Introducción

En esta unidad nos proponemos explicarte las metodologías ágiles de gestión y desarrollo de software. Te contaremos las diferencias que existen entre los modelos tradicionales y los ágiles. También veremos las historias de usuario junto con los roles que intervienen en el modelo ágil.
Finalmente te presentaremos el modelo ágil más utilizado en estos momentos, el Scrum, en donde veremos sus características, roles y componentes que lo integran.

En la Unidad 1 vimos cómo es el proceso de desarrollo de software y te presentamos los principales modelos para gestionar un proyecto. Todos esos modelos se corresponden con los denominados modelos tradicionales, ya que fueron desarrollados al inicio de la Ingeniería de software.
Ahora bien, la experiencia de estas últimas décadas en el desarrollo de software y los cambios en los ritmos de producción han traído como consecuencia que esos modelos muchas veces no se adaptan adecuadamente al ritmo de estos tiempos. Es por eso que ha surgido un nuevo modelo, que lo podemos denominar genéricamente como **Metodología Ágil,** que te presentaremos en esta unidad.

Gestión de Proyectos de Desarrollo Tradicional vs. Ágil

Habrás notado a lo largo de la experiencia en este curso, y también por lo que te habrán contado profesores y profesionales de la ingeniería de software que es muy difícil especificar los requisitos en una única y primera etapa.

Por la complejidad de muchas de las reglas de negocio que automatizamos cuando construimos software, es muy difícil saber qué software se quiere hasta que se trabaja en su implementación y se ven las primeras versiones o prototipos. También, te pasará que muchas veces ni siquiera el propio usuario estará seguro de los requisitos y te irá pidiendo modificaciones a medida que vaya evaluando el software. Resulta también muy difícil documentar de una única vez, antes de la codificación, un diseño que especifique de manera realista y sin variación, todas las cuestiones a implementar en la programación.

Las metodologías tradicionales se basaron en los procesos industriales y de construcción de elementos físicos. De esta manera, las ingenierías clásicas o la arquitectura necesitan seguir este tipo de ciclos de vida en cascada porque precisan mucho de un diseño previo a la construcción, exhaustivo e inamovible como puede ser los planos del arquitecto antes de empezar el edificio.

Como seguramente podrás deducir, una vez realizados los cimientos de un edificio será muy difícil que se vuelva a rediseñar el plano y se cambie lo ya construido.

Con el software no pasa lo mismo ya que, aunque se pretenda emular ese modo de fabricación, el software es algo que tendrá cambios y, por eso, deberás tener muy en claro que es casi imposible cerrar un diseño en una primera etapa para pasarlo a

programación sin tener que modificarlo posteriormente.
En nuestro caso, que desarrollamos software, por su naturaleza, es más fácil de modificar. Cambiar líneas de código tiene menos impacto que cambiar las columnas de un edificio ya construido. De ahí que estas nuevas metodologías ágiles se basan en nuevas propuestas que recomiendan construir rápido una versión software y modificarla evolutivamente.

Diferenciar el cómo se construye software del cómo se construyen los productos físicos es uno de los fundamentos de las metodologías ágiles

Ciclo de vida Ágil

Como seguramente te habrás imaginado, esta nueva metodología también tiene un ciclo de vida. Podríamos decir que sería un ciclo de vida iterativo e incremental, con iteraciones cortas (semanas) y en cada iteración no es necesario que haya fases lineales tipo cascada.

De esta manera, un **proyecto ágil** se podría definir como una manera de enfocar el desarrollo de software mediante un ciclo iterativo e incremental, con equipos que trabajan de manera altamente colaborativa y autoorganizados.
Así,

Un proyecto ágil es un desarrollo iterativo realizado por equipos colaborativos y autoorganizados.

A diferencia de ciclos de vida iterativos e incrementales más “relajados”, en un proyecto ágil cada iteración no es un “mini cascada”.

Veamos porqué...
Esto es así porque el objetivo de **acortar al máximo las iteraciones** (normalmente entre 1 y 4 semanas) lo hace casi imposible. Cuanto menor es el tiempo de iteración más se solapan las tareas. Esto sucede hasta tal punto que muchas veces te podrá pasar que te encuentres en un proyecto donde, se esté diseñando, programando y probando de manera no secuencial, es decir solapada, durante una misma iteración.

Esa característica de simultaneidad de etapas implicará una máxima colaboración e interacción de los miembros del equipo. Los equipos tendrán que ser multidisciplinares, es decir, que no hay roles que sólo diseñen o programen, porque todos pueden diseñar y programar. También significa autoorganización, es decir, que en la mayoría de los proyectos ágiles no hay, por ejemplo, un único jefe de proyecto responsable de asignar tareas.

(hacé clic para acceder a la respuesta)

A modo de resumen, te podemos decir que un **proyecto ágil** lleva la iteración al extremo mediante estos **dos pasos:**

Ciclo de vida Ágil

- **Dividir las tareas del proyecto en incrementos de corta duración** (según la metodología ágil, típicamente entre 1 y 4 semanas).

- **Desarrollar en cada iteración un prototipo operativo.** Al final de cada incremento se obtiene un producto entregable que es revisado junto con el cliente, posibilitando la aparición de nuevos requisitos o la perfección de los existentes, reduciendo riesgos globales y permitiendo la adaptación rápida a los cambios.

El Manifiesto Ágil

El 12 de febrero de 2001, un grupo de 17 destacados y conocidos profesionales de la ingeniería del software escribían en Utah el **Manifiesto Ágil**.

Entre ellos estaban *Ken Schwaber* y *Jeff Sutherland*, los creadores de algunas de las metodologías ágiles más conocidas en la actualidad, entre ellas **Scrum,** metodología que veremos en las próximas secciones. Su objetivo fue establecer los valores y principios que permitirían a los equipos desarrollar software rápidamente y respondiendo a los cambios que pudieran surgir a lo largo del proyecto.

Como te contamos al inicio de esta Unidad, se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

De esta forma se establecieron **cuatro valores ágiles**, que a continuación te detallamos....

Los Valores Ágiles

Así, los cuatro valores ágiles son:

Individuos e interacciones

Valorar a los individuos y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. Se tendrán en cuenta las buenas prácticas de desarrollo y gestión de los participantes del proyecto (siempre dentro del marco de la metodología elegida). Esto facilita el trabajo en equipo y disminuyen los impedimentos para que realicen su trabajo. Asimismo, compromete al equipo de desarrollo y a los individuos que lo componen.

Software funcionando

Desarrollar software que funciona más que conseguir una documentación exhaustiva. No es necesario producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Los documentos deben ser cortos y centrados en lo fundamental. La variación de la cantidad y tipo de documentación puede ser amplia

Los Valores Ágiles

dependiendo del tipo de cliente o del proyecto. El hecho de decir que la documentación es el código fuente y seguir esa idea sin flexibilidad puede originar un caos. El problema no es la documentación sino su utilidad.

Colaboración con el cliente

La colaboración con el cliente más que la negociación de un contrato. Es necesaria una interacción constante entre el cliente y el equipo de desarrollo. De esta colaboración depende el éxito del proyecto. Este es uno de los puntos más complicados de llevar a cabo, debido a que muchas veces el cliente no está disponible. En ese caso desde dentro de la empresa existirá una persona que represente al cliente, haciendo de interlocutor y participando en las reuniones del equipo.

Respuesta ante el cambio

Responder a los cambios más que seguir estrictamente un plan. Pasamos de la anticipación y la planificación estricta sin poder volver hacia atrás a la adaptación. La flexibilidad no es total, pero existen muchos puntos (todos ellos controlados) donde se pueden adaptar las actividades.

Los Principios Ágiles

De los **cuatro valores ágiles** que te presentamos previamente, surgen los **doce principios del manifiesto ágil**. Estos principios son características que diferencian un proceso ágil de uno tradicional.

Los principios definidos en el manifiesto ágil te los comentamos a continuación:

- Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
- Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
- Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
- Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
- 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.
- El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
- El software funcionando es la medida principal de progreso.
- Los procesos ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
- La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
- La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

Los Principios Ágiles

- Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.
- A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para, a continuación, ajustar y perfeccionar su comportamiento en consecuencia.

Product owner

Ahora te presentamos un nuevo rol que surge en la metodología ágil, el **Product Owner.** Y para eso te contamos que el "product owner" (o propietario del producto) es aquella persona con una visión muy clara del producto que se quiere desarrollar, que es capaz de transmitir esa visión al equipo de desarrollo y, además, está altamente disponible para transmitirla.

La figura del **product owner** es clave en la planificación y seguimiento de un proyecto ágil. Es una figura que cuando no realiza correctamente su función el proyecto tiene un serio riesgo, y problema, llegando incluso a dejar de ser ágil, o incluso dejando de ser proyecto.

Es importante explicarte que este rol puede ser ocupado por una persona interna o externa a la organización. Entre sus principales responsabilidades te podemos mencionar:

- Ser el representante de todas las personas interesadas en el proyecto, ya sea internas o externas.
- Definir los objetivos del producto o proyecto.
- Colaborar con el equipo para planificar, revisar y dar detalle a los objetivos de cada iteración.

Historias de usuario

Ahora te vamos a presentar un componente que se utiliza en las metodologías ágiles, las **Historias de usuario.**

Para eso, te contamos que la descripción de las necesidades se realiza a partir de las historias de usuario (user story) que son, principalmente, lo que el cliente o el usuario quiere que se implemente; es decir, son una descripción breve, de una funcionalidad de software tal y como la percibe el usuario.

El concepto de historia de usuario tiene sus raíces en la metodología "**eXtreme Programming**" o programación extrema. Esta metodología fue creada por Kent Beck y descrita por primera vez en 1999 en su libro "eXtreme Programming Explained". No obstante, las historias de usuario se adaptan de manera apropiada a la mayoría de las metodologías ágiles teniendo, por ejemplo, un papel muy importante en la metodología que veremos en la siguiente sección, denominada Scrum.

Es en este momento donde tiene una función preponderante el **Product owner,** ya que en la mayoría de las ocasiones, el negocio o los usuarios, van proporcionando una cantidad de ideas a implementar, que se van convirtiendo en historias de usuario.

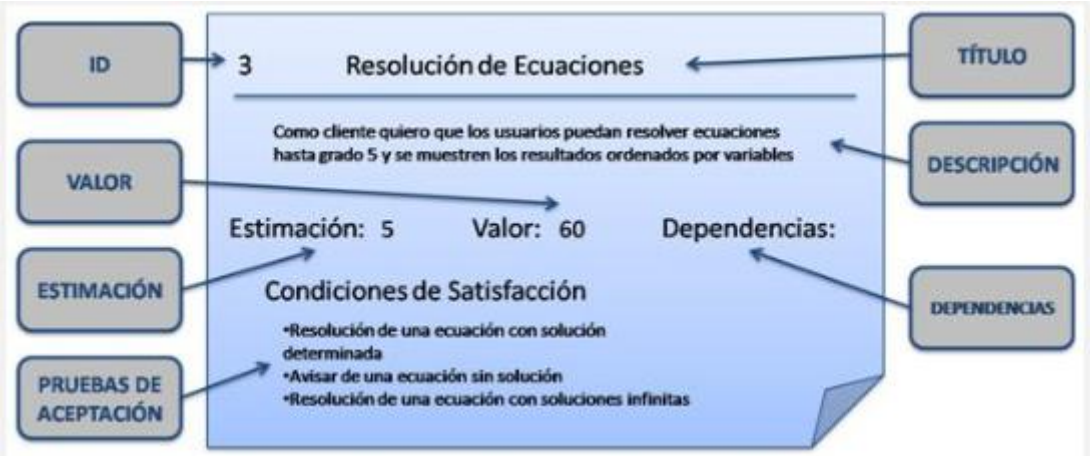
Historias de usuario

La función del product owner es vital, debe ser quien decida las historias de usuario que entrarán en el product backlog (las historias que van a desarrollarse) y cuáles no; además debe fijar la prioridad de las historias del product backlog.

Información de una historia de usuario

Ahora, te contamos aquellos campos que consideramos necesarios para describir de manera adecuada una historia de usuario. Esta es una lista orientativa ya que dependiendo del proyecto, podrás incluir cualquier otro campo que proporcione información útil,

Estos campos los podés observar en la siguiente figura:



De esta manera, una historia de usuario está compuesta por los siguientes elementos:

Hacé clic para conocer los elementos....

- **ID**: identificador de la historia de usuario.
- **Título**: Leyenda descriptiva de la historia de usuario.
- **Descripción**: descripción sintetizada de la historia de usuario. Si bien el estilo puede ser libre, la historia de usuario debe responder a tres preguntas: ¿Quién se beneficia? ¿Qué se quiere? y ¿Cuál es el beneficio? Como ayuda te podemos recomendar que sigas el patrón: Como [rol del usuario], quiero [objetivo], para poder [beneficio]. Con este patrón se garantiza que la funcionalidad se captura a un alto nivel y que se está describiendo de una manera no demasiado extensa.
- **Estimación**: estimación del tiempo de implementación de la historia de usuario en unidades de desarrollo, conocidas como puntos de historia (estas unidades representarán el tiempo teórico de desarrollo/persona que se estipule al comienzo del proyecto).
- **Valor**: valor (normalmente numérico) que aporta la historia de usuario al cliente o usuario. El objetivo del equipo es maximizar el valor y la satisfacción percibida por el cliente o usuario en cada iteración. Este campo determinará junto con el tiempo, el orden con el que las historias de usuario van a ser implementadas.

Información de una historia de usuario

- **Dependencias**: una historia de usuario no debería ser dependiente de otra historia, pero en ocasiones es necesario mantener la relación. En este campo se indicarían los identificadores de otras historias de las que depende.
- **Pruebas de aceptación**: pruebas consensuadas entre el cliente o usuario y el equipo, que deberán ser superadas para dar como finalizada la implementación de la historia de usuario. Este campo también se suele denominar “criterios o condiciones de aceptación".

Es importante que tengas en cuenta que si bien las historias de usuario son lo suficientemente flexibles como para describir la funcionalidad de la mayoría de los sistemas, no son apropiadas para todo.
Si por cualquier razón, necesitás expresar alguna necesidad de una manera diferente a una historia de usuario, podrás utilizar las interfaces o pantallas de usuario.
Del mismo modo puede ocurrir con documentos de especificaciones de seguridad, normativas, etc.

No obstante, lo que sí es importante con esta documentación adicional es mantener la trazabilidad con las historias de usuario. Por ejemplo, a través de hojas de cálculo donde se lleve el control de a qué historia pertenece cada documento adicional, o especificando el identificador de la historia en algún apartado del documento, etc.

Malas interpretaciones sobre el concepto de historia de usuario

Seguramente estarás comparando las historias de usuarios con la documentación de requerimientos funcionales de la etapa de Análisis de los modelos tradicionales y estarás pensando que son lo mismo.
Estos conceptos no siempre quedan bien delimitados y nos puede llevar a la confusión, es por eso que ahora te respondemos algunas preguntas que seguramente te estarás haciendo.

Las historias de usuario ¿equivalen a.....

...requisitos funcionales?

Hacé clic en el botón para ver la respuesta.

En general, se asocia el concepto de historia de usuario con el de la especificación de un requisito funcional. De hecho, muchas veces se habla de que a la hora de especificar una necesidad del cliente o del usuario, las metodologías ágiles usan la historia de usuario y las tradicionales, el requisito funcional. Sin embargo, detrás del concepto de historia de usuario hay muchos aspectos que lo diferencian de lo que es una especificación de un requisito, diferencias que muchas veces son poco conocidas y que llevan a muchos equipos a dudas y confusiones.

Una **historia de usuario** describe funcionalidad que será útil para el usuario o cliente de un software. Y aunque normalmente las historias de usuario asociadas a las metodologías ágiles, suelen escribirse en pósit o tarjetas, son mucho más que eso.
Como te hemos comentado anteriormente, una historia no es sólo una descripción de

una funcionalidad, sino también es de vital importancia para comunicar al usuario el progreso del desarrollo.

Las historias de usuario, frente a mostrar el **"cómo"**, sólo dicen el **"qué".** Es decir, muestran funcionalidad que será desarrollada, pero no cómo se desarrollará. De ahí que aspectos como que "el software se escribirá en Java" no deben estar contenidos en una historia de usuario.

De esta manera, no se deben equiparar las historias de usuario con las especificaciones de requisitos ya que por definición, las historias de usuario no deben tener el nivel de detalle que suele tener la especificación de un requisito.

Una **historia de usuario** debería ser **pequeña, memorizable,** y que pudiera ser desarrollada por un par de programadores en una semana. Debido a su brevedad, es imposible que una historia de usuario contenga toda la información necesaria para desarrollarla, en tan reducido espacio no se pueden describir aspectos del diseño, de las pruebas, normativas, convenciones de codificación a seguir, etc.

Para resolver el anterior problema hay que entender que el objetivo de las historias de usuario es, entre otros, lograr la interacción entre el equipo y el cliente o el usuario por encima de documentar, por lo que tanto no se deben sobrecargar de información.

...casos de uso?

Hacé clic en el botón para ver la respuesta.

Otro concepto que suele crear confusión sobre las historias de usuario son los casos de uso. Aunque hay quien ha logrado incluir casos de uso en su proceso ágil, no quiere decir que las historias de usuario sean equivalentes a los casos de uso.
Básicamente, si pensamos que una historia de usuario es el **"qué"** quiere el usuario, el caso de uso es un **"cómo"** lo quiere.

Generalmente, cuando un proyecto comienza a seguir una metodología ágil, se deberían olvidar completamente los casos de uso y el equipo debería centrarse en la realización de historias de usuario. Sin embargo, esto puede producir los siguientes problemas:

- **Las historias de usuario no proporcionan a los diseñadores un contexto desde el que trabajar.** Pueden no tener claro cuál es el objetivo en cada momento. ¿Cuándo le surgiría al cliente o usuario esta necesidad?
- **Las historias de usuario no proporcionan al equipo de trabajo ningún sentido de completitud.** Se puede dar el caso que la cantidad de historias de usuario no deje de aumentar, lo que puede provocar desmotivación en el equipo. Realmente, ¿qué tan grande es el proyecto?
- **Las historias de usuario no son un buen mecanismo para evaluar la dificultad del trabajo** que está aún por llegar.

Por lo tanto, si en un proyecto ocurre alguno de estos problemas se puede barajar la

Malas interpretaciones sobre el concepto de historia de usuario

posibilidad de complementar las necesidades descritas en las historias de usuario con casos de uso, donde quede reflejado el comportamiento necesario para cumplir dichas necesidades.
En el caso de que se usen las historias de usuario y los casos de uso de manera complementaria, una historia de usuario suele dar lugar a la especificación de varios casos de uso.

Creando buenas historias de usuario

Para asegurar la calidad de una historia de usuario, en el año 2003 se definió un **método llamado INVEST.**

El método se usa para comprobar la calidad de una historia de usuario revisando que cumpla las características que te describimos a continuación:

Independient (independiente)

Es importante que cada historia de usuario pueda ser planificada e implementada en cualquier orden. Para ello debería ser totalmente independiente (lo cual facilita el trabajo posterior del equipo). Las dependencias entre historias de usuario pueden reducirse combinándolas en una o dividiéndolas de manera diferente.

Negotiable (negociable)

Una historia de usuario es una descripción corta de una necesidad que no incluye detalles. Las historias deben ser negociables ya que sus detalles serán acordados por el cliente/usuario y el equipo durante la fase de “conversación”. Por tanto, se debe evitar una historia de usuario con demasiados detalles porque limitaría la conversación acerca de la misma.

Valuable (valiosa)

Una historia de usuario tiene que ser valiosa para el cliente o el usuario. Una manera de hacer una historia valiosa para el cliente o el usuario es que la escriba él mismo.

Estimable (estimable)

Una buena historia de usuario debe ser estimada con la precisión suficiente para ayudar al cliente o usuario a priorizar y planificar su implementación. La estimación generalmente será realizada por el equipo de trabajo y está directamente relacionada con el tamaño de la historia de usuario (una historia de usuario de gran tamaño es más difícil de estimar) y con el conocimiento del equipo de la necesidad expresada (en el caso de falta de conocimiento, serán necesarias más fases de conversación acerca de la misma).

Small (pequeña)

Creando buenas historias de usuario

Las historias de usuario deberían englobar como mucho unas pocas semanas/persona de trabajo. Incluso hay equipos que las restringen a días/persona. Una descripción corta ayuda a disminuir el tamaño de una historia de usuario, facilitando su estimación.

Testable (comprobable)

La historia de usuario debería ser capaz de ser probada (fase “confirmación” de la historia de usuario). Si el cliente o usuario no sabe cómo probar la historia de usuario significa que no es del todo clara o que no es valiosa. Si el equipo no puede probar una historia de usuario nunca sabrá si la ha terminado o no.

Asignando valor a las historias de usuario

Hagamos un repaso, y para eso te recordamos que el Product backlog (también llamado pila de producto) es una lista de tareas que el equipo elabora en la reunión de planificación de la iteración (Sprint planning) como plan para completar los objetivos y requisitos seleccionados para la iteración.

Los ítems del **Product backlog,** deben estar priorizados, es decir, deben tener asignados un valor. Dicho valor es asignado por el Product Owner, y pondera básicamente las siguientes variables:

- Beneficios de implementar una funcionalidad.
- Pérdida o costo que cause posponer la implementación de una funcionalidad.
- Riesgos de implementarla.
- Coherencia con los intereses del negocio.
- Valor diferencial con respecto a productos de la competencia.

Uno de los aspectos más importantes aquí es que la definición de **"valor"** para cada cliente puede ser distinta. Por lo tanto te recomendamos que incluyas algún tipo de escala cualitativa.

Una manera rápida de empezar a asignar valor a las historias es dividirlas en 3 grupos, según sean imperativas, importantes o prescindibles. Dentro de cada grupo te resultará más fácil realizar una ordenación relativa por valor numérico y después asignarlo. Todo ello servirá para que en cada iteración se entregue el producto al cliente maximizando su valor.

Existen otro tipo de ponderaciones, por ejemplo la **técnica MoSCoW.** Esta técnica fue definida en el año 2004. Su fin es obtener el entendimiento común entre cliente y el equipo del proyecto sobre la importancia de cada requisito o historia de usuario. La clasificación es la siguiente:

M - MUST -> Se debe tener la funcionalidad. En caso de que no exista la solución a construir fallará**.**

Asignando valor a las historias de usuario

S - SHOULD -> Se debería tener la funcionalidad. La funcionalidad es importante pero la solución no fallará si no existe.

C - COULD -> Sería conveniente tener esta funcionalidad. Es en realidad un deseo.

W - WON'T -> No está en los planes tener esta funcionalidad en este momento. Posteriormente puede pasar a alguno de los estados anteriores.

A continuación, te presentamos un ejemplo de cómo podés organizar tu lista de historias de usuarios del product backlog:

Requisito	Tarea	Quien	Estado (No iniciada / en progreso / completada)	Dia:	1	2	3	4	5	6	7	8	9	10
				Horas pendientes	1120	1088	1076	1048	1040	1032	1020	1008	992	972
Requisito A	Tarea 1	Joao	Completada		16	8								
Requisito A	Tarea 4	Laura	Completada		4									
Requisito A	Tarea 5	Laura	Completada		4									
Requisito A	Tarea 3	Gabri	Completada		8									
Requisito A	Tarea 2	Laura	Completada		16	8	4							
Requisito A	Tarea 6	Gabri	Completada		8	8	8							
Requisito A	Tarea 7	Joao	Completada		16	16	16	8						
Requisito A	Tarea 8	Laura	Completada		8	8	8							
Requisito A	Tarea 9	Laura	Completada		8	8	8	8	8					
Requisito A	Tarea 10	Laura	Completada		8	8	8	8	8	8	4			
Requisito A	Tarea 11	Joao	Completada		16	16	16	16	16	16	8			
Requisito B	Tarea 12	Gabri	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 13	Laura	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 14	Joao	En progreso		8	8	8	8	8	8	8	8	8	4
Requisito B	Tarea 15	Gabri	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito B	Tarea 16	Laura	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 17	Joao	No iniciada		4	4	4	4	4	4	4	4	4	4
Requisito C	Tarea 18	Gabri	No iniciada		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 19	Laura	No iniciada		16	16	16	16	16	16	16	16	16	16
Requisito C	Tarea 20	Joao	No iniciada		8	8	8	8	8	8	8	8	8	8

Scrum

Como te comentamos previamente, **Scrum es una metodología ágil** que proporciona un marco para la gestión de proyectos. Podríamos decir que hoy en día es la metodología ágil más popular, y, de hecho, se ha utilizado para desarrollar software desde principios de la década de los 90.

El conjunto de buenas prácticas de Scrum se aplica esencialmente a la gestión de proyectos. Y para eso esta metodología se basa en tres pilares fundamentales, que te pasamos a comentar:

Transparencia

Todos los aspectos del proceso que afectan al resultado son visibles para todos aquellos que administran dicho resultado. Por ejemplo, se utilizan pizarras y otros mecanismos o técnicas colaborativas para mejorar la comunicación.

Inspección

Se debe controlar los diversos aspectos del proceso con la frecuencia suficiente para que se le puedan detectar variaciones o desvíos.

Revisión

El producto debe estar dentro de los límites aceptables. En caso de desviación se procederá a una adaptación del proceso y el material procesado.

El equipo en Scrum

Uno de los aspectos más importantes en cualquier proyecto, y por lo tanto también en los proyectos ágiles, es el establecimiento del equipo. Los roles y responsabilidades deben ser claros y conocidos por todos los integrantes del mismo.

Cada **equipo Scrum tiene tres roles**:

Scrum Master

Es el responsable de asegurar que el equipo Scrum siga las prácticas de Scrum. Sus principales funciones son:

- Ayudar a que el equipo Scrum y la organización adopten Scrum.
- Liderar el equipo Scrum, buscando la mejora en la productividad y calidad de los entregables.
- Ayudar a la autogestión del equipo.
- Gestionar e intentar resolver los impedimentos con los que el equipo se encuentra para cumplir con las tareas del proyecto.

Propietario del Producto (Product Owner)

Este rol te lo presentamos en la sección 4. Es la persona responsable de gestionar las necesidades que serán satisfechas por el proyecto y asegurar el valor del trabajo que el equipo lleva a cabo. Su aportación al equipo se basa en:

- Recolectar las necesidades o historias de usuario.
- Gestionar y ordenar las necesidades representadas por las historias de usuario, las que te describimos en la sección 5.
- Aceptar el software al finalizar cada iteración.
- Maximizar el retorno de inversión del proyecto.

Equipo de desarrollo

El equipo está formado por los desarrolladores, que convertirán las necesidades del Product Owner en un conjunto de nuevas funcionalidades, modificaciones o

incrementos del software final. El equipo de desarrollo tiene características especiales, te contamos algunas de las más importantes:

o **Auto-gestionado:** el mismo equipo supervisa su trabajo. En Scrum se potenciarán las reuniones del equipo, aumentando la comunicación. No existe el rol clásico de jefe de proyecto, ya que, como te comentamos antes, el Scrum Master tiene otras responsabilidades.

o **Multifuncional:** no existen compartimientos estancos o especialistas, cada integrante del equipo puede encargarse de tareas de programación, pruebas, despliegue, etc. Asimismo las personas pueden tener capacidades diferentes o conocimientos más profundos en diferentes áreas. Lo importante es que cualquier integrante del equipo sea capaz de realizar cualquier función.

o **No distribuidos:** es conveniente que el equipo se encuentre en el mismo lugar físico. Esto facilita la comunicación y la autogestión que nace del mismo equipo. Sin embargo, hoy en día con las facilidades en las comunicaciones, se pueden realizar proyectos Scrum con equipos distribuidos gracias a herramientas de trabajo colaborativo.

o **Tamaño óptimo:** un equipo de desarrollo Scrum (sin tener en cuenta al Product Owner y al Scrum Master) estaría compuesto por al menos tres personas. Con menos de tres personas la interacción decae y con ella la productividad del equipo. Como límite superior, con más de nueve personas la interacción hace que la autogestión sea muy difícil de alcanzar.

En la siguiente imagen te mostramos los roles que intervienen en el Equipo Scrum:

El Product Backlog

Ahora te proponemos volver a repasar lo que se conoce como la pila de producto o **Product Backlog,** ahora aplicado a Scrum. En esta metodología es uno de los elementos fundamentales.

El **Product Backlog** consiste en un listado de historias del usuario que se incorporarán al software a partir de incrementos sucesivos. Es decir, sería similar a un listado de requisitos de usuario y representa lo que el cliente espera. Una de las principales diferencias respecto de un proceso tradicional es la evolución continua del Product Backlog, buscando aumentar el valor del producto desde el punto de vista del negocio.

A partir del Product Backlog se logra tener una única visión durante todo el proyecto. Y, por lo tanto, los fallos en el Product Backlog repercutirán profundamente en el proyecto.

Seguramente también recordarás que este listado estará ordenado. Aunque no existe un criterio preestablecido en Scrum para ordenar las historias de usuario, el más aceptado es partir del valor que aporta al negocio la implementación de la historia de usuario. El responsable de ordenar el Product Backlog es el **Product Owner,** aunque también puede ser ayudado o recibir asesoramiento de otros roles como, por ejemplo, el Scrum Master y el equipo de desarrollo.

Un Product Backlog cuenta esencialmente, con cuatro cualidades: debe estar detallado de manera adecuada, estimado, emergente y priorizado. Ahora te contamos con mayor detalle cada una de ellas:

Detallado adecuadamente

El grado de detalle dependerá de la prioridad. Las historias de usuario que tengan una mayor prioridad se describen con más detalle. De esta manera las siguientes funcionalidades a ser implementadas se encuentran descritas correctamente y son viables. Como consecuencia de esto, las necesidades se descubren, se descomponen, y perfeccionan a lo largo de todo el proyecto.

Estimado

Las estimaciones a menudo se expresan en días ideales o en términos abstractos. Saber el tamaño de los elementos del Product Backlog te ayudará a darle prioridad y a planificar los siguientes pasos.

Emergente

Las necesidades se van desarrollando y sus contenidos cambian con frecuencia. Los nuevos elementos se descubren y se agregan a la lista teniendo en cuenta los comentarios de los clientes y usuarios. Así mismo, otros elementos podrán ser modificados o eliminados.

Priorizado

El Product Backlog

Los elementos del Product Backlog se priorizan. Los elementos más importantes y de mayor prioridad aparecen en la parte superior de la lista. Puede no ser necesario priorizar todos los elementos en un primer momento, sin embargo sí es conveniente identificar los 15-20 elementos más prioritarios.

El Sprint

Como te hemos comentado anteriormente, una de las bases de los proyectos ágiles es el desarrollo mediante las iteraciones incrementales.

En **Scrum a cada iteración se le denomina Sprint.**

Scrum recomienda iteraciones cortas, por lo que cada Sprint durará entre 1 y 4 semanas. Y como resultado se creará un producto potencialmente entregable, un prototipo operativo. Las características que van a implementarse en el Sprint provienen del Product Backlog.

El equipo de desarrollo selecciona las historias de usuario que se van a desarrollar en el Sprint conformando la pila de Sprint (Sprint Backlog). La definición de cómo descomponer, analizar o desarrollar este Sprint Backlog queda a criterio del equipo de desarrollo.

Algo importante que tenés que saber es que, aunque todos los Sprints dan como resultado un incremento del software, no todos implican un paso a producción. Es responsabilidad del Product Owner y los clientes decidir el momento en el que los incrementos son puestos en producción. Una posibilidad para realizar esa puesta en producción es con los denominados "Sprints de Release". Estos Sprints contendrán, en general, tareas solamente relacionadas con el despliegue, instalación y puesta en producción del sistema. Es decir, que no existen tareas donde se agreguen nueva funcionalidad.

En **Scrum el Sprint Backlog** indica solamente lo que el equipo realizará durante la iteración. El **Product Backlog,** por el contrario, es una lista de características que el Product Owner quiere que se realicen en futuros Sprints.

El **Product Owner** puede visualizar, pero no puede modificar el Sprint Backlog. En cambio, puede modificar el ProductBacklog cuantas veces quiera con la única restricción de que los cambios tendrán efecto una vez finalizado el Sprint.

Para mejorar la gestión de las historias de usuario y las tareas de cada Sprint usualmente se utilizan pizarras u otros mecanismos que brinden información inmediata al equipo.

Reuniones

Las **reuniones** son un pilar importante dentro de Scrum. Se realizan a lo largo de todo el Sprint como muestra la siguiente figura. Estas reuniones están representadas en los cuadros con color gris. Se definen diversos tipos de reuniones:

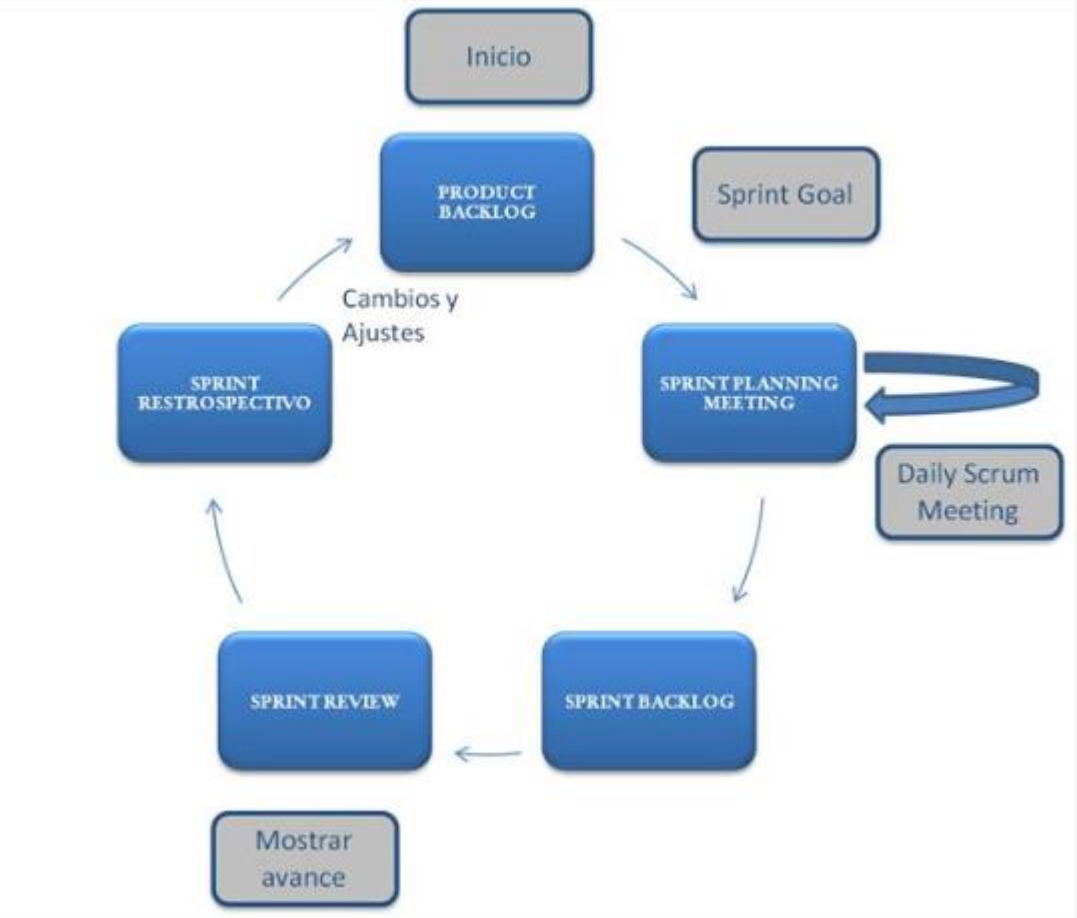
Reuniones

Reunión de planificación del Sprint

Se lleva a cabo al principio de cada Sprint, definiendo en ella qué se va a realizar en ese Sprint. Esta reunión da lugar al Sprint Backlog. En esta reunión participan todos los roles. El Product Owner presenta el conjunto de historias de usuario en el Product Backlog y el equipo de desarrollo selecciona las historias de usuario sobre las que se trabajará. Como resultado de la reunión, el equipo de desarrollo hace una previsión del trabajo que será completada durante el Sprint.

Reunión diaria

Con una duración de no más de 15 minutos, participan el equipo de desarrollo y el Scrum Master. En esta reunión cada miembro del equipo presenta lo que hizo el día anterior, lo que va a hacer hoy y los impedimentos que se han encontrado.



Reunión de revisión del Sprint

Se realiza al final del Sprint. Participan el equipo de desarrollo, el Scrum Master y el Product Owner. Durante la misma se indica qué ha podido completarse y qué no, presentando el trabajo realizado al Product Owner. Por su parte el Product Owner (y demás interesados) verifican el incremento del producto y obtienen información

Reuniones

necesaria para actualizar el Product Backlog con nuevas historias de usuario. No debe durar más de 4 horas.

Retrospectiva del Sprint

También al final del Sprint (aunque puede que no se realice al final de todos los Sprints), sirve para que los integrantes del equipo Scrum y el Scrum Master den sus impresiones sobre el Sprint que acaba de terminar. Se utiliza para la mejora del proceso y normalmente se trabaja con dos columnas, con los aspectos positivos y negativos del Sprint. Esta reunión no debería durar más de 4 horas.

Beneficios del Scrum

La **implementación de las metodologías ágiles**, y, por lo tanto, de los principios ágiles, aporta una serie de beneficios como el aumento de la transparencia a lo largo de la gestión del proyecto, la mejora de la comunicación y la autogestión del equipo de desarrollo. Así mismo, existen otras ventajas que se obtienen al utilizar Scrum, entre las cuales te podemos comentar las siguientes:

Entrega periódica de resultados

El **Product Owner** establece sus expectativas indicando el valor que le aporta cada historia de usuario y cuándo espera que esté completado. Por otra parte, comprueba de manera regular si se van cumpliendo sus expectativas.

Entrega parciales

El cliente puede utilizar las primeras funcionalidades de la aplicación que se está construyendo antes de que esté finalizada por completo. Por lo tanto, el cliente puede empezar antes a recuperar su inversión. Por ejemplo, puede utilizar un producto al que sólo le faltan características poco relevantes, puede introducir en el mercado un producto antes que su competidor, puede hacer frente a nuevas peticiones de clientes, etc.

Flexibilidad y adaptación respecto a las necesidades del cliente

De manera regular el Product Owner redirige el proyecto en función de sus nuevas prioridades, de los cambios en el mercado, de los requisitos completados que le permiten entender mejor el producto, de la velocidad real de desarrollo, etc. Al final de cada iteración el Product Owner puede aprovechar la parte de producto completada hasta ese momento para hacer pruebas de concepto con usuarios o consumidores y tomar decisiones en función del resultado obtenido.

Mejores estimaciones

La estimación del esfuerzo y la optimización de tareas es mejor si la realizan las personas que van a desarrollar la historia de usuario, dadas sus diferentes especializaciones, experiencias y puntos de vista. De la misma manera, con iteraciones cortas la precisión de las estimaciones aumenta.

Beneficios del Scrum

A continuación, te presentamos, a modo de resumen gráfico, el ciclo de la metodología Scrum: