Tomas de notas 24/8

Requerimientos agiles

Si meles pregunta que son los requerimientos agiles o porque son agiles los requerimientos. Que caracteriza a los requerimiento para que podamos hablar de que es una gestion ágil de requerimientos?

No se establecen desde el inicio sino que se agrega conforme hace falta, just in time, el concepto de que yo lo primero que os tiene que salir es que hay un foco, tenemos que hacer foco cuando queremos hacer software con gestión ágil, el valor del negocio. Lo primero que tenemos que tener en cuenta es el valor de negocio. Nosotros no hacemos software como un fin en si mismo, sino que hacemos softawre como un medio para entregarle al cliente valor de negocio. Ese valor de negocio como estamos basados en proceso empíricos, la gran cosa que es desafiante en los requerimientos agiles e sque parto de algo que se llama visión del producto, no parto de un producto completo.

Parto de una vision de producto que tiene que responder al valor de negocio y esa visión de producto es lo que me va a determinar a mi la definición de la primera versión de producto que voy a tener. Esto significa que voy a partir con un objetivo, con un conjunto de alcances y con la conformación inicial del artefacto que nos va a guiar en la construcción del producto. Este artefacto se conoce como Product Backlog. Esto nunca va a esta completo y lo mínimo que necesitamos que tenga el product backlog es la cantidad suficientes de historias para ejecutar la primera iteración del producto que nos va a permitir a nosotros tener retroalimentación del producto.

Esta visión de tener un producto backlog inicializado es la lo que apunta cumplir con el valor del manifiesto que dice que uno va a recibir o bien recibir cambios aun hasta en etapas finales. No vamos a poder hacer una definición completa de antemano del producto sino que vamos a buscar un acuerdo para poder arrancar, la definición del producto y donde queremos ir, y a partir de ahí vamos a ir iterando y ajustando las necesidades y las características finales del producto. Eso en definitiva es la explicación mas larga del primer párrafo, eso de just in time. Just in time significa (viene del Lean) diferir decisiones en realidad, diferir decisiones hasta el último momento responsable. Esta alineado con no especificar de antemano si tenemos el riesgo de que no lo va a necesitar el cliente o quiere otra cosa, generando desperdicio que va en contra del lean que dice desechar desperdicios, solo hay que detallar lo que hace falta. La complicación tiene que ver con como me doy cuenta de que ya es suficiente, que es la problemática principal. Esto es la clave de la gestión del producto backlog. El product backlog es una pila o cola priorizada, la palabra clave es esa, priorizada. Ahí está otro de los grandes problemas que vamos a tener, si no vamos a hacer todo junto que hacemos primero.

Por ahora sabemos que el problema de la priorización es del producto owner que viene del negocio, hay una parte de historia que o tenemos que resolver nosotros, la decisión de lo que necesitamos es del product owner.

La esencia de los requerimientos agiles, que esta resumida en esto que listado está basada en los principios del manifiesto, basado en que la comunicación es cara a cara, también hay otros de los requerimientos que dice que los mejores requerimientos diseños y arquitecturas emergen de equipos auto organizados. Emergen es otra palabra clave, que es que el PO no ve el requerimiento hasta que surja.

Se habla de entregar software funcionando porque facilita la retroalimentación, y facilita la aparición e requerimientos emergentes.

Todo lo anterior es por que son agiles los requerimientos.

Ahí surge la técnica del user storie pero no es un artefacto nativo de scrum, lo adopta pero no es de scrum que tiene 3 artefactos pero ninguna es la user storie. La usamos como un medio para diferir una conversación, para acordarnos de hablar con el usuario, como ítem de planificación ahí viene lo de gestión ágil, cuando usamos a gestión es que estamos planificando y haciendo seguimiento control o monitoreo, la gestión abarca eso. Cuando uno hace gestión con la US es porque las usas como un medio para saber quien se hizo cargo de una, o en que etapa del proceso esta otra, etc. La gestión ágil es binaria, esto significa es gestión de ceros y unos, no hay intermedios no hay porcentajes de avance. Uno no puede estar medio embarazado, estas embarazada o no, eso es la gestión binaria.

Con los requerimientos pasa lo mismo, llega al final de la iteración, tenes la historia implementada o no. La gestión tradicional usa el diagrama de Gantt va mostrando barritas y porcentajes de avance. La gestión tradicional hace para cada tarea porcentajes de avances. Eso es una mentira porque nunca sabes y mucho memos con esa precisión. Esto es un desperdicio de tiempo, las cosas estan o no estan.

Ese es otro de los fundamentos por lo cual la US tiene que ser chiquitas, porque si nos comprometemos a unas US gorda (la interrumpieron y no retomo el tema, f). La gestión ágil usa tableros, con un principio de la transparencia, una de las cosas para lograrlo es usar tableros donde la información está siempre visible, la visualización del trabajo, saber en todo momento donde estamos parados, que tenemos hecho y cuanto nos falta, ese tablero se configura de acuerdo al framework, la más básica tiene 3 columnas, la del todo, una en el medio que dice doing que son la cosas a la que le equipo se compromete, arrastra a doing y cuando termina entonces la pasamos a la columna de done. El tablero se llama sprint backlog, en cada iteración vos arrancas con un … que se llama sprint planning donde decidís estimando que historias son las que vana a pasar de product backlog al tablero, tiene como configuración básica esas 3 columnas.

Hay otro criterio que se llama definition of done (DoD) esto se traduce de hecho o terminando, de cualquiera de als dos formas se le dice o criterio de hecho o criterio de terminado. Define cuando una historia esta en condiciones de ser mostradas al PO. Intenta contrarrestar este enfoque algunos desarrolladores de terminar de teclear y dicen ya termine, pero no le hicieron pruebas unitarias, no está documentado ni está en el repositorio, el código no tiene comentarios (criterios de mínima), entonces no termino, de ahí es como se arma el definition of done que también es un checklist que te dice que tenes que controlarte a vos mismo, la definición la arma el equipo. El equipo crea y se pone de acuerdo respecto al definition of done y ese tablero esta visible para ese equipo todo el tiempo. No se pueden poner user en el medio de las columnas, o esta terminado o no, si no empecé a trabajar está en el to do, la transparencia es lo que tiene que visibilizar todo esto. Cada equipo tiene su propio DoD. Hay una fuerte recomendación de los equipos agiles que es hacer revisiones técnicas para asegurar calidad del producto, revisión del código, de la historia que las hace un par, se suelen cambiar entre ellos. El código se puede subir al repositorio para poderlo subir al repositorio tiene que tener calidad. Esto viene de mano con el principio de la excelencia técnica, esto no se negocia, la calidad esta relacionada con cuando cumple los requerimientos, el producto tiene valor si hace lo que tiene que hacer y lo hace bien, la calidad es la medida donde se satisfacen las expectativas y necesidades de los clientes, la expectativa es algo que no se manifiesta explícitamente, se espera pero no se exige. El cliente asume que te tenes que dar cuenta que las cosas tiene que estar, fácil de usar estos colores, etc, eso es expectativa. Las necesidades si se manifiestan pero hay que cumplir los dos y ahí entra nuestro trabajo de que las expectativas sean explicitas.

Todo esto era para decirles que si uso user muy grandes, con esto de la gestión binarias, si elegimos una muy grande y se compromete todo el equipo a que va a trabajar en esa user porque va a estar al final de la iteración y no la terminan no se entregan nada, porque el 75% es lo mismo que nada, porque

al final de la iteración vuelve al producto backlog. Si el PO no aprueba en done vuelve al PB con la prioridad que el decida. Todo es por algo, no es caprichoso, las prácticas que aparecen acá compensan otras prácticas que fracasaron en otros momentos

Entonces por eso la S de Small de INVEST, de mantener historias pequeñas.

Que pasa con las spikes, las spikes son cuando el equipo no podía estimar cuanto le iba a llevar, lo que caracteriza a una spike es un nivel de incertidumbre tal que esa historia no se puede estimar, esa E de IINVEST es de estimable, entonces si tengo una user que no la cumple porque no la puedo estimar porque no la puedo imaginar por la falta de información que me lleva a no poder estimar, en ese momento eso se transforma en una spike. Hay dos tipos de spikes que pueden estar presentes en una misma user, uno tiene que ver con la tecnología (spike técnica) que tiene que ver con el cómo y la otra es la spike funcional que vienen del lado del Po, del negocio, hay que trabajar un poco mas con el PO, la funcional tiene que ver con qué va a hacer el producto. Siempre hay algo de incertidumbre cuando trabajamos con software, algo que puede fallar, no obstante eso cuando eso excede los limites aceptables ahí es cuando la user se convierte en un spike, entonces la recomendación más importante es no poner en el mismo sprint backlog la investigación con la user que tenes que implementar como resultado de esa investigación, entonces si decidís que parte del equipo tiene que investigar va la spike ahí pero las US relacionadas no van en ese mismo sprint backlog. Así como una user puede tener spike técnicas y de negocio una spike puede afectar a otras user.

<u>Como se estima en ambientes agiles</u>

También tienen concepciones diferentes de lo que plantea la gestión tradicional respecto a las estimaciones.

Frederick Brooks, el habla de que lo mas difícil de todo en hacer software es decidir cual es el software que quiero hacer. Como las estimaciones se basan en los requerimientos, lo siguiente complicado de hacer cuando uno trabaja en un proyecto de software es estimar porque estimar primero tiene asociado un factor de probabilidad, uno estima cuando hay un universo de incertidumbre, si tuvieras certidumbre no estimas porque sabes lo que va a pasar, uno necesita estimar cuando hace una predicción, tenes que dar un valor en un universo donde te falta valor.

En el software funcionan mal las estimaciones, hay una desviación en la gestión tradicional de los proectos, los exitosos que son solo el 40% el 60% se tira a la basura por problemas de requerimiento la mayor aprte de las veces, ese 40% termina desfasado hasta un 40%. Estos enfoques aparecen cuando hay problemas para resolver, intentan identificar los problemas para corregirlos.

Básicamente lo que nosotros llamamos estimaciones agiles tienen sus características, sustento teórico de la misma manera que los requerimientos agiles. La primera es que son relativas, en contraposición a las estimaciones tradicionales que son absolutas, relativa tienen que ver con que somos mejores los seres humanos comparando, o sea puedo contestarte mejor si esta torre es mas alta que la otra, si el faro este que ha puesto schiaretti en el parque sarmiento es mas alto o no es mas alto que el árbol de navidad que pone canal 12, soy mejor comparando que diciéndote cuantos metros mide el faro. SI yo tengo que decir cuantos metros mide o cuanto pesa ese escritorio, cuantos metros tiene el aula, es más difícil a decir si esta aula es más chica o más grande que la de al lado. Ese es el concepto d estimaciones relativas, hacemos estimaciones por comparación en el enfoque ágil.

Relativas significa que estimamos por comparación, para poder hacer la comparación lo básico y fundamental es que vamos a tener algo contra que comparar. Como foco de las estimaciones agiles en relación y diferencia a las estimaciones tradicionales es que se hace foco en la certeza, no en la precisión debido a que la precisión es cara, porque tengo que invertir recursos, esfuerzo, tiempo.

Cuando hacemos software el cliente tiene una presión muy fuerte, el cuando que seria, si yo contesto cuando con precisión tendría que decir el día, hora, minuto y segundo en el cual dejo el producto en condiciones para que uses, certeza es 6 meses, 1 año, 2 meses, etc. Los enfoques tradicionales hacen énfasis en la precisión, invierten recursos dinero y tiempo en eso pero no lo cumplen porque la mayoría de las veces para llegar a esa precisión hubo que hacer suposiciones consecuencia de la falta de información, esa realidad fallo porque lo que asumí no fue así y ya no puedo. Eso sumado a los pecados originales que son mentiras que mucha veces la gente asume cuando estamos estimando como trabajar 8 horas por día. Por eso las estimaciones agiles (relativas) hacen foco en la certeza.

De nuevo viene el principio de diferir las decisiones hasta el último momento responsable, eso significa que al igual que con los requerimientos no hagamos un esfuerzo en estimar toooodo el producto porque no sabes lo que es producto, vamos haciendo estimaciones a medida que vamos necesitando.

Se estima en dos momentos, hay un momento donde se estima mas general, estimaciones de granularidad más gruesa que se llama talle de remera porque dice que esta remera es S, L, XL a ese nivel las user o las épicas y los temas y también en el product backlog que es lo que se suele recomendar. La priorización se hace con el valor de negocio y la estimación tiene que ver con la complejidad apara construir esto.

El seugndo momento en términos de scurm seria durante el sprint planning, hay una ceremonia que se llama así, en esta ceremonia y que formalmente es la primeara de scrum se reúne el equipo que son todos (po, scrum master y developers) y dicen bueno esta es la primera historia que el PO eligió como prioritaria y la vamos a estimar, acá es cuando el equipo tiene que ver hasta donde vamos a estimar, no vas a estimar todo el PB, solo las necesarias para entrar en el sprint. Este sprint planning tiene dos momentos de estimación, uno cuando estimas las historias y luego estimas como vos pensas llevar a la implementación esas historias. Pasan al todo estimadas. Scrum recomienda esta técnica de estimación llamada Pocket Estimation o Pocket Planning.

La siguiente cosa y final de las estimaciones agiles es que estima el equipo, o sea genéricamente estima el que hace el trabajo. En contraposición a la gestion tradicional donde el que estima es el líder del proyecto y la mayoría de las veces asume compromisos basados en esas estimaciones cuando no tiene nada que ver con la gente que va a hacer le trabajo, en las estimaciones agiles es equipos autogestionados , equipos que van a asumir compromisos en función de cosas que ellos decidieron no que les dieron impuestas, estima el que hace el traba, o sea el equipo.

La técnica de Pocket Planning

Es muy sencilla. La unidad de estimación de las historias se llama Story Point o puntos de historia, el story point (SP) es la unidad. Cuando estimo necesito dar un valor cuantificado de alguna manera, tengo que llegar a un numero, algo que se pueda contar.

Ese SP o punto de historia representa el tamaño de la historia de usuario, cuan grande o pequeña es una historia por eso también esto de los talles de remera es asimilable porque la S es mas chiquita que la L, ese es el ejercicio que nosotros queremos hacer. Pero la técnica de PP usa para los SP la serie de fibonnaci, que no tiene fin arranca con un 0,1,1,2,3,5,8,… paramos así porque si queremos cumplir con el invest model no hay que comprometerse con una de mas de 8, pero esto sigue. No le pongamos 4, 6, hay que usar esos números que mencione antes.

El primer elemento es asignar SP numéricamente usando la serie de Fibonacci, el crecimiento es exponencial al igual que el software en su complejidad, asimilable al crecimiento de esta serie, no es la única pero podemos usar cualquier serie que tenga un crecimiento exponencial, no lineal. Lo siguiente que hace la técnica esta de PP es darle valor o reconocimiento o la sabiduría de las

multitudes, en vez de solo estimar en base a la estimación que hice yo, la recomendación es que estime el equipo y aprovechamos la visión que cada uno tiene que es distinta y complementaria y niveladora de la visión que tienen los demás. Se estima en forma colectiva. Luego el siguiente elemento que nos hace falta para aplicar esta de la relativización es encontrar algo que se llama US Canónica, es el elemento que nosotros vamos a utilizar para hacer la comparación, se elige una y contra esa se comparan las demás.

En la versión original había cartitas, cartas, cada persona que iba a participar de la estimación tenía un juego de cartas que tiene los números de la serie de Fibonacci, un infinito y un signo de pregunta.

Cada persona tiene sus cartas, se elige la primera user que queremos estimar, ahí esta el PO presente por las dudas tengamos cierta duda de cómo quiere lo que quiere, el equipo hace las preguntas necesarias, cuando vamos a estimar estimamos para cumplir con el criterio del definition of done porque estoy estimando lo que necesito para implementar esta historia para poder decir termine, porque la mayoría de equipo solo estima el tiempo de programación y nada más y eso esta mal porque vos la podes tener programada pero eso no significa que esté terminado, el DOD es lo que usa el PO para ver si esta terminado o no la US.

Cada uno de nosotros cuando termina la ronda de preguntas con respecto a esta user y tenemos una visión compartida de lo que es, cada uno apuesta y la pone a la carta boca abajo sobre la mesa y estimo que esto tiene este peso y no la muestra hasta que todos terminaron de hacer su estimación individual y estan todas las cartas ocultas, esto es para que nadie se influencie por las decisiones de gente con mas influencia.

Hay otra característica que pide scurm que se llama apertura de que cabe la posibilidad de que estes qeuivocado, en estos momentos uno tiene eque estar permeable a que nos podemos equivocar, tenemos que poder hacer ese ejercicio no esta cosa segada de que puse 5 y es 5 porque eso no construye.

Una vez que todos estimamos hacemos visible la estimación y todas las cartas estan visibles y ahí hay que empezar a explicar, normalmente la explica el que puso el valor más alto, los extremos primeros. Uno que puso 8 y otro que puso 2, yo creo que son 8 puntos por esto esto esto. El que puso 2 hará lo propio y después los demás y entonces nos escuchamos y cuando todos terminamos en función de lo que escuchamos hacemos otra ronda individual pero esta vez con la permeabilidad de considerar lo que no considere, en la siguiente vuelta misma mecánica y ahí es donde se supone que debería haber convergencia, la mayoría de las personas deberían enfocarse hacia el mismo valor de la historia. La técnica no tiene escrito cuantas iteraciones hasta converger, lo aceptable son 2, si el equipo esta acentado necesita 1, las estimaciones tienen que ser convenientes, no nos puede ocupar un día entero la estimación, por eso mas que 2 no es tan recomendable. Esto es lo que tenemos que determinar, si después de 2 iteraciones tiene que haber alguien que diga que se toma la que mas se repite, o se toma el promedio y se avanza porque eso se hace para cada historia que tenemos que estimar por eso no podes estar 5 horas.

La canónica juega porque cuando hacemos esa estimación individual tenemos que pensar en la que se definió como canónica decir esta es, cuantas veces mas complejo mas grande que esta otra que estoy comparando. La gente muchas veces cuando uno no tiene mucha experiencia elige una canónica de 1 porque es lo mas fácil, sencillo chiquito, sin falta de información, ponen la user que me representa eso, no importa que no se implemente en este sprint, elegis una unidad de comparación. El problema de la canónica de 1 es que te puede aparecer una cosa mas sencilla que esta, por eso algunos agregan el medio por las dudas.

Por eso otros equipos usan una canónica de 2 para un equipo que no tiene experiencia es más difícil de determinar que una de 1, o lo que se hace que le gusta a meles es una de 3 que te da margen para abajo y para arriba. Canonica de 1 es lo más fácil, sencillo, conocido, certero que puedo encontrar dentro de la US que tengo que implementar.

Entonces esta famosa SP resulta que se puede analizar, las descomponemos en 3 dimensiones para ayudarnos a signar ese valor numérico y esas 3 dimensiones que tiene un SP son complejidad, esfuerzo, duda (incertidumbre).

Complejidad cuan dificultosa es esta característica que quiero construir, siempre tengo que hacer ese ejercicio de comparación. Como el número que tendría que asignarle a esa user es enorme eso hay que partirlo en cosas más sencillas.

El esfuerzo es trabajo, cuantas horas ideales necesitamos para crear esto, no es tiempo. Tiempo es calendario día mes año, esfuerzo es horas ideales, horas ideales significa horas sacando distracciones, horas efectivas de trabajo cuanto realmente estoy trabajando. Se mide siempre en horas ideales lineales persona, bolsa de horas. Cuantas horas reales necesito para construir esto.

Duda es el nivel de desinformación que tengo relacionado a esta historia, hay veces que lo tengo muy claro y otras que no, cuando esto crece y se hace una bola muy grande se hace una spike.

El SP es un punto que homogeniza estas 3 dimensiones porque puede ser que una US tenga mas o menos de cada uno de los factores teniendo el mismo peso que otra por razones distintas. Entonces como esto de manera individual es difícil de comparar nosotros necesitamos un valor de homogenización y ese numero es el SP, yo puedo agarrar una user de 3 y otra de 3 de otro lado y contra la canónica decidí que estas son 3 veces más complejas, tienen un tamaño 3 veces mayor porque la canónica es de 1. El por qué se justifica en base a estas 3 dimensiones.