

Tareas para el segundo parcial

- Unidad 1: Gestión Lean Agile de Productos de Software**
 - Principios de SCRUM 2018 a nivel ejecutivo y ejecuta (versión)
 - Principios Fundamentales en el contexto del desarrollo de software
 - Unidad 1: Gestión Lean Agile de Productos de Software
- Unidad 4: Aprendizaje del Ciclo de Proceso y de Producto**
 - Conocer las principales etapas
 - Importancia de la etapa para el Ciclo de Vida y Desarrollo
 - Algunas etapas que se realizan en el desarrollo de software
 - Principales Modelos de Ciclos iterativos (CMW - SPICE - ISO) y sus métricas de evaluación.
 - Diferentes tipos de Auditorías: Auditoría de Proyecto y Auditoría del Ciclo de Desarrollo.
 - Procesos de auditoría y su importancia
 - Catálogo de Proyectos: Priorización de pautas para el software - Normas y tipos de pautas para el software
 - Técnicas y Herramientas para controlar software.
 - Normas y Herramientas para la validación de versiones técnicas del software.

Unidad 2: Gestión Lean Agile de Productos de Software

Scrum

Método de trabajo flexible que ayuda a las personas, equipos y organizaciones a generar valor a través de soluciones adaptativas para problemas complejos. Nos permite obtener un incremento del producto durante o al final de cada Sprint.

Se dice que es liviano porque no es un manual cerrado que nos impone como crear software, sino que se definen las reglas de juego.

Scrum no hace fuerza en entornos complejos, donde las incertidumbres superan a las certidumbres y la solución va emergiendo a medida que avanzamos.

Scrum no nos da el paso paso de como llevar adelante un proyecto

Un producto de software no se termina en una iteración

El product Owner ordena el trabajo por hacer en el Product Backlog

El Equipo Scrum realiza en conjunto el Sprint Backlog y se indica que tareas se atenderán durante ese sprint y las prioridades

Para hacer scrum debemos seguir todos los elementos del framework

Agile contiene a Scrum y otras técnicas, es decir Scrum es una de las técnicas Agile

SCRUM

En la actualización 2020 no se habla más de roles, sino que de responsabilidades

En la guía se le asigna responsabilidades al equipo Scrum

- Responsable de todas las actividades con el producto: diseño, desarrollo, pruebas, testing, mantenimiento, etc.
- Responsable de crear incremento de Producto valioso y útil en cada Sprint
- Responsable de tener un DOD acorde al producto en cuestión
- Predica los valores de compromiso, foco, sinceridad, respeto y coraje

Tres responsabilidades (roles)

- Scrum Master
- Product Owner
- Desarrolladores

Otras responsabilidades

- Sprint (contiene los demás eventos)
- Sprint Planning
- Daily Standup
- Sprint Review
- Retrospective del Sprint

Tres artefactos con sus compromisos asociados

- Product Backlog
- Sprint Backlog
- Incremento del Producto

Los roles en Scrum

- Compromiso
- Foco
- Sinceridad o apertura
- Respeto
- Coraje

Responsabilidades

Scrum Master: es quien sabe implementar Scrum tal y como se define en la guía. Es aquél que acompaña al equipo y lo orienta para facilitar una buena aplicación framework Scrum.

Ayuda al PO a definir el Product Backlog y se sienta con el equipo Scrum a definir el DOD. Es el que incentiva a todo el equipo a aplicar Scrum.

Product Owner: encarga el manejo y valor del producto en el que trabaja el Equipo Scrum. Es quién es el representante del cliente y sabe las necesidades (el cliente) y de gestionar el Product Backlog. Define el Product Goal (puede recibir ayuda de Scrum Master), crea los elementos del Product Backlog (dándoles prioridad) y comunicándolo a todos las partes involucradas para su análisis y entendimiento.

Desarrolladores: son quienes crean el trabajo de Sprint y se comprometen a crear el incremento. Junto al Equipo Scrum crean el Sprint backlog, siguen el DOD, adaptan el Sprint Backlog a diario direccinando el progreso hacia el Sprint Goal

Eventos Scrum

El Sprint: es el evento que contiene a los otros 4 eventos.

Cada evento tiene un time-box, es decir una duración máxima predeterminada.

Las ceremonias son oportunidades para inspeccionar y adaptar los artefactos con el objetivo de hacerlos más transparente.

Sprint: para regularidad y minimizar las reuniones no contempladas por el framework.

Sprint: ciclo corto de desarrollo encaminado a la obtención de un incremento de producto, cuya duración no debe exceder de un mes.

Sprint Planning: se plantea el trabajo a realizar durante ese sprint con el objetivo de cumplir el Sprint Goal (definido ahí), se realiza a continuación.

Daily Scrum: reunión por parte de los desarrolladores que se realiza todos los días desde la inspección del progreso hacia la consecución del Objetivo del Sprint y se adapta el Sprint Backlog.

Sprint Review: El Equipo Scrum y los stakeholders inspeccionan sobre todo el Sprint y el incremento creado y se establecen futuras acciones.

Sprint Retrospective: es el último evento de Sprint, es donde para que el Equipo Scrum se inspeccione a sí mismo en lo relativo a procesos, interacciones y herramientas.

Retrospective del PB: está planteada en términos porcentuales porque es el trabajo que se hace continuamente durante todo el Sprint, no tiene un momento determinado o específico.

Artefactos Scrum

Representan trabajo y valor y tienen la función de transmitir lo que pueden ser inspectables y adaptados.

Product Backlog: lista ordenada de todo aquello conocido que el producto puede necesitar. Es gestionado por el PO.

Todos los tipos de producto que se lleva a configurar el product backlog (el PB no es una URN, nunca tendrá todas las características del producto por las que termina y no están o las que todavía no se me ocurrieron o conozco. Aca se definen los releases (durante la planificación de producto) y se dice las características incluidas en cada sprint).

Sprint Backlog: plan de trabajo para el desarrollo durante el Sprint. Se basa en el ultimo elemento del Sprint, Sprint Goal, elementos del Product Backlog seleccionados para este Sprint, plan de ejecución para crear el incremento.

Incremento: iteración de producto que ha de obtenerse cada sprint, incluye la funcionalidad añadida durante el Sprint como la ya creada en los Sprints anteriores. Se define las características que irán a incluirse en cada sprint. Se define una tasa de desarrollo que se necesita.

Compromisos de Scrum

En la guía Scrum 2020 se lleva el valor compromiso a su máxima expresión asignandole a cada artefacto un compromiso Se agrega formalmente el término compromiso.

El Product Backlog debe estar alineado con el objetivo del producto. Lo que nosotros agreguemos o quitemos debe estar alineado con el objetivo del producto.

En el Sprint Backlog se debe definir el objetivo del Sprint. Originalmente no se permitian cambios en el sprint pero aquí se otorga un poco más de flexibilidad diciendo que mientras no se impacte ni modifique el objetivo del Sprint se puede aceptar algún cambio durante el transcurso del mismo.

Incremento de Producto esta asociado al Definition of Done.

Métricas de SCRUM

Capacidad: es la única de las básicas que recomienda SCRUM que se estima y se utiliza para poder cumplir compromiso de trabajo. Los equipos deben estimar lo que pueden hacer en un sprint, la cantidad de trabajo que el cliente quiere que realicen los usuarios del product backlog que asumen el Sprint Backlog para comprometerlos a terminarlos en ese Sprint. Se estima porque se realiza al inicio del Sprint, y se hace con las horas ideales (horas en las que realmente estamos trabajando en un determinado producto en particular, sin los tiempo muertos). Cuando los equipos son mas maduros y tienen mas experiencia se puede estimar en Story Points (se divide cada historia en los Story Points que podrá cubrir). Recomendación: trabajar con rango temporal (no se hace estimando las horas para largo determinar el alcance).

Variables de información para la toma de decisiones: cuando dura el Sprint, el cálculo de capacidad y el objetivo del Sprint. Con esta info determinamos que elementos del PB vamos a trabajar.

Velocidad: Es una métrica que se va obteniendo a lo largo de los sprints contando los SP que el PO me aceptó. La velocidad corrige los errores de estimación.

Burnin Test Estimaciones: casi ni se usa, cuenta cantidad de características funcionando que se desarrollaron en una iteración (o sprint) en SCRUM.

Visibilidad / Ambiente de trabajo

Se relaciona con el principio de que la mejor comunicación es cara a cara

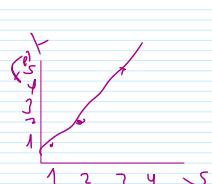
El equipo define sus horarios de trabajo

Herramientas de SCRUM

Taskboard: herramienta visual utilizada para rastrear el trabajo en curso y proporcionar visibilidad sobre las tareas o historias de usuario que un equipo de desarrollo está manejando durante un Sprint.

Hay una configuración de tablero básica pero se puede adaptar de acuerdo a como el equipo quiere trabajar. Puede ser multidimensional (que el miembro del equipo pueda empezar una característica del Sprint Backlog) o funcionales donde se distinguen las tareas por sus funcionalidades.

El código y el río del día es opcional, la estimación pueden ser horas ideales o SP (es lo que recomienda SCRUM porque al estimar en SP se estima sobre el producto, no sobre el trabajo)



Scrum hace mucho énfasis en que sea multifuncional (que cualquiera pueda agarrar una tarea e iniciarla, trabajaria y terminaría)

Tarea 3: columnas

Cuando el equipo necesita hacer una desagregación de las user stories en tareas.

TO DO	IN PROCESS	DONE
Características que salieron del product backlog y entran en Sprint Backlog	Lo que toma una persona para trabajar	Cuando termina y se cumple con el Definition of Done

En este esquema una US va a ser realizada por varias personas.

Se puede agregar una columna Done Done para definir que la US fue finalizada. Es pasar las US a esa columna para saber que están terminadas

Los nombres aparecen en la columna IN PROCEESS porque cualquiera puede iniciarla y se irá llevando con la autogestión del equipo

Granularidad de las tareas:

Cuando tenemos un tablero con pocas tareas se tratan de tareas que provienen de US estimadas con más de 8 puntos.

imposible la gestión binaria (decir si se termina o no) porque perdemos la visibilidad sobre las tareas en las que estamos trabajando.

Si trabajamos con Granularidad más fina, trabajaremos con mas historias más chicas. La ventaja de granularidad más fina es que nos trae mas posibilidad de maniobra pudiendo empezar y terminar las tareas durante un sprint.

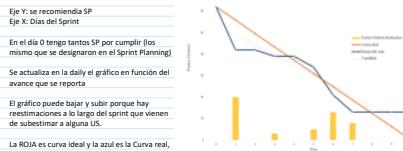
El siguiente apunta a las cosas binarias, las cosas están terminadas o no, no hay nada en el medio.

Gráficos

Sprint Burndown Charts: Gráfico que nos muestra el trabajo terminado o el que nos falta para cumplir con el compromiso del Sprint. Este sigue SCRUM, el BurnUp es opcional.

No es una métrica, es un gráfico.

SPRINT BURNDOWN CHART



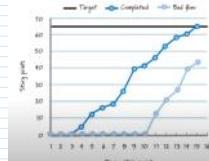
La gestión por horas se enfoca en el esfuerzo, en el trabajo, pero se corre el riesgo de desfasar el consumo de horas con la realidad de lo que se ha trabajado. Se recomienda decir cuantas horas trabajado realmente, no decir cuantas horas trabajado las 6 horas que tenia estimadas para una tarea pero eso no significa que las haya terminado realmente.

Los Burndown Chart se borran al final de cada sprint.

Esa velocidad se utiliza para la estimación del siguiente sprint. La experiencia tomada es válida para el mismo equipo sobre el mismo proyecto, no se puede extrapolar experiencia en otras personas.

Scrum Burndown Chart

Es un gráfico permanente porque hace seguimiento de como voy trabajando en un producto a lo largo del trabajo de los sprints. Es una curva que sube porque va mostrando lo que voy haciendo para llegar al objetivo del producto.



Niveles de planificación

No hay ningún producto de software que se pueda terminar en un sprint.

En la parte de los releases vemos que es la planificación del producto/baúl de release.

La planificación siempre está sujeta a cambios.

- Más tarde es la daily: sincronizamos y vamos viendo la necesidad de hacer ajustes para lograr cumplir con el compromiso.

- Sprint Planning: es la planificación a nivel iteración donde observaremos como resultado el sprint backlog o la configuración del tablero, donde la entrada es el objetivo del Sprint, la duración y la capacidad estimada.

Se planifica que cumplir el sprint backlog, porque cada uno tiene un compromiso en tareas y estimadas en hs ideales

Se planifica que cumplir el sprint backlog, porque cada uno tiene un compromiso en tareas y estimadas en hs ideales

Se planifica que cumplir el sprint backlog, porque cada uno tiene un compromiso en tareas y estimadas en hs ideales

Se define cuantos Sprints voy a necesitar para poder entregar esta cantidad de características del producto estableciendo las condiciones de contexto. Y qué características incluir en cada sprint.

Yo tengo una visión de producto que me lleva a configurar el producto backlog y una definición del release.

Se busca cuantos sprint se necesita para alcanzar el release

Se define una velocidad que puede cambiar

La salida:

Rango de tiempo para cumplir en cada sprint

Cuánto tiempo necesita para terminar el release

Siempre en una base de condiciones o restricciones como cuanto quiero hacer, tiempos que manejo y eso

Cadencia de los Sprints → la define cada organización, es la frecuencia con la que se entregará el release



Mientras mas alto el nivel de planeamiento mas amplio, el horizonte es mas amplio y las personas que intervienen son diferentes, por ejemplo el equipo scrum no participa hasta la planificación del relevo.

- Planificación del Producto: es la sumatoria de releases. Implica tomar decisiones de funcionalidad que quiero ofrecer yo o no en el producto y/o funcionalmente también. Indica en que release voy a entregar cada actividad.

- Planificación portfolio: en una linea de producto es planificar a que producto le quiero dar mas recursos, productos a desarrollar, etc.

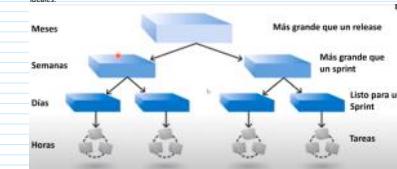
Google Google es todos los productos (Google Keep, Google Drive, Google Maps, ...)

Estimaciones según granularidad

El backlog de portafolio contiene todos los productos que tiene la empresa ordenados prioritariamente

Product backlog: tiene una tendencia cada vez mas fuerte a estimarlos por Story Point, mientras mejor están estimados estos items, mas facil serán estimarlos en el sprint.

Sprint backlog: aquí toman una User Story (estimada en SP) y las pueden desagregar en tareas que se estiman en horas ideales.



Nivel	Horizonte	Qué	Quién	Entregado
Portafolio	1 año o más	Stakeholders y interesados	Administrador de art.	Backlog del Portafolio
Producto	Algunos meses o más	Product Owner y Stakeholders	Visión y evolución del producto	Visión del Producto, Roadmap y Backlog de desarrollo de alto nivel
Relevo	3 (o más) a 8 meses	Equipo Scrum, Desarrolladores	Administración de desarrollo y gerencia de desarrollo	Plan de Relevo
Día	Cada Iteración (de 1 semana a 1 mes)	Equipo Scrum	Que impacto entregar en el desarrollo	Objetivo del Sprint y Sprint Backlog
Hora	Durante el sprint	Equipo Scrum (el desarrollo trabaja en el backlog)	Última compilación lista para entregar	Inspección del progreso y retroalimentación a la fuerza de trabajo y el liderazgo en el trabajo

Escalar Scrum con Nexus

Nexus es un marco para desarrollar y mantener iniciativas de entrega de productos a escala. Se basa en Scrum y lo extiende so lo cuando es necesario. Busca minimizar y administrar las dependencias entre varios Scrum Teams.

Nexus

Es un grupo de aproximadamente de tres a nueve Scrum Teams que trabajan en un mismo producto. Un Nexus tiene un único PO que administra un único Product Backlog.

Este marco de trabajo define responsabilidades, eventos y artefactos que unen y entrelazan el trabajo de los Scrum Teams en un Nexus

Busca preservar e mejorar la inteligencia y el empirismo fundamentales en Scrum, además permite al conjunto de Scrum Teams entregar más valor del que puede lograr un solo equipo

Para esto se debe solucionar los problemas que vienen de resolver desafíos que son comunes entre Scrum Teams:

- Dependencia de equipos
- Preservar la autogestión
- Transparencia del equipo
- Garantizar la responsabilidad

Se da por:

• Estructura compleja del producto

• La comunicación entre equipos no es buena

Nexus extiende Scrum de las siguientes maneras:

• Respaldo integración Team (NIT) se asegura de que el Nexus entregue un incremento integrado valido y útil al menos una vez cada sprint.

• Está formado por el PO quien se encarga del 90% de la administración del valor del producto del trabajo realizado e integrado por los Scrum Teams.

• Master encargado de asegurarse que el marco Nexus se entienda y promueva correctamente y/o miembros separados de los Scrum Teams que pueden ayudar a resolver los problemas que enfrenta el Nexus en cualquier momento (la responsabilidad más importante de estores para que el Nexus integrado Team que para con su Scrum Team).

• El NIT se encarga de sobrever las restricciones técnicas y/o no técnicas del equipo multifuncional que no permitan al nexus entregar su incremento integrado valido

Eventos

• Nexus añade y amplía los eventos definidos por Scrum. Para cada evento se incluye solo a los integrantes justos y necesarios para lograr el objetivo del mismo, si incluimos a gente de más los tiempos de trabajo se extenderán mas de lo necesario.

Consisten en:

- Sprint: igual que en Scrum, los scrum teams en un Nexus producen un único incremento integrado

- Revisión: igual que en Scrum, se divide en 2 partes: el sprint review y el incremento integrado review para identificar las dependencias y/o minimizarlas. Pasamos los items del PB a través de diferentes niveles de descomposición.

Luego definir que equipo se hará cargo de los Scrum Teams para identificar la dependencia entre los mismos pudiendo variar el tiempo y la frecuencia con tal de priorizar estos objetivos.

- Nexus Sprint Review: para planificar las actividades de todos los Scrum Teams dentro de un Nexus para todo el sprint.

- Nexus Daily Scrums: su objetivo es indentificar cualquier problema de integración e inspeccionar el avance hacia el Nexus Sprint Goal (estado actual del incremento integrado). Asisten los representantes adecuados de los Scrum Teams. Los ajustes al plan se realizan durante el desarrollo.

- Nexus Daily Scrums: igual que en Scrum, pero con otra metodología de trabajo.

El Nexus Daily Scrums se complementa con las Daily de cada Scrum Team.

- Nexus Sprint Review: al final de cada Sprint y para proporcionar retroalimentación sobre el Incremento Integrado "Hecho". Se presenta el avance sobre el incremento propuesto a las diferentes partes y se discute el avance hacia el Product Goal.

- Nexus Sprint Retrospective: planificar formas de aumentar la calidad y eficacia de todos en Nexus. Se tiene en cuenta como traerían los diferentes entre los Scrum Teams.

- Artefactos: el PB es el mismo que todos. Mientras mas se refina los items del Product Backlog, tanto se sabe de lo que se trabaja dentro de un Sprint. Nexus Backlog sirve para tener transparencia durante el sprint integrado.

- Sprint Backlog: hay uno único que incluye las tareas del Nexus y de cada Scrum Team que se necesitan para mejorar el producto. En esta escala las dependencias pueden detectarse y minimizarse.

- El compromiso del Product Backlog es el Product Goal que describe el estado futuro del producto y sirve como un objetivo a lo largo del desarrollo.

- Nexus Sprint Backlog: composición del Nexus Sprint Goal y los items del Product Backlog que se incluyen en los Sprint Backlog de cada Scrum Team.

- Serve para resaltar las dependencias y el flujo de trabajo durante el sprint integrado.

- Incremento integrado es la suma de todos los incrementos integrados completado por un nexus que persigue el Product Goal. Debe cumplir con el DOD.

- Se fija en cumplir el definition of Done que define el estado del trabajo integrado cuando cumple con la calidad y las métricas requeridas para el producto. Debe estar integrado, valioso y utilizable.

KANBAN EN EL DESARROLLO ÁGIL

Kanban

Es un framework que sirve para mejorar continua, es un gestión para implementar mejoras continuas.

en las áreas de las empresas que lo aplican. Se basa en los principios de la filosofía Lean.

Kanban asume equipos con experiencia y capacitados, no nos dice como hacer software de calidad.

Las pocas cosas que define Kanban hay que cumplirlas estrictamente.

Kanban → tarjeta de señal o indicación de algo. Se suele utilizar en

Kanban para mejorar la forma de producción.

Unidad 4: Aseguramiento de Calidad de Proceso y de Producto

Testing: una de las actividades que se realizan para el aseguramiento de la calidad del software



Calidad

Atributos y características de un producto o servicio que se relacionan con la habilidad de alcanzar las necesidades manifestadas o implícitas. Que tanto satisfacemos las necesidades del cliente.

La calidad tiene que ver con proyecto, producto, ...

Un software de calidad satisface:

- Expectativas del usuario
- Necesidades del usuario
- Necesidades de la gerencia
- Necesidades del equipo de desarrollo y mantenimiento
- Otros interesados

Las problemáticas que frecuentemente atentan contra la calidad son: los retrasos en el tiempo en su mayor porcentaje, excedidos en presupuestos, no se alcanzan los objetivos o son cancelados o abortados.

Principios

- La "calidad" no es ni invento ni se compra, debe estar embalada.
- El testing no significa calidad, visibleza que bien o mal está echo el producto
- Los errores de todos
- Las personas son las claves para lograrlo (que estén capacitadas)

Existen diferentes perspectivas para otorgar calidad:



Capaz la calidad para el cliente son las características pero para el desarrollador calidad va más por diseño, desarrollo, acoplamiento y cohesión. Al usuario le importa el nuevo lo importa que sea rápido, funcione y sea barato.

La visión transcendental: Es lo de diseño de producto. Lograr cosas grandes. Es un motor para avanzar.

Lo que buscamos al hacer software es llegar al equilibrio de las 3 perspectivas de calidad



Calidad en el software

Las personas necesitan una guía para poder hacer software.

No existen guías que nos indiquen tal como llevar adelante un proyecto, proceso o producto. Nos basamos en modelos para tomar de referencia que sirven para definir procesos en la organización.

Existen modelos que nos ayudan (dando un marco de referencia) a definir proyecto para mejorar los procesos internos. También un framework para mejorar.

Y existen modelos de evaluación que ven que tanto se adhiere el proceso con el modelo que se tomó como referencia.



Definición: el libro: procesos oficiales de software que se aplica de manera que crea un producto que proporciona valor medible a quienes lo producen y quienes lo utilizan.

Ventaja

Software "suficientemente bueno" → se entrega una primera versión del producto pero no con la mejor calidad posible porque nos lleva mucho tiempo de desarrollo, y llegar al mercado a tiempo es mas importante que una calidad excelente.

- No damos nuestra reputación
- No damos resultados
- Diminuimos realizarle mantenimiento extra que nos causa el producto de mala calidad

Desventaja

- Largo desarrollo de software → alto costo de desarrollo, alto costo de mantenimiento, alto costo de tiempo y dinero
- Costos de preventión: costo de administración de las actividades de control y aseguramiento de calidad de software, la planificación de las pruebas.
- Costos de evaluación: investigación de la condición del producto que pasa por primera vez por cada proceso
- Costos de falla: son los errores que aparecen antes y después de enviar el producto a los consumidores.

Mucha gente que satisface

Es importante trabajar con calidad porque nos ayuda a reducir el número de reacciones de las actividades que deben hacerse para desarrollar software. Disminuyendo los costos y mejorando el tiempo de llegada al mercado

Principios

En ágil se asume que todo producto debe ser de calidad y que al tomar los equipos estan capacitados, motivados, que son caracteristicas basicas para hacer producto de calidad.

Visión trascendental → la calidad atraviesa todo, está presente en todos lados

Visión del usuario → calidad en términos de metas específicas del usuario final

Vista de fabricante o manufactura → se define en términos de las especificaciones originales del producto

Vista del producto → calidad tiene que ver con las características inherentes de un producto

Vista Basada en valor → se mide de acuerdo con lo que un cliente está dispuesto a pagar por un producto.

Se espera que la intersección entre las 3 sea lo suficientemente grande para cubrir las 3.

Calidad programada es la calidad que se espera que un producto tenga.

Calidad necesaria son las características mínimas que un producto tiene que tener para satisfacer las necesidades de usuario.

Calidad realizada es lo que realmente hicimos por la calidad del producto.

El resultado en el proyecto es porque nosotros hacemos aseguramiento de calidad tanto de proceso como de producto lo hago siempre en el contexto de un proyecto.

El proceso se mejora para el proyecto en particular

El producto se realiza todas las técnicas en el contexto de un proyecto

Modelo para medir producto → La Sprint Review es la ceremonia que hace mejora en el producto de Scrum

El resultado en el proyecto es porque nosotros hacemos aseguramiento de calidad tanto de proceso como de producto lo hago siempre en el contexto de un proyecto.

El proceso se mejora para el proyecto en particular

El producto se realiza todas las técnicas en el contexto de un proyecto

Auditórium de proyecto → se hace para verificar que el proyecto está respetando el proceso que dijo iba a usar

Pedimos consultar a externos sobre las técnicas que utilizamos para llevar adelante el proyecto para construir el producto

Auditórium funcional (valida y física) (verificación), son las que se hacen a una versión específica de un producto de software señalada en una línea base

Proyectos en evolución y cambios continuos

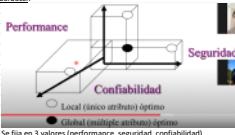
La gente que trabaja sobre definiciones. Si el proceso tiene calidad el proceso es respetado por la gente, entonces el producto va a ser de calidad*. Se puede adaptar los procesos a lo que uno necesita.

La gente que trabaja en empíricos (enfoques ágiles): "La calidad de producto depende totalmente del equipo y su capacidad"

Calidad de Producto

No se puede sistematizar, no existe plantilla que se pueda aplicar a todos los productos para evaluar su calidad.

Modelo de Barbacci:



Cuando creamos un producto es bueno basarnos en uno de los modelos existentes, sin embargo la calidad se termina midiendo por que tanto cumplimos con los requerimientos establecidos por el cliente

Si queremos hacer aseguramiento de calidad utilizamos revisiones técnicas (elemento fundamental), auditorías y el testing para a verificar la calidad.

Calidad de Proceso

No existe una receta común que sirva para todos los casos.

Debemos encontrar un proceso que sea útil para nuestra necesidad particular.

El único factor controlable a la hora de desarrollar software → el proceso

Cómo evoluciona la tecnología no lo podemos controlar, las características del producto no podemos controlarlo, a la gente tampoco podemos controlarla.



Si no hay un proceso definido, no hay un DOD sufriremos el síndrome de la incertidumbre, es la única manera de sentir que tenemos os el control!

Como lo definimos?

Es un conjunto de software y un conjunto de actividades, métodos, prácticas y transformaciones que la gente usa para desarrollar o mantener software y sus productos. No alinea con el libro con las indicaciones solamente, sino necesitamos de gente capacitada que lo siga.

Que es un proceso en PPQA → El proceso no es solamente las tareas escritas, se define al mismo como un triángulo de 3 cosas:

- Si necesitamos que tenga escrito los lineamientos de como vamos a trabajar
- Las personas con habilidades, entrenamiento y motivación
- Y el mejor soporte de herramientas automatizadas y equipamiento.

¿Quiénes deben participar?

• Quiénes deben hacer?

• Cuándo lo deben hacer?

• Cómo lo deben hacer?

¿Cómo es un proceso para Construir Software?



Ingeniería: parte de análisis, diseño y requerimientos.

A esto le debemos agregar disciplinas de gestión y control

Son disciplinas transversales que empiezan desde el momento cero del proyecto y avanza junto con el software

Nos da la visibilidad de como venimos trabajando y como viene cumpliendo con el producto

Planeación y seguimiento de proyectos

Administración de Configuraciones

Aseguramiento de la Calidad

Testing → proceso destructivo que busca descubrir defectos en el software. Un testing es exitoso cuando se logra encontrar defectos; sin embargo, no se dice que no tiene defectos, sino que no se los pudo encontrar. El software tiene defectos.

Testing es una actividad de control de calidad que llega tarde porque el código ya está escrito. No es lo mismo hacer Testing que Aseguramiento de calidad. Cuando se realiza este último se logra con un mejor producto al testing.

Entre el 30% y 50% del costo de un software es Testing.

El testing no introduce los defectos, sino que los visualiza, los saca a la luz.

Una vez que los defectos son visualizados los clasificamos:

- Sección donde le hace este defecto al software
 - Bloqueante
 - Crítico
 - Mayor
 - Menor
 - Cosmético
- Prioridad → definida por el cliente
 - Que tan importante es para el negocio la solución del defecto
 - Urgencia
 - Alta
 - Media
 - Baja

Niveles de Prueba

Pruebas de unidad → la hacen los desarrolladores, es el nivel que encuentra errores. Se

hacen en el ambiente de desarrollo.

Se realizan pruebas de integración → no hay acuerdo, en organizaciones lo hace la gente de desarrollo y terminan con un build que va a producción; también hay gente que hace prácticas de integración continua, obteniendo permanentemente un build que puede ser entrado a las pruebas de sistema.

Pruebas de sistema o de verificación → se lo podemos hacer a versiones o al sistema entero. Hacemos testing de los requerimientos de esa porción del producto. Un desarrollador no puede probar su propia sección, se debe separar la UAT en ágil para probar el sistema.

Tarea de determinar si el producto funciona en su globalidad, por eso el entorno en el que se ejecuta se debe asimilar lo más posible a producción.

Se ven requerimientos funcionales y no funcionales del sistema

Pruebas de aceptación de usuario → el usuario debe estar involucrado en las pruebas, él es

el que lo pidió y sabe lo que quiere

Ambientes para construcción del Software

Desarrollo → donde desarrolladores escriben el código

Pruebas → Pruebas de integración, entorno parecido a producción

Pre Producción → Pruebas de sistema y a veces las de aceptación

Producción → Software funcionando y en operación. Se puede usar para pruebas de

Se descubre en la misma etapa en la que estoy trabajando → Cuando se encuentra se encuentra en una etapa posterior a la que se lo construye

El testing no puede afirmar la ausencia de defectos.

Se sigue realizando testing hasta que llega a su límite, por ejemplo se puede realizar una proyección de defectos y cuando se llega a ese número dejamos de testear.

El testing no introduce los defectos, sino que los visualiza, los saca a la luz

Una vez que los defectos son visualizados los clasificamos:

- Sección donde le hace este defecto al software
 - Bloqueante
 - Crítico
 - Mayor
 - Menor
 - Cosmético
- Prioridad → definida por el cliente
 - Que tan importante es para el negocio la solución del defecto
 - Urgencia
 - Alta
 - Media
 - Baja

Niveles de Prueba

Pruebas de unidad → la hacen los desarrolladores, es el nivel que encuentra errores. Se

hacen en el ambiente de desarrollo.

Se realizan pruebas de integración → no hay acuerdo, en organizaciones lo hace la gente de desarrollo y terminan con un build que va a producción; también hay gente que hace prácticas de integración continua, obteniendo permanentemente un build que puede ser entrado a las pruebas de sistema.

Pruebas de sistema o de verificación → se lo podemos hacer a versiones o al sistema entero. Hacemos testing de los requerimientos de esa porción del producto. Un desarrollador no puede probar su propia sección, se debe separar la UAT en ágil para probar el sistema.

Tarea de determinar si el producto funciona en su globalidad, por eso el entorno en el que se ejecuta se debe asimilar lo más posible a producción.

Se ven requerimientos funcionales y no funcionales del sistema

Pruebas de aceptación de usuario → el usuario debe estar involucrado en las pruebas, él es

el que lo pidió y sabe lo que quiere

Ambientes para construcción del Software

Desarrollo → donde desarrolladores escriben el código

Pruebas → Pruebas de integración, entorno parecido a producción

Pre Producción → Pruebas de sistema y a veces las de aceptación

Producción → Software funcionando y en operación. Se puede usar para pruebas de

Se necesitan ambientes separados
sino es muy caro el error de desarrollo

Caso de Prueba

Es el artefacto principal de testing, es el set de condiciones o variables bajo las cuales el

tester determinará si el software está funcionando correctamente o no.

Una buena definición de casos de prueba nos ayuda a reproducir defectos

Se busca descubrir errores en forma completa con el mínimo esfuerzo y tiempo

La información que contiene la prueba la debemos derivar desde los requerimientos. Si

dominamos el código no debes si saltártalo al cliente o no

Por eso es de lo que se debe aplicar mucho esfuerzo en el testing.

Si el caso de prueba no está actualizado, no se puede ejecutar la prueba.

El éxito de un buen testing está conformado en que tan bien estan definidas las casas de pruebas

y las bases de datos

Querremos la menor cantidad de casos de prueba que sean lo mas eficiente posibles

Para ahorrar costos y tiempos,

Métodos de Testing

Caso Negro: Basados en especificaciones → Testing dinámico

Basados en la experiencia

Caso Blanco: Se basa en análisis y estructura del código → Testing estático

Ciclo de Prueba

«Ejecución de la totalidad de casos de prueba, lo ideal es tener 2 ciclos de prueba (ciclo 0 y 1,

en el 1 tener la salud del producto)

Regrésate

Verificación de la especificación de los casos de prueba que producen error, se fijan que el

código especificado sea el correcto para llegar al defecto señalado.

Todos los ciclos de prueba se van a ejecutar como si fueran un ciclo 0, es decir se realiza

nuevamente todos los casos de prueba en los siguientes ciclos controlando que el resultado

obtenido sea igual al que especificó.

La mejoría es la de regresión pero se suele utilizar el testing sin regresión por temas de

tiempo y recursos.

Proceso de Pruebas



Se prueba de forma inversa a lo que se desarrolla: se realiza prueba unidad (desde lo mas especificado)

Hacia lo general.

Se puede adicionar actividades de prueba cuando ya estoy planificando el proyecto.

Testing a lo largo de vida

- Verificación y validación

 Verificación → vemos si estamos construyendo el sistema correctamente (libre de defectos)

 Validación → Estamos construyendo el sistema correcto? Es decir nos fijamos si es lo que el cliente quiere

Ciclo de vida

Lograr involucrar las actividades de testing lo más temprano posible.

Una vez definido los requisitos ya puedo empezar a trabajar definiendo los casos de prueba

(el código correcto)

Cuanto más temprano mejor porque nos da visibilidad de como se va a probar y utilizar

los productos disminuyendo los costos de correcciones.

Mitos

No empieza cuando termina de escribir el código

No espiar que el software funcione

Testing no infecta calidad de producto o proceso

Tester no es el enemigo del dev

Auditorías de Software

Realiza controles apropiados al software y al proceso de desarrollo

Se realizan para asegurar calidad de Software

Asegura cumplimiento de estándares y proceso de sw y proc de desa

Asegura a que los defectos en el producto sean comunicado a la gerencia para su solución

Para qué auditamos?

Genera una opinión objetiva e independiente

- Permite identificar áreas de ineficiencia del cliente

- Permite identificar oportunidades de mejora

Es la evaluación independiente de los productos o procesos para asegurarnos que seguimos los lineamientos, estándares, especificaciones y procedimientos, basada en un criterio objetivo indiciendo documentación que

especifica:

1. Forma o contenido de los productos a ser desarrollados (qué producto)

2. Proceso o el cual los productos son desarrollador (cómo, decir procesos para desarrollar)

3. Cómo medir el cumplimiento de estándares o lineamientos (métricas)

Por qué realizamos auditoria?

• Evaluar cumplimiento del proceso de desarrollo

- Determinar implementación efectiva del proceso de desarrollo organizacional, proceso de desarrollo del proyecto y actividades de soporte

- Dar visibilidad a la gerencia sobre los procesos de trabajo

Tipos de auditorias

Auditoria de PROYECTO

• Valida el cumplimiento del proceso de desarrollo. Se llevan a cabo de acuerdo a lo establecido en el

PACS (Plan de Aseguramiento de Calidad de Software), donde se debería especificar la persona

responsable de realizar las auditorias y la documentación.

• Su objetivo es garantizar la consistencia del producto mientras evoluciona a lo largo del

proceso de desarrollo (interfaces y software como se especificó en la ERS, requerimientos funcionales de la ERS se validan en Plan de Verificación,...)

Auditoria de PROYECTO:

• Valida que el producto cumpla con sus requerimientos, es decir compara el software que se construyó

con los requerimientos especificados en la ERS. Su objetivo es garantizar que el código implementa sólo

y completamente los requerimientos y las capacidades funcionales descritos en la ERS.

Auditoria de configuración fija:

• Valida que el ítem de configuración tal como está construido cumpla con la documentación técnica que

lo describe.

• Se evalúa el código con la documentación de soporte, su propósito es asegurar que la documentación

que se entregará es consistente y describe correctamente el código desarrollado.

• El PACS debe indicar la persona responsable de realizar la auditoria física.

• El software se puede entregar solo cuando se hayan arreglado las desviaciones encontradas

Responsabilidades

Gerente de SOA: Prepara el plan de auditorias

Calcula su costo

Asigna recursos

Desarrollar las no-conformidades

Acerca la fecha de la auditoria

Comunicó el alcance de la auditoria

Recolecta y analiza evidencia objetiva que es relevante

Realiza auditoria

Prepara el informe de la auditoria

Realiza seguimiento de los planes de acción acordados con el auditado

Auditado

Acerca fecha de auditoria

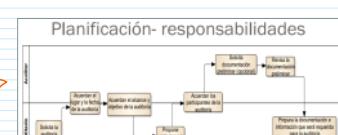
Participa de la auditoria

Proporciona el material de la auditoria

Comunica reporte de auditoria

Propone plan de acción para deficiencias citadas en el reporte

Comunica cumplimiento del plan de acción



Cuando queremos llevar adelante una auditoria los pasos a realizar son:

1) Preparación y planificación

