**Requerimientos Agiles – Clase 24/08/2023**

Hay un foco en el valor de negocio

Valor de negocio es lo primero que tenemos que tener en cuenta

Software como medio para entregar valor de negocio

Como son procesos empíricos, se parte de una visión del producto, no un producto completo

La visión responde al valor de negocio y determina la definición de la primera versión del producto

Voy a partir con un objetivo y un conjunto de alcances, y un artefacto que nos va a guiar

La visión permite generar el producto backlog(artefacto)

El producto backlog nunca va a estar completo

Lo mínimo que necesita el producto backlog para arrancar el producto es la cantidad suficiente de historias para ejecutar la primera iteración

Arrancar con la visión y tener un producto backlog inicializado apunta a cumplir con el principio del manifiesto que dice que voy a recibir cambios en etapas finales

No vamos a poder hacer una definición completa del producto de antemano, así que vamos a buscar un acuerdo para arrancar y a partir de ahí vamos a ir retroalimentando y ganando experiencia con procesos empíricos

Just in time, viene del lean, diferir decisiones hasta el último momento responsable

No especificar de antemano, para eliminar desperdicios

Solo detallar lo que hace falta

La complicación en ese caso es como me doy cuenta cuando es suficiente

El product backlog es una pila priorizada

¿Si no hacemos todo junto que hacemos primero?

La priorización es problema del product owner

El product owner viene del negocio

Lo podemos ayudar, pero la decisión de priorización es de el

La mejor comunicación es cara a cara

Los mejores reqs emergen de equipos autoorganizados

Las user stories no es artefacto nativo de scrum

Las us son un medio para diferir una conversación, para acordarnos de hablar con un usuario,

Cuando uno hace gestión con las us las usa como medio para saber quién se hizo cargo

La DoF se construcye con el INVEST

El sprint backlog es el tablerito de la gestión de la iteración (scrum)

Scrum arranca con spring planning y se decide que historias van a pasar del pb al sb

El sb tiene tres columnas: el to do, el doing y el done

Definition of done, cuando una historia esta en condiciones de ser mostrada al po

DoR condiciones de salir del producto backlog al sprint backlog

DoD, es también un checklist como el DoR, cosas que tiene que controlar

DoD, código testeado, subido al repo, etc

El DoD lo crea el equipo, se ponen de acuerdo

El tablero con el DoR y el DoD están visibles todo el tiempo

La user esta en una columna o en otra, no en mas de una al mismo tiempo

No hay un INVEST para el DoD, el código tiene que estar comentado, documentado, seguir estándar, subido en el repositorio, las pruebas unitarias

El código se debe subir al repositorio, para poderlos subir tiene que tener calidad para que no rompa el repositorio

Atención continua a la excelencia técnica, la calidad no es una variable de decisión

En agiles lo que queda para negociar es cuantos reqs entregas

El tiempo y la calidad no son negociables

Excelencia técnica tiene que ver con la calidad

El producto tiene valor si hace lo que tiene que tiene que hacer y lo hace bien

Calidad es la medida con que se satisfacen las expectativas y necesidades de los usuarios

Expectativas no se manifiestan, es lo obvio, hay que darse cuenta

El cliente asume que te tenes que dar cuenta que hay cosas que tienen que estar

Hay que intentar hacer que las expectativas sean explicitas

Si uso user muy grandes y se compromete todo el equipo a trabajar en esa user y no la terminan, no entregan nada

La user que no termino vuelve al producto backlog y de vuelta el product owner decide si sigue teniendo prioridad o no

Si el producto owner no la aprueba en el done vuelve al producto backlog con la prioridad que el decida

Una epica es una user muy grande que hay que descomponerla en user mas pequeñas

En el pb arriba tiene que haber user stories

Las spikes tienen un nivel de incertidumbre tal que la historia no se puede estimar. La E de INVEST es estimable

Hay tal incertidumbre que la historia no se puede estimar y se convierte en una spike

Hay dos tipos de spike: funcional que viene del negocio y del product owner y técnica

Spiker funcional tiene mas que ver con que va a hacer el producto, la técnica con como

Si hay que averiguar cosas antes es una spike

Siempre hay incertidumbre, cuando excede los limites aceptables la user se convierte en una spike

No poner en el mismo sprint backlog la spike con las user que tienen que ver con esa spike

Hasta que no termine la spike no se cumple el DoR porque no se cumple el estimable del INVEST

Una spike puede afectar a más de una user

El refinamiento del producto backlog es una reunión continua

El pb se define y redefine permanentemente

En cualquier momento el producto owner puede subir y bajar en el pb, en el sb (lo que se esta trabajando) no

Las estimaciones se basan en los requerimientos

Estimar tiene asociado un factor de probabilidad

Uno estima cuando hay incertidumbre

Uno necesita estimar cuando tiene que hacer una predicción

En el soft las estimaciones funcionan mal

En el soft el 40% de los proyectos son exitosos. Ese 40 termina desfasado en las estimaciones iniciales

Estimaciones agiles tiene sustentos teóricos como los reqs agiles

La primera característica de las estimaciones agiles es que son relativas

Las tradicionales son absolutas

Que una estimación sea relativa depende comparando

Las estimaciones se hacen por comparación en el enfoque agile

Poker planning o poker estimation

Relativa significa que estimamos por comparación

Hace falta algo contra que comparar

El foco de las estimaciones agiles es que se hace foco en la certeza, no en la precisión

La precisión es cara

Enfoques tradicionales hacen énfasis en la precisión. Difícil de cumplir

Para llegar a esa precisión hubo que hacer suposiciones

Las estimaciones agiles/relativas hacen foco en la certeza

Diferir las decisiones hasta el ultimo momento responsable

Al igual que con los reqs no hacer un esfuerzo en estimar todo el producto sino cuando hace falta

Se estima en dos momentos

Estimación de granularidad mas gruesa: las user en el producto backlog, como talles de remera: x, xl

Al momento de insertarlo se les hace una asignación inicial

Es recomendado empezar a estimar en el producto backlog

Todo se puede estimar, las épicas, los temas y las user. Es recomendado estimar todo

El segundo momento es durante la sprint planning

Se estiman las historias y como se van a llevar a la implementación

Estima el equipo, el que hace el trabajo

En la gestión tradicional estima el líder del proyecto

Gestión agiles, equipos asumen compromisos en base a cosas que ellos decidieron

Unidad de estimación de las user stories se llama story point

Valor cuantificado de una historia

Ese story point representa el tamaño de la us. Cuan grande o pequeña es una historia

El story point usa la serie de Fibonacci. El número siguiente se obtiene a partir de la suma de los dos anteriores

0, 1, 1, 2, 3, 5, 8, 13

La serie tiene un crecimiento exponencial, al igual que el software

No es la única serie que se puede usar, se puede usar cualquier otra serie exponencial

La recomendación es que estime el equipo todo junto.

Aprovecha una visión complementaria

Se estima en forma colectiva

El siguiente elemento es encontrar la user story canónica. Es el elemento que vamos a utilizar para comparar

Cada persona tiene sus cartas de estimación, tomamos la primer user que queremos estimar, esta el po presente, se estima para cumplir con el DoD.

Tengo que estimar para cumplir con el DoD, no solo el tiempo de programación

El DoD el po lo va a usar para aceptarte o no la user

Cada uno estima que la story tiene tal peso. Se muestra luego que están puestas todas las cartas. Para que la elección no este influida por los otros

Cuando las cartas están visibles se explica porque se eligió cada carta. Por lo general empieza el que puso el valor mas alto (por lo general el 8)

Cuando todos terminan se hace otra vuelta con la misma mecánica hasta converger en un acuerdo

Por lo general no se hacen mas de dos. Las estimaciones tienen que ser convenientes

En el momento que se hace la estimación inicial se hace en base a la canónica. Cuantas veces mas compleja que la canónica es esta que estoy comparando

La mayoría de los equipos eligen una canónica de 1, lo mas fácil, sencillo, certero

Otros equipos eligen de 2 o de 3 para tener mas margen abajo y arriba

La canónica es una user del proyecto

La experiencia no se puede extrapolar, característica de los proyectos agiles

Story point se puede descomponer en 3 dimensiones: complejidad, esfuerzo y duda (incertidumbre)

Esas 3 dimensiones puede tener una user story al asignar un story point

Complejidad es cuan dificultosa es la característica que quiero construir

Esfuerzo es trabajo, cuantas horas ideales necesitamos para crear esto

No es tiempo, tiempo es calendario

Esfuerzo es horas ideales, horas reales de trabajo

El esfuerzo se mide en horas ideales lineales, cuantas horas reales necesito para construir esto

Duda, nivel de desinformación relacionado a la historia

Cuando crece mucho se convierte en una spike

Story point es un número que homogeniza las tres dimensiones

Como cada dimensión individual es muy difícil de comparar, se llevan las dimensiones a un solo punto para comparar

**Gestión de Productos - Clase 31/08/2023**

Necesidad de salir al mercado con una versión del producto para evaluar si vale el esfuerzo hacer el producto entero

Surge con la corriente lean startup

Mvp producto mínimo viable

Parcial practico: identificar mvp, escribir una o dos user stories y estimarlas (con justificación)

Proceso nos dice como hacer las cosas, proyecto se formaliza con unidad de trabajo, como resultado producto

La mejor forma para gestionar el trabajo era el concepto de proyecto y se empezó a usar para hacer software

No confundir proyecto con producto

Roles específicos para productos y proyectos: Product manager y Project manager

Proyecto unidad de gestión y personas y recursos

Project manager administra recursos y personas para cumplir los objetivos del proyecto

Product manager es el Product Owner (a veces). El producto manager a veces no puede tener tanta dedicación para ser PO

Las empresas que su core de negocio no es hacer soft, el área de sistemas es de apoyo a las funciones básicas, se subestima el área y ni siquiera tienen proyectos para gestionar

Con la evolución del la tec se jerarquiza el área y tienen un rol mas relevante, y aparecen los proyectos

Aparece la necesidad de la terciarización y aparecen las consultoras

Las consultoras localizan gente en las oficinas de los clientes para que solucionen algún problema y ayuden en lo que haga falta

No tienen visión de producto sino de proyecto

Empresas tecnológicas (plataformas como uber o mercadopago) el negocio es soft

Hacer soft para ganar dinero, ayudar a la sociedad, cantidad de usuarios

Time to market tiempo que demoramos en sacar algo al mercado

Es acelerado porque sino la competencia llega primero

La mayoría de las características de soft no se usan

De ahí surge el mvp

Foco en las tareas y que estén presentes y sin errores

Con eso no alcanza, porque otro puede hacer lo mismo pero mejor

Eso es solo base

La funcionalidad cumple con la utilidad, no con la usabilidad

La base no es negociable, que el producto haga lo que tiene que hacer

Tenemos que garantizar que el producto es confiable, o dar la sensación de confiabilidad

Usabilidad es cuanto disfruto haciendo lo que tengo que hacer con el producto

Para los productos es difícil cruzar la línea entre lo usable y lo conveniente, porque se puso el foco en las tareas y no en la gente

Un producto de soft significativo le cambia la vida a la gente (como whatsapp, gmail)

Empezar de a poco

No siempre los productos son licenciados

Hay muchos productos exitosos sin licencias, gana con los datos y la información

EL mvp se crea para validar una hipótesis. La hipótesis es que vamos a invertir en un producto que la gente va a querer o va usar. Muchos productos crean una necesidad (ej el streaming)

La característica fundamental del mvp es la hipótesis y el mvp la valida. La hipótesis es si este producto va a conseguir clientes, retenerlos, descargas. Mas que el índice de desacrga se mide el índice de retención (si descargan un producto no lo borran), además de tenerla instalada la usen. Se usa para medir si el producto va a permanecer en el mercado

El uvp es la idea de producto completo

Uno lanza al mercado una partecita, que es el mvp. Se valida el producto con los usuarios

Se pueden hacer variaciones del mvp en base a lo que dicen los usuarios

Mvf, f de feature, una característica. A veces un mvp es una sola feature, en general son mas

Mmf, la m del medio es de marketing, comercializable, tiene que ver con las características mínimas del producto para que pueda salir a comercializar. Muchas veces se confunde el mvp con el mmf. Porque si no lo usas para validar una hipótesis no es un mvp. En el mvp se admite la posibilidad de que el producto se convierta en otro si la hipótesis no es validada.

En el mvp estamos creando el producto y validando si es el producto que el mercado necesita

Mmf es comercializable y apunta a lo mínimo para salir a producción con este producto. Se habla de vender. Mmp producto mínimo comercializable

En el mvp el producto se va para otro lado.

Se necesitan mmfs para salir al mercado

A veces el mvp es un mvf. Un mvp puede ser un video o una presentación de power point, no necesariamente un producto. El mvp no se cobra, se hacen productos para venderlos después. Se buscan inversores.

La hipótesis siempre es basada en el usuario, si tenes usuarios que usen el producto

En el mmf tenemos un cliente claro

El mundo es ambiguo, no funcionan el siempre y el nunca

Hemos ido y vuelto y cambiado de idea, hacemos la planificación de como hacemos para tener finalmente la versión en producción. El producto evoluciona siempre igual

Mvp es el conjunto de características que elegimos para armar una versión del producto que necesitamos salir a validar. La clave de éxito es que este clara la hipótesis. Tenes que saber que validar

Mvf versión mas chiquita del mvp, característica mínima viable. Un conjunto de mvfs arman un mvp. El mvf es una característica. Adentro de un mvp puede haber una o muchas mvfs

El Mrf (reléase, tiene que ver con puesta en producción), características mínimas del realease para ponerse en producción

Mmf, característica mínima comercializable

Cuando parto de un mvp parto de una validación de una hipótesis. Superada esa hipótesis y decididos que vamos, entonces ya cerro el momento de validación aparece la m, vamos a salir al mercado con esto. Mmp tiene mmfs

En un mmp se puede realizar un mvp para validar una mmf

El mmp es igual al mmr cuando es el producto que tenes en esa reléase

Esta gestión la realiza el producto manager. Si las decisiones las toma marketing no es lo mejor para la calidad del producto

Mvp valor suficiente para que lo instale, retenga a los usuarios y evolucione. Si el mvp es exitoso tiene que evolucionar

Un mvp (parte de validación) no es un mmp, hay que validar

Lean su primer principio se llama eliminar desperdicios

Todo lo que no genera valor es deperdicio. Esa es la motivación

La calidad del producto final esta determinada por la calidad del proceso

Si no genera valor es un desperdicio y se elimina

El foco esta en la creación de productos focalizados en la experiencia de los usuarios y que generen valor. El concepto de valor tiene que ser abarctativo. No solo para el cliente/usuario sino también para el negocio. No priorizar el valor de negocio sobre el de usuario. El de negocio es una consecuencia

No tiene sentido gestionar las organizaciones por la cantidad de horas trabajafas

Los valores del manifiesto dicen que hay que satisfacer al cliente con ,o que es de valor para el

El trabajo intelectual se mide con resultados

Lo que nosotros hacemos es soft, no valor. El soft genera valor

La creación de valor se debe enfocar al cliente al que quiero entregarle valor.

Un mvp puede ser producto o no dependiendo de las circunstancias (tiempo y dinero).

Cuando uno se tiene que animar a crear un producto y experimentar mejor le sale a la gente que tiene menos para perder. Dilema de la audacia 0. Trato no pierdo nada.

La audacia cero nos hace formular las hipótesis.

Hiptesis de valor si lo que ya tengo va a seguir estando e Hipótesis de crecimiento si voy a seguir agregando

Si el producto no tiene valor la tasa de retención tiende a 0.

Proceso de preparar un mvp. Encontrar un nicho de mercado, para quien o que porción de mercado es a la que quiero hablarle. Armamos un roadmap, plan de alto nivel que es un recurso muy importante cuando nosotros nos vamos a enmarcar un desarrollo de productos que no están 100% definidos, vamos a sacar un mvp para dentro de 3 meses y no están definidads todas las características del producto; ese roadmap se va a detallar en un plan cuando arranque el proyecto, hay distintos niveles de planificación cuando uno tiene mas nivel de información, el nivel mas abstracto es el roadmap.

**Componentes de un proyecto – Clase 07/09/2023**

El factor clave de éxito son las personas

Hay dos estilos de control de procesos que son los procesos definidos y los empíricos

Proceso definido: establecer de antemano todas las cosas que voy a necesitar hacer para lograr el objetivo del producto. Bien identificado lo que hay que hacer, el orden, los artefactos resultado de las tareas y los roles responsables de las tareas. Esa construcción viene definida por otro -> la organización define, pero no son quienes efectivamente van a hacer el trabajo después. Hacer ajustes al proceso es mas costoso porque los procesos se definen arriba organizacionalmente. Incorporar cambios al proceso en si mismo es mas difícil porque tiene que haber un acuerdo a nivel organizacional. Un proyecto solo no puede cambiar los acuerdos a nivel organización. Los procesos ayudan, no son malos, ayudan a insertarse en la organización. Cuando uno tiene incorporada la forma de hacer las cosas salen mas rápido, las personas son mas resistentes a los cambios, cambio en las formas requiere un esfuerzo adicional. Si no cuidas ciertos detalles de retroalimentación y cosas que pueden cambiar, los procesos se vuelven obsoletos. Los procesos definidos deberían aprender de los ciclos de retroalimentación de los agiles. El sustento de los procesos definidos es tener visibilidad, de donde estamos parados, que es lo que hicimos y cuanto falta para terminar. Es una necesidad natural, para saber que avance tenemos de la situación y si vamos a poder terminar. El líder de proyecto se siente mas cómodo si tiene visibilidad del proyrecto por el que tiene que dat la cara. Los procesos definidos tienen controlado que se hace paso a paso, y si saben que algo en otro proyecto trado tanto tiempo, este mas o menos en las mismas circunstancias nos va a llevar tanto.

El empírico dice acá lo que sirve es la experiencia, no la de antes, sino la de hoy, la de este equipo, este proyecto, de la gente que esta generando la experiencia y conocimiento, el cual me va a servir para hacer las correcciones. La única experiencia que me sirve es la de este equipo en este momento en particular. No puedo traer experiencia extrapolada de otro proyecto. Cambia el contexto, cambian las circunstancias, no es extrapolable la experiencia.

Si nosotros somos un equipo que hace tiempo trabajamos juntos adoptando ciertas practicas, no vamos a dejar de hacer las cosas que nos están saliendo bien, eso es repetibilidad. Si el equipo trabaja siempre de la misma manera, es definido.

La discusión pasa por, el nivel de detalle, quien lo decide, si adentro del proyecto lo puedo cambiar, puedo tomar como equipo decisiones de cambio y cuando las puedo tomar.

Hay ciertas mezclas que se pueden hacer y otras no

Que diferencias hay entre procesos y ciclos de vida

Ciclo de vida tiene el proyecto, y también el producto, son distintos.

El ciclo de vida del producto son todos los estados por los que pasa desde que nace por una idea hasta que se descontinua

El ciclo del producto dura mas que el ciclo del proyecto

El ciclo del proyecto son las fases/etapas, que son estados de evolución del proyecto que determinan que cosas se pueden hacer y en que orden, para entregar el resultado o servicio. Ciclo de vida del desarrollo del software. Cascada, incremental, iterativo incremental, espiral

El ciclo del proyecto es todas las etapsas del proyecto desde que empieza hasta que termina

Tenemos varios proyectos a lo largo del ciclo de vida del producto

La cantidad de proyectos, además ejecutados por equipos distintos que tuvo que hacer evolucionar el producto Word. El ciclo es mas grande a menos que el producto fracase, sino son prácticamente iguales.

Dentro del ciclo de vida del producto se ejecutan varios ciclos de vida de proyecto que dan como resultado una nueva versión del producto.

Tipos de ciclos de vida:

* Secuencial
* Iterativo
* Recursivo

Si el proceso es empírico solo se puede usar el iterativo. De los cuales hay distintos iterativo incremental (Scrum y PUD), que se llaman igual pero no funcionan igual. Secuencial el casacada, con distintas variantes. Recursivo espiral

Proceso define lo que hay que hacer. Ciclo de vida es mas abstracto, dice en que orden se van a ejecutar, cuanto de cada una, en que momento llevar a cabo cada tarea definida del proceso.

El proceso detallamos tal y tal actividad, el ciclo de vida le da información al proceso que es una cosa lineal, para que se pueda ejecutar en el contexto de un proyecto. El ciclo dice el cuando y cuanto. El proceso el que y como

Si hago el diseño en casacada, el ciclo me dice que voy a hacer cada fase el 100% y después no vuelvo.

Los procesos definidos dicen que podes usar cualquier ciclo de vida.

Quien va a liderar un proyecto tiene que saber como hacer su trabajo de acuerdo al ciclo de vida que elige.

El ciclo de vida en cascada es mas fácil de gestionar. Da ilusión de mayor visibilidad

Los ciclos de vida recursivos no hacen entregas parciales como hacen los iterativos, pero tampoco son que hasta el final no vas a tener nada como cascada. Están al medio. Hacen énfasis en la gestión de riesgos. En cada vuelta evaluar si conviene seguir con el proyecto o cambiar. Define, evalua riesgos, construye y entrega. El recursivo da vueltas sobre si mismo hasta que tiene la posibilidad de salir. Espiral toma todo el producto y lo refina en distintas vueltas pero el producto finalmente para operar y poner en producción esta al final. Se justifica con productos muy complejos, muy riesgos, que no tiene sentido liberar el producto por partes (ej poner un satélite en orbita). Hay ciertas cosas que hasta que el producto no este completo no se puede poner en producción. En general no se usa para desarrollo de soft, porque es complejo de gestionar, no tiene mucho sentido.

Los proceso definidos cualquier ciclo, los empíricos si o si iterativos.

Iterativo e incremental en el PUD, e iterativo incremental en Scrum, pero no es el mismo. Conceptualmente quieren decir lo mismo, que el producto lo entrego en distintas iteraciones, donde las primeras van a tener funcionalidad limitada y se va a ir incrementando. Cantidad de iteraciones mínimas 2, porque si tiene 1 es cascada.

El iter incre de PUD, es un ciclo de vida de alcance fijo, mientras que el de Scrum, lo que queda fijo es el tiempo (duración fija). La variable que indica el fin de la iteración. En el PUD me pongo de acuerdo que CU van a entrar en la iteración. Recomienda empezar por los CU que son mas significativos para la arquitectura. La iteración no termina hasta que no implemente todos los CU que me comprometi a implementar. Alcance fijo. No termina la iteración hasta que termino todos los CU, lo que es variable es el tiempo. El cliente no tiene visibilidad de cuando le vas a entregar. Esa falta de visibilidad y previsibilidad molesta al cliente.

En agiles la duración es fija. Esto se presenta tal día, no se negocia. Negocio con el alcance. Si no llegue a todo, entrego menos. Hay que esforzarse por entregar lo que uno se comprometió a entregar.

No metas una mini cascada en cada sprint

Cada persona se compromete con una user, la empieza y la termina y pasa a otra.

Que es un proyecto. Es una unidad de gestión. Tiene un resultado, empieza y termina. Unidad de gestión que nos permite tener un producto o servicio UNICO. El resultado que se obtiene es único. Básicamente el proyecto es un medio de organización. Nos permite administrar recursos, gestionar personas, para obtener un resultado que va a ser único. Lo primero que tengo que destacar es que el resultado es único. Hay varios proyectos para un producto pero se obtienen versiones distintas, se va corrigiendo la hipótesis, va cambiando el producto. Tiene fecha de inicio y de fin, bien identificadas. Cuando se termina el proyecto se reasignan los recursos

Elaboración gradual. Permite lograr el objetivo. El todo lo vamos a dividir en partes y vamos a ir avanzando de a poco. El proyecto se divide en tareas. Vamos a ir desarrollando el producto en partes. Eso se logra porque tenemos actividades relacionadas. Tareas relacionadas es otra característica

Un proyecto tiene que tener un objetivo que tiene que estar relacionado con el resultado. El objetivo describe el resultado que el proyecto quiere tener. Puede tener mas de un objetivo

Objetivo del proyecto tiene que ser claro y alcanzable. Puedo definir un objetivo claro que no sea alcanzable, seria una utopía. Describe el resultado que quiero obtener. Expresarlo en términos que todos entiendan lo mismo.

Lo siguiente que hay que definir para el proyecto es el alcance. Se define como: todo el trabajo y solo el trabajo que hace falta hacer para cumplir con el objetivo. No hagas de mas, no pierdas el objetivo. Y el todo hace referencia a que no te comas cosas (documentación, no mirar solo el tiempo de programación, el trabajo no esta terminado hasta que no esta terminado, Definition of Done)

El proyecto tiene su objetivo y alcance, y el producto también. Están relacionados pero no es lo mismo.

El objetivo del producto definie lo que el producto quiere hacer. El objetivo y alcance de un producto se define en la ers. El del proyecto en un documento llamado Plan de Proyecto (también llamado Plan de Desarrollo)

Cuando empieza a armar el Plan de Proyecto, lo primero que se pone es el objetivo y alcance del proyecto.

El objetivo y alcance de ambos están relacionados.

El alcance dice que cosas tengo que hacer para hacer soft. Me fijo eso en los procesos. Elicitar reqs, hacer entrevista, modelar dominio, etc. Las cosas que tengo que hacer para cumplir el objetvio del proyecto que es hacer un producto. Programar, diseñar.

El alcance (del proyecto) se expresa en términos generales para saber que es responsabilidad nuestra o no

Lo siguiente es definer proceso y ciclo de vida. Definir lo que hay dentro de un plan de Proyecto. Adaptación (el libro le dice instancia). Un proceso se instancia con un proyecto. Cuando temenos que definer el proceso estamos estableciendo un compromise de que del proceso teorico necesito para el Proyecto. Se adapta el proceso al proyecto. Texto y context. Tengo este product, con esta gente, este context. Que cosas del proceso necesito hacer en el Proyecto. Cada Proyecto tiene una adaptación diferente.

Lo siguiente que se hace son estimaciones. Hay que estimar. Que se estima. Costos, tiempo, tamaño. Estimaciones de software. No son las agiles, son otras. Se usan en los procesos definidos. En este context se estima en un orden, no da lo mismo empezar en cualquier lado. Lo primero que se estima es el tamaño. Es dimensionar el tamaño de lo que voy a construer. En el enfoque tradicional se estima en:

* CU
* Líneas de código sin comentarios (LoC). No sirve.

Después de estimar el tamaño se estima el esfuerzo. Horas persona lineales/ideales. Siempre se mide en horas.

Después hay que estimar el tiempo. El tamaño contesta el que, el esfuerzo el como (porque hay que determinar que hacer), el tiempo el cuando (se mide en calendario), después sigue el cuanto con el costo.

Como transformar esfuerzo en tiempo. El esfuerzo es una bolsa de horas lineales. Transformar eso en calendario. Tenemos que ver la gente que tenemos disponible, luego cuantos días a la semana vamos a trabajar y cuantas horas tenes disponibles para trabajar. Índice de solapamiento. Que cosas podemos hacer en paralelo. Necesito hacer solapamiento. No hacer muchas cosas al mismo tiempo, es ineficiente.

Además hay un conjunto de recursos llamados recursos críticos que se recomienda estimar para que no generen desfasajes en el calendario del proyecto. Un recurso critico seria

También se estiman los riesgos. Algo que puede llegar a pasar. La probabilidad de ocurrencia de algo que pueda impactar negativamente en el proyecto que me afecte a mi en mi proyecto. La palabra clave es probabilidad. Puede pasar o no. Tiene un impacto, lo que voy a tener que pagar en mi proyecto si eso ocurre. El riesgo se mide por dos dimensiones: probabilidad e impacto. Cuando necesito comparar iun riesgo con otro multiplico el riesgo por el impacto, y obtengo lo que se llama exposición. Permite comparar un riesgo con otro. Cuando un riesgo tiene probabilidad 1 es una realidad, no un riesgo, y es un problema. No hay que tratar los riesgos como problemas.

La estrategia de reacción a los riesgos es reaccionar a los riesgos cuando son problemas. Eso es gestión reactivas. Si pienso que hago si esto se presenta, es una gestión proactiva, de anticipación. Si pasa esto voy a hacer esto.

Los riesgos tienen en común con las estimaciones la probabilidad. Las estimaciones en si mismas son un riesgo, porque las podes errar.

Faltan las métricas y el seguimiento del proyecto.

Con esto conformamos el plan del proyecto

Las métricas es la declaración de que métricas voy a tomar para tener visibilidad del proyecto

Y el seguimiento es que controles, reuniones y de mas voy a hacer para tener seguimiento del proyecto

**Gestión de la Configuración – Clase 14/09/2023**

Parcial: unidad 1 completa , unidad 3 solo gestión de configuración, unidad 2 gestión ágil de reqs, mvp y estimaciones

Unidad 1 completa

Unidad 2 la mitad, no va scrum. Lo que vimos en clase. Gestión ágil de reqs, user story, manifiesto ágil, mvp, estimaciones

Unidad 3 gestión de configuración de software

Que es, para que sirve, después lo demás

Una de las cosas de planificar es el cronograma. Para armar el cronograma hay que fijarse en las tareas del proceso que se van a adaptar al proyecto. Se adpata en función del equipo, del cliente, del contrato. Eso determina que cosas del proceso hay que hacer

La disciplina de gestión de configuración de soft, de todas las disciplinas asosiadas a la ing de soft, es la disicplia mas concreta que vamos a tener.

El software no solo es código. Hay muchas cosas que conforman el producto que no hay que dejar afuera de la gestión de configuración. Los entregables no siempre son código pero son importantes también. Software == conocimiento, información, que se puede presentar a distintos niveles de abstracción. Una user story es soft, una bd, el diseño de interfaz de usuario, plan de proyecto, todo eso es soft. El proyecto es una sumatoria de tareas, y c/u tiene que tener una salida, un resultado. El resultado de la ejecución de cualquier tarea del proyecto es software, son los artefactos del PUD. A lo largo del proceso de creación de soft voy a generar artefactos asosiados al proyecto, y otros al producto. Todo es importante porque va reflejando las decisiones que vamos tomando y el conocimiento que vamos adquiriendo a lo largo del desarrollo del producto. Automatizar las herramientas para no hacer cosas muy esforzadas a mano. Una de esas cosas que es importante automatizar son las herramientas de gestión de configuración de software.

El soft es fácil de modificar. Por eso te piden muchos cambios. Por eso es importante llevar control de versiones.

La gestión de configuración de soft es una actividad transversal. La ing de soft como disciplina es un concentrador de muchas disciplinas. Tiene que ver con las 3 p. Dentro de la ing de soft hay distintos tipos de disciplinas agrupadas por la función.

1. Disciplinas técnicas: reqs, análisis, diseño, programación, prueba y despliegue.

Se ocupan del producto. No estamos probando un proyecto, estamos probando un producto. Cuando se hace soft se hace un producto.

1. Disciplinas de gestión: La planificación y el seguimiento y supervisión. Estamos hablando del proyecto
2. Disciplinas de soporte: SCM (gestión de configuración de soft), aseguramiento de calidad de proceso y de producto (PPQA), y métricas. También se las llama disciplinas protectoras. Intentan atender y prevenir para que la calidad sea la comprometida con el cliente. Un producto tiene calidad cuando cumple las expectativas. Hay expectativas no solo del funcionamiento, sino también del tiempo en que se va a entregar. Las de soporte ayudan a cumplir la calidad, pero tienen que mirar todas las otras disciplinas.

Gestión de configuración se ocupa de mantener la integridad del producto de soft a lo largo de su vida. La vida del producto, que vive mas que el proyecto. Esto es transversal, hay que hacer SCM desde el momento 0. Cada pieza de código que se crea se llama item de configuración. Puede se cualquier cosa que se pueda meter en el disco de una computadora. Es software.

En el proyecto ejecutas la gestión de conf, se hacen mientras estas con un proyecto activo, pero para el producto a lo largo de su ciclo de vida. Si no tenes un proyecto activo no estas creando ítems nuevos, no hay ningún factor que te haga modificar la gestión de configuración.

La SCM es la estructura mínima que hace falta para generar un producto que tenga calidad y satisfaga al cliente. El testing lo que hace es controlar la calidad, llega tarde, cuando el código ya esta hecho. El testing controla la calidad y hace evidente los defectos que incorporamos en el soft. La SCM trabaja por prevención. Es poder entregar un producto a testing que tenga menos defectos. Igual el testing no se puede evitar. El propósito del testing es la detección de errores cuya prescencia se asume.

La SCM es una disciplina de soporte o protectora cuyo propósito es establecer y mantener la integridad del software a lo largo de toda su vida. Identificar y documentar características funcionales. Controlar los cambios de esas características. Registrar y reportar los cambios de implementación.

Un producto de soft tiene integridad cuando deja contento al usuario, cuando se lo puede rastrear (rastreo de reqs, mantener vinculo con los reqs en que se transformaron a lo largo del ciclo de vida, la rastreabilidad es una decisión, la SCM es una cultura, si hago SCM tengo que definir como mantener la trazabilidad), satisface los reqs no funcionales (la performance sigue siendo importante, pero hay otros importantes también) y cumple los costos tiempos plazos de entrega.

ítem de configuración es cualquier cosa que se pueda guardar en un disco. La cuestión es que el software es volátil. Es importante incorporar un aspecto: su versión. Cada item con su versión en un momento de tiempo. Porque cambia la versión. La SCM busca identificar cada ítem unívocamente en un momento de tiempo, eso lo hace a través de su versión. Un archivo solo con su versión es un ítem de configuración en un momento de tiempo.

Versión tiene muchas variantes respecto a la herramienta que se usa. Pueden ser números correlativos independientemente del ítem. No es un número por ítem. Otras herramientas asignan número por ítem. Se puede configurar que digito es el que va cambiando, dependiendo si el cambio es mayor o menor. Un ítem puede tener evolución lineal

Mas de una persona puede necesitar trabajar en un ítem en un momento de tiempo.

Cuando yo tomo todos los ítems del producto con su versión en un momento de tiempo tengo la configuración de software. La configuración de software es la sumatoria de todos los ítems de configuración con su versión en un momento de tiempo. La SCM administra la configuración. El elemento fundamental que usa para hacer su trabajo es el repositorio

El repo es un contenedor, el espacio que reservaste para contener todos los ítems del producto, la configuración. Es deseable que tenga una estructura.

Cada ítem de configuración tiene que ser único. Las versiones son parte del ítem, pero no tienen que ser parte del nombre.

Los repos son centralizados, y uno baja una copia local en función de los permisos.

Dependiendo de la herramienta es el nombre de la función. Commit es el proceso de subir. Sacar se llama check out o uptdate. Pero eso depende de la herramienta.

De acuerdo a las herramientas puede haber repos centralizados o distribuidos.

Línea base. Es un ítem o un conjunto de ítems con su versión que ha sido aprobado, esta estable, y puede ser usado como referencia para seguir avanzando en la versión del producto. Se marcan ítems como línea base, cada ítem de la línea base se etiquetan como línea base. Cuando tiras de la etiqueta trae todos los ítems de la línea base. Los ítems pasan por la revisión técnica para estar en la línea base. La línea base no se puede cambiar sino a través de un procedimiento establecido. Los que no forman parte de la línea base se cambian fácilmente. Una línea base para cambiarla requiere un procedimiento formal. Podemos tener un documento o muchos. Hay que ponerse de acuerdo con la línea base. Es raro que la línea base tenga un solo ítem. Las versiones de una línea base tienen que ser estables. La definición de línea base dice que fueron probados y tienen estabilidad, si los consulto esta bien lo que voy a consultar. Buscan mantener informada a la gente que cambio. Hay líneas base de especificación que tienen documentación y no código, y las operacionales que si tienen código. Es una etiqueta asociada a c/u de los ítems que queres que formen parte de la línea base, no es un ítem en si mismo, tiras de la etiqueta y tenes toda la línea base.

Ramas. Mas de una persona trabaje sobre un ítem de configuración. Esta el main con su línea base, y por lo general de ahí se abre una rama para cada equipo que esta trabajando.

SCM tiene 4 actividades principales.

1. Identificación de ítems, garantizar identificación univoca de cada ítem. Tiene que ver con asignar un nombre único a cada ítem. Los ítems que tienen que ver con el producto duran lo que dura el producto, los del proyecto lo que dura el proyecto, de una iteración todavía menos. La forma de identificar el ítem varia de acuerdo a eso.

Para incorporar cambios a una línea base hay un comité de control de cambios. El comité lo acepta o no. Lo forman el PO, el arquitecto, el PM, gente de testing, el cliente, desarrolladores. Cualquiera que se pueda ver afectado por ese cambio que estamos afectando puede formar parte del comité. Lo importante es que el comité lo forma el equipo, no un auditor externo.

Auditorias de config de soft se hacen de dos tipos. Una auditoria de configuración física, y otra de configuración funcional. Una auditoria es una revisión objetiva e independiente sobre un proceso o un producto. Quien asuma el rol de auditor no puede ser parte del equipo, ni dependiente de quien es el gerente, suelen ser externos o de otra área de negocio. Ninguna de las dos se puede auditar si no tengo un plan. Una auditoria es un proceso de control. Controlo lo que definí en el plan. Se audita una línea base. Una auditoria se hace en un momento de tiempo. La física se hace primero. Si esa sale bien se hace la funcional. La funcional ve consistencia con los reqs, si lo que estas creando se cumple. En este momento ayuda la rastreabilidad. No se audita todo el producto, una parte. Por lo menos una auditoria de cada tipo antes de liberar una reléase al cliente. La funcional apunta a la consistencia de que la versión del producto que estas auditando, satisfaga los requerimientos. La auditoria física apunta a la verificación

La ultima actividad es informes de estado. Tengo que poder sacar info del repo.

Las auditorias son mas resistidas por los equipos agiles.