**Ins$tuto Profesional AIEP Spa.**
Dirección Nacional de Educación Con1nua
**Programa Talento Digital 2021**

—

Curso Emprendimiento Digital con Tecnologías Web

# Módulo 10: Visión del producto en un emprendimiento digital (6 unidades)

—

**MANUAL DEL PARTICIPANTE**

**Temario de unidades**

1. Unidad 1

**1.1) Metodologías Ágiles**
Temas: Orígenes. Diferencias con Modelos Tradicionales de Desarrollo: Waterfall; RUP; Modelo PMBOK. Diferencias entre “Agile”, “Agilidad” y “Agilismo”. Manifiesto Ágil. Valores y Principios. Otras metodologías ágiles.

Hay algo con lo cual los gerentes de proyectos, y específicamente los gerentes de proyectos digitales siempre han estado obsesionados, es la metodología. Pero las metodologías son un gran problema en el mundo de la administración de

proyectos. ¿Por qué articulamos todo lo que hacemos a su alrededor? ¿Realmente importan tanto? La respuesta es SI.

## Agile Vs Waterfall: Principios

### Principios básicos de Waterfall

**Recopilación de requisitos al inicio:**

Los desarrolladores y los clientes están de acuerdo en lo que se entregará al principio del ciclo de vida del desarrollo. Esto puede hacer que la planificación y el diseño sean más sencillos. El progreso se mide más fácilmente, ya que el alcance completo del trabajo se conoce de antemano. Una buena documentación técnica es parte de los resultados y es más fácil para los nuevos programadores ponerse al día durante la fase de mantenimiento.

**Trabajo completado secuencialmente en fases:**

Cada fase generalmente comienza a medida que termina la anterior. A lo largo del desarrollo, varios miembros del equipo pueden participar o continuar con otros trabajos. Por ejemplo, los evaluadores pueden preparar scripts de prueba a partir de la documentación de requisitos mientras la codificación está en curso.
El enfoque es muy estructurado y puede ser más fácil medir el progreso con hitos claramente definidos, y más fácil de planificar al principio.

**Las pruebas se producen al final del desarrollo:**

Las pruebas son más sencillas de planificar y ejecutar, ya que se puede hacer por referencia a los escenarios definidos en la especificación funcional, al final de la fase de desarrollo.

**Los clientes o partes interesadas no necesitan involucrarse mucho:**

A excepción de las revisiones, aprobaciones y reuniones de estado, la presencia del cliente no es estrictamente necesaria después de la fase de requisitos.

## Principios básicos de Scrum

En 2001, un grupo de desarrolladores se reunió y desarrolló lo que ahora se conoce como el Manifiesto Ágil, que describe los cuatro valores que vieron que respaldan esta forma de abordar el desarrollo. En estas 200 palabras, prácticamente cambiaron el aspecto del desarrollo de software y generaron industrias masivas a partir de él.
Si bien Agile es el marco, Scrum es el enfoque o modelo de desarrollo que utiliza Agile. Como verás, tanto el marco como los aspectos específicos de Scrum tienen sentido cuando piensas en el desarrollo de un proyecto digital.

### Individuos e interacciones por encima de procesos y herramientas:

En primer lugar, se enfoca en individuos e interacciones por encima de procesos y herramientas. La comunicación es clave, no los procesos que ejecutan su proyecto. Dentro de Scrum, esto significa el equipo auto-organizativo y multifuncional.

### Software de trabajo por encima de documentación completa:

Existe un fuerte enfoque en obtener productos que se pueden enviar con mayor rapidez, en lugar de gastar mucho tiempo en escribir los requisitos. En Scrum, las iteraciones de trabajo con límite de tiempo se ejecutan con un producto distribuible al final de cada iteración.

### Colaboración con el cliente por encima de la negociación del contrato:

Los valores de Agile especifican la colaboración del cliente, trabajar con el cliente en todos los puntos y participando activamente en todo el proceso. Scrum tiene una participación constante y regular del cliente.

### Responder al cambio por encima de seguir un plan:

En lugar de considerar al cambio como el enemigo, estar en una posición para considerar al cambio como algo bueno y responder a él es fundamental para el marco ágil. Scrum tiene requisitos que evolucionan constantemente, y el cambio se acepta.

Todas estas cosas realmente tienen sentido cuando se habla de lo digital. La comunicación como equipo, hacer que algo funcione rápidamente, involucrar al cliente y, cuando lo digital es un panorama en constante cambio, poder responder rápidamente al cambio.

...

**1.2) Abanico de Métodos**

Temas: DSDM - Atern; Lean SoUware Development; Extreme Programming (XP);

Kanban.

## Qué es DSDM

DSDM es un método ágil que se enfoca en el ciclo de vida completo del proyecto. DSDM (formalmente conocido como Método de desarrollo dinámico del sistema) se creó en 1994, después de que los gerentes de proyectos que usaban RAD (Desarrollo rápido de aplicaciones) buscaran más gobernanza y disciplina para esta nueva forma iterativa de trabajo.

El éxito de DSDM se debe a la filosofía de que cualquier proyecto debe estar alineado con objetivos estratégicos claramente definidos y centrarse en la entrega temprana de beneficios reales para el negocio. El respaldo de esta filosofía con los ocho principios permite a los equipos mantener el enfoque y alcanzar los objetivos del proyecto.

Existen ocho principios de la metodología de DSDM y estos son:

- Centrarse en la necesidad comercial
- Entregar a tiempo
- Colaborar
- Nunca comprometer la calidad
- Construir incrementalmente a partir de cimientos firmes
- Desarrollar iterativamente
- Comunicarse de forma continua y clara
- Demostrar control

El término “Lean” o “Lean Manufacturing” (cuya traducción sería algo así como fabricación esbelta) es otro término que, al igual que el Kanban , tiene su origen en Toyota. De hecho, “Lean” es sinónimo de Toyota Production System, una estrategia de fabricación aplicada con mucho éxito en Japón y ahora muy famosa en el mundo del software, muchas veces bajo el término de Lean Software Development. En los 50 la industria japonesa estaba recuperándose de la segunda guerra mundial y logró con gran éxito aplicar a sus fábricas de coches los conceptos de calidad en la producción creados por los principales gurús estadounidenses, de entre los que destaca Deming. La paradoja fue que siendo métodos idealmente originados por estadounidenses… fueron aplicados por los japoneses, convirtiendo a Japón líder en la industria automovilística, pasando por encima de los EEUU.

El artífice del Lean, quien introdujo esta nueva manera de fabricar en Toyota, fue

Taiichi Ohno (1912 – 1990), cuya estrategia se fundamentó en tres bases:

– Construir sólo lo necesario.

– Eliminar todo aquello que no añade valor.

– Parar si algo no va bien (lo que está relacionado con el principio de cero defectos).

Además, conviene destacar que el Lean incluye siete importantes principios, los siguientes:

1. Eliminar desperdicios (Eliminating Waste)
2. Amplificar el aprendizaje (Amplifying Learning)
3. Decidir lo más tarde posible (Deciding as Late as Possible)
4. Entrega lo más rápido posible (Delivering as Fast as Possible)
5. Capacitar, potenciar, al equipo (Empowering the Team)
6. Construir con integridad (Building Integrity In)

7. Ver el todo (Seeing the Whole)

Y en lo que refiere al primer punto, los desperdicios, el Lean habla de que un desperdicio es todo aquello innecesario, todo añadido del que se puede prescindir, donde se destacan los siguientes siete siguientes desperdicios:

1. Sobreproducción
2. Tiempo de espera
3. Transporte
4. Inventarios innecesarios
5. Transportes innecesarios
6. Defectos
7. Sobre procesamiento, o procesos inadecuados

Hay quien añade un desperdicio más: el talento humano (desperdiciando la creatividad). En la última edición del libro Lean Thinking se añadió este octavo desperdicio.

**Lean Software Development**

Lean Software Development es una adaptación del “Lean Manufacturing” de Toyota al desarrollo software ágil. Lean Software Development es una metodología ágil desarrollada por los mencionados Mary and Tom Poppendieck. Dicha metodología, como bien dice su nombre, contempla los principios Lean de Toyota.

En Lean Software Development podemos encontrar una adaptación al desarrollo software de los siete principios Lean, los cuales comentamos antes. De hecho, el libro de de Mary y Tom Poppendieck, el “Lean Software Development: An Agile Toolkit”, dedica un capítulo a cada principio Lean. Y así, la metodología tiene como objetivo eliminar desperdicios, seleccionando aquellas características que realmente aportan valor, y da especial importancia a la velocidad y la eficiencia.

## 2.1) Introducción a Scrum

Temas: Fundamentos y principios.

La guía de Scrum lo define como “un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos, a la vez que entregar productos del máximo valor posible productiva y creativamente”.

Desde sus inicios y alrededor de 30 años, este marco de trabajo ha sido usado ampliamente en todo el mundo, **aunque en latinoamerica es posible que lleve alrededor de 10 años en el contexto del desarrollo de productos de software.**

Precisamente, para el desarrollo de nuevos productos de software y la evolución de los existentes, ha demostrado ser una buena alternativa, teniendo en cuenta que la posibilidad de tener un incremento terminado al final de cada iteración, nos confiere la velocidad y la flexibilidad necesarias para reaccionar adecuadamente a los aprendizajes y nuevos retos del mercado que nos confiere la retroalimentación del uso de dichos incrementos.

Para describir los elementos que componen Scrum, basta con darle una buena leída a la guía de Scrum y por este motivo no pretendo transcribir completamente esa guía (su lectura juiciosa la dejo para cada uno), pero sí voy a extraer algunos elementos que juzgo esenciales, de tal forma que podamos complementar el entendimiento de sus fundamentos.

### Los pilares

**Transparencia:** los aspectos significativos del proceso deben ser visibles para aquellos que son responsables del resultado.

**Inspección:** los usuarios de Scrum deben inspeccionar frecuentemente los artefactos y el progreso hacia el objetivo para detectar variaciones indeseadas

**Adaptación:** cuando a partir de la inspección se detectan variaciones, deben

realizarse los ajustes necesarios lo antes posible.

### Los valores

- Compromiso
- Coraje
- Foco
- Apertura
- Respeto

**2.2) Prác$cas de Scrum**

Temas: Sprint Planning; Daily Scrum; Review; Retrospec1va.

### Los eventos

Son bloques de tiempo para crear regularidad y minimizar la necesidad de reuniones no definidas en Scrum.

### Sprint

Es un contenedor de todos los eventos, tiene una duración de un mes o menos durante el cual se crea el incremento de producto “terminado”

Los Sprints contienen y consisten en la Planificación del Sprint (Sprint planning), los Scrums Diarios (Daily Scrums), el trabajo de desarrollo, la Revisión del Sprint (Sprint Review), y la Retrospectiva del Sprint (Sprint Retrospective).

**2.3) Artefactos de Scrum**

Temas: Product Backlog; Sprint Backlog; Incremento.

Los artefactosRepresentan el trabajo o valor en diversas formas que son útiles para proporcionar transparencia y oportunidades de inspección y adaptación.

### Lista de producto (Product Backlog)

Es una lista ordenada de todo lo que se conoce que es necesario en el producto.

### Lista de pendientes del Sprint (Sprint Backlog)

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.

### Incremento

Es la suma de todos los elementos de la lista de producto completados durante el Sprint.

## 2.4) Roles de Scrum

Temas: Scrum Master; Scrum Developers; Product Owner.

### El equipo Scrum

Consiste en un Dueño de producto (Product owner), el Equipo de Desarrollo (Development team) y un Scrum Master. Es auto organizado y multifuncional. Entregan productos de forma iterativa e incremental.

### El Dueño de Producto

Encargado de maximizar el valor y de gestionar la Lista de producto (Product

backlog)

## El Equipo de Desarrollo

Son los profesionales que realizan el trabajo de entregar un incremento de producto “terminado”

## Scrum Master

Es el responsable de promover y apoyar Scrum, ayudando a todos a entender la teoría, prácticas, reglas y valores. Es un líder que está al servicio del Equipo Scrum y de la organización.

3. Unidad 3

**3.1) La visión del Producto en Scrum**

Temas: Producto y su Visión. Cómo Es1mar. Definición de Alcance. Release Management. Scrum Como modelo Itera1vo y Evolu1vo. Mínimo Producto Viable.

## La visión y el por qué

Una de las preguntas que el Chief Product Officer (CPO)deberia hacer constantemente al equipo es: **¿por qué queremos hacer eso?**. Y es en las respuestas donde descubres cuando hay una razón justificada con datos, o simplemente una idea que puede que inspire pero que no esté fundamentada,

## La visión y la priorización

Cuando hablo de la visión y la priorización, me gusta imaginarme a un equipo de escaladores que quieren alcanzar el pico de una montaña. Hay múltiples caminos para conseguirlo, algunos más seguros y largos, otros más cortos y quizá arriesgados, y en función de la experiencia y conocimiento del terreno que tengan, pues trazarán su ruta y seguramente, un plan alternativo por si algo no sale como esperaban. La priorización podría ser algo parecido, en función del tiempo,

complejidad e impacto estimado de lo que queremos hacer – y sobre todo aprender – vamos a poner un orden de pasos o en este caso, un orden en los elementos de nuestro Product Backlog.

El Product Owner y su capacidad para transmitir la visión del producto así como comunicar adecuadamente por qué hacemos lo que hacemos al equipo de desarrollo, se nota muchísimo en el momento en el que el equipo demuestra el incremento de producto en el que ha trabajado. **Un equipo que conoce el producto, demuestra valor y impacto esperado, un equipo que hace lo que el Product Owner le ha dicho presenta acciones implementadas y habla poco del producto e impacto.** Cuando hacemos una demostración del incremento de producto que queremos entregar, todos los roles que trabajan entorno a él, deben saber que el valor que aporta está ligado a alcanzar la visión del producto, esa visión que todos, con unas actividades u otras, intentamos hacer realidad.

**El Product Owner debe conocer y sentir la visión del producto**, debe usarla para **elegir** qué debemos construir en el producto, la debe **transmitir** al equipo y **facilitar** la labor de priorización y decir cómo conseguir el impacto deseado, así como **entregar valor** en la dirección adecuada.

Todo esto de la agilidad en desarrollo de software a implicado que existan incluso nuevos modelos de gestión ágil , hablamos de release management lo que tambien origina que se abran nuevas posiciones de trabajo coo los release managers y esto afecta no solo la industria del desarrollo de software como tal sino a las empresas en general particularmente en contextos de transformación digital.

El *Release Management* puede ser aplicado en varias industrias, y el rol va a variar mucho dependiendo en cuál se encuentre. Por ejemplo, un Software *Release Manager* se enfocará en el proceso de ingeniería de software y en cómo automatizarlo. Este rol generalmente es comparado con el de un *Project Manager*, sin embargo, el rol de *Release Manager* se considera uno de los más retadores del rubro ya que involucra distintos aspectos de la compañía, como planeamiento, seguimiento, manejo de riesgo, testeo, comunicación y lanzamiento, entre otros.

**¿Cuáles son las responsabilidades de un *Release Manager*?**

Algunas funciones típicas que debe cumplir este perfil son:

- Planificación de ventanas de lanzamiento y el ciclo de vida de lanzamiento general.
- Gestión de riesgos que pueden afectar el alcance de la versión.
- Comunicar todos los planes, compromisos y cambios clave del proyecto, incluidos los requisitos.
- Medir y monitorear el progreso.
- Manejar relaciones y coordinar los proyectos entre distintos equipos.

**Pasos clave en el Release Management**

Por otro lado, existen algunos pasos que hacen que un *Release Manager* será exitoso y cumpla satisfactoriamente con sus funciones:

1. Planificación de lanzamiento: en esta etapa, los gerentes de lanzamiento diseñarán pautas de lanzamiento que deben implementarse en toda la empresa.

2. Configuración de lanzamientos: los administradores de lanzamientos supervisarán los diversos aspectos de un proyecto antes de su implementación, asegurando que todos los equipos estén en camino y cumplan con las pautas acordadas.

3. Verificaciones de calidad: la calidad de la versión debe revisarse antes de que se inicie oficialmente un proyecto. El gerente de lanzamiento está a cargo de garantizar que los equipos de Garantía de calidad conozcan las pautas del proyecto y verifiquen que se cumplan estos estándares.

4. Implementación: después de comprobar la calidad, el proyecto está listo para

implementarse. El administrador de versiones sigue siendo responsable de garantizar que un proyecto se desarrolle sin problemas y de manera eficiente.

**¿Qué skills se necesita para ser un Release Manager?**

Los *Release Managers* necesitan una serie de habilidades técnicas para desempeñarse bien en su función, pero quizás lo más importante son las habilidades blandas que les permiten comunicarse, coordinar y liderar múltiples equipos dentro de una empresa.

Habilidades duras:

- Un buen nivel de conocimiento de PC.
- Una excelente comprensión del software y la programación.
- Los gerentes de lanzamiento generalmente se especializarán en un área específica de ingeniería, por lo que necesitarán tener un conocimiento experto de cualquier campo en el que estén trabajando.
- Comprensión de las líneas de entrega continua e integración continua (CD / CI).

Habilidades blandas:

- Capacidad para coordinar múltiples equipos para garantizar que las tareas se completen a tiempo con el calendario de lanzamiento.
- Liderazgo y habilidades analíticas.
- Habilidades de comunicación avanzadas, tanto escritas como verbales.
- Resolución de problemas.

Dado que muchas empresas ahora están trabajando con un desarrollo de software ágil y se realizan muchos más lanzamientos, existe un nuevo enfoque para los lanzamientos de software. Ahora los administradores de versiones deben usar herramientas de automatización de liberación de aplicaciones (ARA), que ayudan a cultivar DevOps e implementar la entrega continua (CD) rápidamente.

4. Unidad 4

**4.1) Historias de Usuario**

Temas: Qué es una historia de usuario. Reconociendo historias de usuario. Escribiendo historias de usuario. Modelo INVEST. Modelo SMART

## ¿QUÉ ES UNA HISTORIA DE USUARIO?

Las historias de usuario son descripciones **cortas y simples** de una característica **contada desde la perspectiva de la persona que desea la nueva capacidad**, generalmente un usuario o cliente del sistema. Por lo general, siguen una plantilla simple:

> Como **<Usuario>**
> Quiero **<algún objetivo>**
> Para que **<motivo>**

Las historias de los usuario a menudo se escriben en fichas o notas adhesivas, se almacenan en una caja y se organizan en paredes o mesas para facilitar la planificación y el debate. Como tal, cambian fuertemente el enfoque de escribir sobre las características a discutir. De hecho, **estas discusiones son más importantes que cualquier texto que se escriba**.

## EJEMPLOS DE HISTORIAS DE USUARIO

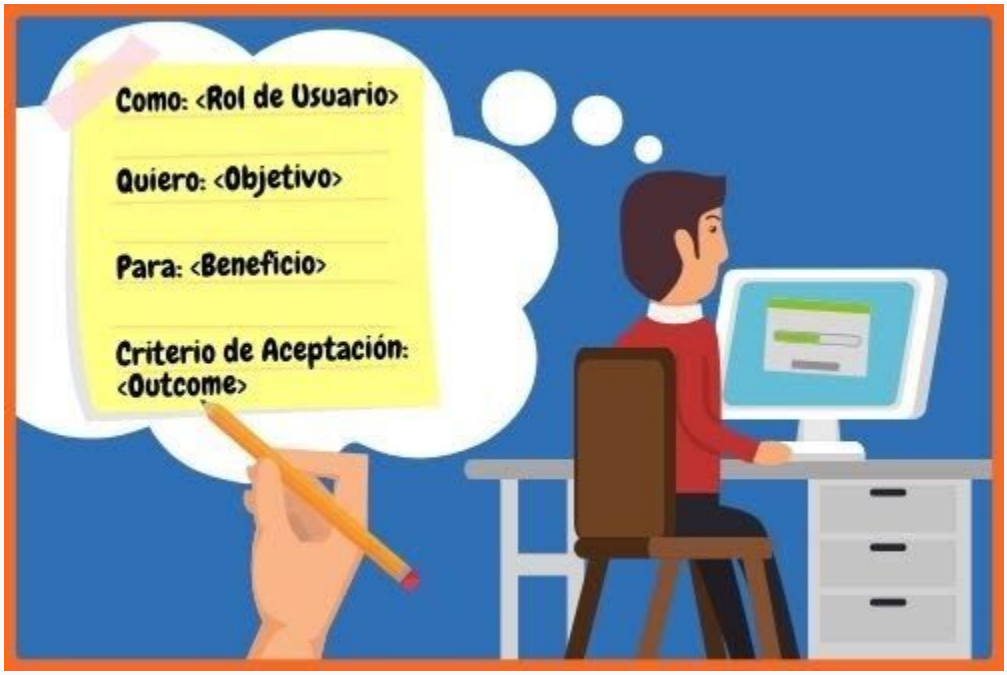
Uno de los beneficios de las historias de usuario ágiles es que **se pueden escribir con distintos niveles de detalle**. Podemos escribir una historia de usuario para cubrir grandes cantidades de funcionalidad. Estas grandes historias de usuario generalmente se conocen como épicas. Aquí hay un ejemplo épico de historia de usuario ágil de un producto de copia de seguridad de escritorio:

Como usuario, quiero hacer una copia de seguridad de todo mi disco duro.

## ¿QUÉ ES UNA HISTORIA DE USUARIO ÉPICA?

Debido a que **una épica en general es una historia de usuario demasiado grande para que un equipo ágil la complete en una iteración**, se divide en varias historias de usuario más pequeñas antes de que se trabaje en ella. La épica anterior podría dividirse en docenas (o posiblemente cientos), incluidos estos dos:

> **Como** usuario de poder, **quiero** especificar archivos o carpetas **para** realizar copias de seguridad en función del tamaño del archivo, la fecha de creación y la fecha de modificación.
> **Como** usuario, **quiero** indicar carpetas que no deben respaldarse **para** que mi unidad de respaldo no esté llena de cosas que no necesito guardar.




## ¿CÓMO SE AGREGAN LOS DETALLES A LAS HISTORIAS DE LOS USUARIO?

El detalle se puede agregar a las historias de usuario de dos maneras:

> Al dividir una historia de usuario en historias de usuarios múltiples y más pequeñas
> Al agregar **"Criterios de Aceptación"**.

Cuando una historia relativamente grande se divide en historias de usuario ágiles y múltiples, es natural suponer que se han agregado detalles. Después de todo,

se ha escrito más.

## ¿QUE ES EL CRITERIO DE ACEPTACIÓN?

Los criterios de aceptación son simplemente una prueba de alto nivel que **será cierta después de que se complete la historia del usuario ágil**. Considere lo siguiente como otro ejemplo de historia de usuario ágil:

Como vicepresidente de marketing, quiero seleccionar una temporada de vacaciones para revisar el rendimiento de campañas publicitarias pasadas para poder identificar las rentables.

El detalle podría agregarse a ese ejemplo de historia de usuario al agregar los siguientes criterios de aceptación:

> Asegurarse de que funcione con las principales fiestas minoristas: Navidad, Pascua, Día de la Madre, Día del Padre, Día del Trabajo, Año Nuevo.
> Las temporadas de vacaciones se pueden establecer de una fiesta a otra (como Día de Acción de Gracias a Navidad).
> Las temporadas de vacaciones se pueden establecer en un número de días antes de las vacaciones.

## ¿QUIÉN ESCRIBE HISTORIAS DE USUARIO?

Cualquiera puede escribir historias de usuario. **Es responsabilidad del Product Owner asegurarse de que exista una Product Backlog actualizado y priorizado** de historias de usuario ágiles, pero **eso no significa que el Product Owner es quien los escribe**. El transcurso de un buen proyecto ágil, debe contar con historia de usuario escritos por cada miembro del equipo.

Además, ten en cuenta que **quién escribe una historia de usuario es mucho menos importante que quién está involucrado en las discusiones de la misma.**


## ¿CUÁNDO SE ESCRIBEN LAS HISTORIAS DE USUARIO?

**Las historias de usuario se escriben en todo el proyecto ágil**. Por lo general, se lleva a cabo un taller de redacción de historias de usuario cerca del inicio del proyecto ágil. **Todos en el equipo participan con el objetivo de crear un Product Backlog** que describa por completo la funcionalidad que se agregará durante el transcurso del proyecto o un ciclo de lanzamiento de tres a seis meses.

Algunas de estas historias de usuario ágiles serán, sin duda, épicas. **Las épicas se descompondrán más tarde en historias más pequeñas que caben más fácilmente en una sola iteración**. Además, las historias nuevas se pueden escribir y agregar al Product Babklog en cualquier momento y por cualquier persona.

## ¿LAS HISTORIAS DE USUARIO REEMPLAZAN UN DOCUMENTO DE REQUISITOS?

Los proyectos ágiles, especialmente los de Scrum, usan un Product Backlog, que es una lista priorizada de la funcionalidad que se desarrollará en un producto o servicio. Aunque el Product Backlog pueden ser lo que el equipo desee, **las historias de los usuario han surgido como la mejor y más popular forma de Product Backlog**.

## UN PRODUCT BACKLOG ES UNA LISTA PRIORIZADA DE LA

## FUNCIONALIDAD QUE SE DESARROLLARÁ EN UN PRODUCTO O SERVICIO

Si bien el Product Backlog puede considerarse como un reemplazo del documento de requisitos de un proyecto tradicional, es importante recordar que **la parte escrita de una historia de usuario ágil ("Como usuario, quiero ...") está incompleta hasta que las discusiones sobre esa historia ocurren**.

A menudo es mejor pensar en la parte escrita como un indicador del requisito real. Las historias de usuario pueden apuntar a un diagrama que representa un flujo de trabajo, una hoja de cálculo que muestra cómo realizar un cálculo o cualquier otro artefacto que el propietario o equipo del producto desee.

5. Unidad 5

**5.1) Refinamiento del backlog**

Temas: Para qué sirve. Importancia de Refinar Historias. Cómo llevar sesiones de Refinamiento. Malas prác1cas.

El refinamiento del Product Backlog es un evento dedicado a agregar detalles, estimar y ordenar las historias de usuario que haya dentro del Product Backlog.

Es normal que los equipos Scrum tengan problemas a la hora de refinar el Product Backlog, ya que no existe una forma predeterminada de hacer este tipo de acciones dentro de un Sprint.

Esto tiene su justificación, ya que el refinement es un proceso continuo, que es único dentro de cada proyecto y de cada Equipo Scrum. Por eso cada equipo tiene que descubrir la frecuencia y la forma de poder agregar detalles y estimar tareas del Product Backlog.

### El refinamiento es un proceso, no un evento

### Qué es el refinamiento del backlog?

*El refinamiento del Product Backlog es el acto de añadir detalle, estimaciones y orden a los ítems del Product Backlog. Este es un proceso contínuo en el que el product Owner y el Development Team colaboran en los detalles de los ítems del Product Backlog.*

**¿Qué se hace en el refinamiento?**

Como define Scrum, en el Refinamiento se trabaja en dar detalle a los ítems del Product Backlog que se realizarán en los próximos Sprints, habitualmente entre los Sprints N+1 y N+3, realizándose en el Sprint actual (N). Para dar detalle se realizan múltiples actividades como:

- **Entender el alcance** de los ítems (p.e. características o requisitos a incorporar al producto).
- **Identificar las dependencias** funcionales y técnicas entre ítems.
- **Analizar funcionalmente** los ítems, usualmente considerando dichas dependencias.
- **Diseñar técnicamente** las soluciones escogidas para realizar los ítems.
- **Definir criterios y pruebas de aceptación**, ya sean más laxos o más esctrictos.
- **Estimar el costo** de realización de los ítems, ya sea en tiempo o en tamaño relativo (p.e. puntos/historia).

**¿Cuándo se hace el refinamiento?**

El Refinamiento se hace cuando se considera necesario y cuando es posible. Algunos factores pueden aconsejar en qué momento hacerlos, p.e.:

- En desarrollos de pocos Sprints puede realizarse principalmente en los primeros ciclos, pero siempre entregando producto al final de cada Sprint.
- En desarrollos mayores puede realizarse de manera periódica para facilitar la asistencia de todos los participantes.
- Las reuniones de refinamiento pueden ajustarse a la disponibilidad de las partes interesadas, como usuarios, expertos u otros roles, especialmente si es difícil convocarlos.

Adicionalmente, la preparación de los siguientes Sprints y la cantidad de Backlog refinado puede dictar la necesidad de realizar más o menos refinamiento.

**¿Quién hace el refinamiento?**

Existe una visión muy extendida que hace al **Product Owner responsable exclusivo de definir los ítems** y **pasárselos al Development Team para que los convierta en software**, derivada de la idea que estos últimos son básicamente técnicos (programadores, testers, etc.) y que no tienen capacidad de análisis. Esta visión como única opción es **falsa**, y **va contra la idea de un Equipo Scrum polivalente y colaborativo**.

6. Unidad 6

**6.1) Priorización**

**En qué consiste la práctica de priorización. Diferenciar lo Urgente de lo Importante. …**

El principio de Pareto establece que el 80% de los efectos provienen del 20% de las causas. Dicho de otra manera: el 80% de los beneficios de la empresa o del proyecto proceden del 20% del tiempo invertido por su personal. En un contexto de desarrollo Agile, la misma regla sería algo como que el 80% del valor del producto proviene del 20% de sus funcionalidades básicas. Pero, ¿cómo definir qué tareas incluir en ese 20% mágico? Obtener ese 20% para contener las tareas más vitales de un producto significa resolver el problema de la priorización de la manera más óptima.

La priorización como principio significa "hacer lo primero, primero". Como proceso, significa "evaluar un grupo de elementos y clasificarlos en orden de importancia o urgencia". En realidad, la priorización es nuestra actividad cotidiana. Continuamente tomamos decisiones y dirigimos nuestra actividad a alguna tarea. A veces es una decisión entre dos tareas: 1) llamar a un cliente potencial o 2) llenar vacíos en un informe regular. A veces, es priorizar entre tres tareas: 1) ir a

un supermercado, 2) cenar con colegas o 3) visitar a un médico.

Pero otras veces, especialmente en el desarrollo de software ágil , tratamos el problema de la priorización de N tareas según los criterios de M. Este es el tipo de problema que no se resuelve fácilmente para el cerebro humano y, por lo tanto, necesita soluciones especiales.

## El problema de la priorización

La priorización de requisitos en todas las metodologías de desarrollo de software se considera una parte vital del proyecto, pero es especialmente importante en el desarrollo de software Agile. Cuando hablamos de algunas de las actividades del propietario del producto en los proyectos de Scrum como "Solicitar artículos en el Product Backlog para lograr los mejores objetivos y misiones" , "Mostrar en qué trabajará Scrum Team a continuación" y "Optimizar el valor del trabajo del equipo de desarrollo" , como se describe en la guía de Scrum, en realidad estamos hablando de la priorización de un Backlog.

Todo lo que estamos tratando de hacer es ordenar los elementos del Backlog de acuerdo con su prioridad. En esencia, estamos tratando de determinar las tareas principales del usuario y ordenarlas de esa manera, al mismo tiempo que tenemos en cuenta algunos otros criterios (organizar las historias de usuarios a partir del Backlog de Funcionalidades N según el criterio M). Por ejemplo, para priorizar las historias de usuario, podríamos utilizar cinco criterios de priorización, tales como el valor que los usuarios asignan a la visión del producto, la urgencia, las limitaciones de tiempo, la complejidad técnica y las preferencias de los interesados.

### 6.2) Modelos de Priorización

Temas: Visión de Negocio; Triage; Moscow. Cómo Negociar

prioridades. **...**

Para lograr el éxito, los proyectos deben tener una prioridad adecuada tanto para

los objetivos principales del proyecto, como para las tareas específicas que lograrán los objetivos. Así que nos ocupamos del problema de priorización en dos niveles:

- Nivel de tareas: defina qué piezas de trabajo (historias de usuario o tareas) se deben realizar, y en qué orden, durante el proceso de desarrollo del producto de software.
- Nivel del producto: determine qué características del producto podrían contribuir mejor al logro de los objetivos principales del proyecto.

Analizaremos diferentes técnicas en función del enfoque anterior.

Opciones para priorizar, teniendo en cuenta las historias de usuario

Existen distintos factores y dimensiones a considerar:

- Tiempo.
- Esfuerzo.
- Valor Funcional.
- Opinión de los stakeholders.
- Opinión del equipo de desarrollo.
- Valor de Mercado.
- Y seguramente varios más...

A partir de los mismos, han surgido distintas técnicas para priorizar los UserStories del backlog.

Las siguientes líneas describen algunas de las técnicas más comúnmente utilizadas. Para facilitar la comparación, se introduce un pequeño ejemplo de un backlog. El mismo luego será priorizado según la visión de cada técnica. El ejemplo consiste en una aplicación de comercio electrónico, con las siguientes UserStories:

US-01. Como usuario registrado quiero poner artículos en el carrito de compras para armar mi pedido.

US-02. Como dueño del comercio quiero que las personas puedan recorrer el sitio sin estar registradas para facilitar la visibilidad del mismo.

US-03. Como dueño del comercio quiero que únicamente los usuarios registrados puedan realizar compras para lograr fidelidad.

US-04. Como jefe de finanzas quiero que los pagos con tarjeta de crédito sean seguros para evitar fraudes.

US-05. Como usuario registrado quiero hacer “check-out” del carrito para finalizar mi pedido.

US-06. Como jefe de marketing quiero que sea sencillo e intuitivo agregar productos al carrito para que los usuarios les resulte atractiva la página.

US-07. Como jefe de marketing quiero que los usuarios puedan realizar búsquedas para que puedan encontrar los productos que desean.

**1. Técnica de clasificación de lista**

Esta es una de las técnicas más usadas y que no requieren ningún tipo de entrenamiento ni preparación. Forma parte de nuestra actividad cotidiana. Tomamos decisiones en base a una priorización simple. Hacemos esto antes que esto otro, por un motivo o una razón y así clasificamos gran parte de las tareas de nuestra vida.

En la gestión de Proyectos, se trata de indicar a cada US un orden de prioridad, empezando por el 1, luego el 2, 3, y continuando hasta n, que es la cantidad total de registros de US.

Las dos grandes ventajas de este tipo de técnica son:

- Solo puede haber un número uno. Evita el escollo de mucho responsables de negocio que quieren que todos los US tengan prioridad 1.
- Aporta precisión y evita la confusión. Se prioriza cada elemento en relación con el resto de elementos, lo que simplifica el proceso y lo hace más claro.

La gran desventaja es que requiere un conocimiento profundo de todas las US definidas y un esfuerzo grande por parte del equipo, para situar cada uno de los US en la posición correcta.

## 2. Técnica Business Value&StoryPoints

Esta técnica proponer priorizar las UserStories basándose en factores como el esfuerzo y la opinión del ProductOwner, del cliente y del equipo de desarrollo, o incluso combinándolas. El Business Value es un valor numérico asignada a cada UserStory (US) donde a más alto el valor, mayor valor para el cliente.

Entonces una manera de priorizar el backlog sería utilizando el Business Value de las US. Sin embargo, llevarlo acabo de manera “aislada” puede traer problemas.

Uno de ellos radica en que el cliente muchas veces se deja llevar por detalles superfluos y esto puede llevar a dejar de lado alguna funcionalidad realmente importante.

Otro problema es que no se considera el esfuerzo que lleva implementar cada US. Esto último puede resolverse con StoryPoints, que son valores numéricos que introduce el equipo de desarrollo estimando el esfuerzo que le llevará desarrollar cada US. Luego, una manera inteligente de priorizar el backlog es utilizando el cociente Business Value/StoryPoints. Es decir, lo más sencillo de implementar que devenga en mayor interés para el usuario. Para ilustrar esta técnica, suponer los siguientes valores para las US del ejemplo:

| US | Business Value | Story Points | Cociente |
|---|---|---|---|
| #01 | 8 | 1 | 8 |
| #02 | 4 | 5 | 0,8 |
| #03 | 7 | 5 | 1,4 |
| #04 | 10 | 8 | 1,25 |
| #05 | 12 | 3 | 4 |
| #06 | 8 | 5 | 1,6 |
| #07 | 6 | 8 | 0,75 |

Dados estos valores, el equipo de desarrollo dará más prioridad a las US #01 y #05, que son las referidas al manejo del carrito. Son las US que más la interesan al usuario que implican menos esfuerzo para el equipo. Luego seguirán las US #06, US#03, y US #04, para terminar con las US #07 y #02. Es importante notar que estos valores no son fijos y se pueden actualizarse al terminar cada sprint.

### 3. Técnica Urgente

Otra técnica para priorizar el backlog es utilizando una tabla de dos dimensiones, pero cambiando el concepto de StoryPoints por otro: la urgencia en tener lista una funcionalidad. Se introducen valores de 1 a 5, donde 5 implica que esa US debe estar lista ya o de lo contrario perderá sentido, mientras que un valor 1 indica que no hay apuro en tenerla lista.

También se tienen en cuenta para este valor de urgencia si la funcionalidad tiene que estar desarrollada por cuestiones contractuales, o porque de ella dependen muchas otras US, por lo debe implementarse antes. Los Business Value siguen la misma escala y se asignan valores de 1 a 5.

Obtenidas todas las asignaciones, se multiplican los valores de urgencia por los de Business Value (obteniendo números entre 1 y 25) y se ubican en una tabla como la que sigue:

| Business Value \ Urgency | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 5 | 5 | 10 | 15 | 20 | 25 |
| 4 | 4 | 8 | 12 | 16 | 20 |
| 3 | 3 | 6 | 9 | 12 | 15 |
| 2 | 2 | 4 | 6 | 8 | 10 |
| 1 | 1 | 2 | 3 | 4 | 5 |

Dada esta tabla, la priorización es evidente: primeros las US en el sector rojo, luego naranja, amarillo y finalmente las verde.

Retomando el ejemplo del comercio electrónico, se pueden asumir los siguientes valores:

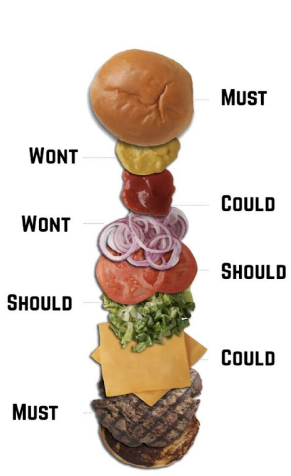
US Business Value Urgencia Sector

| US | Business Value | Urgencia | Sector |
|---|---|---|---|
| 1 | 5 | 4 | Rojo |
| 2 | 3 | 4 | Amarillo |
| 3 | 3 | 2 | Verde |
| 4 | 4 | 3 | Amarillo |
| 5 | 4 | 4 | Naranja |
| 6 | 5 | 4 | Naranja |
| 7 | 3 | 2 | Verde |

Utilizando esta técnica, el equipo de desarrollo comenzará con la US #01, y luego atacará las US #05 y #06. Finalmente, las US #02 y #04, para terminar con las US #03 y #07. Comparado con la técnica anterior, la US #06 ha adquirido mayor prioridad, ya que con esta visión que el carrito sea sencillo de utilizar ha sido catalogado como más relevante.

Razones similares pueden utilizarse para justificar la mayor prioridad de la US #02 en esta ocasión.

La posibilidad de navegar el sitio para usuarios invitados ha sido clasificada con un alto valor de urgencia, por lo que ha subido su prioridad final.



A diferencia de las técnicas numéricas que en ocasiones no resultan efectivas, se ha desarrollado esta técnica que plantea una categorización de las US en función a palabras que tengan un significado concreto, para ello esta técnica propone utilizar las siguientes etiquetas:

M: Esta funcionalidad debe estar (MUST).

S: Esta funcionalidad debería estar (SHOULD).

C: Esta funcionalidad podría estar (COULD).

W: Esta funcionalidad no estará ahora, quizás en un futuro (WON'T).

Luego, las US se llevan a cabo siguiendo esta clasificación: primero las US etiquetadas con M, luego con S, en tercer término aquellas con C y finalmente aquellas con W.

Esta manera de categorizar las US en base a palabras, permite una discusión posterior en función al significado de las mismas como veremos a continuación.

Los requisitos de "MUST" no son negociables si no se entregan, en este caso, el proyecto es un fracaso, por lo tanto, es de interés para todos acordar qué se puede entregar y será útil.

Es bueno tener características que están clasificadas en las otras categorías de "SHOULD" y "COULD".

Los requisitos "MUST" deben formar un conjunto coherente. No pueden ser simplemente "seleccionados" de todos los demás. Si lo son, lo que sucede es que, de forma predeterminada, todos los demás requisitos se convierten automáticamente en "MUST" y se desperdicia todo el ejercicio.

Los requisitos marcados como "WON'T" son potencialmente tan importantes como la categoría "MUST". No es inmediatamente obvio por qué esto es así, pero es una de las características que hacen de MoSCoW una técnica tan poderosa. La clasificación de algo como "WON'T" reconoce que es importante, pero se puede dejar para una versión futura. De hecho, se puede pasar una gran cantidad

de tiempo tratando de producir una buena lista de "WON'T". Esto tiene tres efectos importantes:

- Los usuarios no tienen que luchar para obtener algo en una lista de requisitos.
- Al pensar en lo que se requerirá más adelante, afecta a lo que se pide ahora.
- Los diseñadores que ven la tendencia futura pueden producir soluciones que puedan adaptarse a estos requisitos en una versión futura.

**5. Basados en Riesgos**

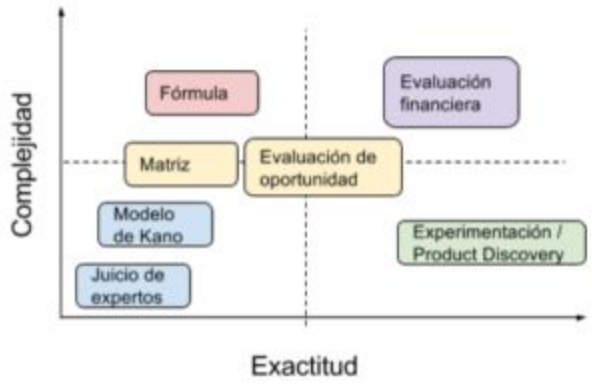
Esta técnica propone simplemente hacer primero las US que involucren mayores riesgos. Lo cual implica realizar un análisis de riesgo completo antes de priorizar el backlog.

Para realizar el análisis de riesgos, ya hay mucha documentación escrita al respecto y por ello no vamos a entrar en este artículo en su análisis.

Las opciones para priorizar, teniendo en cuenta el negocio

Si tenemos en cuenta, principalmente, el Valor del Negocio nos encontramos con distintas técnicas a las ya vistas, en la que priman principalmente el resultado del producto y su impacto en el negocio.

A modo introductorio, podemos observar las distintas técnicas en la siguiente matriz, en función de la complejidad y la exactitud en los resultados:



## 6. Juicio de Expertos

Complejidad: baja

Exactitud: baja

El juicio de expertos es la opción más simple y menos exacta: básicamente se prioriza según la opinión del propio product manager, o de algún stakeholder con conocimiento de la industria (o incluso puede ser en base a juicio de clientes).

Esto se hace (y funciona relativamente bien) en proyectos pequeños con un “dueño de negocio” que tiene mucho conocimiento del problema y la solución que quiere; en estos casos, no vale la pena hacer ejercicios de priorización (considerando incluso que los vetaría si no estuviesen de acuerdo a sus decisiones).

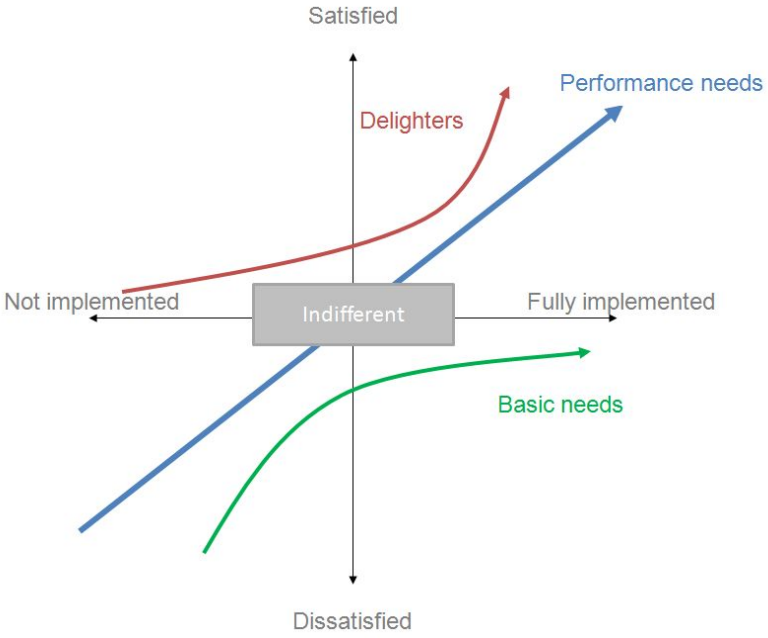
Se deben usar con cuidado, ya que aporta una visión muy sesgada de la visión del negocio.

## 7. El modelo de KANO

Complejidad: medio baja

Exactitud: medio baja

El modelo de Kano propone dividir las funcionalidades que quiero implementar en 3 categorías:




- Necesidades básicas: si no están generan gran insatisfacción, pero si están no generan una satisfacción mayor (por ejemplo, la capacidad de escribir en un editor de texto).
- Necesidades de performance: estos son los que generan mayor satisfacción cuando están y mayor insatisfacción cuando no. No son críticos para el uso del producto, pero serán un factor de decisión contra competidores (por ejemplo, la capacidad de agregar imágenes en un editor de texto).
- Deslumbrantes: son aquellas características no esperadas por el usuario, que generan gran satisfacción si están, pero no generan insatisfacción si no se encuentran (por ejemplo, la capacidad de dictar oralmente a un procesador de texto).

Uno de los principales problemas es que si bien categorizamos los resultados, no nos provee ninguna facilidad para priorizar entre 2 features del mismo grupo. Buen modelo para compararse contra competidores y entender qué y cómo valor el usuario tu producto. Es limitado como técnica de priorización.

—

7. Bibliograka

**1.- Scrum para Dummies Mark C. LaytonWiley, 2015**

**2.- Proyectos agiles con Scrum , Martin Alaimo**