Toma de notas ISW:

17/08: Requerimientos Agiles USER STORIES

técnica para trabajar requerimientos

la ingeniería de software tiene que cubrir estos aspectos: proyecto, procesos y productos. Pero incluimos también un aspecto muy importante que es las personas (tiene un rol fundamental en el éxito o fracaso del desarrollo del software).

Infografía información en imágenes

Lectura obligatoria **no silver bullet** (lo pregunta en el parcial). Parte más difícil de construir software es decidir que construir, es decir establecer los requerimientos.

Procesos empíricos: diferente de proceso definido (PUD O RUD). El foco nuestro es trabajar con procesos empíricos. Filosofía ágil y lean dos enfoques para abordarlos, promueven y utilizan procesos empíricos. Los procesos definidos intentan ser la expresión de completitud ósea todo lo que nosotros necesitamos para llevar a cabo el software tiene que estar descripto ahí, y esta definido de antemano a nivel organizacional y por gente diferente de la que va a hacer el trabajo. El agilismo surgió en el software mientras que lean surgió en la industria automotriz. Encontrar requerimientos agiles, técnica que usamos son las user stories. Los procesos empíricos el que tomas las decisiones es el que hace el trabajo, basado en la experiencia, entonces tenemos que conseguir experiencia de 3 pilares: inspección, adaptación, transparencia. Los ciclos de vida **iterativos** y tienen que ser cortos (es fundamental), ya que yo gano experiencia a través de la retroalimentación de las iteraciones. Quiero decidir que proceso voy a usar y lo definimos con lo menos que necesiten para funcionar. Cada proyecto es distinto de otro, la experiencia se gana en el contexto de este proyecto y de este equipo. La transparencia, no hay nada oculto, la información es de todos, no es solo mi código, si hay un problema pido ayuda, si hago una tarea le cuento a todos, la situación del proyecto debe ser visible para todos en todo momento.

Valoro más las personas y los vínculos, y entregar software funcionando. No prototipos sino versiones del producto que si funciona. El agilismo funciona con equipos que se organizan solo y no que tengan un líder que les dice que hacer a cada uno. Los mejores requeirmientos surgen de la interacción entre el equipo de desarrollo y el product owner. Hay requerimientos emergentes que aparecen a medida que se van desarrollando las iteraciones del producto.

Product owner no es parte del equipo de desarrollo, sino que es del negocio y define las características que necesito de mi producto ahora y que voy a necesitar después. Es representante del cliente.

El manifiesto ágil es una ideología con un conjunto definido de principios que guían el desarrollo del producto. Siguen siendo ordenados y rigurosos. Tiene varios frameworks por ejemplo scrum, crystal, etc.

Los enfoques ágil y lean te dan buenas prácticas o lineamientos para cómo resolver cosas, pero no arman un proceso por eso no podemos decir que son una metodología.

Lo importante es el valor de negocio que le entregamos al cliente y no el software particular que hacemos. El software es una herramienta, un medio para un fin. Las user stories están en un nivel de abstracción más alto, a nivel de usuario, la descripción tiene que ser cortas y describir una necesidad que tiene el usuario. Priorización entender lo bueno y suficiente, saber cuándo parar. Aferra a la idea de que hay características del software que no se usan nunca, entonces tenemos que hacer foco en lo que si se va a usar. El product owner es el encargado de la priorización, artefacto product backlog, las cosas mas importantes se ubican arriba (ver PowerPoint 😊). Concepto **just in time** contrario a la descripción de ante mano de todos los requerimientos, sino solo los que voy a hacer ahora y que le tengo que dar al cliente. Eliminar desperdicios no invertir tiempo en escribir requerimientos que después voy a cambiar, ósea desperdicio tiempo. Es mucho más efectiva la comunicación cara a cara, o sea todos trabajando en la misma locación.

Requerimiento de negocio y usuario son los que contemplan las user stories.

Direccionan el proyecto tres aspectos: alcances, costos y tiempo. Tradicional fijan los requerimientos al principio y a partir de esos requerimientos deciden los recursos y el tiempo de trabajo del equipo. Hay resistencia en cambiar los requisitos ya que implica un cambio en el costo y el cliente no quiere. Para contrarrestar esto, esta el enfoque del agilismo donde dejan libre los alcances, pero en cada iteración de desarrollo voy a dejar fijo los recursos (equipo) y el tiempo (iteraciones de duración fija, time box), y se acuerda en el equipo que requerimientos se van a entregar.

User stories: a través de una historia tratan de expresar una necesidad. Tres partes de una historia de usuario: tarjeta, confirmación y conversación. La conversación es la mas importante, pero no queda en ningún lado, ósea lo que queda visible son las otras dos. **Tiene que contestar quien, que necesita ese quien y para o porque lo necesita** (valor de negocio que recibo, mas importante). No describen como, nada de tecnología ni como voy a hacer para resolverlo. El usuario solo expresa su necesidad. Complementar las user stories con los criterios de aceptación (no es lo mismo que pruebas de aceptación, estas muestran que se cumplen los criterios de aceptación) que completan. La frase verbal nos dice el que. Lo otro describe partes del producto que tengo que desarrollar. Hay niveles de granularidad diferentes que define el equipo. Las user stories son porciones verticales, o sea tengo que entregar software funcionando, cada user story tiene un poco de cada capa para que pueda funcionar, solo desarrollas lo que te hace falta para que funcione.

Modelado de roles usamos tarjeta de rol de usuario.

Proxies aparecen cuando el product owner es inútil, no deberían estar en general (son roles del negocio no de desarrollo de software).

Criterios de aceptación son una formalización de lo que el product owner quiere. Definen limites para una user story. Es un requerimiento no funcional de calidad. De acá se derivan las pruebas de aceptación y debemos tener una que lo cumpla y otra que no. Los detalles no están en la carta, sino que están en la conversación y en las pruebas (debe tener muchos detalles y estar actualizadas). Las pruebas nos ayudan a decidir si las condiciones se aceptan o no. Pensar cosas positivas y negativas. Las pruebas de aceptación son un acuerdo, si funcionan como tienen que funcionar se acepta la user story.

Definition of ready se aplica a la user story, cuando esto es aceptado, la user puede entrar en una iteración y puede ser implementada. El equipo lo define porque es técnico. Es un checklist de todas las cosas que tiene que cumplir. Tiene un acuerdo mínimo, para estar lista tiene que cumplir con el INVEST model (independent, negotiable, valuable, estimable, small, testable), en cualquier orden pero lo prioriza el product owner.

* Independiente
* Negociable: explica el que y no el cómo. No voy a escribir en la user el cómo.
* Valuable: debe tener valor para el cliente, o sea el valor de negocio de esa user.
* Estimable: debo poder estimar una user para que este bien. Si no es estimable se transforma en una spike. Puedo asignarle un valor numérico.
* Small: se puede realizar en una sola iteración. Debo iniciar y terminar esa user en esa iteración.
* Testeable: demostrar que fueron implementadas en los términos que el product owner dijo.

**24/08:**

¿Porque son agiles los requerimientos agiles? No se establecen desde el inicio, se van estableciendo a lo largo de que avanza el proyecto. Lo primero que debemos tener en cuenta es el valor del negocio, no hacemos software como un fin sino es un medio para entregarle valor al cliente. El desafío es que yo parto de la visión de un producto y no un producto completo, me determina la definición de la primera versión del producto que voy a tener, es decir parto de un objetivo y algunos alcances que me van a permitir generar el producto backlog (artefacto) que nunca va a estar completo que necesita la cantidad necesaria de user stories para desarrollar la primera iteración. No corro el riesgo de hacer algo que capaz no hace falta, eliminar desperdicio, porque solo detallamos lo que hace falta.

La gestión ágil es una gestión binaria. Las cosas están o no están, entregan valor o no entregan valor.

Definition of ready es cuando esta lista para sumarse al backlog.

Definition of done define cuando una historia esta lista para ser mostrada al product owner. ¿Quién lo hace? El equipo se pone de acuerdo con el definition of done.

Una épica es una user muy grande, que hay que descomponer en varias users. Esta abajo del backlog.

**Spikes cuando el equipo no la puede estimar**, nivel de incertidumbre alto. Hay mucha desinformación por eso no la puedo estimar. Dos tipos de spike: spike técnica (tiene que ver con la tecnología y esas cosas) y spike funcional (viene del lado del product owner, o sea del negocio). Siempre hay incertidumbre igual.

Estimaciones como se estima en los entornos agiles.

Estimación diferente de priorización.

Las estimaciones se basan en los requerimientos, lo difícil es estimar, porque tiene asociado un factor de probabilidad, hay incertidumbre por eso estimas. Se estiman en dos momentos cuando hay una granularidad gruesa o cuando hay una granularidad fina. Lo que nosotros llamamos estimaciones agiles tienen ciertas características:

* Son relativas, somos mejores comparando, dependen. Hacemos estimaciones por comparación.
* Se hace foco en la certeza y no en la precisión.
* No hagamos un esfuerzo de estimar todo el producto, igual que con los requerimientos just in time. Durante el sprint planning se reúne el equipo y estiman las user stories, definen cuales y hasta donde vana a estimar.
* Estima el que hace el trabajo, o sea el equipo y todos asumen el compromiso. En lo tradicional, el que estima es el líder del proyecto y asume un compromiso.

Story point es la unidad de las estimaciones. Representa el tamaño de la historia de usuario. Se usa la serie de Fibonacci para estimarla numéricamente, tiene un crecimiento exponencial igual que el crecimiento de complejidad del software. Tiene que estimar todo el equipo junto, así aprovechamos la visión de todos, se estima de forma colectiva. Encontrar user stories canónica es el elemento que nosotros vamos a usar para realizar las comparaciones.

Poker planning:

Se hace una ronda de estimación individual y luego se revela y se explica porque se da ese puntaje. Liego se hace otra vez y ahí es cuando hay convergencia. Las estimaciones tienen que ser convenientes. Se tiene que hacer esto por cada story que haya que estimar. Se usa la canónica, generalmente de un puntaje de uno (es l amas fácil o conocido) o de dos, (a la profe le gusta de tres porque da margen para arriba y para abajo), para comparar.

Se descompone en tres dimensiones que pueden tener una story en el momento de asignar un story point son complejidad, esfuerzo y duda o incertidumbre.

* Complejidad: cuan dificultosa es para realizarla.
* Esfuerzo: cuantas horas ideales (sacando distracciones, horas de trabajo posta) necesitamos para crear esto.
* Duda: que nivel de desinformación tengo. Técnica o de negocio. Levanta más la estimación porque no puedo medir lo que no se.

Pueden tener mismo peso varias stories, como por ejemplo una es más compleja pero otra tiene más duda y tienen el mismo peso. Necesito un valor de homogenización que es el story point. El story point lo vamos a tener que justificar con las tres dimensiones.

**31/08:**

Gestión de productos

Necesidad de salir al mercado para justificar el esfuerzo y los recursos invertidos. Lean startup.

MVP producto mínimo viable. (es del practico y va al parcial practico). En el parcial encontrar los mvps, describir dos user stories y estimarlas con justificación.

Hacemos productos con servicios. Product manager (producto owner) y Project manager (actividades para gestionar un proyecto, recursos y personas).

Consultoras surgen de la terciarización porque las áreas de sistemas de las empresas colapsan, entonces llaman a las consultoras para que ayuden con los proyectos de software.

Porque queremos crear productos de software: para satisfacer a los clientes, para obtener dinero, realizar una gran visión y cambiar el mundo, para tener muchos usuarios logueados (ganar usuarios que usen tu aplicación).

Time to market o sea cuando sacar el producto al mercado. Concientizar de que hay muchas características de software que no se usan por eso elegimos puntualmente que si vamos a desarrollar.

Evolución de los productos de software: focalizar el producto en la experiencia en quien va a usar el producto. La funcionalidad del producto, base de la pirámide, cumple con lo que tiene que hacer y no tiene errores. Después tenemos que garantizar de que el producto es confiable, después usabilidad que está relacionado con la experiencia de usuario o sea que tanto disfruto de usar el producto. Un producto de software significativo es un producto que le cambia la vida a la gente por ejemplo WhatsApp o mercado pago.

A diagram of a pyramid

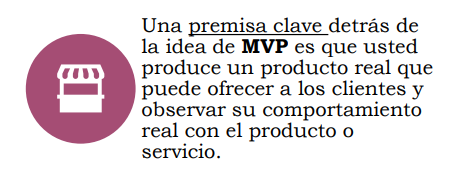
Description automatically generated

Producto que tenga valor y que sea único. **Hipótesis,** el mvp se crea para validar a una hipótesis o sea si voy a conseguir cliente y si los voy a poder mantener. Introducir variantes o propuestas que vienen del usuario, que nosotros tenemos que decidir si queremos modificar el mvp. MVF minimal viable feature, o sea una característica. Un mvp puede ser un solo MVF, que nosotros queremos probar en el mercado. MMF es comercializable, tiene que ver con las características mínimas que debe tener el producto para que yo pueda comercializarlo, muchas veces se confunden el MVP y MMF, si no vas a resolver la hipótesis no es un mvp. Estamos creando el producto, entonces validamos si ese es el producto que tenemos que crear, con las características que debe tener. El MMF es marketing o sea las características que yo tengo que cumplir para poder vender el producto. Vamos obteniendo retroalimentación entonces el producto se va modificando. El mvp puede ser un video o una presentación, no necesariamente tiene que ser codigo. Ambigüedad.

* Producto mínimo viable es el conjunto de característica que nosotros elegimos para validar la hipótesis.
* Característica mínima viable es una sola característica que se quiere validar, el MVP puede ser una o varias MVF.
* Características mínimas del ralease que características mínimas debe tener el reléase para poner en producción.
* MMP es el primero que sea hace para poder comercializarlo.

Parto del mvp parto de la hipótesis. Supero la hipótesis y sigo con el producto, ahora tengo que fijarme en la mmp que tiene las mmf. Si una de las características salió bien vamos a generar un nuevo reléase y ahí volvemos a agregar nuevas características.

Libro de Eric Ries lean startup.



Errores comunes: confundir mvp con mmf

Valor vs desperdicio:

Pricipio lean: eliminar desperdicios.

Hay una relación entre la calidad del producto y la calidad del proceso. Analizar de todos los esfuerzos que nosotros hacemos definir cual vale la pena y cuales no. Si no te genera valor es un desperdicio y si es desperdicio hay que eliminarlo y así poder ser más eficiente. Valor para el cliente y para mi negocio, no puedo priorizar el valor del negocio por el valor del cliente porque si no el cliente no me va a comprar. No tiene sentido gestionar la organización por la cantidad de horas trabajadas, porque hay horas que no generas valor.

La fase de experimentación: mejor le sale a la gente que menos tiene para perder. La audacia de cero. Saltos de fe.



**7/09**

Procesos: como vamos a hacer las cosas

* Definidos: establecer de antemano todas las cosas que yo voy a necesitar para cumplir con el objetivo del proceso. Que cosas hacer, en que momentos, los artefactos, los roles, etc. La definición viene definida por otro, la organización define todas estas cosas, pero ellos no son los que van a desarrollarlo. Hacer ajuste y cambios son costosos porque el proceso viene de la organización y si queres incorporar cambios al proceso es más difícil porque tiene que haber un acuerdo organizacional. La resistencia al cambio resulta en procesos obsoletos que no le aportan a la organización. El sustento de los procesos definidos es tener visibilidad, los que tiene que hacer las cosas están enterados. Podes usar cualquier tipo de ciclo de vida.
* Empíricos: dicen que lo que sirve es la experiencia de estos procesos y este equipo. Este conocimiento es lo que voy a usar para hacer las correcciones. No es extrapolable la experiencia. Si o si tiene que usar un ciclo de vida iterativo.

Que necesitamos para hacer software: captura de requerimientos, análisis, diseño, implementación, prueba y despliegue.

Diferencia entre proceso y ciclo de vida: hay tres tipos genéricos de ciclos de vida secuencial, iterativo y recursivo. El ciclo de vida es mas abstracto que el proceso, nos dice en que orden vamos a ejecutar las tareas, en qué momento, y cuanto vamos a ejecutar. Le da información para que el proceso se pueda ejecutar dentro de un proyecto. El ciclo de vida afecta la forma en que se gestiona el proyecto.

El proyecto tiene un ciclo de vida y el producto también tiene un ciclo de vida. El del producto es mas grande y dura mas que el del proyecto. El ciclo de vida define estado del proyecto por los que pasa desde que comienza hasta que termina.

Iterativo incremental en SCRUM: al producto lo voy a entregar en distintas iteraciones que cada iteración va a tener funcionalidades limitadas y en cada iteración se va a incrementando la funcionalidad. Cantidad mínima de iteraciones dos porque si no seria en cascada. el tiempo queda fijo, o sea es de duración fija. Variable que indica el fin de la iteración. Si no llegue a todo entrego menos pero si o si hay que cumplir con el tiempo.

El PUD el alcance es fijo y lo que varía es el tiempo.

Que es un proyecto: unidad de gestión que nos permite obtener un producto o servicio único. Importante: el resultado que se obtiene es único. Es un medio de organización.

Características de un proyecto:

* Resultado único

Estimación de software:

* Tamaño: dimensionar el producto que tenemos que construir. Se puede medir en casos de usos. Contesta el que
* Esfuerzo: cantidad de horas personas útiles que va a necesitar el producto para terminarse. Contesta el como.
* Tiempo: contesta el cuándo.
* Costo: las decisiones que yo tomo con respecto al tiempo me van a cambiar el costo.
* Recursos críticos: es recomendación, para que no generen desfasajes en el calendario.

Problema de la gestión de proyectos cuando mezclamos el tiempo con el esfuerzo.

Los riesgos también se estiman. La probabilidad de ocurrencia de algo que pueda impactar negativamente en mi proyecto. La palabra clave es probabilidad. Se mide por dos dimensiones: una la probabilidad y la otra es el impacto. Multiplico la probabilidad con el impacto y obtengo la exposición, que es valor numérico que me permite compara un riesgo con otro. Un riesgo con probabilidad 1 es un problema.

Métricas y seguimiento del proyecto.

**14/09:**

Scrum no va, va al otro parcial.

Unidad 3: gestión de configuración.

Parcial teórico: preguntas multiple choice (con solo una respuesta correcta) y máximo dos preguntas a desarrollar. Primero teórico y después el práctico (2hs).

Siempre es importante de un concepto saber qué es y para qué sirve.

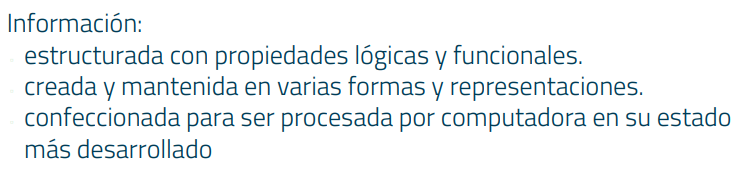
Software configuration management: es una disciplina de soporte concreta cuyo propósito es mantener la integridad del software a lo largo de toda su vida. Gestión de configuración de software. Hay herramientas automatizadas para la gestión de configuración. Es una actividad transversal que abarca todo. Se hacen mientras vos tenes un proyecto activo, pero se deben mantener para soportar el producto por más que allá terminado el proyecto. Es la estructura mínima básica para poder desarrollar un producto de calidad asegurada. Que es que un producto se software tenga integridad cuando deja contento al usuario, se lo puede rastrear entre las etapas (debo definir cómo voy a mantener la integridad del software), cumple con sus expectativas de costo y plazos de entrega, satisface los requerimientos no funcionales (performance, mantenibilidad, seguridad, etc.).

Las disciplinas técnicas como análisis, diseño, programando, prueba, etc. está relacionado con la creación del producto.

Las disciplinas de gestión que son la planificación de proyecto y el seguimiento y supervisión del proyecto, relacionado con la dimensión de proyecto.

Las disciplinas de soporte (también se les dice protectoras) que son SCM, aseguramiento de calidad de proceso y producto PPQA, métricas, pone el foco el proyecto y el producto. Intentan atender y prevenir situaciones que afecten la calidad del producto (el producto tiene calidad cuando cumple las expectativas de la funcionalidad y el tiempo en el que lo vas a entregar). Abarcan todo porque debe mantener la integridad del producto de software a lo largo de la vida del producto que vive mas que el proyecto. Se tiene que arrancar desde el momento cero, por eso lo primero que tenemos que crear es el repositorio.

El software, **no solo es código**, tiene un montón de cosas que hacen al software más allá del código. El software es igual a información que se puede presentar a distintos niveles de abstracción. El resultado de la ejecución de cualquier tarea del proyecto es software (artefacto como le decíamos antes). Voy a generar artefactos asociados al proyecto y al producto, y todo esto es software y es importante. El software tiene un problema, es que es fácil de modificar (ósea que lo podés borrar), la gestión de configuración viene a solucionar esto: problemas de versiones, cual es la real o quien tiene la última versión.



Palabras clave relacionadas:

* **ítem de configuración:** todos los artefactos que forman parte del producto o proyecto que tengan una versión en un momento de tiempo, porque los ítems se modifican, la única manera de identificarlos unívocamente es a través de la versión. Todo lo que se puede guardar en un disco. Ejemplo: user story 28 versión 1.2.
* **Repositorio:** contiene los ítems de configuración. Mantiene la historia de cada IC con sus atributos y relaciones. Debe tener una estructura para que quede ordenado. En una base de datos.
* **Línea base:** es un ítem o un conjunto de ellos con su versión que ha sido aprobado, es estable y que puede ser usado como referencia para la construcción del producto. Las etiquetamos. Si pedís te trae todos los ítems con su versión que forman parte de la línea base. Tienen que pasar todas las revisiones técnicas y controles de calidad que fueron definidas previamente. No se puede cambiar, solo a través de un procedimiento formal establecido para cambiarlo. Mantener informada a la gente sobre que se cambió.
* **Ramas:** hay una rama principal, y sirven para bifurcar el desarrollo. Permiten la experimentación. Pueden ser descartadas o integradas. Es principalmente para código.
* **Configuración del software:** un conjunto de ítems de configuración con su correspondiente versión en un momento determinado. La gestión de configuración administra esta mierda.

Versión: forma particular de un artefacto en un instante o contexto dado, ya que el artefacto evoluciona a través del tiempo.

Variante: es una versión de un ítem de configuración que evoluciona por separado.

Cuatro actividades de la gestión de configuración de software:

* Identificación de ítems: identificación univoca de cada ítem. Asignar un nombre único a cada ítem. Generar ítems que tiene distinto ciclos de vida (producto y proyecto). Se le da una ubicación dentro de la estructura del repositorio.
* Control de cambios: comité de control de cambios para incorporar cambio a una **línea base**. El comité evalúa el cambio y se autoriza o rechaza, si se autoriza deben definir qué día va a suceder. ¿Quién forma parte del comité? El PO, el arquitecto, el líder del proyecto, analista funcional, testing, el cliente si el cambio es algo muy grande, o sea cualquiera que se pueda ver afectado por ese cambio.
* Auditoria de configuración: auditoria de configuración física y funcional. Una auditoria es una revisión objetiva e independiente sobre un proceso o producto. El auditor no puede ser parte del equipo, por eso se contrata auditores externos. Funcional vela por la integridad de los ítems ve consistencia con los requerimientos y lo que estas creando. Necesito un plan para auditar. Se audita una línea base. Primero se hace la auditoria física y después la funcional. La funcional a la validación y la física a la verificación.
* Registro e informe de estado.