**Resumen Segundo Parcial:**

A white background with black text

Description automatically generated

**28/09**

**SCRUM (ver clase grabada)**

(Completar)

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

Trabajar en sprint (una iteración fija de máximo 4 semanas) o un contenedor también se les dice.

Setear un conjunto de condiciones (en el sprint planning) y luego durante un sprint no podemos aceptar cambios.

El ciclo de vida de SCRUM es el ciclo de vida iterativo incremental de tiempo fijo.

Soluciones adaptativas para gestionar

Timebox: el tiempo encerrado en una caja, todas las actividades deben tener duración fija. Cada equipo va a acordarlo, de preferencia menos tiempo. Daily 15 minutos. El tiempo es difícil de organizar. Nos obliga a ser respetuosos con nuestro tiempo y el tiempo de los demás.

La decisión de cancelar un sprint es del producto owner.

* Sprint planning: definimos el objetivo del sprint. Todos asisten al planning, incluido el producto owner.
* Daily
* Sprint demo/review: tiene que estar el producto owner que es el que define si esta bien o no. Se mejora el producto.
  + Se calcula la velocidad que mide producto en story points (la cantidad de story points que acepto el producto owner, la cantidad de producto que hice bien). Se da generalmente en la review. Si no se acepta la user se vuelve al producto backlog.
* Retrospective: se mejoran los procesos. Mejoramos los tres aspectos de los procesos empíricos, sobretodo la transparencia. Solo el equipo, puede venir un modelador de afuera así el scrum máster puede participar y no moderar. Seria mejor que este el producto owner, pero no obligatorio. Relacionado con el principio 12 del manifiesto ágil. Mejoras.
* Story time

Cuatro actividades más una que es continua o sea cuando yo la quiero usar y es la de refinamiento del producto backlog.

Categorías de responsabilidades:

* Developers: los del equipo de trabajo, crean producto.

Compromisos asociados a los artefactos.

El producto backlog tiene un compromiso con el objetivo del producto formalizado.

El sprint backlog tiene un compromiso con el objetivo del sprint. Los cambios los aceptamos si no ponen en peligro el objetivo del sprint.

El product increment tiene compromiso con el objetivo del definition of done (acuerdo con el equipo lo que debe tener el producto para que lo pueda ver el product owner).

Potentially shippable product increment es que si el PO quiere ponerlo en producción debería poder hacerlo.

Capacidad del equipo en un sprint:

Cálculo de capacidad del equipo en un sprint

Se hace en el planning y si se estima, tenemos que ver el trabajo que vamos a necesitar. Se puede medir en puntos de historia (vamos a poder hacer tantos puntos en tanto tiempo), otra manera de medirla es en horas ideales. El equipo al inicio de la planificación hace el cálculo.

Ejemplo:

A table with text and numbers

Description automatically generated

Asume developers que puede garrar una user y pueden arrancarla y terminarla.

Tablero o taskboard: duran lo que dura el sprint, generalmente se hacen en el planning.

Tiene tres columnas básicas:

* To do
* Doing: cumple con el definition of ready
* Done: cumple con el definition of done

A diagram of a process

Description automatically generated with medium confidence

Se pone las stories en una columna y se divide por filas cada user. Las user se dividen en tareas y son las tareas las que se ponen en las tres categorías. Las tareas se pueden asignar a varias personas del equipo. En el done tienen que estar todas las tareas terminadas para que la user pase a estar done.

Granularidad del tablero tiene que ser fina, porque si son muy grandes no nos podemos equivocar. Tener muchas tareas chiquitas es mejor que pocas grandes. Hay mas avances porque al trabajar con más chicas terminas más rápido.

Sprint burndown charts: herramienta que sea usa para el seguimiento del progreso del producto. Gestión por horas está mal.

**Clase 5/10:**

**Testing de software o prueba de software**

Se tiene que hacer en el desarrollo de software

Hacemos foco en el aseguramiento de calidad del producto.

Aseguramiento de calidad de proceso y de producto: prevención (más barata que la corrección), tiene que ver con incorporar acciones durante el proceso de creación de software con un propósito claro que es económico. Conjunto de acciones que podemos hacer para evitar los errores y que estos se conviertan en defectos.

Edad de defecto tiempo que pasa entre que aparece el efecto hasta que se resuelve. Los defectos de más edad son más costosos de resolver.

La herramienta para hacer aseguramiento de calidad de producto son las revisiones de pares o técnicas, no las hace el jefe ni el cliente, sino que lo hacen los compañeros de equipo donde no hay relaciones jerárquicas. Se puede hacer revisiones técnicas

Inspección de software técnica que usamos para la revisión de pares formales. Hay relaciones informales son las walkthrough o recorrida. Le hacemos revisiones técnicas al código, al diseño, a la documentación, a los palanes, requerimientos, estimaciones, todos los artefactos que alguien construyan se le pueden hacer revisión.

La otra técnica de aseguramiento de calidad son las auditorias que a la agilidad no le gusta mucho.

El testing entra cuando el producto ya está listo o sea el código ya se puede ejecutar. El testing no es aseguramiento, sino que es control. El testing hace validación y verificación. Lo tengo que hacer con un producto que ya está listo, entonces detecto defectos.

Automaticemos lo más que podamos en el testing.

Aseguramiento de calidad de proceso es donde aparecen los modelos de calidad, por ejemplo, ISO 9001 o la ISO 27000. Hacer auditorias y también las revisiones (pueden ser las mismas que las de producto, pero el foco es diferente). Un modelo de calidad te dice que, y no te dice como, o sea te dice que cosas debe tener para que se de buena calidad. Se hace en dos niveles, a nivel de proceso y luego si es que se usa ese proceso en el proyecto. Y así realizamos la adaptación. Controlo el proyecto, pero lo que controlo es el cumplimiento del proceso.

Auditorias informáticas: auditorias que se le hacen al producto cuando está en producción. Peritaje analizar sistemas en funcionamiento.

Si el proceso que yo uso para crear el producto tiene calidad entonces el producto que yo creo entonces también va a tener calidad.

**TESTING**

Diciplinas de soporte trabajan para que 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 exitoso cuando encontremos defectos.

**Diferencia entre error y defecto**, el error se descubre a partir de técnicas especificas 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 traslado 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 esta bueno 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:** cuan 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. Cuanto 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 probas 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 mas 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. Continuos 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:**

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 especifico y se utiliza en diferentes etapas del proceso de desarrollo y despliegue.

Tienen que estar separados porque uno tiene que llegar con una versión limpia para probar. Los desarrolladores no tienen acceso al ambiente de testing. Las pruebas de aceptación de usuario se hacen en el ambiente de preproducción, pero en general como las empresas son re ratas lo hacen en el ambiente de producción (no está muy bien).

Caso de prueba: set de condiciones o variables bajo las cuales un tester determinara si el software esta funcionando correctamente o no. Nos permite reproducir los defectos, si no se puede reproducir no es un defecto. Hace que una prueba sea sistemática. Es imposible probar todo entonces hay que elegir, de ahí surgen las técnicas. Tener menor cantidad de casos de prueba. Los datos son importantes, porque son casos específicos para cada caso de prueba. En el parcial poner datos concretos.

La información la saco de los requerimientos porque definen lo que el sistema tiene que hacer.

A diagram of a process

Description automatically generated

Condición de prueba: es la reacción esperada de un sistema frente a un estímulo particular, este estimulo está constituido por las distintas entradas.

Ejemplo caso de prueba:

A paper with blue writing

Description automatically generated

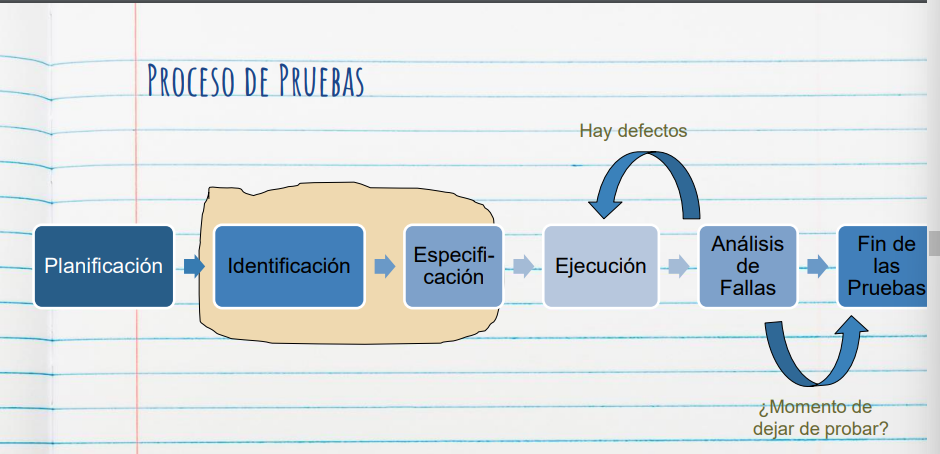
Estrategias de prueba: caja negra y caja blanca. (ver en el practico). Esto se evalúa en el segundo parcial.

Testing exploratorio se usa para conocer el sistema cuando no tengo documentación.

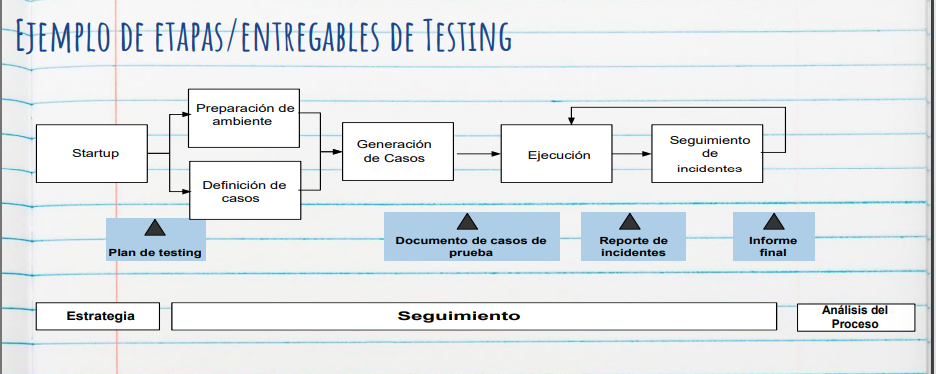
Ciclo de prueba: es la ejecución de un conjunto de casos de prueba sobre una versión del sistema a probar. Cantidad de ciclos de prueba ideales son dos (menos utópico que uno). El primer ciclo de prueba se llama ciclo cero. La calidad de productos esta relacionada con la cantidad de ciclos de pruebas.

Concepto de regresión: prueba sin regresión es en donde yo solo pruebo los errores que yo detecte y no todo. Con regresión se prueba todo ya que se puede haber pasado algún error que no hayamos visto pero que si esta, todo de nuevo como si fuera el ciclo cero, todos los ciclos se comportan como el ciclo cero.

Proceso de pruebas: se planifica, es difícil definir cuando dejar de probar.



Entregables que se generan en un proceso de prueba:



Es responsable quien hace el testing definir la severidad y la prioridad.

Clase 12/10

Filosofía lean

framework Kanban

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.

Amplificar el aprendizaje: transparencia, busca transformar el conocimiento en prácticas compartidas. O sea que el conocimiento sea del equipo de todos.

Embeber la integridad conceptual: atención continua de excelencia técnica, velar por la calidad del producto.

Diferir compromisos hasta el ultimo momento responsable: tiene que ver con la toma de decisiones, que una buena toma de decisiones esta basada fuertemente en información. Postergar el asumir un compromiso hasta que tengas más información para poder asumir el compromiso. Tiene que ver con el just in time. Tiene que ver también con la planificación. 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.

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.

Entregar lo antes posible, retroalimentación, antes entregue al cliente antes puedo obtener retroalimentación. La entrega continua es una forma de entrega lo antes posible.

Gastos en producción en lean: 7+1

A screenshot of a diagram

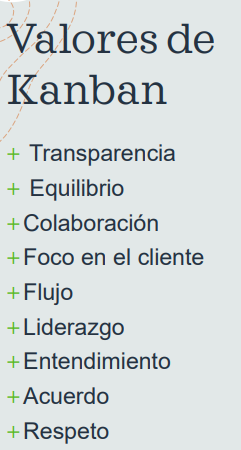
Description automatically generated

El +1 es el talento no utilizado y es el más complejo, genera mucho desperdicio.

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.



El flujo de trabajo tiene que fluir y evitar cuello de botella.

Equilibrio cuanto vamos a hacer de cada cosa y en que momento para logra el flujo y que no haya atascamiento.

Principios de Kanban: organizados en dos grandes grupos

* 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

Practicas generales de Kanban: método menos prescriptivo que Scrum. Las practicas nos dicen el que y no el como asique nosotros decidimos como mierda queremos hacerlo.

1. Visualizar: tiene que ver con esta necesidad de tener que ver todo el proceso y tener toda la información para hacer el trabajo accesible. Utiliza como herramienta fundamental el tablero igual que en Scrum, pero son diferentes. Nuestro proceso se tiene que mapear en el proceso. Los kanban con minúscula son los post its que nos dicen que hacer.
2. Limitar el trabajo en progreso: WIP (work in progress), limites la cantidad de trabajo que podés hacer en cada columna. Establecer límites es decisión nuestra, debería haber en todas las columnas.
3. Gestionar el flujo: lograr que el trabajo fluya, las colas acumulan el trabajo y se controla el control de flujo con los límites. Las colas te trabajo son las colas de progreso y las otras son de acumulación.

A red and white rectangular box

Description automatically generated with medium confidence

1. Hacer explicitas las políticas: pocas políticas, pero explicitas en el tablero. Definition of done y definition of ready son políticas que tienen que estar explicitas 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).
2. Implementar ciclos de retroalimentación o feedback
3. Mejorar de manera colaborativamente: como hacemos para que nuestro sistema funcione cada vez mejor.

Como aplicar Kanban es dividir el trabajo en piezas.