Ingeniería y calidad de software – Toma de notas en clase

17/8/23

Procesos Empíricos

La ingeniería de software tiene que cumplir conocimientos de los procesos proyectos y de producto. Estamos incorporando dentro de la disciplina de ingeniería de software el elemento mas importante que son las personas. Hay una corriente que tiene unos a los de existencia pero recién ahora y con el agilismo y con el movimiento del managment 3.0 se formalizo que el peoplement es una parte esencial de la ingeniería de software porque somos una profesión humano-intensiva, eso significa que lo más importante para hacer software es la materia gris. Las personas tienen un rol fundamental decisivo del éxito o fracaso de cualquier proyecto. Hay corrientes que se han popularizado en este último tiempo que hacen un reconocimiento que tiene la gente en el contexto del software.

Dentro de los procesos que hemos estudiado, venimos estudiando procesos definidos, como el PUD, son definidos en contraposición a los empíricos de los cuales hay dos corrientes que los adoptan y utilizan que son los enfoques de filosofía ágil y Lean que su traducción es liviana pero usamos Lean. Tanto el enfoque Ágil como el lean forman parte de lo que se conoce como procesos empíricos, los promueven y utilizan.

Los procesos definidos intentan ser la expresión de completitud, de todo lo que nosotros necesitamos para hacer un producto de software que este escrito ahí, que diga quien lo va a hacer, que entradas necesito y que voy a generar d cada actividad y de acuerdo a las organizaciones y a las costumbres las métricas, herramientas asociadas a las actividades del proceso y está definido de antemano a nivel organización y p por gente diferente de la que va a hacer el trabajo. Esas serían las características de un proceso definido, y nos acercamos bastante con el PUD.

Desde el surgimiento de la corriente AGIL, la que se puso mas cercana a nuestra industria fue primero el movimiento Ágil que nació en el software, el LEAN no nace en el software sino en el automotriz, se conoce como Toyota System y después se adapta al software. El agilismo surge en el 2001 donde hubo un ´primer acuerdo formal y de ahí surge el manifiesto ágil

El PUD se malinterpreta porque nunca intento ser una receta de cocina, que el proceso se adapta al proyecto, ese se adapta es pensar lo que le conviene al proyecto, la clave es que venia definido desde afuera el proceso entonces el movimiento ágil lo que dice es que hay un afuera muy grande que dice que el que piensa el software es el que lo hace, no queremos recibir estimaciones impuestas que ocurría en la gestión tradicional.

El empirismo es basado en la experiencia la experiencia como se consigue? Porque si es la primera vez que trabajamos en un equipo y no tenemos experiencia no podemos hacer un proceso empírico? No, entonces el proceso dice que tenemos que conseguir la experiencia. Los procesos se basan en 3 pilares, 1 es inspección, 2 adaptación, 3 transparencia.

Hay una cosa que dijo Manuel que es cierta y es que los ciclos de desarrollo tienen que ser cortos, todos los procesos empíricos se basan en ciclos de vida iterativos, del tipo que quieran pero iterativos, después aparece cada marca, cada framework. Ágil no es una metodología, es una filosofía, un pensamiento y de ahí se derivan frameworks muchos y ninguno es un proceso completo, no lo son porque casualmente estan tratando de pelear contra los otros que si lo son. Entonces lo que hacen esos frameworks es darte lineamientos, algunas buenas pautas para lagunas cosas. Scrum te da pautas para gestionar el proyecto por ejemplo. No arman un proceso completo, no podemos decir que Scrum sea un proceso o una metodología.

El empirismo sale y dice yo como equipo acá quiero decidir que proceso voy a usar y entonces el proceso lo definen con la intención de lo menos que el equipo necesite para funcionar y lo que hace aferrándose a ciclos de vidas iterativos. Un proceso definido te recomienda un ciclo de vida iterativo pero podes elegir no usarlo y echarle la culpa al PUD. En el empírico no hay opción a elegir, los empíricos si o si tienen que usar ciclos de vida iterativos debido a como yo gano experiencia, que es debido a la realimentación rápida, frecuente. Ciclos de iteración cortos donde yo controlo (inspección) determino si tengo que corregir algo, es decir me adapto y entonces estos ciclos cortos de inspección adaptación hace que yo pueda retroalimentar y en esa retroalimentación esta la experiencia, ahí aprendo no importa si cuando arranque no sabía nada porque de hecho cada proyecto es distinto de otro porque puede se otro producto, otro cliente, otro equipo y cualquiera de esas cosas, cada vez que arranco un proyecto la experiencia se gana en el contexto de este proyecto con este equipo cada uno es uno y sus circunstancias Cada proyecto tiene su particularidad y tenemos que aprender de esa particularidad y por eso necesitamos ciclos cortos de realimentación.

Transparencia es fundamental para hacer funcionar esto, la palabra transparencia quiere decir que es visible para todo el mundo, no hay nada oculto, la transparencia dice no escondamos las cosas abajo de la alfombra, la información es de todos, no es mi código, es del equipo, me hago cargo de hacer una tarea y voy a blanquear mi situación si tengo problemas pido ayuda, esas son las bases de las que habla la transparencia, claridad, el objetivo tiene que ser visible para todo el mundo, la situación de avance tiene que ser visible para todo el mundo todo el tiempo. Si no hay transparencia lo anterior no funciona. Por eso estos son los 3 pilares del empirismo.

Una infografía es un elemento que comunica pero con dibujos, gráficos y con poco texto.

Manifiesto agiles y sus valores

Valoramos lo que esta arriba pero lo valoramos fuertemente que algo que está bajo pero no significa que lo de abajo no lo valoramos sino que lo valoramos menos y ahí es cuando empieza a haber unos choques de los empíricos con los definidos porque los definidos son estrictos respecto del respeto y cumplimiento del proceso y el agilismo valora mas las personas y el vinculo que ese establece entre ellas que los procesos y las herramientas. Valoramos mas el software funcionando que tanta cantidad de documentación completa, extensiva y sobretodo desde el principio el proyecto y ahí viene la gran confusión que todos llegan al año que viene y todos eligen SCRUM para no documentar pero en ningún lado dice que no haya que hacerlo sino que eso lo decide el equipo.

Yo como enfoque necesito valorar lo que valora el cliente que es el software funcione, entonces nos alineamos a entregar software frecuente que es lo que quiere el cliente, iteraciones cortas que generar productos que se pueden poner en producción, no es un prototipo es una versión del producto.

El responsable de escribir las user stories el producto owner, que es alguien que conoce suficientemente el producto para saber que va a hacer el producto en el negocio, no es un rol que este en el equipo de software, son gente del negocio, son los no técnicos del manifiesto ágil, el producto owner tiene que poder conocer su negocio, sus necesidades y tener capacidad de decisión para poder tomar las decisiones ahora y en el tiempo futuro, se llama priorización, es importante decidir que se hace primero y que se hace después. Es un representante del cliente, tiene que hacer un trabajo importante el negocio para elegir a esa persona porque el éxito de un proyecto de desarrollo ágil tiene que ver con este rol.

La base de los requerimientos esta sustentada en la comunicación cara a cara y que de las mejores arquitecturas, requisitos, requerimientos y diseños emergen de equipos autoorganizados, esto está en LEAN y AGIL. Tiene que ver con hacerme cargo, tomar decisiones y comprometerme con esa decisión

porque el equipo espera eso de mí y lo necesita. El agilismo funciona con equipo autoorganizados, autogestionados donde no hay un jefe que asigna trabajos, sino que hay gente que es buena y se dedica a algo que es bueno y asume el compromiso. Emergen tiene que ver en contraposición de esta definición completa de los requerimientos que se espera en los enfoques definidos. Acá dice que lo mejor es que vaya surgiendo, los mejores requerimientos van a salir de ahí de ese vínculo que se establece entre el PO y el equipo. Hay un porcentaje de requerimientos que se llaman requerimientos emergente, que significa que aparecen mientras el producto está naciendo, creciendo y evolucionado, no es que el PO no lo dijo antes porque no se le ocurrió porque muchas cosas hasta que no ve algo no se da cuenta que necesita todo esto y eso son más o menos el 50% de los requerimientos de un proyecto, el resto está entre los conocidos y no conocidos obtenidos de las preguntas hechas al PO y las respuestas que este dio, esos son los conocidos y los desconocidos son el margen de error por no haber preguntado.

Que es AGIL

Es ese paraguas con los valores y principios que nos van a permitir desarrollar software de mejor calidad, no es una metodología o un proceso. Lo que quiere hacer es un equilibrio entre demasiado proceso y nada de proceso que pasaba antes de los procesos definidos, era darse la cabeza contra la pared o me arreglo con esto pero eso no significa que no sea ordenado o riguroso, el agilismo es mas estricto con las pocas prácticas que pide con respecto a su forma de ejecutarse, es exigente en ese sentido.

No es como dice mi colega, el agilismo no es la legalización del quilombo.

Requerimientos en AGILE

Nosotros una de las primeas cosas que queremos entendiendo que vamos a hacer desarrollo de a poco, una de las cosas que nosotros queremos incorporar en nosotros es que nosotros no hacemos líneas de código pero lo importante no es el software sino el valor de negocio que entregamos a nuestro cliente. El software no funciona en el éter sino inserto en un negocio y si nos contratan es porque quieren generar valor, no es un fin en si mismo, es un medio para un fin.

Al escribir historias de usuario vemos los requerimientos un nivel mas arriba porque el caso de uso definía requerimientos de software, las user stories estan arriba en un nivel de abstracción más alto a nivel de usuario y negocio entonces la descripción de la historia tiene que ser corta y segundo apuntar a describir una necesidad que tiene el usuario en términos del usuarios con terminología del usuario y siempre recuerden que en la medida de lo posible lo escriba él.

Otra de las cosas que nos cuesta es la priorización, es entender que lo bueno es suficiente, cuando parar? Hay que aprender a terminar cuando es bueno y suficiente para entregarle al cliente en esta vuela.

El 45% del software hay características que no se usan nunca, el 7% se usa siempre y 13% usualmente. La inmensa mayoría de los productos tienen características que no se usan nunca. Acá se cumple lo de la curva de Pareto que el 80% del valor esta en el 20% de las acciones, tenemos que construir juntos con el PO ese 20% y lo mostramos realimentando y aparecen esos emergentes. Eso es la adaptación, adaptarse a esos emergentes, donde debemos tener al cabeza predispuesta a hacer cambios cada vez que hagan falta, los cambios son bienvenidos aun en etapas finales del proyecto.

Con esto de que el producto se entrega por partecitas aparece la importancia de la priorización que la tiene que hacer el que sabe, es decir, el PO que va a decir lo más importante o no para el negocio, su ámbito de trabajo y responsabilidad esta sobre la base de un artefacto llamado lista priorizada de

producto, es decir, un Product Backlog. Se llama cola priorizada porque lo más importante se ubica arriba y hacia abajo lo que es menos importante y el dueño de este artefacto es el PO y puede hacer lo que quiera, como agregar o quitar características o cambiarle su prioridad.

Otro de los conceptos que hay que incorporar es Just In Time que viene a pelear contra la definición completa y de antemano de todos los requerimientos y dice que vamos a descubrir, analizar y describir requerimientos a medida que haga falta. NO voy a describir los requerimientos al inicio sino a los que les voy a entregar al cliente en esta iteración entonces analizo solo cuando necesito y de esta manera se agrega el concepto de eliminar los desperdicios que es uno de los principios fundamentales de LEAN que tienen que ver con no escribir requerimientos que van a cambiar porque sino todo tu trabajo de descripción se fue a la basura entonces se hace lo que necesitas en el momento.

Es mas eficiente si podemos estar en equipo colocados, significa que todos trabajamos en el mismo espacio de locación. La razón es porque en el 100% de la efectividad de la comunicación lo que uno dice tiene una incidencia del 17% el otro 83% son las formas, como posturas, tono de voz, etc. El contenido aporta en la comunicación solo el 17% si mandamos un mail al cliente y el cliente contesta perdieron el 83%.

Los requerimientos de negocio y requerimientos de usuario son los user stories. Los casos de uso son requerimientos de software.

Esto lo llevamos adelante de la siguiente manera. Esto se llama triple restricción o también se lo conoce como el triángulo de hierro pero en definitiva hablamos de las dimensiones en las que se mueve un proyecto en función de la decisiones que tenemos que tomar cuando trabajamos.

Primero alcance, luego costos que se derivan de los recursos y el tiempo, entones cuando nos dan una responsabilidad por un proyecto como manejamos esas variables, que dejamos fijo y que negociamos es la calve de éxito de nuestro proyecto. Si nos meten en un proyecto donde las 3 estan condicionadas para que me queres a mi dónde no tengo ningún margen para tomar decisiones.

El tradicional fijan los requerimientos al principio y a partir de esos requerimientos determinan los recursos que hacen falta siendo el más importante le esfuerzo, es decir las horas de trabajo de la gente que son el 80% del valor del producto.

Los recursos que se derivan en costos se determinan desde los requerimientos. Si cambian los requisitos debería recalcular, pero el cliente no te deja. Para eliminar esto aparece el agilismo donde deja libre el producto backlog pero en cada iteración del desarrollo voy a dejar fijo los recursos que vendría a ser el equipo y voy a dejar fijo el tiempo, es decir, voy a apuntar a iteraciones de duración fija. La iteración dura lo que el equipo decida, pero lo que se decide no se cambia. Se acuerda que vamos a entregar de requerimientos con este tiempo y estos recursos, se deriva el alcance. La priorización es fundamental. El tiempo de fin no se modifica, es fijo, SCRUM le llama Time Box sino porque el tiempo está encerrado en una caja, la fecha no se corre.

User stories

Apuntan a tratar de contando una historia resolver el problema histórico y viejo de un articulo No silver bullet. Ese artículo que tiene muchos años era de un señor que trabaja en IBM haciendo hardware y lo ponen a gestionar Software donde cuenta todas las cosas que hizo mal, porque fracaso su proyecto de software.

Entre otras cosas lo que dice este señor es que la parte más difícil de hacer software es decidir que software quiero construir. Lo más difícil es los requerimientos. Aparecen técnicas, y le damos vuelta y

no hay caso porque es un proceso social y comunicacional, todo tiene que ver con el vinculo que establece el PO y el equipo para ver si logramos una visión compartida de lo que hay que hacer.

En este contexto aparecen las user stories, que no resuelve todos los problemas que tenemos ya que eso no existe pero la clave es incorporar la mayor cantidad de herramientas que podamos. Las 3 partes de una historia de usuario son esas 3 que estan ahí que en ingles son las 3 C. Card, Conversation y Confirmation.

De esas 3C, la mas importante es la conversación. La conversación queda en algún lado? Si. La parte visible, permanente, que queda reflejada son las otras 2, la tarjeta y la confirmación. Este método apareció con las tarjetas bibliográficas.

En el frente la tarjeta y en el dorso la confirmación, que es el concepto teórico y las pruebas de aceptación es lo práctico.

La sintaxis de la user stories, tienen 3 partes, tiene que contestar quien, que necesitas ese quien y por qué, para que lo necesita. Fundamentales, si tuviéramos que apostar la mas importantes la de valor de negocio, que esta reflejado en el why, o sea en el porqué.

Como docente yo quiero puedo necesito registrar las notas del parcial de forma tal que informe a mis estudiantes su situación académica, eso ultimo es el valor de negocio. Lo describí de tal forma que nunca dije como, no mencione tecnología, nada de software, lenguaje de negocio entonces le estamos dando la libertad al PO de que exprese su necesidad.

Como conductor quiero buscar un destino a partir de una calle y altura para poder llegar al lugar deseado sin perderme.

La frase verbal es como el nombre corto, el nombre corto tiene que ver casualmente con el que y la idea de esto es poder buscarlas más rápido o registrarla en alguna herramienta de seguimiento pero no reemplaza a la user. Las user stories es producto, describe partecitas del producto que nosotros tenemos que construir, entonces si voy a una empresa y veo una user stories que di e que tengo que migrar la base de datos eso no puede ser una user stories, eso es un atarea, esta a nivel de proyecto, es parte de lo que tengo que hacer en el precio para que el producto tenga la calidad que necesito.

En el product backlog se acomodan las stories para saber la prioridad de cada una. Es físico, uno abajo de la otra y asumo que la de arriba es la que queres primero. Esto es el origen. La user storie tiene que ser corta en el frente tiene que ir eso que dijimos antes, si no entra en la tarjeta estamos mal…

La parte de atrás de la tarjeta la parte de confirmación.

Las user stories son multipropósito en el sentido de para que las puedo usar en un proyecto, las puedo usar para acordarme que tengo que hablar de esto, cuando hay que estimar también son un ítem de planificación porque se gestiona a través de user stories el SCRUM, y fundamentalmente expresan una necesidad que tiene nuestro PO respecto de algo que tiene que estar en nuestro producto, por eso las US son de producto

El PO es el que prioriza las historias, hay niveles de granularidad distinto en las historias y puede que haya historias grande o chica según el equipo en función de si la puede terminar en una iteración entonces tiene muchos depende. Cuando son muy grande y no se puede terminar en una iteración se llaman épicas. Tenemos formas de acomodar las épicas e historias en lo que se conoce como temas, se acomoda abajo en el producto backlog.

Una cosa fundamental de que esto tiene que ser si o si para que funcione es que las user stories tiene que ser con porciones verticales significa que le tengo que entregar a mi PO software funcionando, esto significa que cada una de estas es chiquita pero es una unidad funcional, de esta manera tenemos que entregar las piezas de software de manera que puedan ser consumida, si la user storie la corto horizontalmente no puedo entregar algo que necesite el usuario funcionado, por eso se corta verticalmente.

El modelado de roles es importante, es el único lugar donde tenemos permitido poner usuario como nombre. Es una maqueta que explica las características de este usuario y que va a ser importante para diseñar la experiencia de ese usuario. Hay varias técnicas que ya vamos a estudiar.

Uusarios representantes

A veces es necesario, aunque cada vez que podamos lo tenemos que evitar y sor los Proxies Product Owner que aparecen cunado el PO que sabe no está disponible, ahí es donde viene la figura de los usuarios representantes o los proxies po.

Hay algunos roles de personas que podrían funcionar como usuarios representantes, pero ninguno es un rol técnico de software, porque el PO es del negocio en cualquier caso. Entonces esa asociación que hacen las empresas de meter analistas funcionales como PO es un gran error.

Criterios de Aceptación de User Stories

Son una puesta en blanco sobre negro, una formalización de las cosas que el PO exigen que estén implementadas en esa user, y nos sirven para definir las pruebas de aceptación. Entonces para bajarlo a algo concreto, presento la user que vimos recién del GPS donde estaba el conductor y el destino.

Decimos la altura de la calle es un número y podríamos ser un poco mas fino al decir que es un numero natural. Estos criterios pueden ser cosas muy sencillas, destierren la palabra obvio porque nada es obvio.

Otro es que la búsqueda no puede demorar más de 30 segundos es un criterio de aceptación, lo acepta si respeta eso, sino no lo acepto. De acá se derivan las pruebas de aceptación, debiendo tener como mínima una que la cumpla y otro que no la cumpla.

Cada vez que el usuario se pueda equivocar lo va a hacer, por tanto hay que pensar en todos los casos de error. Estan escritos en la user y son de negocio, tiene que estar expresado en un termino que le hubiera salido a usuario.

Entonces básicamente los detalles que nos hacen falta no estan en la card, no tiene detalles. Gran parte de los detalles está en la conversación y la otra parte esta en las pruebas, hay un enfoque bastante fuerte en esto del agilismo de movernos a las pruebas porque deben ser detalladas porque sino no sirven, si no hago un caso de prueba detallado no sirve para probar el sistema.

Primero las derivamos de los criterios y segundo expresan detalles que no estan expresados en al card y nos ayudan en el momento de la implementación de la card a si se acepta o no se acepta. Al final le aclaramos si esta prueba tiene que fallar o no. Entonces si leo la user derivo todo porque charle con el usuario y los detalles quedan reflejados ahí.

En algunos prácticos vamos a tener que escribir historias y para hacerlo debemos aplicar la técnica donde la conversación es importante y tenemos que preguntarle al profe que funcionan como PO lo que tenemos que hacer, ergo, pregúntenles porque nosotros no preguntamos y después nos quejamos de que el PO no acepta.

Se hace un acuerdo entre el equipo y la PO de si al momento de presentar la US todo pasa, pruebas exitosas (falla donde debe y va bien donde debe) entonces el PO me la acepta. Se debe hacer un acuerdo porque el testing no es exhaustivo, por el tema de costos beneficios.

Definición de listo

Formas de controlar la calidad, de lo que hacemos. Este primer criterio que se llama DoR (Definition of Ready), definicion de listo. Esto se aplica a la user, cuando esto es OK, significa que esa user esta en codniciones de entrar a una iteración y ser implementada, si no lo cumple pues queda mas abajo en el producto backlog hasta que se cumpla.

La llave de que salga de un producto backlog y entre a una iteración es cumplir este criterio y lo define el equipo. Es un check list, tenes esto, esto, esto, ese check list tiene una base de mínima, una cuerdo mínimo. Cualquier definition of ready tiene mínimamente lo que se llama invest model.

De mínima una user para cumplir con el criterio de ready debe cumplir el modelo Invest.

I es de independiente, significa que cada US de una porción vertical se puede implementar en cualquier orden definido por el PO, eso significa que esta US no depende de ninguna otra.

N es de neogicable, singifica que la historia está escrita en términos de que quiero, no de como. No hay que dejarlo al PO que diga como tiene que implementarse, eso no se escirbe en una user, el como queda para negociarlo después

V es de valuable, es la parte del por que o para que de la user storie, si no lo tiene no cumple con esto, tengo que establecer el valor de negocio en la US.

E es de estimable, hay veces que una historia es tan compleja o tiene tanta incertidumbre o hay tantas cosas que no tiene respuesta no la puedo estimar, si no la puedo estimar no es una user sino un spike, es el proceso que pasa cuando nos damos cuenta de que es más lo que no sabemos que lo sabemos. Cuando no se puede estimar no es una user, se transforma en un spike. Necesito ese valor para ver si me puedo comprometer a desarrollarlo o no, esa incertidumbre puede venir de parte del negocio o puede ser tecnica. El spike también se la conoce como item de investigación, es la traducción mas parecida. Es estimable cuando puedo asignarle un valor numérico, si no cumple no anda bien con el definition of ready.

S es de pequeño es consumible en una iteración, iteración que scrum llama stream. Esto signficia que tengo que empezar y terminar la user en esa iteración por eso debe cumplir con ese tamaño de small. Hay patrones para partir US para que cumpla con esto y ahí esta el riesgo de las historias tan cortas para cumplir con esto pero no tiene valor de negocio.

T es de testeable, que significa que puedo demostrar que esa user se implementó en los términos que el producto owner se necesita.

A esto también se le pueden sumar otros criterios como tener un prototipo, pero eso viene dado por el equipo.

Algo más sobre las user stories

No son una ERS, no es requerimiento de software, no es lo mismo que un caso de uso y no hacemos juicio de valor de cual es mejor o peor. El producto backlog no es lo mismo sobre la ERS.

Diferentes niveles de abstracción.

Una US es una descripción corta de una funcionalidad, la épica es una user storie larga que se puede descomponer en varias US. El tema o tema de inversión es una colección de US relacionadas. Esta clasificación no es rígida.

Las historias son las que entran al proceso de construcción luego de haber refinado ideas objetivos, épicas para obtener estas historias.