U2: GESTIÓN LEAN-ÁGIL DE PRODUCTOS DE SOFTWARE

Temas:

- ✓ Manifiesto Ágil/Filosofía Lean
- ✓ Requerimientos en ambientes lean ágil
- ✓ Introducción al Desarrollo Ágil.
- ✓ Requerimientos en ambientes ágiles – User Stories
- ✓ Estimaciones en ambientes ágiles
- ✓ Frameworks de SCRUM a nivel equipo y escala
- ✓ Métricas Ágiles
- ✓ Gestión de Productos de Software - Planificación de Productos - Herramientas para Definición de Productos de Software
- ✓ Lean UX
- ✓ Design Thinking

MANIFIESTO ÁGIL

INDIVIDUOS E INTERACCIONES sobre procesos y herramientas
(Es más importante la interacción (y vínculos) constante con los clientes y entre los miembros del equipo que un proceso muy estructurado.)

SOFTWARE FUNCIONANDO sobre documentación extensiva
(Es importante la planificación por sobre el plan, y que las decisiones queden documentadas; Ejemplo: la arquitectura. Además, no toda la documentación debe permanecer mucho tiempo)

COLABORACÓN CON EL CLIENTE sobre negociación contractual
(En qué punto hay fricción entre cliente y trabajadores; esa fricción está en los cambios de los requerimientos; las tres dimensiones: tiempo, costos, alcance; El cliente debería estar involucrado en el trabajo ágil; Es importante que el contexto donde trabaja el equipo sea también ágil; Los clientes van aprendiendo y entendiendo el proceso)

RESPUESTA ANTE EL CAMBIO sobre seguir un plan

Se valora mucho más lo marcado EN CAPITAL que lo de la derecha, pero los dos son importantes.

Principios del Manifiesto Ágil

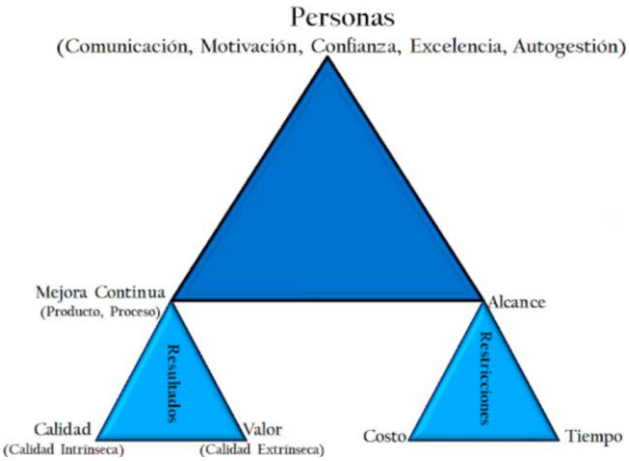
1) Nuestra mayor prioridad es *satisfacer al cliente* mediante la entrega temprana y continua de software con valor
2) Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3) Entregamos Software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4) Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5) Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6) El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7) El software funcionando es la medida principal de progreso.
8) Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida. (Hay que lograr que el equipo de trabajo tenga un ritmo constante para tener mejor previsibilidad de cuánto SW es capaz de entregar en cada iteración)
9) La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad ("La calidad del producto no se negocia"; Hay que ajustar, por ejemplo; la cantidad de SW entregado, pero nunca la calidad. Se asume también que los equipos están lo suficientemente capacitados para hacer el trabajo)
10) La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.(Simplicidad="cantidad de trabajo no realizado"; Entregar solo lo que se pide y no extra)
11) Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados. (Se refiere a que la toma de decisiones la debe hacer el equipo; La calidad emergen conforme el equipo se familiariza con el producto)
12) A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para la continuación ajustar y perfeccionar su comportamiento en consecuencia

Restricciones respecto a los proyectos en el enfoque ágil

Triángulo de Hierro tradicional: Hay tres dimensiones a tener en cuenta: el alcance, costo y tiempo

Triángulo ágil: El anterior pasa a ser la pata derecha de otros dos aspectos: Valor (Calidad Extrínseca y la más importante) y Calidad (Valor Intrínseca)

Versión final del triángulo: pone a las personas como lo más importante. Mejora continua (Traingulo anterior), y el Alcance (primer triángulo) son las patas de este último triángulo.



GESTIÓN ÁGIL DE REQUERIMIENTOS

La gestión ágil cambia el enfoque de la gestión y desarrollo de los requerimientos.

Pilares (o enfoques) de los Requerimientos Agile:

- Hace mucho énfasis en construir valor de negocio. Es decir; construir el producto correcto que genere ese valor.
- Los requerimientos se descubren de a poco y hay que empezar con lo mínimo necesario para poder empezar el proceso. Después se va refinando.
- El desarrollo de Requerimientos está enfocado a construir junto con el cliente (técnicos y no técnicos trabajando juntos), que tiene el conocimiento del producto y trabaja con el equipo. Esto se hace usando historias y modelos para construir el producto.

El segundo pilar se justifica en que en la realidad solo el 7% de las características del SW lo usa el cliente siempre. Toda la otra funcionalidad se usa en mucha menor medida.

Para no agregar funcionalidad sin sentido la gestión ágil agrega el concepto de que el dueño del producto (Product Owner), quién es el que mejor conoce sus necesidades, determina el orden de los requerimientos qué el considere más importantes en el Product Backlog. El Product Owner debe estar disponible para funcionar. Va en línea con el principio de "Just In Time". Esta agilidad en cuanto a la comunicación hace no necesario la documentación extensiva.

El principio de comunicación cara-a-cara se justifica en:

El objetivo final es entregar valor al cliente que se entrega a través de la característica del Software.

Diferencia respecto a la Gestión Tradicional

En el enfoque tiempo-alcance-recursos; el tiempo y los recursos se mantienen constantes y es el alcance el que se estima y cambia.

Tiempo: dos semanas a un mes como lo definen los métodos ágiles (Son iteraciones de duración fija)

Recursos: Es el equipo de trabajo con una determinada capacidad más los recursos materiales y herramientas, que se mantiene.

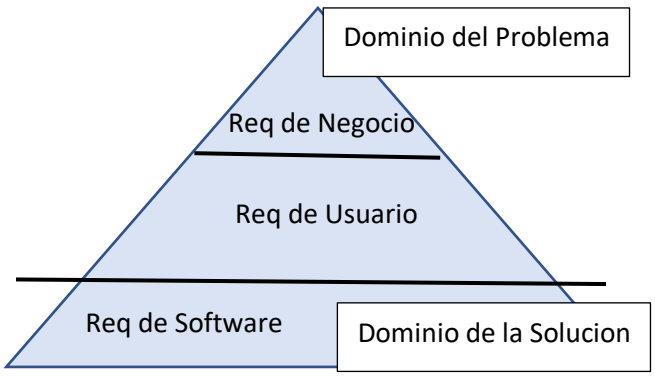
¿Cuánto de este producto se puede construir en tal cantidad de tiempo con este equipo?

Gracias a que el cliente puede cambiar la prioridad de los requerimientos, es como el equipo planifica las iteraciones.

Tipos de Requerimientos

La gestión ágil trabaja a nivel de Requerimiento de Usuario alineado al Requerimiento de Negocio (y el gestor del proyecto de mantener esta alineación). El Product Owner es el encargado de identificar las User Stories. Así se entiende las necesidades del negocio que es lo más importante.

El requerimiento del usuario (en formato US) debe resolver el Requerimiento de negocio correspondiente.



A tener en cuenta:

- Los cambios son la única constante
- No todos los Stakeholders están presentes. El único Product Owner los representa y debe trabajar con ellos para poder ayudar al equipo.
- Siempre se cumple eso de que: "El usuario dice lo que quiere cuando recibe lo que pidió".
- No hay técnicas ni herramientas que sirvan para todos los casos.
- Lo importante no es entregar una salida, un requerimiento, lo importante es entregar, un resultado, una solución de "valor".

Principios relacionados a los Requerimientos ágiles

1- La prioridad es satisfacer al cliente a través de releases tempranos y frecuentes (2 semanas a un mes)

2 -Recibir cambios de requerimientos, aun en etapas finales

4 - Técnicos y no técnicos (Product Owner) trabajando juntos todo el proyecto.

6 - El medio de comunicación por excelencia es cara a cara

11 - Las mejores arquitecturas, diseños y requerimientos emergen de equipos autoorganizados.

USER STORIES

"(...)se las llama 'stories' porque se supone que Ud. cuenta una historia. Lo que se escribe en la tarjeta no es importante, lo que Ud. habla, ¡sí!" - Jeff Patton

Es una descripción corta respecto a una necesidad que tiene el usuario.

La parte más difícil de construir SW es decidir precisamente qué construir. No es lo mismo que el Usuario haya decidido una necesidad a que el equipo lo decida. La responsabilidad recae en distintas personas.

Partes de las User Stories

- Conversación: es la parte más importante y no se registra toda.
- Tarjeta: Es la parte que se registra de la conversación y la parte visible. Es el soporte físico.
- Confirmación: Se escriben al dorso de la tarjeta y son las pruebas de usuario que son necesarias para que el Product Owner acepte una determinada característica de una User Story.

Forma de Expresar Historias de Usuario

<Frase Verbal>

Como *<nombre del rol>*, yo puedo *<actividad>* de forma tal que *<valor de negocio que recibo>*

Contesta el Who, What & For What.

El **rol** representa a quién está realizando la acción o quién recibe el valor de la actividad (No siempre de Rol "Usuario").

La **actividad** es la acción que realizará el sistema.

El **valor de negocio** comunica por qué es necesaria la actividad

La **Frase verbal** permite ponerle un nombre a la User Story que puede ser usado en conversaciones. (No es obligatorio)

Ejemplo:

Buscar Destino por Dirección

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

Las historias son:

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

No es una especificación de requerimientos ni tampoco reemplaza a la ERS.

El Product Owner es el que prioriza las User Stories en el Product Backlog.

Las User Stories abarcan esto (en el patrón de capas):

De esta forma pueden entregar una funcionalidad (Si fuese horizontal no puede generar valor)

Modelado de Roles

Se describen perfiles de usuarios que van a usar el software y son protagonistas de las User Stories.

Usuarios Representantes o Proxies

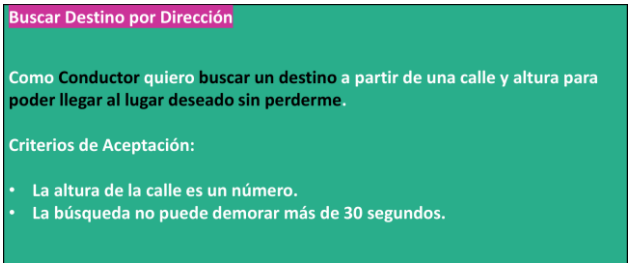
Son candidatos a Product Owner que debe ser un rol de negocio.
Tipos de usuarios representantes:

- Gerentes de Usuarios
- Gerentes de Desarrollo
- Alguien del grupo de marketing
- Vendedores
- Expertos del Dominio
- Clientes
- Capacitadores y personal de soporte.

CRITERIOS DE ACEPTACIÓN

Es información concreta para saber si lo que implementamos es correcto o no (según el Product Owner).

Definen límites para la User Story.
Ayudan a que los PO respondan lo que necesitan para que la US provea valor (requerimientos funcionales mínimos).
Ayudan a que el equipo tenga una visión compartida de la US
Ayudan a desarrolladores y testers a derivar las pruebas.
Ayudan a los desarrolladores a saber cuándo parar de agregar funcionalidad en una US.
Aparecen debajo de la story (son parte de la Tarjeta).



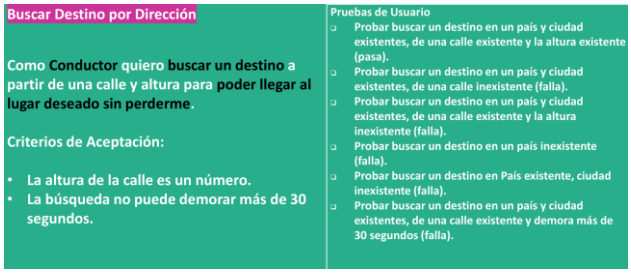
Se escriben a nivel de usuario. Mejor si tienen una valoración cuantitativa.
Definen una intención, no una valoración.
Son independientes de la implementación.
Relativamente de alto nivel, no es necesario que se escriba cada detalle.
Los detalles (de implementación) deben estar en las pruebas y en el diseño de los casos de prueba, que es donde se necesitan.

Pruebas de Aceptación

En el dorso de la tarjeta y expresan detalles resultantes de la conversación. Son las pruebas que ejecutará el PO para comprobar que se construyó lo que se pidió. Son los escenarios mínimos e indispensables para aceptar el software.

Se escriben en infinitivo (Probar buscar, Probar ...)

Complementan la User Story. La User Story completa queda como:



Se deben identificar escenarios de éxito (pase) y de fracaso (falla).

En teoría el usuario es el que define las pruebas de aceptación porque es el que sabe qué casos alternos pueden existir. La US las definen entre todos en sí.

DEFINITION OF READY - DEFINICIÓN DE LISTO

Es una medida de calidad que define el equipo para poder determinar que la User Story puede entrar a la iteración de desarrollo.

El modelo de calidad de base es el INVEST Model.

INVEST Model

- Independent - calendarizables e implementables en cualquier orden. El PO decide cuáles US se van a desarrollar.
- Negotiable - el "qué" no el "cómo". Debe escribirse la US en términos de qué necesita el User.
- Valuable - debe tener valor para el cliente
- Estimatable - para ayudar al cliente a armar un ranking basado en costos. Es asignarle un peso a la US para poder compararla con otras.
- Small - deben ser "consumidas" en una iteración. Depende del Equipo.
- Testable - demostrar que fueron implementadas. Relacionada a las pruebas de aceptación.

Es un modelo de mínima para construir el Definition of Ready. Permite determinar si la US entra al Sprint.

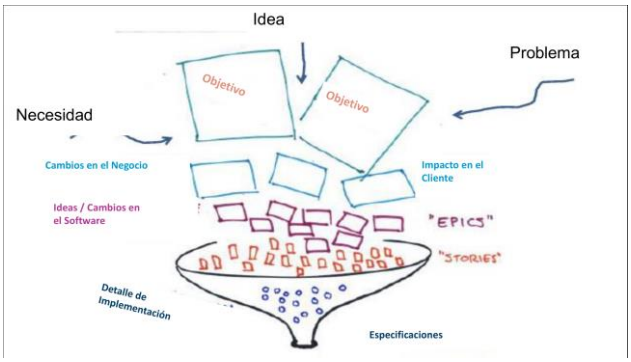
Las User Stories ...

- No son especificaciones detalladas de requerimientos (como los casos de uso)
- Son expresiones de intención, "es necesario que haga algo como esto..."
- No están detallados al principio del proyecto, elaborados evitando especificaciones anticipadas, demoras en el desarrollo, inventario de requerimientos y una definición limitada de la solución.
- Necesita poco o nulo mantenimiento y puede descartarse después de la implementación.
- Junto con el código, sirven de entrada a la documentación que se desarrolla incrementalmente después.

Niveles de Abstracción

Épica: es una US muy grande (no cumple con el INVEST) y no se puede consumir en una iteración.

Temas: pueden agrupar US que van para un mismo módulo del Sistema.



De la Idea, Necesidades, Problemas se refinan las US a Épicas, luego a Stories y por último a Especificaciones y Detalles de Implementación.

SPIKES

Tipo especial de historia, utilizado para quitar riesgo e incertidumbre de una User Story u otra faceta del proyecto.

Se clasifican en : técnicas y funcionales. Pueden necesitarse de los dos tipos para una misma US.

Pueden utilizarse para:

• Inversión básica para familiarizar al equipo con una nueva tecnología o dominio.

• Analizar un comportamiento de una historia compleja y poder así dividirla en piezas manejables.

• Ganar confianza frente a riesgos tecnológicos, investigando o prototipando para ganar confianza.

• Frente a riesgos funcionales, donde no está claro como el sistema debe resolver la interacción con el usuario para alcanzar el beneficio esperado.

En general se usan para tener mayor conocimiento sobre cierto tema del que se tiene incertidumbre.

Técnicas

• Utilizadas para investigar enfoques técnicos en el dominio de la solución.
  - Evaluar performance potencial
  - Decisión hacer o comprar
  - Evaluar la implementación de cierta tecnología.

• Cualquier situación en la que el equipo necesite una comprensión más fiable antes de comprometerse a una nueva funcionalidad en un tiempo fijo.

Funcionales

• Utilizadas cuando hay cierta incertidumbre respecto de cómo el usuario interactuará con el sistema.
• Usualmente son mejor evaluadas con prototipos para obtener realimentación de los usuarios o involucrados.

Lineamientos para el Spike

Deben ser:

- Estimables, demostrables y aceptables
- La excepción, no la regla
  - No se deben usar siempre que haya incertidumbre porque siempre la hay.
- Implementar la spike en una iteración separada de las historias resultantes. (y antes de la US en la que se quiere implementar)

Para tener en cuenta sobre US:

Ejecutar spikes inmediatamente antes de donde se aplique su resultado.
Diferir el análisis detallado tan tarde como sea posible, lo que es justo antes de que el trabajo comience.
Hasta entonces, se capturan requerimientos en la forma de "user stories".
Las user stories no son requerimientos de software, no necesitan ser descripciones exhaustivas de la funcionalidad del sistema.

Esto es porque si van a cambiar los requerimientos entonces es mejor detallarlos cuando se los implemente.

Tips para que las US sean útiles

Un paso a la vez (evitar la palabra "Y")

No olvides la parte invisible: la conversación

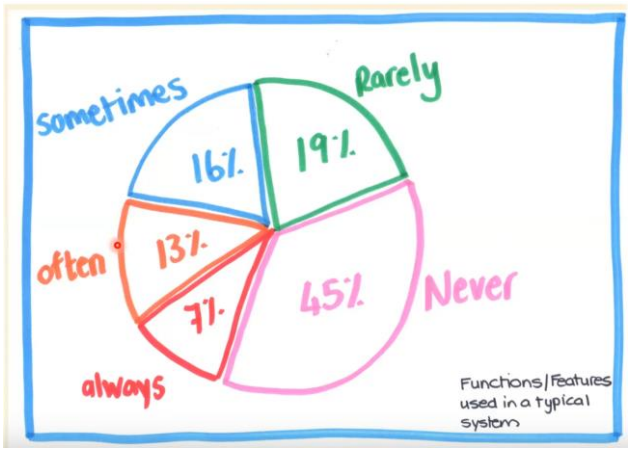
Usar palabras claras en los criterios de aceptación

Las user stories se escriben desde la perspectiva del usuario

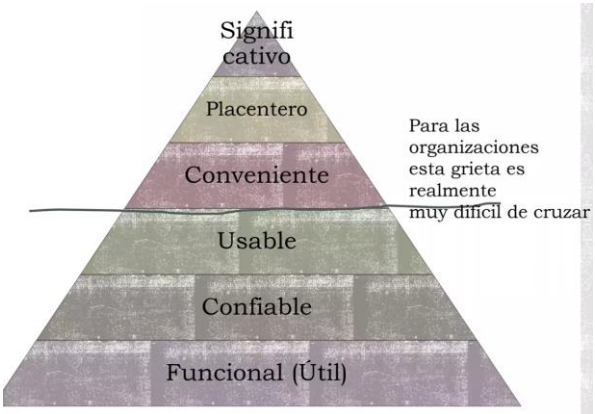
No forzar todo para escribirlo como user stories

GESTIÓN DE PRODUCTOS (EN AMBIENTES ÁGILES)

Los productos se crean por distintas motivaciones.



Evolución de los productos de Software:



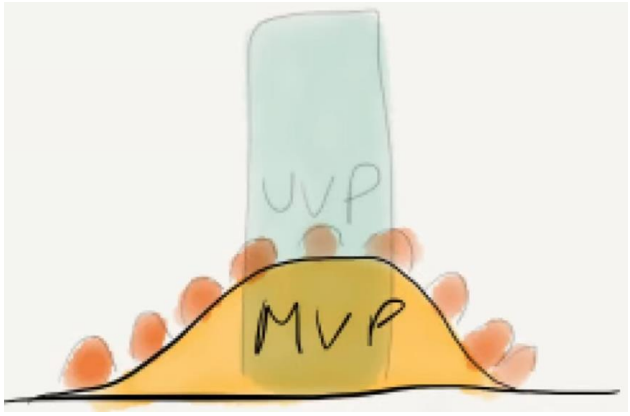
Conveniente es el SW que genera diferencia y acelera productividad.

MVP

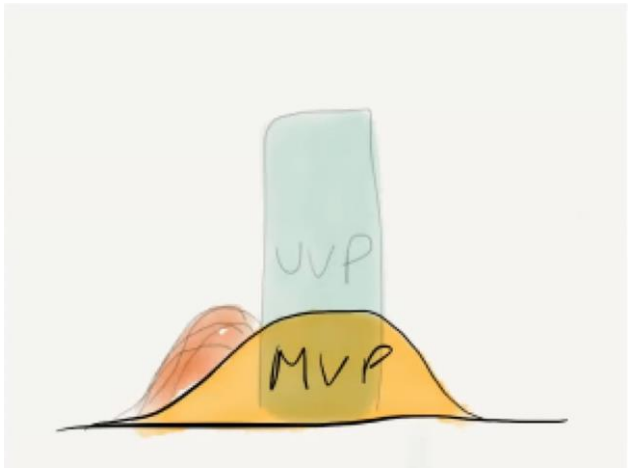
Producto mínimo viable. Es el producto mínimo que se necesita para validar un concepto, propósito o hipótesis de valor único. Debe ser barato. "Es una hipótesis por validar". Se necesita realimentación rápido. Se necesita saber si va a haber un mercado o un cliente (se busca un subconjunto de clientes potenciales) dispuesto a "comprarlo". Es casi un prototipo.

Con este MVP salgo a buscar clientes. Los clientes luego van definiendo nuevas necesidades para poder lanzarlo al mercado en algún momento.

Ese es su propósito principal. Buscar Feedback. Cada cliente puede estar buscando distintas cosas (origina los puntos naranjas).



Puede pasar que al revisar la hipótesis o propósito se vaya alejando del foco o hipótesis inicial (los clientes apuntan a una misma dirección en cuánto a qué es lo que quieren).



Si eso pasa puede convertirse en un segundo MVP y surge también una nueva hipótesis. Y si se quiere aumentar el crecimiento o si se sabe que puede haber mayor alcance en otras áreas, se pueden añadir nuevas características.

MVF

Característica (Feature) Mínima Viable. Es parte del MVP porque un producto tiene un conjunto de características. Es la pieza más pequeña de funcionalidad que puede ser liberada. Tiene valor para la organización y para los usuarios.

MMF

Es el conjunto de Características (Feature) Mínimas Comercializables (Marketable). Aquí el propósito es buscar lo mínimo para poder salir a vender el producto al mercado. Esto esta incluido en el MMP (Producto Mínimo Comercializable). Su objetivo es aportar valor. Supone cierta certeza de que existe valor en esta área y que sabemos cuál debe ser el producto para proporcionar ese valor.

Se divide la gran característica en MMFs más pequeños debido al tiempo de comercialización (Time to market) que es acotado.

MRF

Característica mínima Desplegable (Entregable/Publicable). Se encamina a un producto que saldría al mercado. Es el conjunto de características mínimas para cumplir con el release. Van a formar parte del MMR (Producto Mínimo Entregable/Desplegable)

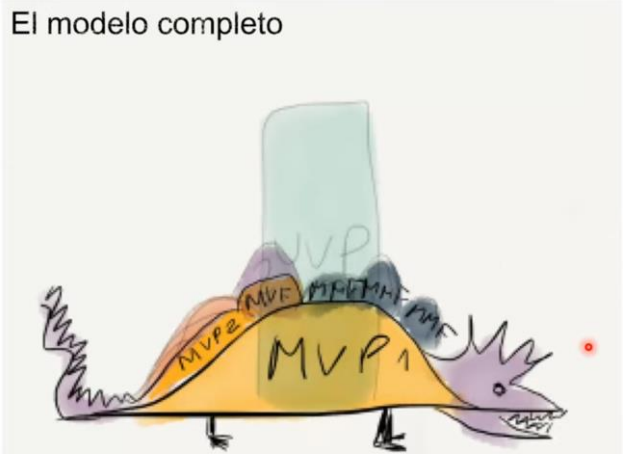
MMP

Es el primer MMR dirigido a primeros usuarios (early adopters). Focalizado en las características clave.

MMR

Release de un producto que tiene el conjunto de características más pequeño posible. Es el incremento más pequeño que ofrece valor nuevo a los usuarios y satisface sus necesidades actuales.

El modelo completo

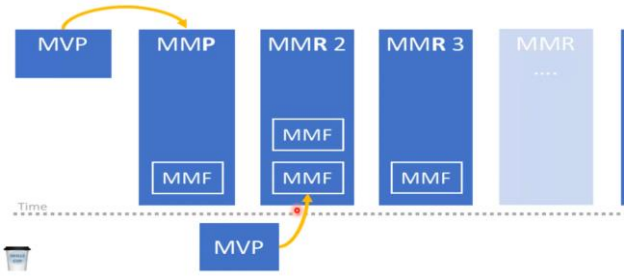


Las US van a determinar el MVP que se desarrollará y que se debe validar.

Mientras menos tiempo se invierta, menos plata y menor el riesgo.

Relación entre MVP, MMF, MMP, MMR

Se arranca con el MVP para validar un conjunto de características y da origen al 1er MMP formado por un conjunto de MMF. Ese primer MMP es el primer MMR. A partir de ahí salen nuevos MMR. Puede ocurrir que aparecen nuevas características (que quiera implementar o que el mercado requiera) que dan origen a un nuevo MVP (para validar la nueva hipótesis), que va a ser parte de los MMR subsiguientes.



A tener en cuenta:

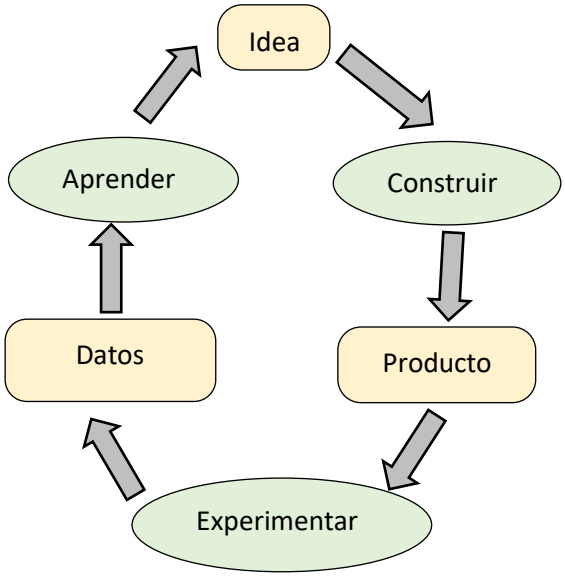
No confundir MVP con MMP o MMF.

Se debe generar valor. Sino es desperdicio.

El MVP es capaz de eliminar (o no generar) desperdicio puesto que genera retroalimentación rápida ("Just In Time") por lo que en teoría solo se implementan lo que genera valor.

MÉTODO LEAN START - UP

Es un modelo tipo: Build-Experiment-Learn Feedback Loop.

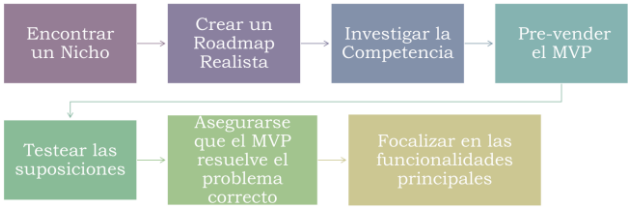


Es un ciclo de construir, experimentar y aprender. Es el ciclo que se usa para construir el MVP.

Lo que nos motiva a crear valor es definir una Hipótesis. Es nuestro motor.

El MVP puede moverse en estos extremos, pero no una aplicación terminada. Es decir, no necesita ser código ejecutándose. Solo necesita poder mostrar esa característica que se quiere comprobar de la hipótesis.

Una vez cumplido el Ciclo, se puede evaluar que se va a hacer. La audacia de cero. Si no se tienen ingresos, clientes, etc... que se pueda perder, es cuando se anima el emprendedor a presentar el MVP. El "Salto de Fe" es el que se hace para presentar el MVP.



Roadmap es el "plan" de como voy a ir implementando nuevas características.

ESTIMACIÓN

La idea es entender cuál es el tamaño de lo que nosotros queremos hacer.

Se debe hacer como equipo.

Utilizan una medida llamada "Story Points"; que es una medida de estimación relativa. Se comparan distintos aspectos.

Tips respecto de las estimaciones

Si las estimaciones se utilizan como compromisos son muy peligrosas y perjudiciales para cualquier organización.

Lo más beneficioso en las estimaciones es el "proceso de hacerlas".

La estimación podría servir como una gran respuesta temprana sobre si el trabajo planificado es factible o no.

La estimación puede servir como una gran protección para el equipo.

Tamaño (Medido con Story Poits)

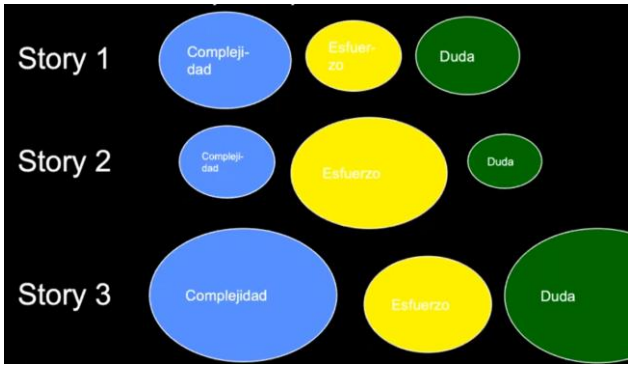
Para medirlo, se usa el Story Point.

El tamaño se indica con:

- La *complejidad* (compleja/simple) de una Story (no es proporcional al tiempo requerido necesariamente)
- Qué tanto *esfuerzo* se necesita para hacer esa story. Cuánto tiempo (cantidad de horas) lleva hacerlo
- La *duda o incertidumbre* intrínseco a la US respecto del negocio o detalles técnicos.

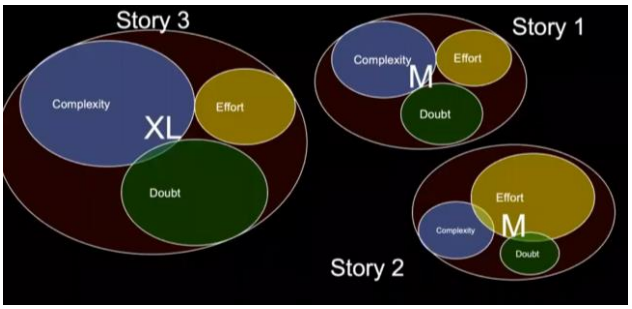
Cómo Estimar

Distintos tipos de Stories

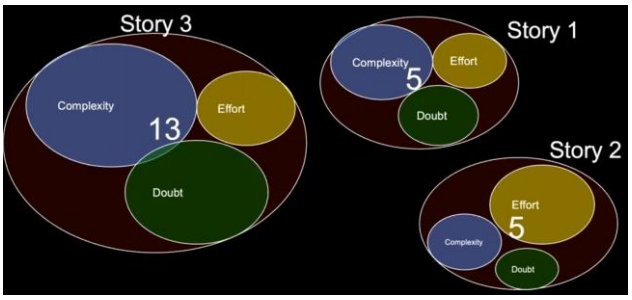


(Los óvalos del medio, son esfuerzos)

Se elige también una escala para estimar el tamaño como: de 1 a 10, Tallas de Remeras, potencias de 2, Serie de Fibonacci.



O con números (fibonacci):



Velocidad (Velocity)

Es una medida (métrica) del progreso de un equipo. Se calcula sumando el número de story points (asignados a cada user story) que el equipo completa durante la iteración.

Se cuentan los story points de las Users Stories que están completas, no parcialmente completas.

La Velocidad corrige los errores de estimación.

PROPUESTA DE MÉTODO DE ESTIMACIÓN: POKER ESTIMATION O POKER PLANNING

Los participantes tienen tarjetas de Poker.

Se usa la serie de Fibonacci. Se determina por la suma del numero anterior más el actual (1,1,2,3,5,8 ...). Existe una diferencia visible entre 3 y 5 por ejemplo.

El puntaje es proporcional al tamaño de la US. Se trata de que sea menor o igual a 13.

- 0: Quizás ud. no tenga idea de su producto o funcionalidad en este punto.
- 1/2, 1: funcionalidad pequeña (usualmente cosmética).

- 2-3: funcionalidad pequeña a mediana. Es lo que queremos. 😊
- 5: Funcionalidad media. Es lo que queremos 😊
- 8: Funcionalidad grande, de todas formas lo podemos hacer, pero hay que preguntarse sino se puede partir o dividir en algo más pequeño. No es lo mejor, pero todavía 😊
- 13: Alguien puede explicar por qué no lo podemos dividir?
- 20: Cuál es la razón de negocio que justifica semejante story y más fuerte aún, por qué no se puede dividir?.
- 40: no hay forma de hacer esto en un sprint.
- 100: confirmación de que está algo muy mal. Mejor ni arrancar.

Prerrequisitos

- Lista de features/stories a ser estimadas
- Cada estimador tiene un mazo de cartas.

Pasos

1. Determine la base story (la canónica) que será usada para comparar con las otras stories. Digamos, Story Z.

    1.1 La story a ser estimada se lee a todo el equipo.

    1.2 Los estimadores discuten la story, haciendo preguntas al product owner (las que se necesiten).

    1.3 Cada estimador selecciona una carta y pone la carta boca abajo en la mesa.

    1.4 Cuando todos pusieron las cartas, las mismas se exponen al mismo tiempo.

    1.5 Si todos los estimadores selecciona el mismo valor, ese es el estimado. Si no, los estimadores discuten sus resultados, poniendo especial atención en los más altos y los más bajos. Después de la charla, GOTO to 1.3.

2. Se toma la próxima story, se discute con el product owner.

3. Cada estimador asigna a la story un valor por comparación contra la base story. "Cuan grande/pequeña, compleja, riesgosa es esta story comparada con Story Z?". GOTO to 1.3

A tener en cuenta:

La Base Story (base canónica) se busca que no sea tan grande para poder compararla con las otras Stories. Se trata de que la duda sea bien chica.

Los spikes no se estiman. Las US grandes no se estiman tampoco porque no son posibles de hacer en una sprint.

CLASE DEL TEÓRICO - ESTIMACIÓN

(Falta la clase del teórico de estimación)

FRAMEWORK SCRUM

Definicion de scrum

Marco ligero que ayuda a: personas, equipos y organizaciones a generar valor a través de soluciones adaptables para problemas completos. Requiere scrum master para fomentar entorno donde:

- Propietario del producto -> Ordena el trabajo de un problema complejo en product backlog
- Equipo de scrum-> convierte selección del trabajo en un incremento de valor durante el sprint
- Equipo de scrum y partes interesadas inspeccionan los resultados y realizan ajustes necesarios para el próximo sprint
- Repetir

Marco deliberadamente incompleto, basado en inteligencia colectiva de personas que lo utilizan. No proporciona instrucciones detalladas, sino que guía sus relaciones e interacciones. Se pueden emplear: procesos, técnicas y métodos y hace visible la eficacia relativa de la gestión actual, entorno y técnicas permitiendo mejoras.

Teoría de scrum

Basada en empirismo y pensamiento Lean.

Empirismo-> afirma que el conocimiento proviene de la experiencia y toma de decisiones se basa en lo observado.

Pensamiento Lean: reduce desperdicios y se centra en lo esencial.

Scrum emplea enfoque iterativo e incremental para optimizar previsibillidad y controlar el riesgo. Scrum involucra a grupos de personas que colectivamente tienen todas las habilidades y experiencia para hacer el trabajo o adquirirlas si es necesario.

Scrum combina cuatro eventos formales para-> inspección y adaptación dentro de un evento contenedor llamado sprint.

Eventos funcionan porque implementan los pilares empíricos de Scrum: transparencia, inspección y adaptación

Transparencia

Proceso y trabajo emergentes deben ser visibiles para quienes realizan el trabajo y los que lo reciben. Decisiones importantes se basan en estado percibido de sus tres artefactos formales. Artefactos con poca transparencia pueden conducir a decisiones que disminuyen el valor y aumentan el riesgo. Permite inspección, inspección sin transparencia genera engaños y desperdicios.

Inspección

Los artefactos de Scrum y el progreso hacia objetivos acordados deben ser inspeccionados con frecuencia y diligentemente para->detectar varianzas o problemas potencialmente indeseables.

Scrum proporciona cadencia en forma de sus cinco eventos para ayudar con inspección: Permite la adaptación, inspección sin adaptación se considera inútil. Eventos de Scrum están diseñados para provocar cambios.

Adaptación

Si-> aspecto de un proceso se desvía fuera de límites aceptables o producto resultante es inaceptable-> proceso que se está aplicado o materiales que se producen deben ajustarse -> ajuste debe realizarse lo antes posible para minimizar la desviación adicional.

Si personas involucradas no están empoderadas o no poseen capacidad para autogestionarse -> adaptación es más difícil

Se espera que: equipo de scrum se adapte cuando -> aprende algo nuevo por medio de inspección.

Valores de scrum

Dan dirección al equpo de Scrum con respecto a su trabajo, acciones y comportamiento. Decisiones, medidas y forma en que se utiliza scrum deben reforzar estos valores, no disminuirlos o socavarlos. Cuando estos valores son asimilados->los pilares empíricos de Scrum cobran vida construyendo .

Confianza, Compromiso, Enfoque, Apertura, Respeto, Coraje

Equipo de scrum se compromete a lograr sus objetivos y apoyarse mutuamente. Su enfoque principal es el trabajo del Sprint para hacer mejor el progreso posible hacia estos objetivos. Equipo de scrum y partes interesadas están abiertos sobre el trabajo y los desafíos.

Miembros del equipo de scrum se respetan mutuamente para ser personas capaces e independientes, tienen el valor de hacer lo correcto y de trabajar en problemas complejos.

Equipo de Scrum (Scrum team)

Consta de un Scrum master, propietario del producto y desarrolladores. No hay subequipos ni jerarquías. Es una unidad cohesionada de profesionales enfocada en un objetivo a la vez el objetivo del Producto
Son multifuncionales -> miembros tienen todas las habilidades necesariras para crear valor en cada Sprint.
Son autogestionados -> internamente deciden quíen hace qué, cuándo y cómo.
Lo suficientemente pequeño para permanecer ágil
Lo suficientemente grande para completar un trabajo significativo dentro de un Sprint, 10 o menos personas generalmente.
Si equipos se vuelven demasiados grandes->reorganizamos en varios equipos cohesionados, centrados en el mismo producto, comparten el mismo objetivo del producto, trabajo pendiente del producto y propietario del producto

Responsable de Todas las actividades relacionadas con los productos:

Colaboración, Verificación, Mantenimiento, Operación, Experimentación, Investigación, Desarrollo, Trabajar en sprints a un ritmo sostenible mejora el enfoque y la consistencia del equipo de scrum.

Equipo es responsable de crear un incremento valioso y útil en cada Sprint.

Responsabilidades específicas: Desarrolladores, propietario del producto, Scrum Master.

Desarrolladores

Personas que se comprometen a crear cualquier aspecto de un incremento útil (funcional) en cada Sprint

Habillidades específicas que necesitan son a menudo amplias y varían con el dominio del trabajo

Responsables de:

- Crear plan para el sprint, el Sprint Backlog
- Inculcar calidad adhiriéndose a definición de Hecho
- Adaptar su plan cada día hacia el Objetivo Sprint
- Responsabilizarse mutuamente como profesionales

Propietario del producto

Responsable de maximizar el valor del producto resultante del trabajo del equipo de Scrum (forma puede variar) y de gestión eficaz de la pila del producto:

Desarrollar y comunicar explícitamente el objetivo del producto. Creación y comunicación clara de elementos de trabajo pendiente del producto.
Pedido de artículos de trabajo pendiente del producto.
Asegurarse que trabajo pendiente del producto sea transparente, visible y comprendido.
Puede hacer el trabajo anterior o puede delegar la responsabilidad a otros.
]Organización debe respetar sus decisiones-> visibles en contenido y orden del trabajo pendiente del producto y a través de incremento inspeccionable en la revisión de Sprint
Persona, no comité

Puede representar las necesidades de muchas partes interesadas en el trabajo pendiente del producto, cambios del mismo se negocian con el propietario del producto

Scrum Master

Responsable de establecer Scrum tal como se define en la guía de Scrum.
Ayuda a todos a comprender la teoría y la práctica de Scrum (dentro del equipo y organización)
Responsable de la efectividad del Scrum Team -> permite que el equipo mejore sus prácticos dentro del marco de Scrum.
Líderes que sirven al equipo Scrum y toda la organización
Capacitar a miembros en autogestión y multifuncionalidad.
Ayudar a enfoque en creación de incrementos de alto valor que cumplan con la definición de hecho.
Promover eliminación de impedimentos para el progreso del equipo.
Asegurar que eventos del scrum se lleven a cabo, sean positivos, productivos y se respete el tiempo establecido para ellos.
Ayuda al PO: Encontrar técnicas para definición eficas de objetivos del producto y la gestión de los retrasos en el producto. Ayudar a que elementos del trabajo pendiente de productos sean claros y concisos. Ayudar a establecer la planificación empírica de productos para un entorno complejo. Facilitar la colaboración de las partenes interesadas
Ayudan a la organización:Liderar, capacitar, mentorizar a la organización en adopción de Scrum
Planificar y asesorar implementación de Scrum
Eliminar barreras entre partes interesadas y equipos de scrum

Eventos de Scrum

Sprint es un contenedor para todos los eventos. Cada evento es una oportunidad formal para inspeccionar y adaptar los artefactos de Scrum. Diseñados para permitir la transparencia necesaria. Crean regularidad y minimizan la necesidad de reuniones no definidas en Scrum. Óptimamente se llevan a cabo al mismo tiempo y lugar para reducir la complejidad

Sprint

Donde las ideas se convierten en valor. Eventos de longitud fija de un mes o menos para crear consistencia. Nuevo sprint comienza inmediatamente después de la conclusión del Sprint anterior. Dentro del sprint ocurre todo el trabajo necesario para alcanzar el objetivo del producto:

- Planificación (sprint planning)
- Scrums Diarios
- Revisión del Sprint (Sprint Review)
- Retrospectiva (Sprint Retrospective)

No se hacen cambios que pongan en peligro el Objetivo Sprint. Calidad no disminuye. Se refina el trabajo pendiente del producto según sea necesario. Alcance se puede clarificar y renegociar con el Propietario del Producto a medida que se aprende más. Garantizan inspección y adaptación del progreso hacia un objetivo del Producto como mínimo una vez al mes, permiten previsibilidad. Sprints más cortos-> pueden generar más ciclos de aprendizaje, limitar riesgo de coste y esfuerzo a

un período de tiempo más pequeño. Cada sprint puede considerarse un proyecto corto. En entornos complejos se desconoce lo que sucederá. Puede ser cancelado si el Objetivo del Sprint se vuelve obsoleto, solo por el Propietario del Producto.

Planificación de Sprint

Inicia el sprint estableciendo el trabajo que se realizará para el mismo. Trabajo colaborativo de todo el equipo -> plan resultante.
Se discuten los elementos de trabajo pendiente de producto más importantes y cómo se asignan al objetivo del producto. Se puede invitar a otras personas a asistir para proporcionar asesoramiento. Temas:

Por qué este Sprint es valioso.
PO propone cómo el producto podría aumentar su valor y utilidad en el sprint actual.
Equipo *define un objetivo de Sprint que comunique por qué el Sprint es valioso para las partes interesadas, objetivo debe finalizarse antes del final de la planificación de sprint
Qué se puede hacer este Sprint

Debate con el PO, desarrolladores seleccionan elementos del Product Backlog para incluir en el sprint actual. Equipo de scrum puede refinar estos elementos aumentando la comprensión y confianza. Cuanto más sepan los desarrolladores sobre su rendimiento pasado, su capacidad futura y su definición de hecho, más seguros estarán en sus pronósticos de Sprint. Cómo se realizará el trabajo elegido. Para cada elemento de trabajo pendiente de producto seleccionado los desarrolladores planifican el trabajo necesario para crear un incremento que cumpla con la definición de hecho. Se descomponen elementos de trabajo pendientes del producto en otros más pequeños que se puedan realizar en un día o menos, según la discreción de los desarrolladores. Trabajo pendiente de Sprint : objetivo de Sprint + elementos de trabajo pendiente de producto seleccionados para el sprint + plan. Duración máxima de ocho horas para un sprint de un mes

Scrum Diario

Propósito: Inspeccionar el progreso hacia el Objetivo Sprint y adaptar el Sprint Backlog según sea necesario, ajustando el próximo trabajo planeado. Evento de 15 minutos máximo para desarrolladores del equipo de Scrum. Al mismo tiempo y lugar todos los dias laborales para reducir complejidad. PO o SM participan como desarrolladores si están trabajando activamente en los elementos del Trabajo pendiente de sprint. Desarrolladores pueden seleccionar cualquier estructura y técnica que deseen mientras se centre en el progreso hacia el objetivo de Sprint y produzca un plan accionable para el siguiente día de trabajo. Crea enfoque y mejora autogestión. Mejoran la comunicación, identifican impedimentos, promueven una rápida toma de decisiones y eliminan necesidad de otras reuniones. No es la única vez que los desarrolladores pueden ajustar su plan.

Revision del Sprint (Sprint Review)

Propósito: Inspeccionar el resultado del Sprint y determinar futuras adaptaciones. El equipo de Scrum presenta los resultados de su trabajo a las partes interesadas clave y se discute el progreso hacia el Objetivo de Producto. Se revisa lo que se logró en el Sprint y lo que ha cambiado en su entorno. Trabajo pendiente del producto se puede ajustar para satisfacer nuevas oportunidades.Es una sesión de trabajo no una presentación. Máximo de cuatro horas para un Sprint de un mes

Retrospectiva del Sprint

Propósito: Planificar formas de aumentar la calidad y la eficacia

Equipo de Scrum inspecciona cómo fue el último Sprint con respecto a-> individuos, interacciones, procesos, herramientas, definición de hecho. Suposiciones que los desviaron se identifican y se exploran sus orígenes. Se analiza qué fue bien durante el Sprint, problemas encontrados y cómo fueron o no resueltos. Se identifican los cambios más útiles para mejorar su eficacia, mejoras más impactantes se abordan lo antes posible y se pueden agregar al sprint Backlog para próximo Sprint. Concluye el Sprint. Máximo de 3 horas para un sprint de un mes.

Artefactos de Scrum

Representan trabajo o valor

Diseñados para maximizar la transparencia de la información clave

Inspección se da en misma base de adaptación

Cada artefacto tiene compromiso para garantizar que proporcione información que mejore la transparencia y el enfoque frente al cual se pueda medir el progreso: ● Para el Product Backlog, es el Objetivo del Producto. ● Para el Sprint Backlog, es el Objetivo del Sprint. ● Para el Increment es la Definición de Terminado.

Product Backlog

Es una lista emergente y ordenada de lo que se necesita para mejorar el producto. Es la única fuente del trabajo realizado por el Scrum Team. Existe el refinamiento del PB. Developers que realizarán el trabajo son responsables del dimensionamiento.

Compromiso: Objetivo del Producto

Describe un estado futuro del producto que puede servir como un objetivo para que el Scrum Team planifique. El Objetivo del Producto está en el Product Backlog. El resto del Product Backlog emerge para definir "qué" cumplirá con el Objetivo del Producto.

Sprint Backlog

Se compone de:

Objetivo del Sprint (por qué). El conjunto de elementos del Product Backlog seleccionados para el Sprint (qué). Plan de acción para entregar el Increment (cómo). El Sprint Backlog es un plan realizado por y para los Developers.

Compromiso: Objetivo del Sprint

El Objetivo del Sprint es el único propósito del Sprint. Proporciona flexibilidad en términos del trabajo exacto necesario para lograrlo. Crea coherencia y enfoque. En el sprint planning se crea se agrega al SB. Pueden colaborar con el PO para negociar alcance del SB.

Incremento

Un Increment es un peldaño concreto hacia el Objetivo del Producto. La suma de ellos hace al objetivo del Producto (Por eso funcionan juntos). Para proporcionar valor, el Increment debe ser utilizable. Se pueden crear múltiples Increments dentro de un Sprint. El trabajo no puede considerarse parte de un Increment a menos que cumpla con la Definición de Terminado.

Compromiso: Definición de Terminado

Es una descripción formal del estado del Increment cuando cumple con las medidas de calidad requeridas para el producto. Si un elemento del PB cumple con la DefOfRead (Sino vuelve al PB). La Definición de Terminado crea transparencia al brindar a todos un entendimiento compartido de qué trabajo se completó como parte del Increment.. Es parte de los estándares de la organización. Los distinto STs de la misma Org deben cumplir el mismo DefOfReady.

MÁS SOBRE SCRUM (CLASE DEL TEÓRICO)

Compromiso de los artefactos

Es un concepto que no estaba en las versiones de Scrum anteriores y qué hace visible el valor del artefacto en sí.

Se agrega formalmente el concepto de objetivo del producto.

El objetivo del Sprint nos da mayor flexibilidad en cuánto a las USs que podemos meter en el sprint (incluso durante el sprint).

Timebox

Existe entonces un tiempo definido para cada actividad porque no se quiere negociar con el tiempo para que sea constante. Eso permita que tengamos más confianza en cuanto al ritmo de trabajo y mayor previsibilidad. Todas las ceremonias son Timeboxes. Lo importante es respetar el timebox no la duración en sí misma (que recomiende Scrum específicamente) que se debe definir con el equipo.

La nueva guía de scrum Define una ceremonia obligatoria: el refinamiento del Product Backlog que debe durar el 10% del tiempo del Sprint.

Definición de Hecho

Es un criterio que determina si una característica entra al producto que se le va a mostrar al Product Owner. Es un acuerdo que define el equipo. Tiene su propia checklist. Si no se cumple con la checklist está característica no puede entrar a la Sprint Review. Permite trabajar con los valores de compromiso y respeto.

Capacidad del equipo

Es una métrica. Es la que se estima y es la que usa el equipo para ver cuánto compromiso de trabajo puede llegar a asumir en un Sprint. Se realiza en la Sprint planning. Habiendo estimado esa capacidad Entonces dice cuántos cuántas historias de usuario del producto Backlog van a pasar al Sprint backlog. Se ve Cuántas horas ideales cada miembro del equipo puede llegar a dedicar al proyecto al trabajo. También se puede estimar la capacidad en puntos de historia. Se recomienda trabajar por rangos de tiempo y no por una cantidad definida de horas. Arriba también hace una mejor estimación. Entonces el alcance se determina

habiendo definido la capacidad del equipo y el timebox en sí y a partir de eso es que se puede decir qué es lo que se puede llegar a completar en el Sprint.

Las variables que debe definir el equipo es la *duración del Sprint, cálculo de capacidad y el objetivo del Sprint* y con esas tres variables se define que se va a hacer y qué va a pasar al Sprint backlog.

El concepto de visibilidad implica también que la forma de trabajar del equipo sea visible; como por ejemplo que trabajen el mismo espacio y que puedan ver qué es lo que está haciendo cada uno.

El problema de la gestión en horas es que estima esfuerzo y no sobre el producto en sí.

Herramientas de Scrum

Tablero de Scrum o Taskboard:

(Tareas)Tiene la frase verbal, código de ítem del backlog, el tiempo estimado en horas ideales y el nombre de la persona que va a realizar esa actividad que en principio la empieza y termina ella misma. A veces también tienen el número del día del Sprint en el que se va hacer.

Scrum recomienda estimar con puntos de historia porque en definitiva Esto es lo que estima sobre el producto y lo valioso es el producto.

La configuración básica del tablero tiene tres columnas dos: La columna de las historias por realizar, la columna de las tareas que se están realizando y la columna de las tareas completadas. Se le puede llegar a agregar una cuarta columna en dónde se anotan las tareas que se están por realizar más específicas que vienen del Sprint backlog. En esta última columna estás tareas se estiman en horas ideales. Puede pasar también que hasta una columna más que se llame "listo listo" en donde se coloquen las historias de usuario en sí que ya se terminaron. Scrum propone que sean la misma persona las que se asignen así mismas las tareas por realizar por sus principios de autosuficiencia autogestionados y motivados.

Granularidad

Es mala cuando en el tablero existen tareas muy grandes Entonces es poca la granularidad o es mala. Son historias de 8 puntos o mas.

La gestión ágil apunta una gestión binaria Qué quiere decir que las tareas están terminadas o no están terminadas y no hay puntos medios.

Burntdown Charts

Es un gráfico que nos permite visualizar el trabajo. En el eje "y" están la cantidad de puntos por hacer y en el eje x está en la cantidad de días dentro de Sprint. Idealmente el gráfico toma forma de recta de pendiente negativa en el cuadrante primero. Este gráfico solo existe durante una Sprint.

Sprint Burnup Chart

Muestra cuánto trabajo se va terminando. Tienen el eje de las y los puntos de qué se van realizando y en el eje de las x los días del Sprint. Se mantienen por la duración del proyecto porque toma las medidas de todos los Sprints.

Niveles de planificación

El más chico de los niveles de planificación es el Diario está en las Daily scrum. Pero donde se hace ajustes diarios.

El siguiente nivel de planificación es el Sprint planning Y es a nivel de iteración.

Por encima está la planificación del Release qué es una versión del producto que se libera. La planificación del Release implica estimar Cuántos xprint voy a necesitar para tener las características que integran el Release.

Por encima de esta esta la planificación del producto qué es la sumatoria de los releases que voy generando en el tiempo.

Planificación de portfolio se refiere ya a un nivel en dónde se planifica una familia de productos como por ejemplo Office.

Y por último está el nivel de estrategia.

Cada uno de estos niveles tiene un horizonte es decir un determinado intervalo de tiempo por el que se realiza la planificación.

En base a los distintos niveles de decisión podemos tener distintos niveles de backlog. Uno de portfolio, otro de producto y otro del Sprint, por ejemplo. Se pueden usar también distintos tipos de medidas para cada uno.

De la visión del producto obtengo el producto Backlog. Es necesario decir que nunca tienen todas las características del producto. Luego parte de ese backlog va a formar un Release específico para el cual se debe estimar cuántos Sprints va a llevar desarrollarlo. Cuántos Release voy a sacar es planificación de producto y cuántos Sprints me va a llevar a hacer un Release es planificación de release. Luego está la planificacion del Sprint para ver un conjunto de historias que se quieran realizar y ese conjunto de historias se pueden dividir en tareas más pequeñas qué son las que se van a realizar también en el mismo Sprint.

La cadencia de los Realeses la define la organización.

La velocidad corrige los errores de estimación. La velocidad se calcula como la cantidad de historias aceptadas en un Sprint.

La velocidad es parecida otra métrica llamada rtf o características testeadas corriendo por iteración.

Solo la capacidad se estima.