***Procesos de Software***

En el camino desde que se tiene una idea hasta que se materializa se dan varios pasos importantes, tales como:

* Organizar el equipo de desarrollo (tamaño del mismo, definir los roles, …)
* Decidir la tecnología que va a usarse.
* Gestionar el tiempo:
  + Predicción de futuro.
  + ¿Qué hacer?

Hay que tener cuidado con el cambio en los requisitos, ya que puede acarrear diversos problemas como aumento de tiempo, erratas en el producto, etc.

Concepto de adaptabilidad: en un ejemplo, repartir el trabajo en zonas para poder reducir así costes de producción.

Como introducción, abarcaremos el modelo de vida software en cascada.

Este consta de ciclos iterativos, y es actualmente el más usado, aunque no sea el mejor, ya que lleva demasiado tiempo implantado.

Se divide en 4 fase generalmente: obtención de requisitos, diseño del problema, codificación del mismo y, por último, testing del producto.

El trabajo del testing en cascada es el de comprobar si se cumplen los requisitos. Es un trabajo de sufridor en este modelo, ya que se realiza demasiado tarde y cuando se encuentra el problema ya ha adquirido una gran complejidad. Además, cargará con los retrasos de las demás etapas.

El ciclo en cascada se basa en un modelo contractual: la relación entre cliente y la consultora es cerrada, es decir, se tiene un tiempo estipulado de salida del producto. Se busca cerrar los requisitos y, una vez hecho, se comienza a buscar el desarrollador.

El tiempo y el dinero requeridos se estiman suponiendo el tiempo necesario, siguiendo a veces un proceso de licitación, es decir, que entre empresas con competencia se ajustan las necesidades al mínimo.

También hay penalizaciones en caso de no cumplir con los objetivos pactados-.

Estos errores son, comúnmente, los siguientes: repetición de código, falta de comentarios, código muerto, etc.

Por otra parte, hay una gran dependencia del desarrollador que diseño el código a l ahora de querer realizar algún cambio.

**DIFERENCIAS ENTRE VERSION Y RELEASE**

Versión: no necesariamente la ve un usuario.

Release: lo que ve el usuario a la hora de descargar.

TODAS LAS RELEASE SON VERSIONES, PERO NO AL REVÉS NECESARIAMENTE.

Un buen modo de prevenir errores y mejorar la eficiencia es realizar integraciones continuas, lo cual implica testing casi a diario.

Lo que pretenden los modelos de desarrollo ágiles es romper el muro creado por otros modelos entre le testing y las operaciones

**INFORMACION DE AGILEMANIFESTO.ORG:**

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

* **Individuos e interacciones** sobre procesos y herramientas.
* **Software funcionado** sobre documentación extensiva.
* **Colaboración con el cliente** sobre negociación contractual
* **Respuesta ante el cambio** sobre seguir un plan.

Esto es, aunque valoramos los elementos de la derecha, valoramos mas los de la izquierda.

A su vez, contemplamos los siguientes doce principios sobre el desarrollo ágil:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua del SW con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo.
3. Entregamos SW funcional frecuentemente con preferencia al periodo de tiempo mas corto posible.
4. Los responsables del negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados.
6. El éxodo mas eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El SW funcionando es l a medida principal de progreso.
8. Los procesos ágiles promueven el desarrollo sostenible.
9. La atención continua a la excelencia técnica y al buen diseño mejora la agilidad.
10. La simplicidad, arte de maximizar el trabajo no realizado, es esencial.
11. Las mejores arquitectura, requisitos y diseños emergen de equipos auto-organizados
12. A intervalos regulares el equipo reflexiona sobre cómo ser mas efectivo para, a continuación, ajustar y perfeccionar su comportamiento en consecuencia..

**PEOPLEWARE**

Management 3.0 – es lo que se busca. Estructura pensada para que el equipo y la persona sean lo más importante. Además, en los equipos ágiles se distinguen varios niveles: una persona es la mejor y los demás miran, todos son los mejores, … Hay 5 niveles de equipos de desarrollo ágil, y hay que intentar estar siempre entre el 4 y el 5.

La diferencia entre los modelos tradicionales frente a los ágiles es que, estos últimos, se centran en la humanidad, mientras que los primeros lo hacen en la especialización, lo cual no es óptimo.

Por otro lado, hay que procurar seguir la ley de Brooks si se quiere llevar a cabo un equipo ágil competente. Esta dice que, a más personas dentro de un proyecto, menos productividad habrá.

Trampa de la productividad: alguien lleva un producto tarde, llega el del PowerPoint y mete a 10 personas más para intentar adelantar el proyecto, cuando lo que está haciendo realmente es retrasarlo. Este acto de meter personas sin cualificar en un proyecto se denomina BODYSHOPPING, y lo hacen bastantes empresas.

Frase a recordar: MODERN AGILE MAKE PEOPLE AWESOME, DELIVER VALUE CONTINOUSLY, EXPERIMENT AND LEARN RAPIDLY, MAKE SAFETY A PREREQUISITE.

Tipos de un grupo de desarrollo ágil:

* Ley del más fuerte (1).
* Lo hago por trabajo (2).
* Yo soy el mejor (3).
* Todos somos los mejores (4): donde hay identidad de grupo, tienen el mismo título, actúan por unos valores y una cause noble.

**PEOPLEWARE EN 6 AREAS:**

* Recruitment: cómo hacer que venga gente más idónea para un proyecto (meetups).
* Equipo de alta productividad: se pone el foco en el equipo o en el proyecto. En muchos sitios se montan proyectos donde la gente se va metiendo. Tiene un problema, que es que el producto no tiene una fecha fija. En peopleware hay que refrescar la idea de equipo, von ayudas como el numero personas (a partir de 9 personas un grupo es improductivo).
* Felicidad (mediante NIKO NIKOS y KUDOS): un equipo feliz y motivado es mucho más productivo. Los NIKO NIKOS y los KUDOS se suelen gestionar en cortos plazos de tiempo como, por ejemplo, de un SPRINT a otro.
* Entorno: importa en la productividad del equipo. Cada vez que se interfiere a alguien, hay aproximadamente 15 minutos de perdida de concentración.
* Auto Organización: consistente en que el equipo de desarrollo se encargue de repartirse los trabajos en función de sus capacidades. Para ello se pueden crear tablas de multifuncionalidad y de delegación, las cuales ayudan bastante a esta tarea.

**RECRUITMENT**

**FELICIDAD**

**VISION DE EQUIPO**

**TAMAÑO**

**ENTORNO**

**MULTIFUNCIONALIDAD**

**AUTO ORGANIZACION**

**PEOPLEWARE**

A la hora de realizar los SPRINTS, es importante realizar los mapas de historias de usuario, crear un MVP – producto mínimo viable -, realizar un desarrollo dirigido por hipótesis y llevar a cabo un continuous delivery, consistente en intentar entregar versiones lo más rápido posible. Más tarde explicaremos el porqué de realizarlo así.

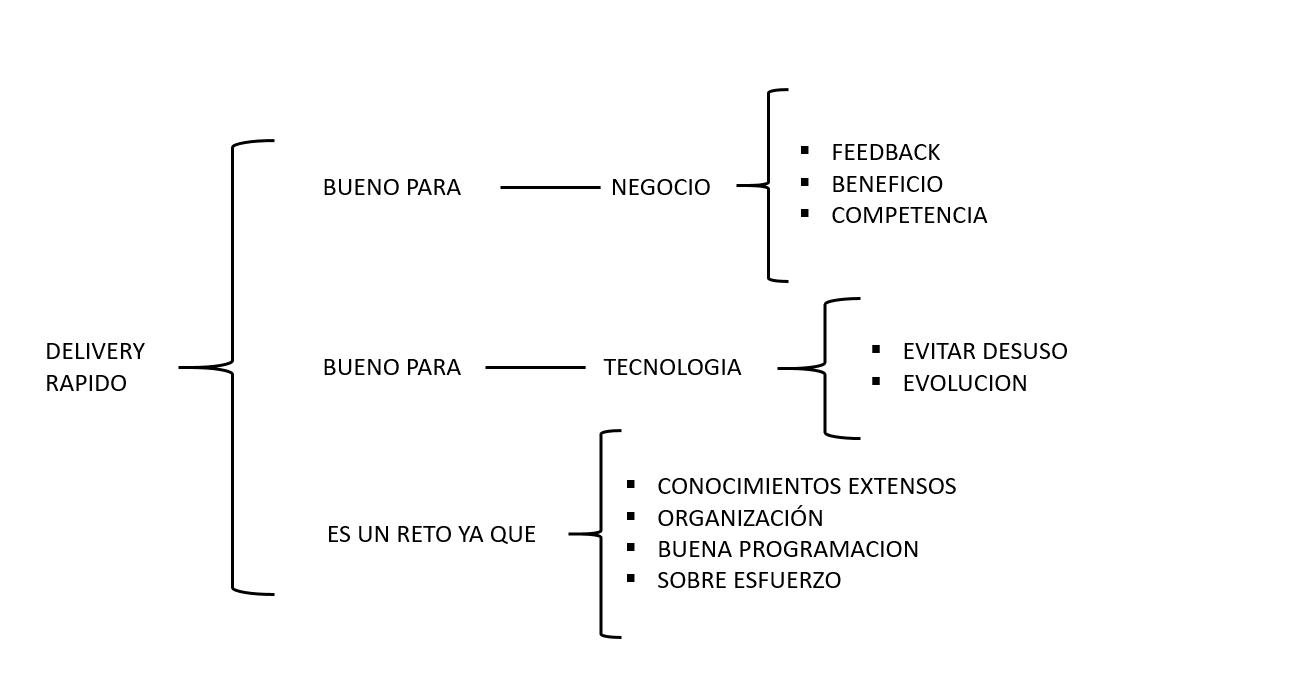
En cuanto a los mapas de historias de usuario se puede optar por realizar un product backlog bidimensional, que en la parte de arriba tenga las EPIC o epopeyas, que son historias de usuario muy importantes para el proyecto, mientras en la parte de debajo se pondrán algunas más secundarias, y los desgloses de las epopeyas. Se debe especificar también cuál será el MVP.

De cara a las historias de usuario se puede seguir el modelo de las 3 C’s: CARD; CONFIRMATION; COLLABORATION.

La figura del SCRUM Master en detalle se podría definir así:

* No es el jefe del equipo, es decir, no despide a nadie.
* El proceso es su principal responsabilidad: se tiene que encargar de las duraciones de los SPRINT y de mejorar los dailys.
* Defiende al equipo de otros integrantes, como el product owner.

El proceso de entrega de versiones hasta que se lleva a cabo la primera release pasa por los puntos de: versión, incremento, working software y producto potencialmente entregable.



**BUENAS PRACTICAS PARA EL DESARROLLO SOFTWARE:**

* Trazabilidad.
* Documentación.
* Testing
* Refactorización.
* Regresión.
* Modularización.
* Control de versiones.
* Integración continua.
* Nombres identificativos.

Y un largo etcétera.

**ESTIMACIONES (HEART OF AGILE)**

El proceso ágil tiene 4 partes:

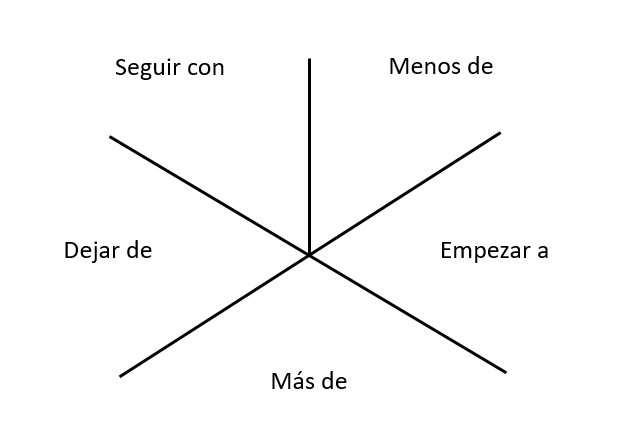
* Entregar.
* Colaborar.
* Mejorar.
* Reflexionar -> retrospectivas.

**DIFERENCIA ENTRE RETROSPECTIVA Y REVIEW**

La retrospectiva se encargar de evaluar el cómo has hecho el SPRINT, mientras que la review se encarga de evaluar el qué has hecho, los objetivos logrados y demás.

El líder encargado de que se realice una retrospectiva eficiente es el SCRUM Master, ya que es el encargado del proyecto., aunque todos participan en la creación de ésta.

Uno de los métodos de retrospectiva más utilizados es el método en estrella:



**ESTIMACIONES**

Deberían realizar las estimaciones el equipo de desarrollo, con ayuda del SCRUM Master y el PRODUCT OWNER; pero sin la incursión de ambos en ellas.

Tienen varias maneras de realizarse: bien por puntos de historia (dificultad o tiempo siguiendo series de numeros), o mediante planning Poker.

Se deben realizar en el planning de cada SPRINT, ya que se debe estimar cuánto se va a tardar en realizar cada tarea propuesta para ese SPRINT.

Cabe decir que se deben asignar puntos de historia por comparativa entre las distintias historias de usuario.

Un error frecuente es el de creer que por meter más actividades en el product backlog se mejora el producto, siendo totalmente al revés, es mejor tener pocas H.U. valiosas que muchas que sean secundarias. En la review del producto, se muestra cómo va al StakeHolder, es decir, se enseña lo realizado en ese SPRINT.

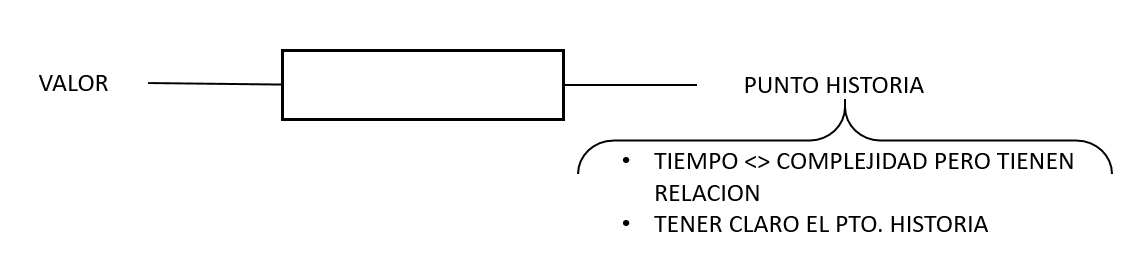
Posibles malas prácticas que te puedan llevar a realizar un mal SPRINT son que el equipo de desarrollo tenga “miedo” a las historias de usuario que hay que crear, y que se premie más la cantidad que el valor de las mismas.

Aspectos que se pueden englobar dentro del término TIME BOX:

* Tiempos orientativos: se sabe cuándo empieza y termina el SPRINT.
* SPRINT
  + PLANNING: con tiempo exponencial p.e. 8 h/ 30 días.
  + DAILY.
  + REVIEW.
  + RETROSPECTIVA.

El SCRUM Master es el que se encarga de cortar (si son 15 minutos de daily, a los 15 minutos se e para)

COMO ESTIMAR



Estimar en PS es como adivinar, tanto el cuándo voy a tener el producto (tiempo que se tardará) como el cuánto voy a poder hacer conforme a ese tiempo.

La estimación siempre es al futuro y se realiza en las primeras fases del sprint, es decir, en el sprint planning. Por el contrario, en el modelo en cascada, se realiza mucho antes, una vez tomados los requisitos.

La estimación es importante debido a su relación con el dinero. A veces, la presión externa llevará a malas estimaciones, ya que te comienzan a estrenar con tiempo, cambios en los requisitos y demás.

Para realizar una estimación más exacta, se deben tener en cuenta estos conocimientos:

* Necesidades del proyecto: qué tiene que hacer.
* Equipo y su experiencia.
* Cuánto cambiará el proyecto a lo largo del tiempo.

En el modelo en cascada, por el contrario, nada cambiaba y todo lo que se debía hacer estaba en la fase de toma de requisitos.

Por tanto, en un modelo ágil, cuanto menos se cumplan los conocimientos puestos arriba peor se podrá realizar la precisión del proyecto.

Por otro lado, está bien conocer unas “diferencias” entre albañiles e informáticos ya que, de cara a las empresas, muchas te compararán con uno a la hora de realizar trabajos:

* Trabajo físico vs. Intelectual.
* Complejidad.
* El albañil puede haber hecho ese trabajo muchas veces.
* Evolución continua del SW.
* Física, matemáticas.
* El SW está preparado para el cambio.
* Dependencia entre tareas
* A más número de trabajadores, no tiene por qué disminuir la productividad.

**INCEPTION**

Una Inception, es un conjunto de dinámicas orientadas a enfocar a todas las personas involucradas en un proyecto hacia un mismo objetivo, reduciendo muchas de las incertidumbres, ayudando a explicar los riesgos más evidentes y poniendo en común las expectativas de todos.

El porqué de hacer una inception se podría resolver con las preguntas de ¿qué es lo que mata a la mayoría de proyectos? Y ¿qué podríamos haber hecho?

Para preparar una inception es necesario reunir a todas las personas necesarias, buscar como objetivos los MVP del proyecto y conocerse mutuamente.

Una inception tiene una serie de tareas que se pueden englobar en el “INCEPTION DECK”:

1. ¿Por qué estamos haciendo este proyecto?: ir dando razones hasta llegar a la principal.
2. The elevador pitch: método que se compone de la siguiente tabla:

|  |  |
| --- | --- |
| PARA | CLIENTE OBJETIVO |
| QUE | OPORTUNIDAD/NECESIDAD |
| EL PROYECTO | NOMBRE DEL PRODUCOT |
| ES UN | CATEGORÍA/DESCRIPCIÓN |
| QUE | RAZÓN PARA COMPRAR |
| AL CONTRARIO DE | PRODUCTO ALTERNATIVO |
| NUESTRO PRODUCTO | DIFERENCIA PRIMARIA |

1. Product Box: caja que contendrá el producto. Debe ser llamativa para el cliente.
2. Lista de NO hacer: también divisible en esta tabla

|  |  |  |
| --- | --- | --- |
| EN ALCANCE | FUERA DE ALCANCE | EN DUDA |
|  |  |  |
|  |  |  |

1. La comunidad del proyecto: compuesta por los Shakeholders, testera, usuarios y el quiso de desarrollo
2. Tiempo: estimación del tiempo que llevará el terminar el producto.
3. Enseñar la solución: en la que se establecen las expectativas tecnológicas y de herramienta, se visualiza el alcance del producto y se comunican los posibles riesgos a tener en cuenta (¿Qué nos mantiene despiertos por las noches?).
4. Slogan.
5. Logo del producto.

Por tanto, la técnica inception se podría resumir así:

* Qué estamos construyendo y por qué.
* Qué es diferenciados en nuestro proyecto.
* Qué está dentro del alcance.
* Quiénes son los interesados en el proyecto.
* Cómo construiremos la solución.
* Cuáles son lo riesgos e impedimento a los que nos enfrentamos.
* Cómo de grande vemos el proyecto.
* Dónde debemos ser firmes y donde podemos ser flexibles.
* Aproximadamente cuánto nos costará una primera entrega.

**LEAN Y METODO KANBAN**

LEAN comenzó con un modelo que implementó Toyota a la hora de trabajar en la automoción. Nace como un modelo PULL, frente a los PUSH que se habñian asentado en el cascada, esto consiste en descartar o tirar (waste) todo lo que no aporta nada al usuario, al considerarse desperdicio. Esto es similar al desarrollo de un MVP en SCRUM. A raíz de este modelo trabajo, aproximadamente nación en 2001 el término de Agilidad.

El método LEAN ha diversificado en varios tipos o campos, como el LEAN STARTUP, LEAN CHANGE MANAGEMENT (gestión de cambio LEAN), LEAN LEARNING, LEAN CANVAS (modelos de negocio visuales [canvas = lienzo]), LEAN SOFTWARE DEVELOPMENT, …

KANBAN es la técnica estrella del modelo LEAN. Sus 3 principios son:

1. Visualiza las fases/proceso del producto.
2. Limita el WIP (work in process/progress) mediante el uso de números en las columnas de las historias de usuario.
3. Mide tiempos mediante leads y ciclos.

En KANBAN no hay SPRINTS, solo limitaciones. Limita las columnas de las historias de usuario por valor, pudiendo variar según la importancia que se le de (al contrario de SCRUM, donde se dejaba el valor de las áreas decidido en el SPRINT PLANNING y no se tocaba salvo emergencias).

KANBAN es una técnica más desordenada pero necesita más madurez del equipo.

¿WIP en SCRUM? -> se puede representar como poner un máximo en tareas/persona, centrarse en las historias de usuario de mayor a menor valor.

EN KANBAN, SE DEBE EVITAR EL CAMBIO DE CONTEXTO. Como última característica, comentar que en KANBAN no tiene por qué haber una secuencialidad predefinida, al ser una técnica más abierta a cambios.

**RESUMEN DE LA GUÍA SCRUM**

SCRUM se puede definir como un marco de trabajo por el cual las persona pueden acometer problemas complejos adaptativos.

SCRUM se caracteriza por ser:

* Ligero.
* Fácil de aprender.
* Extremadamente difícil de llegar a dominar.

El equipo SCRUM está formado por los siguientes componentes:

* Product Owner: es el responsable de maximizar el valor del producto y del trabajo del equipo de desarrollo. Además es la única persona responsable de gestionar la Lista del Producto (Product Backlog)
* Equipo de Desarrollo: consiste en los profesionales que desempeñan el trabajo de entregar un incremento de producto “terminado”, que s epoda poner en producción, al final de cada SPRINT.
* SCRUM Master: responsable de asegurar que SCRUM es entendido y adoptado. Es un líder que está al servicio del equipo SCRUM.

Eventos de SCRUM:

En Scrum existen eventos predefinidos con el fin de crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum. Todos los eventos son bloques de tiempo (*time-boxes*), de tal modo que todos tienen una duración máxima. Una vez que comienza un Sprint, su duración es fija

**El Sprint**

El corazón de Scrum es el Sprint, es un bloque de tiempo (*time-box*) de un mes o menos durante el cual se crea un incremento de producto “Terminado”, utilizable y potencialmente desplegable.

Durante el Sprint:

* No se realizan cambios que puedan afectar al Objetivo del Sprint (*Sprint Goal*);
* Los objetivos de calidad no disminuyen; y,
* El alcance puede ser clarificado y renegociado entre el Dueño de Producto y el Equipo de Desarrollo a medida que se va aprendiendo más.

**Reunión de Planificación de Sprint (*Sprint Planni***

El trabajo a realizar durante el Sprint se planifica en la Reunión de Planificación de Sprint. Este plan se crea mediante el trabajo colaborativo del Equipo Scrum completo. Se realizan las siguientes preguntas:

* ¿Qué puede ser terminado en este Sprint?
* ¿Cómo se conseguirá completar el trabajo seleccionado?

**Scrum Diario (*Daily Scrum*)**

El Scrum Diario es una reunión con un bloque de tiempo de 15 minutos para que el Equipo de Desarrollo sincronice sus actividades y cree un plan para las siguientes 24 horas.

**Revisión de Sprint (*Sprint Review*)**

Al final del Sprint se lleva a cabo una Revisión de Sprint para inspeccionar el Incremento y adaptar la Lista de Producto si fuese necesario. Durante la Revisión de Sprint, el Equipo Scrum y los interesados colaboran acerca de lo que se hizo durante el Sprint

**Retrospectiva de Sprint (*Sprint Retrospective*)**

La Retrospectiva de Sprint es una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

La Retrospectiva de Sprint tiene lugar después de la Revisión de Sprint y antes de la siguiente Reunión de Planificación de Sprint. Se trata de una reunión restringida a un bloque de tiempo de tres horas para Sprints de un mes. Para Sprints más cortos se reserva un tiempo proporcionalmente menor. El Scrum Master se asegura de que el evento se lleve a cabo y que los asistentes entiendan su propósito. El Scrum Master enseña a todos a mantener el evento dentro del bloque de tiempo fijado. El Scrum Master participa en la reunión como un miembro del equipo ya que la responsabilidad del proceso Scrum recae sobre él.

**Artefactos de Scrum**

**Lista de Producto (*Product Backlog*)**

La Lista de Producto es una lista ordenada de todo lo que podría ser necesario en el producto, y es la única fuente de requisitos para cualquier cambio a realizarse en el producto. El Dueño de Producto (*Product Owner*) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

**Lista de Pendientes del Sprint (*Sprint Backlog*)**

La Lista de Pendientes del Sprint es el conjunto de elementos de la Lista de Producto seleccionados para el Sprint, más un plan para entregar el Incremento de producto y conseguir el Objetivo del Sprint. La Lista de Pendientes del Sprint es una predicción hecha por el Equipo de Desarrollo acerca de qué funcionalidad formará parte del próximo Incremento y del trabajo necesario para entregar esa funcionalidad en un Incremento “Terminado”.

**Incremento**

El Incremento es la suma de todos los elementos de la Lista de Producto completados durante un Sprint y el valor de los incrementos de todos los Sprints anteriores. Al final de un Sprint, el nuevo Incremento debe estar “Terminado”, lo cual significa que está en condiciones de ser utilizado y que cumple la Definición de “Terminado” del Equipo Scrum. El incremento debe estar en condiciones de utilizarse sin importar si el Dueño de Producto decide liberarlo o no.

**Definición de “Terminado” (*Definition of “Done”*)**

Cuando un elemento de la Lista de Producto o un Incremento se describe como “Terminado”, todo el mundo debe entender lo que significa “Terminado”. Aunque esto varía significativamente para cada Equipo Scrum, los miembros del Equipo deben tener un entendimiento compartido de lo que significa que el trabajo esté completado, para asegurar la transparencia. Esta es la definición de “Terminado” para el Equipo Scrum y se utiliza para evaluar cuándo se ha completado el trabajo sobre el Incremento de producto.