ISW 17/08/2023

Procesos definidos: RUP, PUD.

Procesos empíricos: Ágil (filosofía, pensamiento. NO metodología ni proceso), Lean.

Definidos: expresión de completitud, que diga claramente quien lo va a hacer (roles), que entradas (artefactos), que voy a generar, qué métricas voy a tomar, que herramientas voy a utilizar. Está definido de antemano y por gente diferente de la que va a hacer el trabajo.

Empíricos: Basados en experiencia, conseguida a través de la retroalimentación del ciclo. Ciclos de desarrollo cortos, iterativos del tipo que sea, si o si (para obtener esa retroalimentación). De esta forma los ciclos cortos aportan experiencia.

3 pilares:

1 – Inspección: (cicla con adaptación obteniendo retroalimentación.)

2 – Adaptación

3 – Transparencia: Todo visible para todos, sin ocultar, blanqueando las tareas realizadas, la situación de avance del proyecto tiene que ser visible para todos en todo momento.

Los frameworks ágiles o lean te dan pautas y lineamientos de como construir y gestionar los productos. NO arman un proceso completo, entonces SCRUM no es una metodología como tal.

Cada proyecto tiene su particularidad y se debe aprender de el, para ello necesitamos esos ciclos de vida cortos.

Manifiesto ágil: 12 principios y 4 valores.

De los 4 valores, valoramos fuertemente lo que esta arriba sobre lo que esta abajo, sin embargo, lo que esta abajo tiene su importancia. Por ejemplo: Valoramos que el software esté funcionando (lo que valora el cliente) sobre la documentación extensiva, PERO se debe documentar.

[Libro USER STORIES, Myke Cohn?]

Product Owner: es de negocio, tiene que tomar decisiones, conoce en detalle el negocio, representa al cliente.

User Stories

Requerimientos en Agile

Las user stories ven un poco más arriba que los casos de uso, un nivel de abstracción mas alto a nivel usuario y negocio, entonces la descripción de la historia tiene que ser corta y apuntar a describir una necesidad que tiene el usuario, con terminología del usuario, escrita por el (en medida de lo posible).

Product Backlog: posee una cola priorizada, donde lo mas importante se ubica arriba, el dueño de esa cola es el PO y puede tomar las decisiones que sean necesarias.

Just In Time: Analice cuando lo necesite, no antes. Describimos y analizamos requerimientos conforme nos hagan falta, al comienzo nos concentramos en los pocos requerimientos que le debo entregar al cliente en esta iteración. Dedicar tiempo a cosas que no son urgentes puede derivar en desperdicio si cambian los reqs.

Comunicación: El cara a cara es la mejor forma de comunicación. Permite que fluya información vocal, subvocal, gestual con realimentación rápida. El contenido de la comunicación es el 17% del resultado, el resto es gestual, actitudinal, etc.

Dominio del problema – Dominio de la solución? Pegar foto filmina.

Triangulo de fijos y variables.

Alcance: reqs

Recursos: costo

Tiempo: tiempo xd

Los tradicionales fijan los requerimientos, a partir de ahí, determinan los recursos que van a necesitar (80% 10% 10%) y lo mismo el tiempo.

Ágil: deja libre el alcance, pero en cada iteración, deja fijo los recursos (el equipo que va a trabajar) y el tiempo, teniendo iteraciones de duración fija definidas por el equipo. El equipo y el tiempo no deben cambiar. Se van cumpliendo las prioridades hasta que se termina el tiempo y se entrega ese producto en esa iteración.

[Leer articulo obligatorio, parcial toman preguntas de ahí] Fred Brooks - “No Silver Bullet - Essence and Accidents of Software Engineering”.

User Stories:

Partes: CCC - Conversación, Tarjeta, Confirmación

La conversación es la parte más importante.

En el frente la tarjeta con who what why, en el dorso la confirmación (teórico, restricciones?)

Diagrama

Descripción generada automáticamente

La mas importante es el valor de negocio, reflejada en el why. Ojo con nombre del rol, no siempre poner como usuario, poner un quien que diga algo.

Se usa un nivel de abstracción que nunca se dice como, se usa lenguaje natural de negocio. Ej:

Interfaz de usuario gráfica, Texto

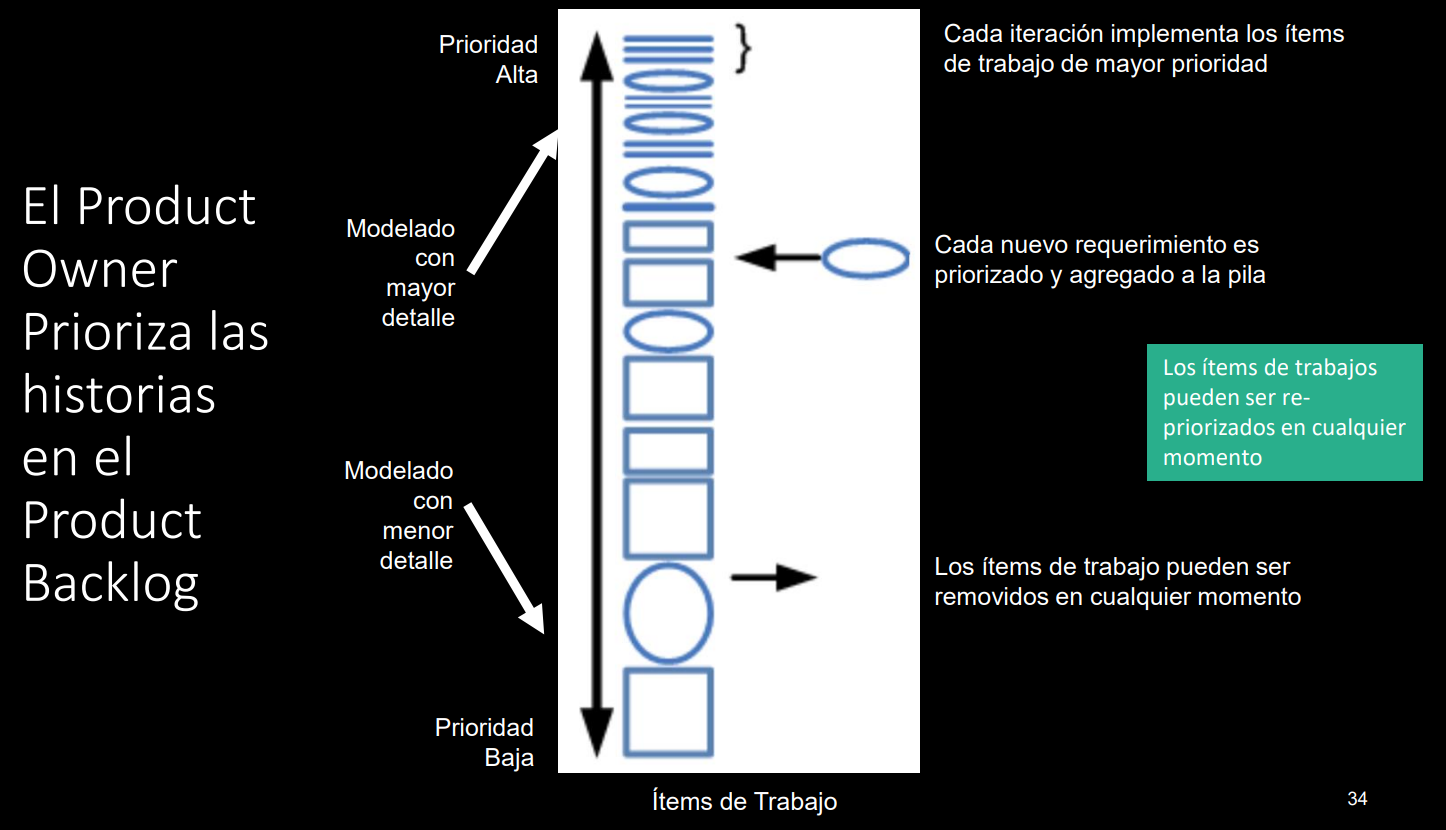
Descripción generada automáticamente

La user story es producto, NO es proyecto.

En el product backlog se acomodan las stories, ¿cómo? Respetando prioridad, ¿quién? El Product owner.

Las User Stories son multipropósito, son:

* Una necesidad del usuario
* Una descripción del producto
* Un ítem de planificación
* Token para una conversación
* Mecanismo para diferir una conversación



Los grandes van abajo porque aparte de la prioridad, todavía no son user stories, necesitan ser trabajadas y convertidas en varias user stories (líneas chicas) para ser implementadas en una iteración.

Porciones verticales.

Modelado de roles

Colgaste, preguntar.

Criterios de aceptación de user stories

Formalización de las cosas que el product owner nos va a pedir que ¿???.

Requerimientos no funcionales

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

De esto se derivan las pruebas de aceptación, con criterios para aceptarla o no (pasa o falla).

Detalles: expresadas en la documentación, no están en las cards, pero hacen falta para entender como hacer la user story.

Pruebas de Aceptación de Historias de Usuario

Expresan detalles resultantes de la conversación, complementan la User Story.

Proceso de dos pasos: 1. Identificarlas al dorso de la US. 2. Diseñar las pruebas completas

Para escribir historias bien: hay que preguntarles a los profes pq en el parcial son los PO, y de ahí sacamos los detalles y pruebas de usuario.

Imagen que contiene Diagrama

Descripción generada automáticamente

Las pruebas de aceptación son un acuerdo, ¿el usuario acepta la user story? Entonces sí.

Formas de controlar la calidad.

Definición de listo – Definition of ready

Se aplica a la user. Definition of Done.

Tiene mínimamente un INVEST model, de mínima, para cumplir con el ready, una user debe tener INVEST.

INVEST Model

* Independent – calendarizables e implementables en cualquier orden.
* Negotiable – user escrita en términos de “qué” no el “cómo hacerlo”.
* Valuable – debe tener valor para el cliente. ¿Por qué? o ¿para qué?
* Estimatable – para ayudar al cliente a armar un ranking basado en costos, cuando no se puede estimar sus valores, NO es una user. Debe poder estimarse valor numérico.
* Small – deben ser “consumibles” en una iteración.
* Testeable – demostrar que la user fue implementada en los términos que el PO necesita.

Otras:  
Cada user debe tener su prototipo.  
SI no hay asociadas reglas de negocio, especificarlas.

Niveles de abstracción: Stories, temas y épicas.

Story: …

épica: una story grande. Que deben desglosarse en US mas chicas.

Tema: user relacionadas?.

SPIKE: lo que la caracteriza es un nivel de incertidumbre tal, que esa historia no se puede estimar, por ejemplo, por no conocer la tecnología.   
Tiene dos tipos, uno tiene que ver con tecnología (cómo lo va a hacer) y la otra es funcional que son indefiniciones que vienen por parte del negocio (qué va a hacer), del PO. Hasta que esa spike no se trate no puede pasar el DoR, por la E de INVEST.

ISW T 24/08/2023

¿Qué son los requerimientos ágiles?

Requerimientos ágiles

Just in Time: diferir decisiones hasta el ultimo momento responsable. (no especificar de antemano si después puede no necesitarse o cambiarse, generando desperdicios). Solo detallar lo que hace falta.

Valor de Negocio -> Visión de producto -> Product Backlog

Foco en el **valor de negocio**, se parte de la **visión del producto** que responde al valor de negocio, y esa visión determina la primera versión del producto que voy a tener, es decir, voy a partir con un objetivo, conjunto de alcances y con la conformación inicial del artefacto que nos va a guiar en la construcción del producto que genera el valor de negocio.

El **product backlog** (pila priorizada) va a requerir una cantidad de user stories para poder construir el producto.

No se puede hacer una definición completa del producto de antemano, y tampoco se busca.

Los mejores? requerimientos emergen de los grupos autoorganizados.

Gestión binaria: 0 y 1, o no está hecho o está hecho. Los requerimientos, en una iteración, o están hechos o no están hechos, no hay porcentajes de avance. (en la gestión tradicional si hay, pero acá NO).

Tema futuro:   
Definition of Done (DoD): cuando está listo para mostrar al PO, lo crea el equipo, cada equipo puede tener el suyo.

Del **Product Backlog (PB)**, con **Definition of Ready** (que utiliza de mínima **INVEST**) pasan las **US** a un sprint backlog de 3 columnas, **To Do**, **Doing** y **Done**. De la **To Do** pasan a **Doing** cuando se están desarrollando, y cuando se terminan, con todas las pruebas, comentarios y demás, mediante una **Definition of Done** (DoD) pasan al **Done**. Las US son chicas porque si fueran grandes, pueden no terminarse y vuelven al PB.

Estimaciones ágiles

- Son **relativas,** estimamos por comparación. Es mas fácil comparar que decir cuanto X tiene algo. Por ejemplo, comprar 2 edificios es mas fácil que decir cuantos metros tiene de alto uno de ellos.

- Se hace **foco en la certeza** no en la precisión, (enfoques tradicionales si enfocan en precisión y fallan). Es preferible decir en tantos meses tengo una iteración, que decir tal día voy a tener el producto.

**- Diferir las decisiones** hasta el último momento, evitar hacer un esfuerzo en estimar todo el producto, se debe ir estimando conforme se necesite.

- Todo se puede estimar 👍

Sprint planning: reunión del equipo donde estiman la cantidad de historias que consideran necesarias para entrar al sprint, utilizando el PB. Se usa **Poker Planning**.

**- Estima el que hace el trabajo**, equipos autoorganizados.

Poker Planning

Unidad de estimación de user stories: **Story Point** (punto de historia), representa el tamaño de la historia de usuario.

Utiliza la serie de Fibonacci. **0, 1, 1, 2, 3, 5, 8, 13**, … (Usar más de 8/13 quedan muy grandes).

Se usa esta serie de crecimiento exponencial porque se asimila al crecimiento exponencial que tiene el software, no así una lineal.

Estima el equipo de forma colaborativa.

**US Canónica**. La misma es el elemento que vamos a utilizar para comparar, se elige una historia que va a ser la canónica y contra esa se comparan las demás.

Como se hace: cada persona tiene sus cartas de estimación, se toma la primer US a estimar, con el PO presente para preguntarle, se estima teniendo en cuenta el DoD ya que es lo que el PO va a usar para aceptar o no la US.

Luego de hacer las preguntas, cada uno elige individualmente su “apuesta” de estimación, con una carta boca abajo. Cuando todos eligen se muestra y ahí cada uno explica su decisión, normalmente empieza el que eligió más alto.

Cuando todos terminan de opinar, se hace otra ronda individual de “apuesta”, se muestra y deberían converger, se pueden hacer más pero se pierde el tiempo. Se puede elegir la que más se repita o promediar, para no perder el tiempo. La canónica juega pensando cada vez que se estima individualmente, la US analizada, cuantas veces mas que la canónica.

[Tamaño de la canónica, equipos inexpertos eligen una canónica de 1 pero puede traer complicaciones, se suele elegir canónica de 2 o 3 (Meles le gusta 3) ya que tiene valores abajo y arriba para comparar.]

**3 dimensiones** que puede tener una US al momento de asignarle un story point

* Complejidad: cuan dificultosa es la característica a implementar
* Esfuerzo: cuantas horas ideales (sin distracciones) de trabajo se necesitan para implementar, no es tiempo calendario.
* Duda (Incertidumbre): Que nivel de desinformación tengo relacionado a esta historia, tipos: **técnica** o **de** **negocio**.

ISW 31/8/2023 TEORICO

MVP: Producto mínimo viable – Minimal Viable Product.

**Proceso -> Proyecto -> Producto**

Proyect Manager: dueño del proyecto, se encarga de hacer las actividades para gestionar los proyectos, administra los recursos y gestiona las personas, para poder cumplir con los compromisos que asumió con el proyecto.

Product Manager: dueño del producto, dentro de los equipos ágiles es el product owner, a veces coincide el rol, a veces son dos distintos.

¿Por qué creamos productos de sw?

* Para satisfacer a los clientes
* Para tener muchos usuarios logueados
* Para obtener mucho dinero
* Realizar una gran visión, cambiar el mundo

**¿Qué características realmente utilizamos de un producto de sw?**

Imagen que contiene Diagrama

Descripción generada automáticamente

**Evolución de los productos de sw**

Diagrama

Descripción generada automáticamente

Focalizar en experiencias: Focalizar en quien va a usar el producto, hoy en día aun les cuesta a las empresas darle atención a ese requerimiento.

Focalizar en tareas: que debe hacer el producto, que estén presentes esas funciones y que no tenga errores.

Que sea **Funcional** es base, relacionado con utilidad, no es algo negociable, el producto debe hacer lo que tiene que hacer. Luego tenemos que garantizar que el producto es **confiable**, por lo menos darle la sensación al usuario de que lo es jaja, por último, que sea **usable** es cuando disfruto haciendo lo que tengo que hacer usando el software, ejemplo de Netflix y claro/Flow.

Es difícil evolucionar debido a que siempre se focaliza en tareas y no en la gente y su experiencia de usuario.

Un producto **significativo** le cambia la vida a la gente. Ej: Whatsapp, mercadopago.

**MVP, MVF, y MMF**  
Dinosaurio Lean/Agile

Un producto nuevo tiene una hipótesis de valor único: el producto/servicio será único.

Imagen que contiene Icono

Descripción generada automáticamente

Empezamos a crear el producto que queremos, empieza la motivación del producto mínimo viable.

El **MVP** se crea para validar una **hipótesis**, y la hipótesis fundamental es que vamos a invertir en un producto que la gente va a querer y necesitar, ¿Vamos a tener clientes para esto? ¿Esto realmente va a satisfacer una necesidad? Muchos productos crean una necesidad, por ejemplo, streaming (Netflix). Producto que sea único y tenga valor.

Índice de retención - Índice de uso

El dibujo del dinosaurio dice que se puede lanzar al mercado una parte del producto final, para ir viendo que pasa, validándolo con los usuarios para que ellos planteen variantes o disconformidades. Luego debemos analizar y tomar una decisión, no siempre se hace lo que quiere el cliente, depende lo que queremos transmitir con el producto. ¿A que perfil de usuario queremos dejar contentos? No se puede a todos.

MVF: Minimal Viable **Feature**: UNA característica, a veces un MVP es un solo feature, para ver si se prende al mercado y se puede salir con eso.

MMF: Minimal **Marketable** Feature: características mínimas que debe tener un producto para salir a comercializarlo (ya tenemos el cliente), o los requisitos mínimos de las stores que debe cumplir. Áreas de marketing. Ej: en whatsapp, que se puedan mandar mensajes y no necesariamente videos.

Se pueden hacer varios MVP, y con la hipótesis podemos validar si es el producto que se estaba esperando.

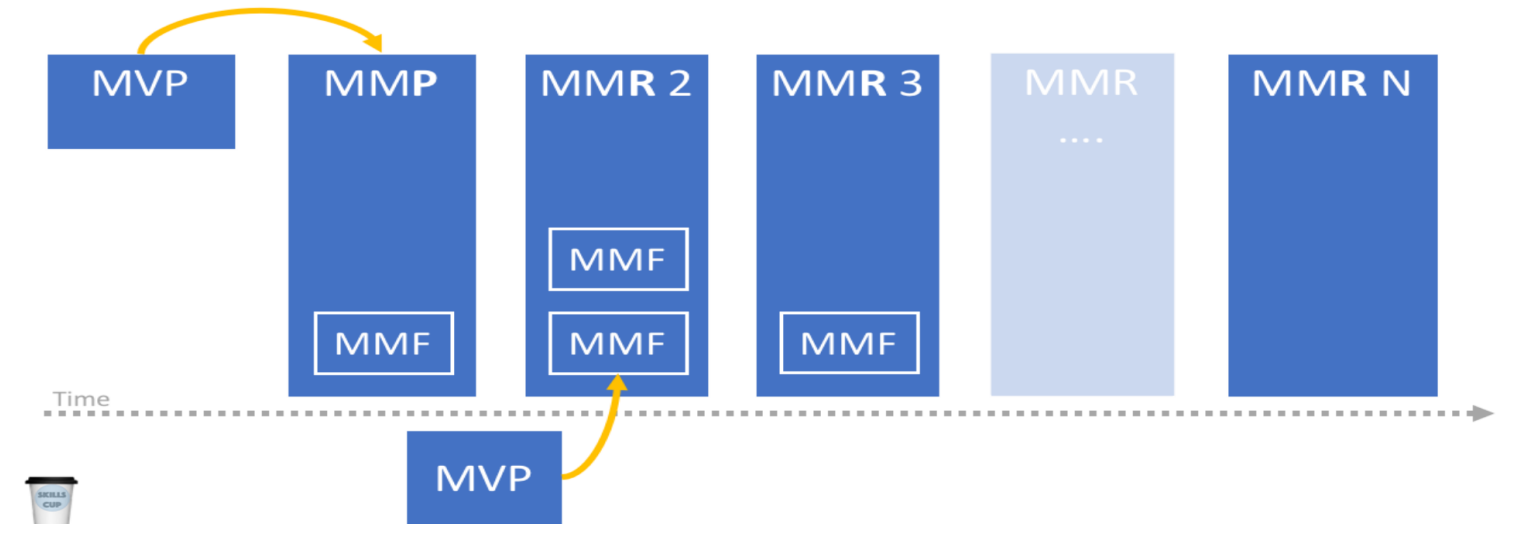
Un MVP puede ser un video o un ppt, no necesariamente un producto, porque es para validar una hipótesis, y si se puede validar de otra forma también se puede.

MVP: Es el punto de características que elegimos para armar una versión del producto que necesitamos salir a validar, si lo que se nos ocurrió esta bueno mediante una validación. La clave para esto es la hipótesis, sin la hipótesis es muy difícil validar.

MVF: Versión más pequeña del MVP, es una característica de el, un conjunto de MVF arman un MVP, en algunas ocasiones un solo MVF arma un MVP.

MRF: Minimal **Release** Feature: que características mínimas tiene que tener el reléase para ponerse en producción.

**Relación entre MVP, MMF, MMP, MMR.**



Al partir de un MVP, se parte de una validación de hipótesis. Superada la hipótesis, los inversores deciden seguir invirtiendo, y al cerrar la validación, se cambia la V por M, MMP (minimal marketing product) conteniendo una feature (MMF), sale al mercado y es exitosa. Se procede a generar otro reléase (MMR2) donde agregamos otra característica (MMF) la cual puede ser agregada mediante otro MVP, y así sucesivamente (ejemplo no siempre es así).

**MVP**

* Versión de un nuevo producto creado con el menor esfuerzo posible
* Dirigido a un subconjunto de clientes potenciales
* Utilizado para obtener aprendizaje validado.
* Más cercano a los prototipos que a una version real funcionando de un producto.

**MMF**

* Es la pieza más pequeña de funcionalidad que puede ser liberada
* Tiene valor tanto para la organización como para los usuarios.
* Es parte de un MMR or MMP.

**MMP**

* Primer release de un MMR dirigido a primeros usuarios (early adopters).
* Focalizado en características clave que satisfarán a este grupo clave.

**MMR**

* Release de un producto que tiene el conjunto de características más pequeño posible.
* El incremento más pequeño que ofrece un valor nuevo a los usuarios y satisfice sus necesidades actuales.
* MMP = MMR1

Completar mas con filminas y libros¿?¿?

Errores comunes

…

**Valor vs Desperdicio**

**Todo lo que no genera valor, es desperdicio**, con esta motivación se mejoran procesos y la calidad de los productos y servicios.

Valor para el usuario y para el negocio.

El problema es que si se prioriza el valor de negocio por sobre el del usuario no se va a llegar porque el usuario no va a comprar, el valor de negocio es una consecuencia del valor de usuario, ambos están relacionados.

La productividad de un Startup no puede medirse en términos de cuánto se construye cada día, por el contrario, se debe medir en términos de averiguar la cosa correcta a construir cada día.

El trabajo intelectual se mide en resultados y no en horas.

**Hacia la creación de valor**

El sw es un medio para obtener valor, entregamos una característica que el cliente va a usar y eso nos da valor, no lo bien que este hecho.

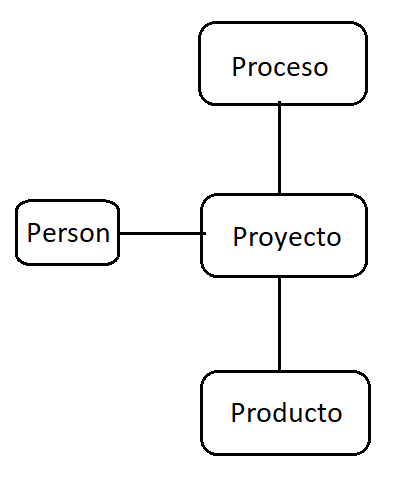
**Fase de experimentación**

… DELIRÓ.

**“Saltos de Fe”**

ISW 7/9 TEORICO

**Procesos**



[Piensa lento, piensa rápido. Coleman]

**Proceso:** conjunto de actividades interrelacionadas que indican lo que hay que hacer para cumplir el objetivo.

-Definidos: Establecer de antemano todas las cosas que voy a necesitar hacer para cumplir el objetivo del producto a realizar. Bien identificadas las cosas a hacer, el orden, los artefactos que hay que generar, los roles responsables de llevar adelante esas tareas, tiene una descripción muy detallada del paso a paso del proceso. Esa construcción de proceso viene definida por otro. La organización define y quienes definen son roles dedicados a hacer eso que **NO** van a ser los que hacen el trabajo después (cuando pase a proyecto), y hacer cambios al proceso es más costoso, porque como los procesos se definen en la parte de proceso, al querer incorporar cambios requiere un acuerdo a nivel organizacional. Característica de tener visibilidad de donde estamos, que hicimos y que falta para terminar. Busca predecir tiempos de procesos futuros en base a procesos pasados muy similares.

-Empíricos: Ágil, Lean. Ciclos de vida **cortos**, enfoque en el valor. Sirve la experiencia, la experiencia del equipo y proyecto en este momento en particular, no de otra gente. La experiencia no es extrapolable tan lejos como plantean los procesos definidos.

Todo lo realizado en **proceso** es teórico, y es puesto en práctica en **proyecto**.

**Proyecto** y **producto** tienen ciclo de vida, el de proyecto es el que estudiamos en diseño, el de producto son las distintas versiones que va teniendo el producto hasta que se descontinúa, el cual dura más y tiene muchos proyectos, muchos ciclos de vida de proyectos dentro de ese solo ciclo de vida de producto.

Tipos de ciclo de vida

* Secuencial -> Cascada
* Iterativo -> Iterativo e incremental
* Recursivo -> Espiral

Ciclo de vida: en que orden se ejecutan las tareas, cuanto de cada tarea si toda o una parte, en que momento se van a llevar a cabo esas tareas.

Cascada: realiza todo lo de **R** para después pasar a **A**, y así sucesivamente completando todas las tareas de cada letra. Es más fácil de gestionar y da ilusión de visibilidad de avance más ordenada estando en UN solo lugar a la vez, pero **NO** es recomendable.   
Iterativo e incremental: va realizando partes de **R** y de **A** en iteraciones hasta completar todas las iteraciones. **Recomendados👍**. Iteraciones mínimas: dos, porque si fuera una seria cascada xdxd. No tiene máximas como tal.

Recursivo: no hacen entregas parciales como los iterativos y tampoco es como el cascada que te da un producto recién al final. El recursivo hace énfasis en la gestión de riesgos, lo va refinando y mejorando en cada vuelta, pero el producto es obtenido recién al final (como el cascada). Se usa para productos con mucho riesgo que no tiene sentido ir liberando entregas, como ser un satélite que requiere ir refinándolo y largarlo al final bien hecho. No tuvo mucho éxito.

**R**

**A**

**D**-> Diseño Arquitectura, D Base de datos, D Experiencia Usuario.

**I**

**P**

**D**

**Iterativo e Incremental**

PUD**:** Alcance fijo. Variable que indica el fin de la iteración es implementar todos los casos de uso comprometidos a realizar (alcance fijo). Variando recursos y **tiempo**. El cliente no sabe cuando va a recibir el producto.

SCRUM**:** Duración fija. La variable que indica el fin de la iteración es la fecha de presentación del incremento (duración fija). Variando el **alcance** y recursos. El cliente sabe cuando va a recibir el producto, pero puede recibir poco en esa entrega (o nada en las primeras entregas). Hay que tener cuidado de no hacer una mini cascada dentro de un sprint, pasando por cada etapa RADIPD las users y al terminar el tiempo quedar a la mitad sin tener nada para entregar.

**Proyecto**

Unidad de gestión que permite obtener un **producto** o **servicio** **ÚNICO**, es un medio de organización, permitiéndonos administrar recursos, gestionar personas para obtener un resultado que va a ser único (en un producto, cada versión es diferente a la anterior por lo cual también son únicos).

Tiene fecha de inicio y de fin bien identificados.

**Elaboración gradual**: el todo se divide en partes (tareas) y se avanza de a poco, se logra debido a tener **tareas interrelacionadas**.

Definir un **objetivo** del proyecto que es el resultado, el objetivo debe ser claro(no ambiguo) y alcanzable(realizable). Todos deben entender el objetivo y se debe poder realizar.

**Alcance** del proyecto: todo el trabajo y solo el trabajo que hace falta hacer para cumplir con el objetivo. **NO hacer de más ni de menos**. (no sólo programación).

No mezclar proyecto con producto, ambos tienen objetivo y alcance, pero NO es lo mismo. De un producto se lee el objetivo y alcance en la **ERS**, y en el proyecto en el **Plan de proyecto**

**¿Cómo están relacionados? ej**

Producto: Gestión notas estudiantes. Obj: gestionar notas de estudiantes… alcance (no a quien va dirigido sino lo que debe tener): …?

En una casa serian: dormitorios, salas, muebles, colores.

Proyecto: obj: desarrollar un producto de sw que gestione las notas de los estudiantes, alcance(lo que hay que hacer **NO** a quien va dirigido): elicitar reqs, hacer entrevistas, modelar dominio, casos de uso, programar, cosas a hacer para cumplir el objetivo de desarrollar un producto.

Obj: crear especificaciones de reqs para un producto q gestione notas… Alcance: elicitar reqs, entrevistas, casos de uso.

Entonces en un plan de proyecto se deben definir: Objetivo, Alcance, Proceso y ciclo de vida, Estimaciones.

Estimaciones de SW

Lo primero que se estima es el **tamaño (qué)**:

* LoC (líneas de código sin comentarios) (ya no se usa).
* Casos de Uso
* Cantidad de clases, métodos, alcances.

Lo segundo a estimar es el **esfuerzo (cómo)**: siempre se mide en horas persona lineales (ideales sin desperdicios).

Luego se estima el **tiempo (cuando)**: la unidad es variable, usando tiempo calendario, se suele recomendar usar semanas.

La cuestión es como transformar el esfuerzo de horas en tiempo calendario, viendo la gente disponible, sus perfiles, cuantos días a la semana vamos a trabajar y cuantas horas de esos días, luego se ve que cosas podemos hacer en paralelo y que dependencia hay entre las tareas (índice de solapamiento) gestión de dependencias. Cada persona debe hacer una tarea a la vez, no sirve el multitasking.

luego se estiman **Recursos críticos**: licencias. (las pc y cosas básicas se supone que están)

después se estiman **Riesgos:** probabilidad de ocurrencia de algo que pueda impactar negativamente en mi proyecto. Dos dimensiones: probabilidad e impacto. Para comparar dos riesgos se multiplica la probabilidad e impacto de ambos y te da exposición para saber cual es mas peligroso. Riesgo con probabilidad 1 no es riesgo, es problema. Gestión proactiva: anticiparse, gestión reactiva: reaccionar ante el problema.

------ fin estimaciones ------

**Métricas**

**Seguimiento de proyecto**

ISW 14/9 TEORICO

U1 completa

U2 “lo que vimos”, gestión ágil de reqs, users, mvp, estimaciones. (la mitad u2)

U3 gestión de configuración.

2 a desarrollar

Multiple choice (a b c d, UNA sola)

US completa, frase verbal, criterio aceptación, prueba de usuario.

En otro punto estimación y encontrar canónica.

MVP justificando

**Gestión de configuraciones de software (SCM)**

¿Qué es? ¿Para qué sirve?

Es una disciplina de soporte o protectora, cuyo objetivo es mantener la integridad del producto de software a lo largo de toda su vida. (con esa le basta je)

Un producto de sw tiene integridad cuando: deja contento al usuario, se lo puede rastrear, satisface criterios de performance (RF y RNF) y cumple con sus expectativas de costo, tiempos y plazos de entrega.

¿Para qué sirve?

Su propósito es establecer y mantener la integridad de los productos de software a lo largo de su ciclo de vida.

El proceso es una definición teórica que se adapta en función de obligaciones contractuales, el nivel de expertiz del equipo, entre otras, al proyecto ¿el cual requiere un cronograma?

¿Por qué es importante que se haga gestión de configuración de software?

Resulta que el software, (que no es solo código), es mucho más amplio que solo código, “software = conocimiento = información”, que se puede presentar a distintos niveles de abstracción. US, BD, diseño de UI, plan de proyecto es software. El proyecto es una sumatoria de tareas, y cada tarea debe tener una salida, resultado, el cual es software. (en PUD era artefacto) hay artefactos asociados al proyecto (ej) y al producto (casos de prueba, documentación), reflejan las decisiones tomadas y el conocimiento adquirido.

“Automatizado con herramientas” como ser el control de versiones, utilizando la herramienta git, por ejemplo.

SW = conjunto de programas, procedimientos, reglas, documentación, datos. (def. de libro). (a la profe le gusta más la que dio ella)

Aborda la problemática del control de versiones, de no saber quien tiene la última versión del software o quien borro y perdió todo lo realizado (debido a no tener el versionado)

SCM como disciplina de soporte. Es una actividad “paraguas”, transversal a todo el proyecto (en la cual se ejecutan estas actividades de soporte), relevante para el producto a lo largo de su ciclo de vida. Pero se realizan durante el proyecto, mientras se hacen ítems de configuración.

Ing. De Software abarca disciplinas:

* Técnicas (se ocupan del **producto**): Reqs, Análisis y diseño, programación, prueba, despliegue. Creación de producto con calidad.
* Gestión (se ocupan del **proyecto**): Planificación de proyecto y el seguimiento y supervisión.
* Soporte/protectoras (se ocupan del **proyecto** y **producto** (todo)): SCM, PPQA (aseguramiento de calidad de proceso y de producto), métricas. Intentan asegurar que la calidad del producto sea la acordada con el cliente. Deben tener en cuenta todas las otras disciplinas, porque mantiene la integridad del producto de sw a lo largo de su vida. Se debe hacer SCM desde el momento 0.

Cualquier artefacto = **ítem de configuración** (software).

Gráfico, Diagrama

Descripción generada automáticamente

El testing controla el sw una vez hecho, el aseguramiento de calidad de sw lo controla con anticipación, previniendo. SIEMPRE se debe hacer testing para detectar errores, porque somos humanos👍.

Problemas en el manejo de componentes (filmina)



Tener claro cada uno de esos ítems.

Ítem de configuración: cualquier cosa que se pueda guardar en un disco (artefactos que forman parte del producto) **con su versión en un momento de tiempo** (es la forma de identificarlos unívocamente). Ej: (filmina) Plan de desarrollo, manual de usuario, requerimientos. [Ej: User1 v1.2]

Variante: varias personas versionando

Configuración de sw: conjunto de ítems de configuración con su correspondiente versión en un momento determinado, es una foto de las versiones de todos los ítems en un momento determinado.

Repositorio: contenedor, espacio en un disco, reservado para contener todos los ítems de configuración del producto, es deseable que tenga una estructura. Analogía con placard sin estantes ni cajones, tirando la ropa, lo cual dificulta recuperar después una remera👍. En el repositorio el número de versión no debería ir en el nombre del ítem, para algo está la versión xd, la herramienta lo versiona, al cambiarle el nombre lo toma como un ítem nuevo.

Un repositorio por proyecto. Si un producto tiene varios proyectos se pueden usar varias carpetas que después van a parar a la raíz.

Repositorios centralizados y descentralizados. (filmina)

Línea Base: conjunto de ítems con su versión que ha sido aprobado y es estable, y que puede ser usado como referencia para seguir avanzando en la construcción del producto. Para marcarla se la etiqueta, en el repositorio se eligen un conjunto de ítems y se los marca como línea base etiquetándolos, al tirar de una etiqueta te traen todos los ítems de configuración con la versión que tenían en el momento de etiquetarlo. Se pone cuando pasa todas las revisiones técnicas que tenia que pasar (cada ítem tiene sus revisiones, no son las mismas para todos) no se puede meter algo defectuoso a una línea base. La línea base no se puede cambiar sin usar un procedimiento formal de cambio.

Gráfico, Gráfico de dispersión

Descripción generada automáticamente

El propósito de las líneas bases es mantener informadas a las personas de qué cambió en la nueva línea.

Pueden ser:

▪ De especificación (Requerimientos, Diseño). Tiene solo documentación.

▪ De productos que han pasado por un control de calidad definido previamente. Tiene código.

Ramas: sirven para bifurcar el desarrollo, por lo general se abre una rama por cada equipo que esta trabajando y trabajan sobre esa rama hasta que se haga la integración de la misma (merge). No se sugiere dejar merges automáticos xd. Al finalizar deberían cerrarse las ramas y quedar solo la principal.

Actividades fundamentales de la SCM: Control de cambios, Informes de Estado, Identificación de Items, Auditorias de configuración.

Identificación de ítems: identificación **unívoca,** asignarle un nombre único, para un proyecto de software se tienen distintos ítems a identificar que varia de acuerdo a:

Diagrama

Descripción generada automáticamente

Por ejemplo:

* ERS\_DeliverEat.docx (identifica el ítem sin poner versión)
* Ítem de producto cuando son varios (como CU): DeliveryEat\_CU\_<<nombreCU>>.docx
* Plan de proyecto (uno solo): AD\_PlanProducto.docx

En el plan de gestión de configuraciones debería ir toda la estructura. (NO EN EL README)

Cambios en la **línea base**: comité de control de cambios.

Diagrama

Descripción generada automáticamente

¿Quién puede formar parte del comité de cambios, quién debería?

El PO, el PM, el arquitecto, analistas funcionales, gente de testing, el cliente. Cualquiera que se pueda ver afectado por el cambio evaluado, debe formar parte del comité.

Auditorías de configuración de software: se hacen de dos tipos: **física** y **funcional. NO** son sorpresa, y lo que se va a auditar tampoco.

¿Qué es una auditoría? Es una revisión **objetiva** e **independiente** sobre un proceso o un producto. El que va a ser auditor no puede ser parte del equipo ni dependiente del gerente del equipo (para buscar la objetividad), se suelen contratar externos o de otra área que no tenga dependencia funcional con el jefe del equipo. La auditoría de configuración funcional vela por la integridad…

Sin plan no se puede auditar porque es lo que se revisa para corroborar que se trabajó correctamente sobre lo que se definió en el plan. ¿Qué se audita? Una línea base. Si la auditoría física dio mal resultado, no se hace la funcional, porque va a estar mal. Corrobora la consistencia de lo que se debía hacer con lo que se hizo.

En una auditoría de configuración física (se hace primero) se ve la integridad del repositorio y que este consistente con lo que se planificó.

Se debería hacer una auditoría de cada tipo, mínimo, para liberar un producto.

Informes de estado.

En un enfoque ágil la actividad que el equipo podría resistirse son las auditorías, porque el equipo piensa que un equipo autoorganizado lo puede resolver el equipo y la auditoría al ser objetiva e independiente, de afuera del equipo, entonces creen que de afuera no deberían opinar sobre lo que hicieron, pero pueden hacerse tranquilamente, es compatible SCRUM con auditorías. El cliente puede exigirlo.