**Resumen segundo parcial ISW**



SCRUM: (leer la guía de scrum)

* Equipo
* Ceremonias
* Artefactos
* Nexus

Es un framework para gestión que no apunta a ninguna actividad de la ingeniería.

Trabaja con proyectos en iteraciones finitas llamadas sprints.

Equipo: SCRUM se basa en la idea de equipos autoorganizados y multifuncionales. El equipo SCRUM consta de un Scrum Master, un Product Owner y el Equipo de Desarrollo. El Scrum Master actúa como facilitador y líder del proceso, el Product Owner es responsable de definir y priorizar el trabajo, y el Equipo de Desarrollo es el grupo encargado de realizar el trabajo real.

Ceremonias: SCRUM incorpora varias ceremonias para facilitar la comunicación y el seguimiento del progreso. Todas las ceremonias son timebox no solo el sprint. Las principales ceremonias son:

* Sprint: un período de tiempo (generalmente de 2 a 4 semanas) durante el cual se lleva a cabo el trabajo.
* Reunión de Planificación del Sprint: al comienzo del Sprint, el equipo selecciona las tareas que se abordarán durante ese período.
* Reunión Diaria de SCRUM: una breve reunión diaria donde el equipo comparte actualizaciones sobre su trabajo y discute los obstáculos.
* Revisión del Sprint: al final del Sprint, el equipo muestra el trabajo completado al Product Owner y otras partes interesadas. Nexus reemplaza a las sprint reviews individuales de cada equipo.
* Retrospectiva del Sprint: una reunión después de la revisión del Sprint en la que el equipo analiza lo que funcionó bien y qué se puede mejorar.

Artefactos: SCRUM utiliza varios artefactos para mantener un registro del trabajo y el progreso:

* Product Backlog: una lista priorizada de todas las características, mejoras y tareas pendientes del proyecto.
* Sprint Backlog: una lista de elementos seleccionados del Product Backlog para ser abordados durante el Sprint.
* Incremento: el trabajo completado y potencialmente entregable al final de cada Sprint.

Testing de Software:

* Concepto
* Objetivo
* Niveles de prueba
* Ambientes
* Casos de prueba
* Tiempos
* Principios de Testing
* Tipos de prueba

**TESTING**

Disciplinas de soporte trabajan para que el producto final tenga la mayor calidad.

Proceso destructivo que trata de encontrar defectos en el software (cuya presencia se asume), su objetivo no es asegurar que el software funcione sino encontrar defectos. Identifica defectos, si no encontraste no significa que no hay, sino que no buscaste bien. Decimos entonces, que la prueba va a ser exitosa cuando encontremos defectos.

**Diferencia entre error y defecto**, el error se descubre a partir de técnicas específicas que me permiten encontrar los mismos dentro de la misma etapa en la que estoy trabajando. Mientras que los defectos nos demuestran un error no detectado que se trasladó a una etapa siguiente. En el testing encontramos defectos ya que encuentran cosas incorrectas que se realizaron en la etapa de implementación que es la etapa anterior.

* No asegura la calidad.
* Ninguna actividad de control está bien que lo haga alguno que hizo el código.
* El testing es la actividad más cara, 30 a 50% del costo de un software confiable.
* El testing unitario si lo hace el programador porque es parte de su trabajo.
* El testing no certifica nada, solo identifica defectos.

**Severidad:** cuán grave es el defecto y me ayuda a determinar el impacto. 1 – Bloqueante (no permite seguir con el caso de prueba) 2 – Crítico (compromete la ejecución del caso de prueba) 3 – Mayor 4 – Menor 5 - Cosmético

**Prioridad:** urgencia que tenemos para resolver este defecto, lo decide el product owner. 1 – Urgencia 2 – Alta 3 – Media 4 – Baja. Cuánto impacto técnico me va a llevar corregirlo.

**Niveles de prueba o testing** (existen tanto en ágil como en tradicional)

**Pruebas unitarias:** son aquellas en donde pruebo un componente individual, algo acotado que tiene que ver con el desarrollo que estoy realizando. Normalmente se ejecutan por el mismo desarrollador, ya que probamos componentes individuales de forma independiente. Encontramos errores.

**Pruebas de sistema:** testing de versión, porque probamos una versión del producto que no está terminada (en un ciclo de vida iterativo incremental). Busca asegurarse que el sistema en su totalidad funcione de manera satisfactoria. Se tienen en cuenta también requerimientos no funcionales. El ambiente de prueba tiene que ser lo más parecido al ambiente de producción.

**Pruebas de integración**: hace foco en integrar los componentes que ya fueron aceptados en sus pruebas unitarias, se integran para ver su funcionamiento conjunto. Se hace una integración de manera incremental para lograr una identificación más correcta de los errores. Continuous integration es la automatización de las pruebas de integración. Aparece como una actividad del workflow de testing.

**Pruebas de aceptación de usuario:** lo hace al usuario a este testing, está en el workflow de despliegue. El foco no es encontrar defectos, sino que es establecer confianza en el sistema. En la review de scrum.

**Ambientes: (completar esta mierda)**

Los ambientes son los lugares en donde se trabaja para el desarrollo de software. Los ambientes para la construcción del software se refieren a los diferentes entornos utilizados en el ciclo de vida del desarrollo de software. Cada ambiente cumple un propósito específico y se utiliza en diferentes etapas del proceso de desarrollo y despliegue.

**Casos de prueba**: tiene que ver con un conjunto de condiciones o variables que nos van a permitir determinar si el software está funcionando correctamente o no. La buena definición de casos de prueba nos ayuda a reproducir defectos. **Consiste en ejecutar una serie de pasos o acciones en una determinada funcionalidad determinando completamente con qué valores voy a hacer la ejecución para ver si el software me va a dar los resultados esperados o no.** Ayuda a identificar defectos y controlarlos, solucionarlos. Debo lograr definir la menor cantidad de casos de prueba pero que estos pueda cubrir el testing completamente, pero no podemos identificar de manera infinita. Tratamos de apuntar a los lugares que son más propensos a tener errores o defectos.

Hay que usar estrategias de prueba se dividen en dos: Caja blanca y Caja negra.

Caja Blanca: es una técnica de testing que examina la estructura interna de un programa y deriva los datos de testeo desde la lógica del programa, evaluando las sentencias y condiciones.

Lean y Kanban:

* Lean que es
* Principios Lean
* Kanban que es
* Prácticas generales Kanban
* Cómo aplicar Kanban

Desperdicios evitables e inevitables, por ejemplo las meetings son evitables.

**Filosofía Lean** es una metodología de gestión que se originó en el sistema de producción de Toyota en Japón en la década de 1950. Su principal objetivo es maximizar el valor para el cliente mientras minimiza el desperdicio y los recursos mal utilizados. Busca también desarrollar las capacidades relacionadas con la resolución de problemas.

Tiene un lema muy importante que dice debemos “tomar lo que tengas, tomar el proceso que tengas y a partir de ahí trata de aplicar estos principios y de mejorar lo que hoy ya tenes”

**Principios Lean:**

* **Eliminar desperdicios** (principio más importante), se mide en términos de todo lo que no genera valor es desperdicio. Buscar en el proceso cosas que no generan valor al negocio, implica trabajo innecesario o re trabajo. Surge del concepto de Just In Time, y está relacionado con los principios ágiles de Software funcionando y Simplicidad.
* **Amplificar el aprendizaje:** transparencia, busca transformar el conocimiento en prácticas compartidas. O sea que el conocimiento es del equipo de todos. Nosotros podemos aprender a partir del trabajo que estamos haciendo y que ese conocimiento sea accesible para todos, y así poder lograr la mejora continua. Relacionado con el principio de Equipos autoorganizados ya que se puede lograr un espiral de conocimiento accesible a todos los miembros del equipo, el hecho de que no haya un jefe hace que la información no se quede en ese líder sino que se comparte. No perdemos tiempo en definir un diseño por completo previamente ya que esto no prevé que el diseño va evolucionar en el tiempo.
* **Embeber la integridad conceptual:** contemplar todas las partes del producto/servicio haciendo foco en lo que tiene que ver con los requerimientos funcionales. Incluir todas las partes desde el principio y no olvidarse de alguna. Atención continua de excelencia técnica, velar por construir con calidad un producto desde el principio, prevenir defectos. *“Encastrar todas las partes del producto o servicio, que tenga coherencia y consistencia (tiene que ver con los requerimientos no funcionales). La integración entre las personas hace el producto más íntegro”.* INTEGRIDAD PERCIBIDA: el producto total tiene un balance entre función, uso, confiabilidad y economía que le gusta a la gente. INTEGRIDAD CONCEPTUAL: todos los componentes del sistema trabajan en forma coherente en conjunto.
* **Diferir compromisos hasta el último momento responsable:** tiene que ver con la toma de decisiones, que una buena toma de decisiones está basada fuertemente en información. Postergar el asumir un compromiso hasta que tengas más información para poder asumir el compromiso, no antes debido a que puede haber incertidumbre. Tiene que ver con el concepto de Just In Time. Tiene que ver con el Product Backlog, en donde podemos arrancar con las user stories e ir avanzando sin tenerlo todo definido. Asignación de trabajo que es dinámica.
* **Dar poder al equipo**, equipos autoorganizados, el equipo debe tener capacidad de decisión. Darle la oportunidad al equipo de empoderarse. Se busca que las personas que son las responsables de hacer las cosas tengan la suficiente delegación de decisiones y de responsabilidades para que ellos se puedan hacer cargo.
* **Ver el todo**, debemos tener una visión completa no solo del proceso, sino que del producto que tenemos que crear. Kanban tiene prácticas que nos permiten a nosotros saber donde estamos parados. *“Tener una visión holística, de conjunto (el producto, el valor agregado que hay detrás, el servicio que tiene todos los productos como complemento)”*
* **Entregar lo antes posible**,acortar el ciclo de desarrollo y los tiempos y entregar cuanto antes al cliente. Retroalimentación, antes entregar al cliente antes puedo obtener retroalimentación. La entrega continua es una forma de entrega lo antes posible.

Gastos en producción en lean: 7+1 (hay más pero no se que tan importantes son)

* El +1 es el talento no utilizado y es el más complejo, genera mucho desperdicio.
* El movimiento es de personas y de materia prima.
* Sobre procesamiento: hacer de mas al pedo (relacionado con el principio yagni)

**Kanban** es un **framework o método** que trata de incorporar **mejoras evolutivas** a nuestro proceso (foco en productos intangibles). Plantea una mejora gradual. Arranca con el concepto de Just In Time (trabajar a partir de la demanda del cliente). Es un proceso para gestionar el cambio en procesos de desarrollo de software en proyectos, pero es un ENFOQUE para la gestión de cambios. No es para gestión de proyectos ni tampoco modela un proceso de desarrollo de software sino que sirve para introducir cambios en un proceso de desarrollo ya existente.

El flujo de trabajo tiene que fluir y evitar el cuello de botella. Relacionado con el WIP, limitar el trabajo en progreso. Equilibrio cuánto vamos a hacer de cada cosa y en qué momento para lograr el flujo y que no haya atascamiento.

Principios de Kanban: organizados en dos grandes grupos (no se que onda esto lo dijo la meles)

- Gestión de cambios: Comenzar por lo que tienes ahora, trae tu proceso y con eso arrancamos, le saca las excusas a la gente. Facilitador de la mejora de procesos. Fomentar actos de liderazgo: empoderar el equipo, no jefe. Acordar la búsqueda de la mejora a través del cambio evolutivo.

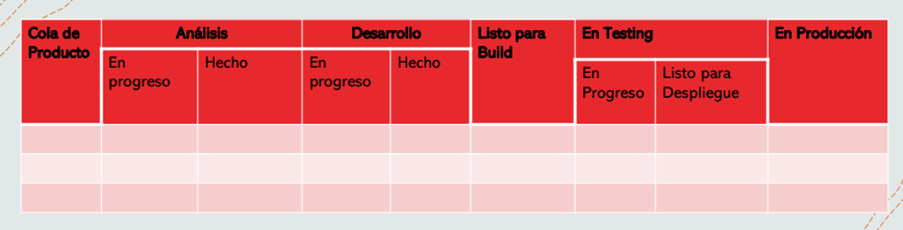
- Entrega de servicios: Comprender y enfocarse en cumplir las necesidades y expectativas del cliente: tiene que ver con entregarle al cliente software funcionando. Gestionar el trabajo. Revisar periódicamente la red de servicios y sus políticas para mejorar los resultados entregados

**Prácticas** generales de Kanban: método menos prescriptivo que Scrum. Las prácticas nos dicen el qué y no el cómo así que decidimos qué mierda queremos hacerlo.

1. **Visualizar el trabajo:** tiene que ver con esta necesidad de tener que ver todo el proceso y tener toda la información para hacer el trabajo accesible (transparencia). Utiliza como herramienta fundamental el tablero visual para representar el flujo de trabajo, igual que en Scrum, pero son diferentes. Nuestro proceso se tiene que mapear en el tablero, se muestran las tareas en progreso, pendientes y terminadas. Los kanban con minúscula son los post its que nos dicen qué hacer.

2. **Limitar el trabajo en progreso:** WIP (work in progress), límites la cantidad de trabajo en curso. Establecer límites es decisión nuestra, debería haber en todas las columnas del tablero. Esto evita la sobrecarga del equipo y ayuda a mantener un flujo de trabajo equilibrado y constante.

3. **Gestionar el flujo de trabajo:** lograr que el trabajo fluya de manera continua y eficiente, las colas acumulan el trabajo y se controla el control de flujo con los límites. Las colas de trabajo son las colas de progreso y las otras son de acumulación. Debemos identificar y resolver los cuellos de botella y los problemas que afectan el flujo de trabajo, con el objetivo de optimizar y garantizar una entrega rápida y constante.



4. **Hacer explícitas las políticas:** pocas políticas, pero explícitas en el tablero para todos los miembros del equipo. Definition of done y definition of ready son políticas que tienen que estar explícitas en el tablero. Todas las políticas deben ser acordadas entre todos los involucrados. Las políticas sirven si cumplen con las siguientes características: pocas, sencillas, bien definidas, visibles, aplicables en todo momento, fácilmente modificables por los que prestan el servicio (importante para la adaptación al cambio).

5. **Implementar ciclos de retroalimentación o feedback:**

6. **Mejorar de manera colaborativa:** como hacemos para que nuestro sistema funcione cada vez mejor. Kanban fomenta la mejora continua a través de la colaboración y el aprendizaje. Los equipos deben buscar constantemente formas de mejorar el proceso y adaptarlo a medida que surgen nuevas ideas y desafíos.

**Cómo aplicar Kanban:** primero analizo donde estoy parado ahora, luego acuerdo los límites de trabajo en progreso y en cada etapa definir la cantidad de tareas máxima (tarjetas) que puedo tener en progreso (limitar el WIP).

1. **Dividir el trabajo en piezas:** podemos utilizar las User stories. Luego de dividir el trabajo podemos categorizar las tarjetas que representan las piezas de trabajo, como por ejemplo, asignarles colores a los distintos tipos de trabajo.
2. **Visualizar el flujo de trabajo:** utilizamos nombres en las columnas del tablero para ilustrar dónde está cada ítem en el flujo de trabajo, cada una de las piezas definidas en el punto anterior fluye de izquierda a derecha por las columnas. Importante en Kanban es que son los recursos los que se autoasignan a las piezas de trabajo y no que alguien le asigne trabajo a un recurso (los recursos son las personas del equipo).
3. **Limitar el WIP:** asignar límites explícitos de cuántos ítems puede haber en progreso en cada etapa del flujo de trabajo. Se deben establecer límites realista basados en la capacidad del equipo y las necesidades del proyecto.

Métricas: (estudiar ejemplos)

* Métricas enfoque tradicional
* Métricas enfoque Ágil
* Metricas de Lean - Kanban

Una **métrica** es un número que se obtiene de un proceso de medición, para tener visibilidad de una realidad. La medida o la presencia o el grado de valor sobre un atributo que yo quiero medir. determinación cuantitativa.

En el **enfoque tradicional**: muchas no tienen un propósito conductivo al objetivo del desarrollo de software. basados en procesos definidos.

El dominio de las métricas de software se divide en: métricas de proceso, de proyecto y de producto (no se incluye a la gente porque a la gente no se mide).

Las **métricas de proceso** son públicas, son de la organización, consolidando las métricas del proyecto, denotan un comportamiento organizacional, no hace referencia a ningún producto, proyecto ni persona, se usan para mejorar el proceso. Más eficientes en la remoción de los defectos cuando los detectamos antes y no el cliente. Las voy sacando de cada proyecto y sacó promedio y luego publico.

* Tasa de defectos encontrados en pruebas internas: Esta métrica mide la eficiencia de su equipo en la detección temprana de problemas antes de que el software llegue al cliente. Cuanto más temprano se detecten los defectos, más eficiente será el proceso.
* Tiempo promedio para completar una tarea en el proceso de desarrollo: Mide la eficiencia en términos de cuánto tiempo se tarda en completar tareas individuales en el proceso de desarrollo. La reducción del tiempo promedio puede indicar una mayor eficiencia.
* Tasa de retrabajo: Esta métrica cuantifica cuánto trabajo adicional se debe realizar para corregir errores o defectos una vez que se han detectado. Una tasa de retrabajo más baja indica una mayor eficiencia en la producción.

Las **métricas de proyecto** son visibles para los que tienen que opinar sobre el proyecto pero no se publican a nivel organizacional por eso son privadas, dentro del equipo si deben poder acceder al valor de las métricas para cumplir con la transparencia.

* Horas trabajadas en el proyecto: Puede ayudar a evaluar el uso del tiempo y recursos en el proyecto y comparar el progreso con las estimaciones iniciales.
* Costo real versus presupuesto: Mide si el proyecto se mantiene dentro del presupuesto original o si está sobrepasando los costos previstos.
* Calidad de la comunicación del equipo: Puede incluir la evaluación de la efectividad de las reuniones, la claridad de la comunicación, y la satisfacción general del equipo.

Las **métricas de producto** mide el software, cantidad de líneas de código (no sirve), cantidad de requerimientos, calidad, necesidades cumplidads del cliente. Tenemos que elegir algo alrededor de la calidad por ejemplo porque a esta no la podemos medir. por ejemplo la cantidad de requerimientos cumplidos, suelen ser cosas indirectas. tenemos la problemática de no poder medir ciertas cosas. Cantidad casos de uso por complejidad, cantidad de defectos, tienen que ver directamente con el software. contar algo que tenga que ver con el producto. El líder de proyecto define que se va a medir, cuánto se va a medir, definir de donde sacar los datos, que cálculos vas a hacer, para que todos hagan los mismo cálculos o de la misma manera.

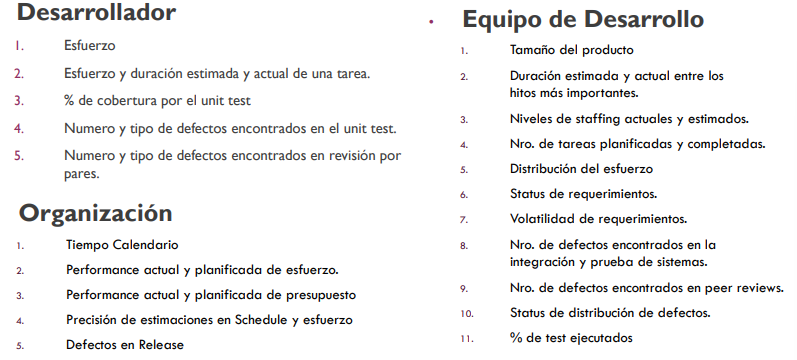
* Número de defectos por línea de código: Mide la calidad del software en términos de errores o problemas encontrados en el código. Menos defectos por línea de código indican una mejor calidad.
* Porcentaje de requerimientos cumplidos: Evalúa en qué medida el software cumple con los requerimientos del cliente. Un alto porcentaje indica un cumplimiento satisfactorio de las necesidades del cliente.
* Complejidad ciclomática del código: Mide la complejidad del código, lo que puede indicar cuán fácil es de mantener y cuántos posibles puntos de fallo puede tener.
* Tasa de defectos encontrados por el cliente: Esta métrica mide la calidad percibida del producto por parte del cliente y se basa en los problemas que el cliente encuentra después de la entrega del producto.
* Cantidad de casos de uso por complejidad: Evalúa la complejidad de los casos de uso en el software y puede proporcionar información sobre la facilidad de uso y la calidad del producto.

No son perfectas pero siempre hay que tenerlas. ¿se puede agregar métricas? si tienes los datos y tienes la necesidad se puede agregar. Todo se resume en la cantidad de horas que va a llevar hacerlo porque el esfuerzo de la gente es lo más costoso.

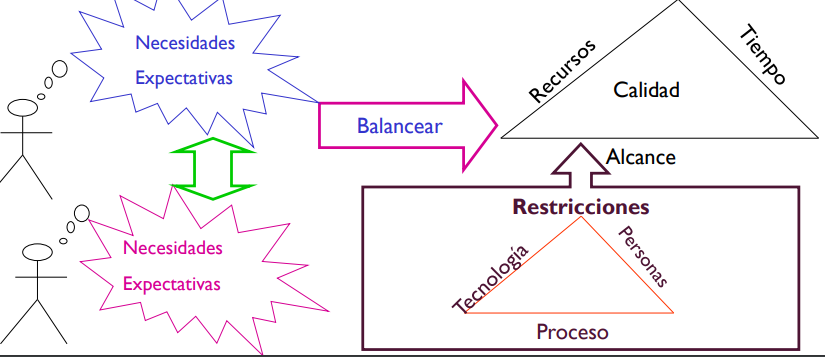
Métricas básicas en el enfoque tradicional: tamaño en el producto (que tenga que ver con los requerimientos), esfuerzo (hora personas lineales), tiempo calendario, defectos (por severidad o densidad de defectos o sea cuantos defectos tenes por bloque de código).

**Tomas la métrica a nivel de proyecto**, el ámbito de captura de un proyecto. Y para el proceso entrego las métricas de cada proyecto y alguien de la organización las consolida.

**Ejemplos de métricas:**



Las métricas tienen un costo, la precisión es cara. Mantengamos las cosas lo más simple posible. debemos satisfacer necesidades y expectativas principalmente de los clientes, pero estas son diferentes entonces debemos buscar un equilibrio por eso debemos armar las métricas que vamos a utilizar para cumplir con el equilibrio. tiene que estar equilibrado porque sino lo primero que se sacrifica es la calidad y eso no esta bueno. defectos materialización más básica y concreta de la calidad.



Las **métricas en ambientes ágiles** filosofía muy simplista de las métricas basado en el principio de que lo más importante es satisfacer al cliente con software funcionando en entregas tempranas y continuas.

La **velocidad** que mide la cantidad de story points que presentamos y que fueron aceptados por el Product Owner. No se estima se calcula (muy importante). Es la métrica de ágil que mide software funcionando. Mido en un sprint, métrica de producto.

La **capacidad** métrica de consumo interno, la usamos en el sprint planning para definir cuánto nos podemos comprometer como equipo en el sprint, alguna bibliografía dice que velocidad estimada pero no confundir. Esta si se estima. Se mide en horas ideales o en puntos de historia (diferencia en equipos maduros y no tan maduros o con menos experiencia). Relacionada con una práctica que es desagregación de la user story en tareas. Mido en un sprint, medida de proyecto.

**Running tested features**, ya no se utiliza ya que mide cantidades absolutas, no tiene en cuenta los story points sino que mide la cantidad de user stories hechas en una iteración. El hecho de que cuenta cantidades absolutas no es muy representativo porque no tiene en cuenta los story points de cada una.

No hay de proceso porque cada equipo define su proceso. No tiene sentido lo que hacemos en tradicional. La experiencia no se extrapola.

Las **métricas en Kanban** en procesos empíricos con enfoque Lean, apuntan a los procesos ya que Kanban se basa en la mejora de procesos.

El **Lead time** es desde que ingresó algo hasta que yo lo entrego, desde el momento que se está pidiendo un ítem de trabajo hasta el momento de que se entregó. Se miden en días.

El **Cycle time** mide el ritmo de producción, a partir de que yo empecé a poner foco en esto que el cliente me pidió hasta que lo terminó. Se mide en días.

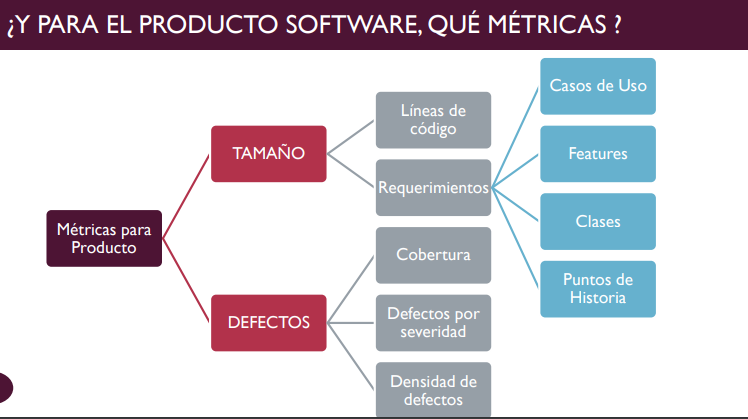
El **Touch time** toma solo las columnas de trabajo, cuantas horas son reales de trabajo sobre una pieza. Se mide en días.



La **eficiencia del ciclo de proceso**, mejor cuando se acerca a uno ya que indica que el touch time se acerca más al elapsed time (tiempo transcurrido), o sea significa que no hay desperdicio.



**Métricas orientadas a servicio**, miden la calidad en la entrega de servicio: expectativa de nivel de servicio, capacidad de nivel de servicio que el sistema puede entregar, acuerdo de nivel de servicio que es acordado con el cliente, umbral de la adecuación del servicio el nivel por debajo del cual este es inaceptable para el cliente. Se hace vinculado a los defectos, de acuerdo a la prioridad y se establece el tiempo máximo que le dan al cliente para resolverlo. Estas son más métricas para el producto.



Siempre se miden defectos, la cobertura es cuanto probamos de todo lo que tenemos que probar , osea cuantos cubriste con tus casos de prueba que probaste de todo lo que hay que probar. Las organizaciones deben tener un mínimo de cobertura.

Volatilidad de requerimientos mide cuánto cambian los requerimientos en el proyecto. Sirve para hacer análisis de impacto de cambios, para saber que margen de estabilidad vos tenes en los planes que haces.

Status de los requerimientos cuantos requerimientos tengo en cada estado.

Aseguramiento de calidad:

El **aseguramiento de calidad de procesos y productos** se centra en la implementación de medidas preventivas para garantizar que un producto alcance la máxima calidad posible. Esto implica tomar acciones anticipadas y realizar actividades antes de que el producto esté terminado, con el objetivo de asegurar que la calidad esté integrada en todas las etapas del proceso. En lugar de detectar y corregir errores o defectos después de que se hayan producido, el enfoque se centra en prevenirlos desde el principio.

La **CALIDAD** se refiere a 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 están implícitas o no se mencionan explícitamente. La calidad implica la capacidad de un producto o servicio para satisfacer y superar las expectativas y requisitos del cliente. La calidad es relativa a las personas.

El **aseguramiento de calidad de procesos** implica establecer y mantener estándares y procedimientos claros para llevar a cabo las actividades de manera consistente y eficiente. Esto incluye definir las mejores prácticas, documentar los procesos, capacitar al personal en la ejecución correcta de las tareas y establecer controles para monitorear y medir el desempeño.

Por otro lado, el **aseguramiento de calidad de productos** se centra en garantizar que los productos cumplan con los estándares de calidad establecidos. Esto implica realizar inspecciones, pruebas y evaluaciones durante el proceso de fabricación para identificar y corregir cualquier defecto o problema antes de que el producto final sea entregado al cliente. También se pueden implementar sistemas de gestión de calidad, como ISO 9001, para asegurar que se cumplan los requisitos y se sigan las mejores prácticas.

Cuando hablamos de la **calidad esperada** de un producto debemos definir desde qué perspectiva. Para tener calidad tenemos que integrar todas estas perspectivas. En el aseguramiento de calidad nos enfrentamos al desafío de tener que lograr que se alineen de la mejor manera posible la calidad programada, la calidad necesaria y la calidad lograda. La **calidad programada** se refiere a las expectativas y estándares de calidad que se establecen para el producto durante su planificación y diseño, es la calidad que se busca alcanzar y que se establece como objetivo a lo largo del proceso de desarrollo. La **calidad necesaria** es el nivel mínimo de calidad requerido para que el producto sea considerado correcto y pueda satisfacer los requerimientos y expectativas del usuario, conjunto de características y funcionalidades esenciales que el producto debe de tener para cumplir con su propósito. La **calidad lograda** se refiere a la calidad real que se ha alcanzado en el producto final una vez completado el proceso de desarrollo y las pruebas correspondientes, es el resultado concreto y medible de las actividades de desarrollo, aseguramiento y control de calidad. Al lograr que estas tres perspectivas de calidad coincidan disminuimos efectivamente el riesgo de desperdicio y de insatisfacción de los clientes.

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 aseguramiento de calidad, como revisiones técnicas y auditorías, para evaluar la aplicación de los modelos seleccionados. El proyecto a su vez es el ámbito donde uno trabaja para obtener el producto, la versión de producto que yo someto a evaluación se va generando en el contexto de un proyecto. Para el contexto del producto, existen técnicas y herramientas tanto para el aseguramiento de calidad, como la revisión de pares o las auditorías (De configuración, auditorías físicas y funcionales), como para el control de calidad, como es el Testing.

El aseguramiento de calidad de proceso se enfoca en el proceso de desarrollo de software y busca mejorar la forma en que se trabaja, mientras que el aseguramiento de calidad de producto se centra en evaluar y garantizar la calidad del producto de software resultante

El proceso es el único factor controlable para mejorar la calidad del Software y el rendimiento como organización. Podemos instanciar un proyecto que tenga calidad lo que resulta en un software de calidad.

Ámbitos por los que se mueve la ingeniería de software: Hacemos software utilizando un conjunto de personas y un conjunto de herramientas tenemos como unidad de gestión al proyecto y este es el que genera como resultado un producto de software. Este producto se genera de manera iterativa e incremental en sucesivos proyectos a lo largo de sus vidas y los proyectos para generar como resultado los productos usan procesos.

En los tres ámbitos está la motivación de hacer las cosas de la mejor manera posible e ir mejorando en el tiempo, se han diseñado mecanismos para ver cómo se va implementado esta mejora continua. Es decir si estamos en el ámbito del producto que este este de la mejor manera.

Los **Modelos de mejora** son esquemas o plantillas de trabajo para encarar un proyecto de mejora de un proceso. El propósito de los modelos de mejora es analizar este proceso que tiene la organización y armar un proyecto cuyo resultado no es un producto sino un proceso mejorado que se incorpora en la organización.

El **Modelo IDEAL** es un modelo que nos da el contexto para crear un proyecto cuyo resultado va a ser un proceso definido. Arranca con buscar un apoyo en la organización, que es muy importante porque este tipo de proyectos no suelen ser críticos o de más alta prioridad, entonces si no tengo alguien que me avale a realizar este proyecto no suelen completarse exitosamente. Se hace un diagnóstico para ver en dónde estoy parado, ver que hace bien y que se hace mal, se llaman Análisis de Brecha o sea para ver a dónde quiero llegar y en donde estoy ahora. Se definen planes de acción que deben de aprobarse, luego se escriben los procesos con sus roles y actividades y todo lo que regula el proceso, también se define qué prácticas. Que tenes que hacer vos para tener como resultado un proceso con las prácticas que vos quieras que esté listo para ser implementado. Lo empezamos a utilizar en los proyectos en donde está involucrado este proceso, y documentamos los resultados los analizamos y decidimos si tenemos que volver a ejecutar el modelo. Trabajar con proyectos pilotos, no voy a hacer la implantación de todos los procesos y después hacer la implantación masiva.

Los **modelos de calidad** se usan como referencia para bajar lineamientos que nuestro proceso tiene que cumplir para poder llegar a un cierto objetivo. Son modelos que usamos para crear.

El **modelo CMMI** es un modelo descriptivo que nos dice el qué y no el cómo. Las siglas significan **integración de los modelos de madurez y capacidad**. Proporciona una guía de mejores prácticas para mejorar la capacidad y madurez de los procesos organizacionales. El modelo se basa en buenas prácticas recopiladas de la industria y está diseñado para ayudar a las organizaciones a alcanzar niveles superiores de calidad, eficiencia y satisfacción del cliente. Me ayuda a ver qué prácticas tengo que incluir en el proceso para cumplir con los objetivos.

Representaciones

Por etapas: mide la madurez de la organización, determina que tanto puede cumplir la organización.

Se organizan en **niveles de madurez** de la organización en etapas predefinidas. En cada nivel de madurez hay un conjunto de prácticas y capacidades específicas que deben alcanzarse para avanzar al siguiente nivel. Se divide a las organizaciones en maduras e inmaduras. Mientras más madura es la organización más capacidad tiene para cumplir con los objetivos, entonces mejoraba la calidad de sus productos y disminuye sus riesgos. Una ventaja que tiene es que provee una única clasificación que facilita las comparaciones entre organizaciones.

* Nivel 1 INICIAL: organización inmadura, donde los procesos son impredecibles, pobremente controlados y reactivos (atacar el problema una vez sucedió). No existe visibilidad respecto del proceso. No sabemos cuando va a terminar, cuanto nos va a costar, ni la calidad de lo que se obtiene.
* Nivel 2 ADMINISTRADO: ya son maduras. En este nivel, la organización comienza a tener controles básicos para la gestión de proyectos y procesos. Se busca una mayor estandarización y se inicia la recopilación de métricas para el seguimiento y la toma de decisiones. Si mi organización llega a este nivel puedo decir que tiene madurez para administrar sus proyectos y que el resultado de sus proyectos va a ser un producto de software de lo que se sabe que se espera de él. Área de proceso opcional que tiene que ver con la terciarización.
* Nivel 3 DEFINIDO: En este nivel, la organización ha establecido y documentado procesos definidos y estandarizados. Se definen roles y responsabilidades, y existe retroalimentación.
* Nivel 4 CUANTITATIVAMENTE ADMINISTRADO: En este nivel, la organización tiene la capacidad de cuantificar y medir su rendimiento. Se establecen métricas, y se recopilan datos para analizar el desempeño de los procesos y procedimientos
* Nivel 5 OPTIMIZADO: En este nivel, la organización se enfoca en la mejora continua y en la optimización de sus procesos.

Continua:

Elige áreas de proceso dentro de lo que el modelo te ofrece, y yo elijo qué procesos son los que yo quiero mejorar por separado, entonces en vez de medir la madurez de toda la organización mido la **capacidad de cada área de proceso en particular que puede estar en cualquier nivel**. En mi organización puede haber procesos en nivel cero que quiere decir que no se ejecuta o sea que no hay nada en la empresa que lo esté haciendo. Las áreas de procesos que se enfocan en aspectos específicos del desarrollo y gestión de software, como la planificación, el monitoreo y control, la gestión de requisitos, la implementación y la evaluación. Los niveles dicen que áreas de procesos debe cumplir obligatoriamente.

Tiene 3 dominios o ámbitos de mejora llamado constelaciones: son especializaciones

* CMMI DEV: provee la guía para medir, monitorizar y administrar los procesos de desarrollo. Se enfoca en el desarrollo de software.
* CMMI SVC: provee la guía para entregar servicios internos o externos.
* CMMI ACQ: es de adquisición, cuando la empresa no hace cosas si no que contrata gente que va a hacer cosas por ellos, entonces provee la guía para permitir seleccionar y administrar la adquisición de productos y servicios. Tercializas.

Roles y grupos

Los grupos se refieren a existencia de roles que cubran ciertas tareas, estos roles se adaptan al tamaño de la organización, al nivel de madurez que la organización quiere alcanzar. Lo importante es que exista alguien responsable de cubrir las actividades de cada uno de los roles o grupos.

Comparación CMMI y Agile:

1. Filosofía y enfoque:

CMMI (Capability Maturity Model Integration):

* Filosofía: CMMI se basa en un enfoque de mejora continua y madurez organizacional. Se centra en establecer procesos bien definidos y gestionados para mejorar la calidad y la eficiencia en el desarrollo de software.
* Enfoque: CMMI se enfoca en la estandarización de procesos y la medición constante del desempeño para alcanzar niveles predefinidos de madurez en la organización.

Agile:

* Filosofía: Agile se basa en la flexibilidad y la colaboración. Promueve la entrega de valor al cliente de manera rápida y continua, adaptándose a los cambios y priorizando la satisfacción del cliente.
* Enfoque: Agile se centra en equipos autoorganizados, iteraciones cortas y en la adaptación a los requisitos cambiantes a lo largo del ciclo de desarrollo.

1. Gestión de proyectos:

* CMMI: CMMI pone un fuerte énfasis en la gestión de proyectos, incluyendo la planificación detallada, la gestión de riesgos y la medición de la calidad del proceso y del producto. Utiliza una estructura jerárquica para la asignación de responsabilidades.
* Agile: adopta un enfoque más ligero en la gestión de proyectos. Se basa en la autoorganización de equipos, la colaboración continua y la adaptación a medida que se descubren nuevos requisitos. Los roles de gestión tradicionales son menos prominentes en Agile.

1. Entrega de software:

* CMMI: se enfoca en entregar productos basados en procesos definidos y controlados. Se espera que los entregables cumplan con estándares predefinidos y sean resultado de procesos bien gestionados.
* Agile: se centra en la entrega continua e incremental de software funcional. Los productos se entregan en ciclos cortos, generalmente de 2 a 4 semanas, con la prioridad en la satisfacción del cliente y la adaptación a sus necesidades cambiantes.

1. Flexibilidad y adaptabilidad:

* CMMI: tiende a ser menos flexible y adaptable a cambios en los requisitos a lo largo del ciclo de desarrollo. Se basa en procesos definidos y puede requerir esfuerzos significativos para modificarlos.
* Agile: se destaca por su capacidad de adaptarse rápidamente a los cambios. Se espera que los equipos Agile sean flexibles y capaces de ajustar su trabajo según las necesidades cambiantes del cliente y el mercado.

Las **auditorías** son evaluaciones independientes de los productos o procesos de software para asegurar el cumplimiento con estándares, lineamientos, especificaciones y procedimientos, basada en un criterio objetivo incluyendo documentación que especifique la forma o contenido de los productos a ser desarrollados, el proceso y cómo debería medirse el cumplimiento.

Hay **tres tipos de auditorías**: de proyecto, de configuración funcional y de configuración física. La **auditoría de proyecto** es la responsable de ver si el proyecto se ejecutó con el proceso de desarrollo que se dijo que se iba a ejecutar, al principio del proyecto elegís un proceso al cual te comprometes, nivel de cumplimiento del proceso que nos comprometemos a utilizar. Las auditorías

Los **roles de la auditoría** son el auditado, alguien del equipo suele ser el líder del proyecto. El auditor tiene como principal característica que tiene que ser de fuera del proyecto (y de la organización) para que las auditorías sean objetivas e independientes. El gerente de SQA es el responsable de manejar las personas que tiene a su cargo que hacen auditorías en los proyectos, el arma un plan general de todas las auditorías que se van a hacer los equipos, administra los recursos y costos, hace seguimiento si puede haber alguna discrepancia.

El **proceso de auditoría** comienza con la preparación y la planificación de forma conjunta el auditado y el auditor, se ejecuta y el auditor pide documentación y hace preguntas, esta luego es analizada y se hace un reporte del resultado que se entrega al auditado, este lo analiza y puede no estar de acuerdo o si pero en función de eso queda el reporte final. Después puede pasar que el auditor haga un seguimiento de las desviaciones que encontró hasta que él considere que han sido resueltas.

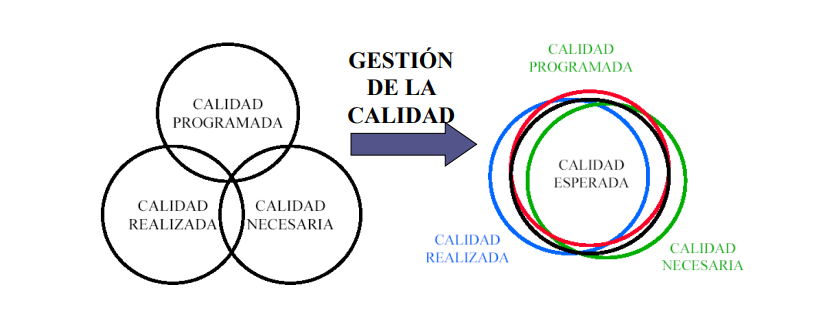
Los **tipos de resultados** de las auditorías son buenas prácticas cuando el auditor encuentra algo superador, o sea algo que se ha desarrollado mucho mejor de lo esperado. Las desviaciones son cualquier cosa que no se hizo o que no se hizo como el proceso dijo que se iba a hacer. Las observaciones son cosas que advierte el auditor que no llegan a ser desviaciones pero son riesgosas entonces el auditor las destaca.

Las **métricas de la auditoría** para obtener información de lo que pasa en las auditorías que pueden ser esfuerzo por auditoría, cantidad de desviaciones, duración de auditoría, cantidad de desviaciones clasificada por PA de CMMI.

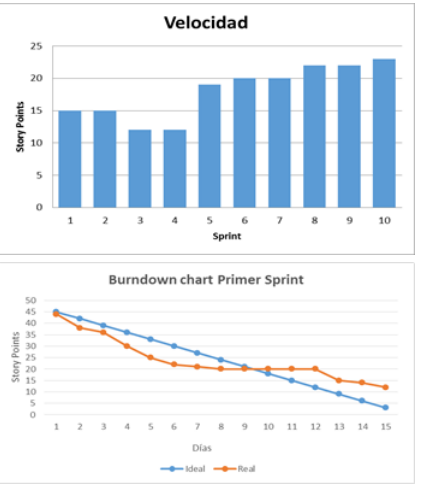
ISO se certifica CMMI se acredita, si hace esto es porque se está usando un proceso definido (porque si o si miran el proceso porque los productos no se puede estandarizar para el producto). Primero debo definir un proceso compatible y después ver que ese proceso se use. debemos armar un proyecto de mejora de proceso para poder acreditarlo que va a dar como resultado un proceso, tiene que tener un plan y en algún punto vamos a explicar las actividades que me van a dar como resultado acá aparecen los modelos de mejora porque son una guía para armar el cronograma del proyecto de mejora. Los modelos de calidad son el espejo que yo necesito para mirar el proceso, tengo que ver cual es compatible con el proceso. Los proyectos van a tener que ejecutar las actividades, pero lo voy a tener que adaptar porque hay guías de adaptación donde se definen los criterios para ajustarlo a nuestro proceso. Para hacer la acreditación tenes que tener un proceso definido. La evidencia objetiva son los artefactos que vas generando a lo largo del proyecto, ágil no deja evidencia por ejemplo las daily no dejan evidencia o ningún resultado. para hacer las evaluaciones también se usan modelos pero son modelos de evaluación.

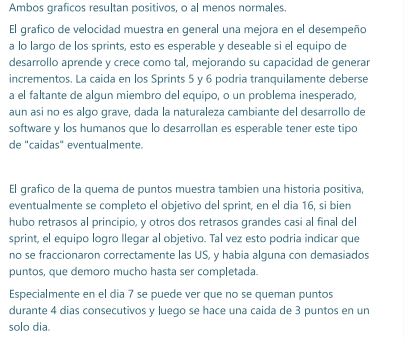
Preguntas teóricas:

1. Dado el siguiente gráfico explique la razón por la cual todo lo que está fuera de coincidencia es considerado desperdicio.



1. Suponiendo un desarrollo ágil, analice en los siguientes gráficos el comportamiento del proyecto manifestando en los mismos y explique detalladamente la interpretación que hace luego del análisis.





1. Realice un análisis de los principios del manifiesto ágil y explique las prácticas que propone el framework Scrum para aplicar en forma concreta los principios.
2. Realice un análisis de los principios lean y explique las prácticas que propone Kanban para aplicar en forma concreta los principios.