW.
  + Aumentar las pruebas no aumenta la calidad automáticamente: además de esto se deben implementar técnicas y enfoques adecuados para asegurar la calidad.
  + El aseguramiento de calidad debe planificarse: es necesario contar con un plan de aseguramiento que establezca los objetivos, actividades y recursos necesarios.
  + Simplicidad, empezar por lo básico: en lugar de complicar los procesos, se busca simplificar y comenzar por los fundamentos básicos del aseguramiento
  + Debe ser razonable para mi negocio: debe adaptarse a las necesidades y características de cada negocio, siendo realista con cada contexto y viable en términos de recursos y objetivos alcanzables.
* ¿Calidad para quién?
  + Calidad del proceso para construir el producto, calidad del producto en si misma, calidad que se espera que tenga. Todo es desde diferentes perspectivas. Tenemos que integrar todas estas expectativas o visiones, desde cuales son las del usuario (una buena interfaz gráfica) hasta las de los desarrolladores (buena cohesión y acoplamiento).
  + Visión del usuario: es la mas complicada. Cumplir las expectativas del usuario y conocerlas. Principio de comunicación constante es crucial.
  + Visión del producto: nivel de satisfacción de los R particulares de cada producto
  + Visión del proceso: si el proceso de desarrollo usado es el correcto para ese producto, si aporta valor al mismo y no produce desperdicios
  + Visión de valor: encontrar un equilibrio entre costo-beneficio. Para obtener el mayor valor para el cliente, pero generando ganancias.
  + Visión trascendental: responde a la pregunta de para que quiero construir este producto. Tiene que ver con querer lograr cosas más allá de lo que uno se imagina, como una utopía, objetivos difíciles de alcanzar, como 0 defectos, pero que al fin y al cabo son lo que nos impulsa.
* 3 calidades: se busca hallar un equilibrio entre las tres, todo lo que quede fuera de dicha coincidencia se considera desperdicio
  + Calidad programada: se refiere a las expectativas y estándares de calidad que se definieron y establecieron en la planificación y diseño. Es la ideal y la que se busca alcanzar.
  + Calidad necesaria: es el nivel minimo de calidad requerido para que el producto sea considerado correcto y pueda satisfaces las necesidades, expectativas y R del usuario.
  + Calidad lograda: es la calidad realmente alcanzada en el producto final, una vez completado el proceso de pruebas. Se busca obviamente que sea igual a la calidad programada. Es el resultado que se mide.

El proceso se materializa en proyectos, los cuales son la unidad de trabajo que da vida a dicho proceso. Al iniciar la ejecución de un proyecto, es esencial incorporar actividades de calidad, como revisiones técnicas y auditorias, para evaluar la aplicación de los modelos seleccionados.

El proyecto a su vez es el ámbito donde un trabaja para obtener el producto. Cuando hago actividades de aseguramiento de calidad de proceso o producto, se hacen en el contexto de un proyecto específico.

* Aseguramiento de Calidad de Proceso vs Aseguramiento de Calidad de Producto
  + Proceso: se enfoca en los procesos de desarrollo de SW y busca una mejoría en la forma de trabajo.
  + Producto: se enfoca en evaluar y garantizar la calidad del producto de SW resultante.
* Técnicas y herramientas de proyectos
  + Revisiones técnicas: se hacen entre pares, se somete a evaluación el producto. Se hacen sobre cualquier artefacto que queramos.
  + Auditorias: son realizadas por personas externas. Los procesos empíricos estan en contra porque justamente creen en equipos autoorganizados y capaces de reconocer y corregir errores por si mismos.
* Técnicas y herramientas de productos
  + Para aseguramiento de calidad
    - Revisiones técnicas: realizadas entre pares, con el foco en detectar defectos lo antes posible.
    - Auditorias (de configuración): valida una versión de la línea base
  + Para control de calidad
    - Testing: para controlar la calidad del producto cuando ya esta listo.

-los modelos de calidad dicen: hace tu proceso de la manera en que tenga calidad para vos, pero ese proceso debe ser compatible con lo que yo te digo (modelos de referencia)

-la calidad del proceso nunca es un fin en si mismo, pero es necesaria porque lo que realmente interesa es la calidad del producto final-

* Actividades de Administración de Calidad de SW
  + Aseguramiento de calidad: se establecen estándares y procedimientos de calidad. Se elige contra que estándares se van a efectuar las comparaciones (revisiones técnicas y auditorias)
  + Planificación de calidad: se seleccionan los procedimientos y estándares de calidad para un proyecto y se los modifica de ser necesario. Por ejemplo, en que momento se van a hacer las revisiones, auditorias, etc.
  + Control de calidad: se ejecuta control de calidad contra los estándares y procedimientos establecidos. Se hace Testing.
* Procesos basados en Calidad
  + El trabajo de las personas encargadas de la calidad en el desarrollo de SW es establecer y definir procesos que se van a seguir para conseguir la calidad de producto final. Estos procesos incluyen actividades de planificación, diseño, desarrollo o implementación, pruebas y entrega del SW. Una vex que se han establecido estos procesos, es importante evaluar constantemente su calidad y efectividad, esto se loegra a través de la evaluación del producto resultante. Se analiza lo entregado para determinar si cumple con la calidad, y satisfacen los R y criterios establecidos. En funcion de estos resultados, se toman decisiones sobre la mejora de los procesos, si se identifican deficiencias o defcetos en el producto, se realiza una revisión exhaustiva del proceso realizado para implementar mejoras donde sea necesario, y evitar que se repitan los errores. Por el contrario, si la evaluación resulta exitosa y se cumple con todo lo establecido, el proceso utilizado se estandariza. Esto implica documentar el paso a paso realizado, las actividades y las buenas practicas empleadas durante el desarrollo. Estos procesos estandarizados se distribuyen al resto de la organización para que puedan ser seguidos y aplicados de manera consistente en proyectos futuros
  + Diagrama

    Descripción generada automáticamente
* Modelos de Mejora de Procesos: la mejora continua de procesos significa comprender los procesos existentes y modificarlos para incrementar la calidad del producto o reducir los costos y el tiempo de desarrollo. Los procesos definidos estan de acuerdo con esta definición, ya que consideran que la calidad del producto depende de la calidad del proceso. Los modelos NO DICEN COMO hacer las cosas, son modelos descriptivos. El propósito de un modelo de mejora es analizar un proceso que se lleva a cabo, y armar un proyecto cuyo resultado, en lugar de ser un producto, es un proceso mejorado que se vuelca de nuevo a la organización con la idea de que se produzca un producto mejor.

Los modelos de mejora son recomendaciones o estándares para encarar un proyecto de mejora de proceso. Son 2: IDEAL, SPICE

* IDEAL – Initiating, Diagnosis, Establishing, Acting/Ejecución, Learning: modelo de mejora que nos sirve para definir en una organización, un proyecto que ayuda a mejorar un proceso que esa empresa ya tiene. Nos da el contexto para crear un proyecto cuyo resultado va a ser un proceso definido. Es un modelo cíclico. Debemos hacer un analisis de brecha, es decir, ver a donde estamos parados y donde queremos ir. Comienza buscando un sponsor, que es un apoyo en la organización en términos económicos. Una mejora de proceso nunca es crítica, por eso es importante tener a alguien que avale este tipo de proyectos.
  + 1. Initiating: se comienza por reconocer las necesidades de cambio en la organización, razones para iniciar, determinar metas y objetivos buscados al proponer un cambio en el proceso. Se busca sponsor
  + 2. Diagnosis: se analiza la madurez actual de la organización y los riesgos asociados al proceso de mejora. Análisis de brecha (modelos de calidad)
  + 3. Establishing: se elabora un plan detallado de como se va a llevar a cabo el proceso de mejora, acciones específicas, entregables y responsabilidades basadas en el resultado del diagnóstico.
  + 4. Acting/Ejecución: se efectúan los cambios y se reúne información para aprender de la mejora. Se implementan las acciones planeadas en un proyecto piloto (no en todos los procesos de la organización, es una prueba). Si la solución es satisfactoria se implanta en la empresa.
  + 5. Learning: busca garantizar que el próximo ciclo sea más efectivo aún. Durante esta fase se revisa toda la información recaudada en los pasos anteriores, se evalúan logros y objetivos alcanzados. Extrapola la mejora al resto de procesos en caso de que sea exitosa la ejecución.
* SPICE: es un modelo creado para la mejora de procesos de SW (ISO). Es una adaptación para la evaluación de procesos por niveles de madurez. Es un modelo dual, teórico. Tiene dos partes
  + Modelo de calidad
  + IDEAL + SPICE

Este modelo establece conjuntos predefinidos de procesos con el objetivo de definir un camino de mejora. Los niveles son

* + Nivel 0 – organización inmadura = la organización no tiene una implementación efectiva de los procesos
  + Nivel 1 – organización básica = la organización implementa y alcanza los objetivos de los procesos
  + Nivel 2 – organización gestionada = la organización gestiona los procesos y los productos de trabajo se establecen, controlan y mantienen
  + Nivel 3 – organización establecida = la organización utiliza procesos adaptados basados en estándares
  + Nivel 4 – organización predecible = la organización gestiona cuantitativamente los procesos
  + Nivel 5 – organización optimizando = la organización mejora continuamente los procesos.
* Modelos de Calidad
  + CMMI – Capability Maturity Model Integration: surge en el ambiente del SW y luego se abre a otras áreas, quedando todo bajo un mismo modelo (CMMI)

Nos dice el qué, no el cómo. Es un modelo descriptivo, no prescriptivo.

El objetivo principal es brindar una guía para mejorar la capacidad y madurez de los procesos organizacionales.

El modelo se basa en buenas practicas recopiladas de la industria y esta diseñado para ayudar a las industrias a alcanzar niveles superiores de calidad, eficiencia y satisfacción del cliente.

CMMI se estructura en niveles de madurez y áreas de proceso. Los niveles de madurez representan etapas de mejoras evolutivas en la organización. Las áreas de proceso se enfocan es aspectos específicos del desarrollo y gestión del SW.

Las organizaciones pueden obtener beneficios como la reducción de defectos, la mejora de la productividad, la gestión de riesgos mas efectiva, y la mejora en la entrega de productos y servicios.

Es un modelo flexible y adaptable a las necesidades de cada organización.

Tiene 3 dominios o ámbitos de mejora

* + - DEV: provee la guía para medir, monitorizar y administrar los procesos de desarrollo
    - SVC: provee la guía para entregar servicios internos o externos
    - ACQ: provee la guía para administrar, seleccionar y adquirir productos o servicios
  + Representaciones de CMMI
    - Por etapas: se organizan los niveles de madurez en etapas predefinidas. Donde para pasar al siguiente nivel se necesita cumplir con cierto conjunto de prácticas y capacidades.
      * Nivel 1 Inicial: no hay visibilidad sobre los procesos. Hay mucha incertidumbre sobre costos y tiempos, y, por ende, mucho riesgo. Tiene procesos no estandarizados
      * Nivel 2 Gestionado: resuelve el problema del nivel 1. La org. establece prácticas de gestión básicas y utiliza enfoques mas disciplinados.
      * Nivel 3 Definido: la org. establece y documenta procesos estandarizados y repetibles
      * Nivel 4 Cuantitativamente Gestionado: la org. establece mediciones cuantitativas para el control y la evolución y mejora de los procesos.
      * Nivel 5 Optimizado: la org. se enfoca en la mejora continua y la optimización de los procesos.
      * Mientras mas madura la org. (nivel 5) mas capacidad tiene para cumplir con los objetivos
    - Continua: esta representación lo que hace es elegir áreas de proceso dentro de las que ofrece el modelo (22) y yo elijo cuales son los que quiero mejorar por separado. Entonces en lugar de medir la madurez de la empresa mido la capacidad de un proceso en particular.

Mide la capacidad de las distintas áreas de poder cumplir con los objetivos.

La medición de la capacidad se realiza en una escala del 0 al 5, donde cada nivel indica mayor capacidad de una implementación y desempeño exitoso. 0 es el nivel que no se ejecuta.

* SCAMPI: es el método formal para evaluar y que una empresa sepa si adquirió un determinado nivel o capacidad de CMMI.
* CMMI y Agil: lo que quiere decir básicamente es que tienen que ceder o encontrar un punto medio porque CMMI tenemos un proceso definido x cumplir, y se evalúa si se cumplió o no. En agil no se evalúa

AUDITORÍAS DE SOFTWARE

* Son evaluaciones independientes de los productos y procesos, para el aseguramiento de calidad del SW, es decir asegurar que se cumplan estándares, lineamientos, especificaciones y procedimientos, basada en un criterio objetivo y documentación que especifique
  + La forma o contenido de los productos a ser desarrollados
  + El proceso por el cual los productos van a ser desarrollados
  + Como debería medirse este cumplimiento de estándares.

Es una actividad incluida dentro de las disciplinas de soporte, y son realizadas por un equipo externo al proyecto, que no tienen nada que ver con el equipo de desarrollo.

Implican cierto costo, pero los beneficios son superiores.

* Beneficios:
  + Se obtienen opiniones desde otra perspectiva, objetivas e independientes
  + Permiten identificar áreas de insatisfacción potenciales del cliente
  + Permite asegurar que se esta cumpliendo con las expectativas
  + Permite visibilidad sobre los procesos de desarrollo: identificar puntos fuertes y puntos que son oportunidades de mejora.
  + Da visibilidad a la gerencia sobre los procesos de trabajo
  + Resultado: mejores productos conllevan a clientes satisfechos y crecimiento del negocio.

Tipos de Auditorias

* Auditoria de proyecto: tiene que ver con validar el cumplimiento del proceso de desarrollo. Es responsable de evaluar si el proyecto se ejecuto en base al proceso que se dijo que se iba a utilizar. Apunta a ver el cumplimiento del proceso que como equipo nos comprometimos a utilizar.
  + Las auditorias de proyecto se deberían realizar de acuerdo con lo establecido en el PACS (plan de aseguramiento de calidad de software)
  + El PACS debería indicar quien es el encargado de llevar a cabo las auditorias.
  + Las inspecciones de SW y las revisiones de la documentación de diseño y prueba deberían incluirse en la auditoria.
  + El objetivo de esta auditoria es verificar objetivamente la consistencia del producto a media que este va evolucionando a lo largo del desarrollo.
* Auditoria de configuración funcional: tiene que ver con validar si el producto cumple con los requerimientos. Compara el software que se construyo con lo especificado en la ERS. El propósito es asegurarse de que el producto cumple solamente con los requerimientos y capacidades descriptas.
* Auditoria de configuración física: verifica si los ítems de configuración cumplen con la documentación que los describen. Esto permite la trazabilidad y la satisfacción de los R. La auditoría física compara el código con la documentación de soporte.

Su propósito es asegurar que el producto que se entrega es consistente y coincide con la documentación también entregada

El PACS debería definir quien realiza esta auditoría.

Roles en una Auditoría

* 1. Auditado: generalmente es el Líder de Proyecto, pero puede ser otro.
  + Participa de la auditoria
  + Propone una fecha
  + Entrega evidencia
  + Contesta las dudas del auditor
  + Propone cursos de acción para las desviaciones encontradas
  + Responde el reporte de auditoría
  + Propone el plan de acción para las deficiencias citadas en el reporte.
* 2. Auditor: es una o dos personas externas al proyecto, pueden ser personas del grupo de aseguramiento de calidad.
  + Acuerda la fecha de auditoria
  + Recolecta y analiza la evidencia objetiva que es relevante y suficiente para tomar conclusiones acerca del proyecto auditado
  + Realiza la auditoria
  + Realiza un reporte
  + Realiza el seguimiento de los planes de acción acordados con el auditado.
* 3. Gerente de SQA (Software Quality Assurance): es quien prepara el plan de auditoría, calcula su costo, asigna recursos, resuelve las no-conformidades entre el auditor y el auditado.

Proceso de Auditoría

* Preparación y planificación: se planifica y prepara la auditoria en forma conjunta entre el auditado y el auditor. Generalmente el líder de proyecto la convoca
* Ejecución: el auditor realiza la auditoria, hace preguntas al auditado, pide documentación, busca evidencia objetiva (lo que está en la documentación) y subjetiva (lo que el equipo expresa que hace)
* Analisis y reporte de resultado: se analiza la documentación que contiene los resultados obtenidos, se genera un reporte y se lo entrega al auditado.
* Seguimiento: según lo que acordaron el auditado y el auditor, el auditor puede realizar un seguimiento de los planes de acción tomados para corregir las desviaciones encontradas, hasta que se consideren que han sido resueltas.

Herramientas y técnicas utilizadas en auditorías

* Checklists: contiene una lista de preguntas tipo para garantizar que independientemente de quien realice la auditoria, el foco de esta de quienes la realicen sea el mismo. Son de mínima, es decir que se debe garantizar mínimamente contestar esas preguntas, pero el auditor puede solicitar más, osea hacer más preguntas.
* Muestreo: se toma una muestra representativa de los productos y/procesos para realizar la auditoria.
* Revisión de registros
* Herramientas automatizadas

Resultados de Auditoría – Tipos

* Buenas prácticas: practica, procedimiento o instrucción que se ha desarrollado mejor de lo que se esperaba. Supera lo esperado. Cuando se hace el reporte de auditoria se comienza comentando las buenas prácticas.
* Desviaciones: requieren un plan de acción por parte del auditado. Cualquier cosa que no se hizo como el proceso indicaba que debería hacerse.
* Observaciones: condiciones que podrían mejorarse pero que no llegan a ser una desviación. No requieren un plan de acción. Pueden llegar a traer problemas, por eso se destacan, para que el equipo considere mejorarlas.

Métricas de auditoria

* 1. Esfuerzo de auditoria: mide la cantidad de tiempo y recursos que fueron dedicados a la realización de la auditoria. Expresado en horas o días de trabajo
* 2. Cantidad de desviaciones: mide la cantidad de desviaciones identificadas durante la auditoria. Es decir, cantidad de incumplimientos de estándares, normas o requisitos.
* 3. Duración de auditoria: mide la cantidad de tiempo que llevo, desde su planificación hasta la entrega de resultados. Puede ser útil para evaluar la eficiencia y la capacidad de gestión de proceso de auditoría
* 4. Cantidad de desviaciones clasificadas por área de proceso de CMMI: desglosa las desviaciones encontradas según las áreas de procesos definidas por el modelo CMMI

REVISIONES TÉCNICAS

* Es una actividad realizada por un colega, cuyo propósito es mejorar la calidad del SW, mediante la detección temprana de errores en cualquier artefacto que se genere. Es una técnica de aseguramiento de calidad. Es un proceso de validación y verificación, pero NO corrige errores.
  + Validación: estoy haciendo el producto correcto
  + Verificación: estoy haciendo el producto de la manera correcta

Busca evitar el retrabajo, introduciendo la cultura de validación y verificación. Motiva a realizar un mejor trabajo, sin poner nunca en juicio al autor del artefacto, ni buscando culpables cuando se encuentre un error.

Tipos de revisiones

* Formales: tienen procesos definidos con roles
  + Inspecciones: objetivos son detectar y remover todos los defectos encontrados eficiente y efectivamente. Es una actividad de garantía de calidad de software. Es una actividad formal donde se usa un checklist que funciona como ayuda memoria para saber que cosas controlar. Se toman métricas y se realiza un informe de la revisión final.
* Informales: cuando no existe un proceso que diga cómo realizarlo
  + Walkthrough o recorrido: objetivos son mínima sobrecarga, rápido entorno, capacitación de desarrolladores. Es un analisis estático donde un diseñador o programador dirige a miembros del equipo a través de un producto de SW, y estos pueden hacer preguntas, comentarios acerca de posibles errores o violaciones de estándares y otros problemas. Es una reunión informal a modo de debate entre colegas. No obtiene métricas, pero es lo mas elegido dentro de enfoques agiles.