ESCUELA SUP. Nº 49 “J.J. de URQUIZA”

Tec. Sup. en Desarrollo de Software

Gestión de Proyectos de Software

Metodologías Ágiles: Scrum

3° Año

El surgimiento de las Metodologías Ágiles

Repasamos un poco de la historia del desarrollo de software

- La mayoría de las metodologías fueron introducidas desde la ingeniería civil, lo que resultó en un exhaustivo control sobre los procesos y las tareas.
- El Modelo en Cascada (Waterfall), se convirtió en el modelo metodológico más utilizado dentro de la industria. Data de principios de los años setenta y tiene sus orígenes en los ámbitos de la manufactura y la construcción, ambientes físicos altamente rígidos donde los cambios se vuelven prohibitivos desde el punto de vista de los costos, sino prácticamente imposibles.
- A medida que han pasado los años, y con el advenimiento de las economías globalizadas y los entornos web, el contexto de negocio de los sistemas ha pasado de ser relativamente estable a convertirse en un contexto altamente volátil, donde los requerimientos expresados hoy, en muy pocas oportunidades son válidos unos meses más tarde. Bajo esta nueva realidad, las metodologías Waterfall resultaron muy “pesadas” y prohibitivas para responder satisfactoriamente a los cambios de negocio.

El surgimiento de las Metodologías Ágiles

Repasamos un poco de la historia del desarrollo de software

- Bajo estos nuevos contextos, donde investigaciones sugieren que el involucramiento del usuario y el empleo de periodos de tiempo más cortos son claves para incrementar las tasas de proyectos exitosos, fueron surgiendo nuevas metodologías, como por ejemplo:
    - Metodologías en Espiral
    - Metodologías Iterativas
    - Metodologías Ágiles

El surgimiento de las Metodologías Ágiles

Repasamos un poco de la historia del desarrollo de software

- En los ‘90s surgieron varios movimientos identificados con el nombre de Metodologías Livianas (Lightweight Methodologies), entre estos se encuentran Extreme Programming (XP), Scrum, Software Craftmanship, Lean Software Development, etc.
- Más tarde, en febrero de 2001, se reunieron en Utah (EEUU) un grupo de diecisiete profesionales reconocidos del desarrollo de software, y referentes de las metodologías livianas existentes al momento, con el objetivo de determinar los valores y principios: Manifiesto Ágil, compuesto de 4 valores y 12 principios (desarrollado anteriormente).
- El objetivo era generar un marco que les permitirían a los equipos desarrollar software de forma más acertada con las necesidades del cliente y responder mejor alos cambios que pudieran surgir a lo largo de un proyecto de desarrollo.
- Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales, caracterizados por la rigidez y dominados por la documentación.

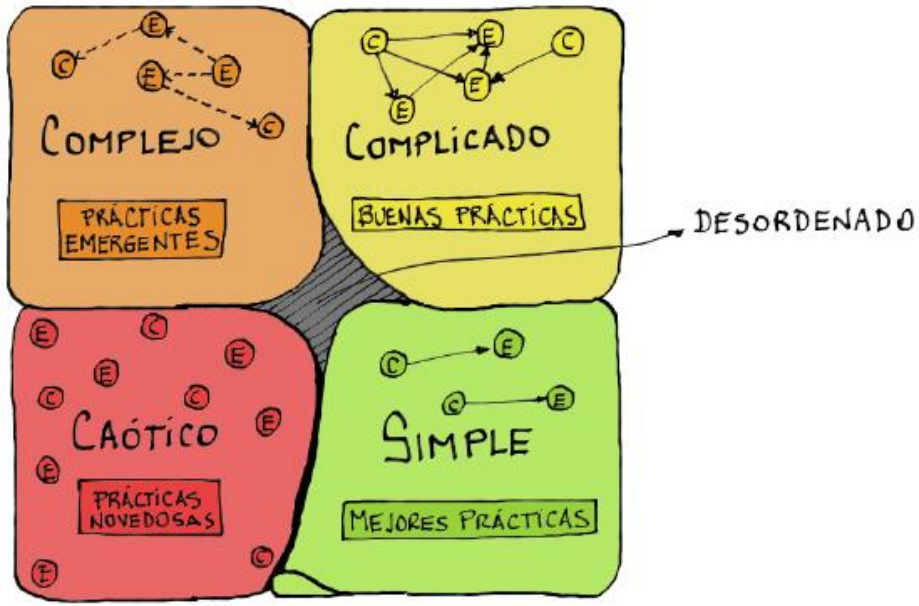
SCRUM:

- Antes de comenzar a desarrollar Scrum, vamos a entender el contexto en el cual es más eficiente.
- Utilizando el marco Cynefin para comprender las diferentes situaciones en las que nos podemos encontrar operando, y cuál es, según este enfoque, la manera más eficiente de responder a cada una de ellas.

SCRUM:

El marco Cynefin (Dave Snowden – 1999):

- La palabra Cynefin viene del galés y no tiene traducción literal. Lo más parecido en español sería hábitat, pero sería más correcto definirlo como “the place of our multiple belongings”: el lugar al que pertenecemos y que nos define.
- El marco Cynefin compara las características de cinco dominios de complejidad diferentes: simple, complicado, complejo, caótico y desordenado. Nos ayuda a entender cuál es la diferencia entre un contexto y otros; en qué contextos nos movemos y cómo debemos actuar en cada uno de ellos.



Marco Cynefin:

Dominio Simple:

- En este dominio se opera con problemáticas simples. Es muy fácil identificar las causas y sus efectos.
- Por lo general, la respuesta correcta es clara, conocida por todos e indiscutible.
- En este dominio existen las mejores prácticas, soluciones conocidas para problemas conocidos.
- Los procesos más eficientes en este dominio son aquellos que especifican una serie lógica de pasos y se ejecutan de manera repetitiva, una y otra vez.
- Ejemplos de este dominio son la construcción en serie de un mismo producto, la instalación en muchos clientes de un mismo sistema.
- Si bien Scrum puede funcionar en este contexto, los procesos compuestos por pasos bien definidos son mucho más eficientes.

Marco Cynefin:

Dominio Complicado:

- En este dominio encontramos problemas complejos, buenas prácticas y perfiles expertos.
- Hay múltiples soluciones correctas para una misma problemática, pero se requiere del involucramiento de expertos para poder identificarlas.
- Un ejemplo típico de este escenario es la solución de un problema de performance en un software o base de datos, la sincronización de semáforos en un cruce de 3 avenidas, la búsqueda de eficiencia en la distribución logística de mercaderías, etc.
- Si bien Scrum podría emplearse, no necesariamente sea la forma más eficiente de resolver estas situaciones, donde funcionaría mejor un conjunto de expertos en la materia que releven la situación, investiguen diferentes alternativas y planteen la solución en base a las buenas prácticas.
- Una práctica habitual de este dominio es el mantenimiento de sistemas y soporte técnico.

Marco Cynefin:

Dominio Complejo

- Cuando nos enfrentamos a problemas complejos, los resultados se vuelven más impredecibles.
- No existen ni mejores ni buenas prácticas catalogadas para las situaciones frente a las cuales nos podemos encontrar. Simplemente, no sabemos con anticipación si una determinada solución va a funcionar. Solo podemos examinar los resultados y adaptarnos.
- Este es el dominio de las prácticas emergentes. Las soluciones encontradas rara vez son replicables, con los mismos resultados, a otros problemas similares. Para poder operar en la complejidad necesitamos generar contextos donde haya lugar para la experimentación y donde el fallo sea de bajo impacto.
- Se requieren niveles altos de creatividad, innovación, interacción y comunicación.
- El desarrollo de nuevos productos o la incorporación de nuevas características en productos existentes es un contexto complejo en el que Scrum se utiliza mucho para actuar, inspeccionar y adaptar las prácticas emergentes de un equipo de trabajo.

Marco Cynefin:

Dominio Caótico

- Los problemas caóticos requieren una respuesta inmediata. Estamos en crisis y necesitamos actuar de inmediato para restablecer cierto orden. Imaginemos que el sistema de despacho de vuelos en un aeropuerto de alto tráfico deja de funcionar.

- Este no sería un escenario para utilizar Scrum.

- Aquí debemos actuar de inmediato, alguien debe tomar el control y mover la situación fuera del caos. Por ejemplo, solucionar el problema inmediatamente (sin importar la forma técnica), para luego, fuera del caos, evaluar y aplicar una solución más robusta, de ser necesario. Este es el dominio de la improvisación.

Marco Cynefin:

Dominio Desordenado:

- Nos movemos en el espacio desordenado cuando no sabemos en qué dominio estamos. Se la clasifica como una zona peligrosa, ya que no podemos medir las situaciones ni determinar la forma de actuar.

- Es muy típico en estas situaciones que las personas interpreten las situaciones y actúen en base a preferencias personales.

- El gran peligro del dominio desordenado es actuar de manera diferente a la que se necesita para resolver ciertos problemas. Por ejemplo, mucha gente en el ámbito del desarrollo de software está acostumbrada al desarrollo secuencial, por fases, detalladamente planificado utilizando las mejores prácticas de la industria, y este enfoque, que corresponde al dominio Simple, muchas veces se aplica en el dominio complejo.

- Si nos encontráramos en el espacio desordenado, todo lo que hagamos debe estar enfocado netamente a salirnos de ese espacio hacia uno mejor identificado, para luego actuar de la manera en que dicho dominio lo requiera.

Scrum:

¿Qué es Scrum?

- Scrum es un marco de trabajo que nos permite encontrar prácticas emergentes en dominios complejos, como la gestión de proyectos de innovación.

- No es un proceso completo, y mucho menos, una metodología.

- En lugar de proporcionar una descripción completa y detallada de cómo deben realizarse las tareas de un proyecto, genera un contexto relacional e iterativo, de inspección y adaptación constante para que los involucrados vayan creando su propio proceso.

- Esto ocurre debido a que no existen ni mejores ni buenas prácticas en un contexto complejo. Es el equipo de involucrados quien encontrará la mejor manera de resolver sus problemáticas.

- Este tipo de soluciones serán emergentes.

Scrum:

Generalidades

- El equipo de desarrollo se encuentra apoyado en dos roles: el ScrumMaster y el Product Owner.

- El ScrumMaster es quien vela por la utilización de Scrum, la remoción de impedimentos y asiste al equipo a que logre su mayor nivel de performance posible. Puede ser considerado un coach o facilitador encargado de acompañar al equipo de desarrollo.

- El Product Owner es quien representa al negocio, *stakeholders*, cliente y usuarios finales. Tiene la responsabilidad de conducir al equipo de desarrollo hacia el producto adecuado.

Scrum:

Generalidades

- El progreso de los proyectos que utilizan Scrum se realiza y verifica en una serie de iteraciones llamadas Sprints.

- Estos Sprints tienen una duración fija, pre-establecida de no más de un mes.

- Al comienzo de cada Sprint el equipo de desarrollo realiza un compromiso de entrega de una serie de funcionalidades o características del producto en cuestión.

- Al finalizar el Sprint se espera que estas características comprometidas estén terminadas, lo que implica su análisis, diseño, desarrollo, prueba e integración al producto.

- En este momento es cuando se realiza una reunión de revisión del producto construido durante el Sprint, donde el equipo de desarrollo muestra lo construido al Product Owner y a cualquier stakeholder interesado en participar.

- El feedback obtenido en esta reunión puede ser incluido entre las funcionalidades a construir en futuros Sprints.

Scrum:

Principios de Scrum: los valores del Manifiesto Ágil desde la perspectiva de Scrum.

1. **Individuos e interacciones por sobre procesos y herramientas:** Scrum se apoya en la confianza hacia las personas, sus interacciones y los equipos. Los equipos identifican lo que hay que hacer y toman la responsabilidad de hacerlo, removiendo todos los impedimentos que encuentren en su camino y estén a su alcance. Los equipos trabajan en conjunto con otras partes de la organización cuando los impedimentos están fuera de su ámbito de control.

2. **Software funcionando por sobre documentación exhaustiva:** Scrum requiere que al final de cada Sprint se entregue un producto funcionando. La documentación es entendida, en Scrum, como un producto intermedio sin valor de negocio. Los equipos pueden documentar tanto como crean necesario, pero ninguno de estos documentos pueden ser considerados cono el resultado de un Sprint. El resultado de un Sprint es, nuevamente, el producto funcionando. El progreso del proyecto se mide en base al producto funcionando que se entrega iterativamente.

Scrum:

3. **Colaboración con el cliente por sobre la negociación de contratos:** El Product Owner es el responsable de la relación que existe con los usuarios finales, stakeholders y áreas de la organización que van a obtener el beneficio del producto. El Product Owner es parte del Equipo Scrum y trabaja colaborativamente con el resto de los individuos dentro del equipo para asegurarse que el producto construido tenga la mayor cantidad posible de valor al final de cada iteración.

4. **Respuesta al cambio por sobre el seguimiento de un plan:** Scrum, por diseño, se asegura que todo el mundo dentro de un equipo tenga toda la información necesaria para poder tomar decisiones informadas sobre el proyecto en cualquier momento. El progreso es medido al final de cada Sprint mediante software funcionando y la lista de características pendientes está visible continuamente y para todos los miembros. Esto permite que elalcance del proyecto cambie constantemente en función de la retroalimentación provista por los *stakeholders*. Fomentar el cambio es una ventaja competitiva.

Scrum:

Cinco Valores de Scrum:

1. **Foco:** Los Equipos Scrum se enfocan en un conjunto acotado de características por vez. Esto permite que al final de cada Sprint se entregue un producto de alta calidad y, adicionalmente, se reduce el *time-tomarket*.

2. **Coraje:** Debido a que los Equipos Scrum trabajan como verdaderos equipos, pueden apoyarse entre compañeros, y así tener el coraje de asumir compromisos desafiantes que les permitan crecer como profesionales y como equipo.

3. **Apertura:** Los Equipos Scrum privilegian la transparencia y la discusión abierta de los problemas. No hay agendas ocultas ni triangulación de conflictos. La sinceridad se agradece y la información está disponible para todos, todo el tiempo.

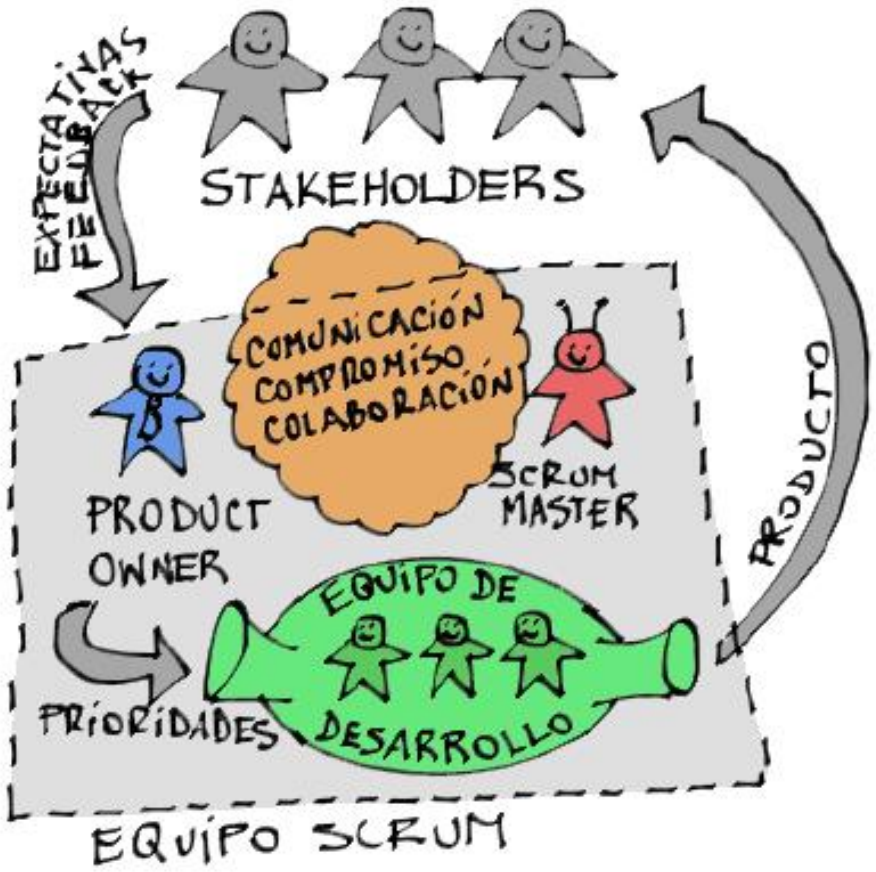
4. **Compromiso:** Los Equipos Scrum tienen mayor control sobre sus actividades, por eso se espera de su parte el compromiso profesional para el logro del éxito.

5. **Respeto:** Debido a que los miembros de un Equipo Scrum trabajan de forma conjunta, compartiendo éxitos y fracasos, se fomenta el respeto mutuo, y la ayuda entre pares es una cuestión a respetar.

Roles de Scrum

En un Equipo Scrum se espera que intervengan tres roles:

- Product Owner
- Equipo de Desarrollo
- ScrumMaster



Roles de Scrum

Product Owner

El Product Owner es la persona responsable del éxito del producto desde el punto de vista de los *stakeholders*.

Entre sus principales responsabilidades destacan:

- Determinar la visión del producto, hacia dónde va el equipo de desarrollo
- Recolectar los requerimientos
- Determinar y conocer en detalle las características funcionales de alto y de bajo nivel, y sus prioridades.
- Generar y mantener el plan de entregas (*release plan*)
- Cambiar las prioridades de las características según avanza el proyecto, acompañando así los cambios en el negocio
- Aceptar/rechazar el producto construido durante el Sprint y proveer feedback valioso para su evolución
- Participar de la revisión del Sprint junto a los miembros del Equipo de Desarrollo para obtener feedback de los *stakeholders*.

Roles de Scrum

Product Owner

- El Product Owner se focaliza en maximizar la rentabilidad del producto. La principal herramienta con la que cuenta para poder realizar esta tarea es la priorización. De esta manera puede reordenar la cola de trabajo del equipo de desarrollo para que éste construya con mayor anticipación las características o funcionalidades más requeridas por el mercado o la competitividad comercial.

- Otra responsabilidad importante del Product Owner es la gestión de las expectativas de los stakeholders mediante la comprensión completa de la problemática de negocio y su descomposición hasta llegar al nivel de requerimientos funcionales.

Roles de Scrum

Equipo de Desarrollo:

- Está formado por todos los individuos necesarios para la construcción del producto en cuestión. Es el único responsable por la construcción y calidad del producto. Deberá transformar las funcionalidades comprometidas en software funcionando y con calidad productiva, o en otras palabras, producir un incremento funcional potencialmente entregable.

- Es auto-organizado. Esto significa que no existe un líder externo que asigne las tareas ni que determine la forma en la que serán resueltos los problemas. Es el mismo equipo quien determina la forma en que realizará el trabajo y cómo resolverá cada problemática que se presente.

- Es recomendable que un equipo de desarrollo se componga de hasta nueve personas. Cada una de ellas debe poseer todas las habilidades necesarias para realizar el trabajo requerido (multi-funcionalidad), lo que significa que dentro del equipo de desarrollo no existen especialistas exclusivos, sino más bien individuos generalistas con capacidades especiales.

- Lo que se espera de un miembro de un equipo de desarrollo es que no solo realice las tareas en las cuales se especializa sino también todo lo que esté a su alcance para colaborar con el éxito del equipo.

Roles de Scrum

Equipo de Desarrollo:

El equipo de desarrollo tiene tres responsabilidades:

- La primera es proveer las estimaciones de cuánto esfuerzo será requerido para cada una de las características del producto.

- La segunda responsabilidad es comprometerse al comienzo de cada Sprint a construir un conjunto determinado de características en el tiempo que dura el mismo.

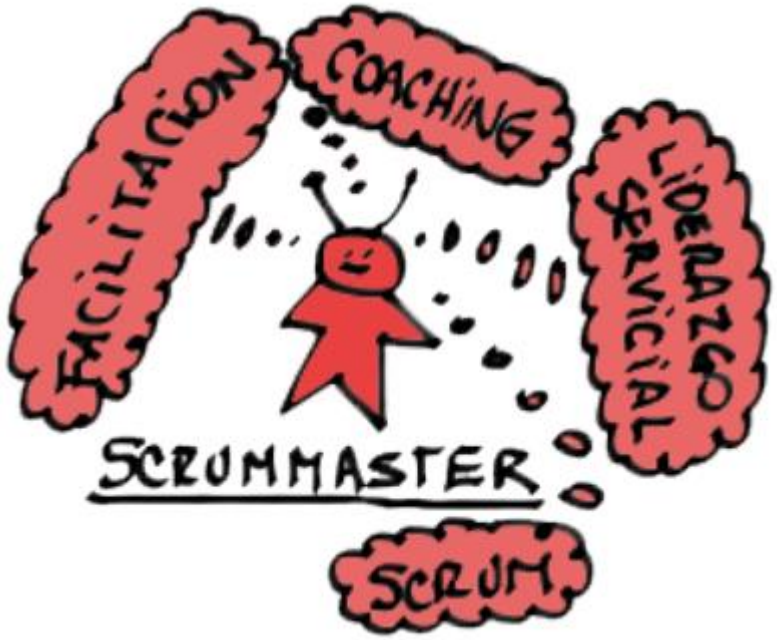
- Y finalmente, también es responsable por la entrega del producto terminado al finalizar cada Sprint

Roles de Scrum:

ScrumMaster

El ScrumMaster es el Coach del equipo y es quien lo ayuda a alcanzar su máximo nivel de productividad posible. Se dice que es un *líder, facilitador, provocador, detective y motivador*.

Se espera que acompañe al equipo de trabajo en su día a día y garantice que todos, incluyendo al Product Owner, comprendan y utilicen Scrum de forma correcta.



Roles de Scrum:

ScrumMaster

Las responsabilidades principales del ScrumMaster son:

- Velar por el correcto empleo y evolución de Scrum
- Facilitar el uso de Scrum a medida que avanza el tiempo. Esto incluye la responsabilidad de que todos asistan a tiempo a las daily meetings, reviews y retrospectivas.
- Asegurar que el equipo de desarrollo sea multifuncional y eficiente
- Proteger al equipo de desarrollo de distracciones y trabas externas al proyecto
- Detectar, monitorear y facilitar la remoción de los impedimentos que puedan surgir con respecto al proyecto y a la metodología
- Asegurar la cooperación y comunicación dentro del equipo

Además de estas cuestiones, el ScrumMaster debe detectar problemas y conflictos interpersonales dentro del equipo de trabajo, e involucrarse en caso de que los mismos no sean resueltos por el mismo equipo.

Elementos de Scrum:

El proceso de Scrum posee una mínima cantidad necesaria de elementos formales para poder llevar adelante un proyecto de desarrollo:

- Product Backlog
- Sprint Backlog

Elementos de Scrum:

Product Backlog

El Backlog del Producto es básicamente un listado de ítems (Product Backlog Ítems, PBIs) o características del producto a construir, mantenido y priorizado por el Product Owner.

Es importante que exista una clara priorización, ya que es esta priorización la que determinará el orden en el que el equipo de desarrollo transformará las características (ítems) en un producto funcional acabado.

Elementos de Scrum:

Product Backlog: Priorización

Priorización por valor de negocio de cada PBI:

Una forma de priorizar los ítems del Product Backlog es según su valor de negocio. Podemos entender el valor de negocio como la relevancia que un ítem tiene para el cumplimiento del objetivo de negocio que estamos buscando.

Priorización por retorno de la inversión (ROI) de cada PBI:

Un enfoque diferente de medir la prioridad de un determinado ítem del Backlog es calcular el beneficio económico que se obtendrá en función de la inversión que se deba realizar. Esto implica encontrar o conocer el valor económico ganado por la incorporación de una determinada característica a un producto. Una vez identificada, el cálculo es relativamente simple:

ROI = valor de negocio/costo

Donde el costo representa el esfuerzo necesario para la construcción de una determinada característica de un producto y el valor de negocio es el rédito económico obtenido por su incorporación.

Elementos de Scrum:

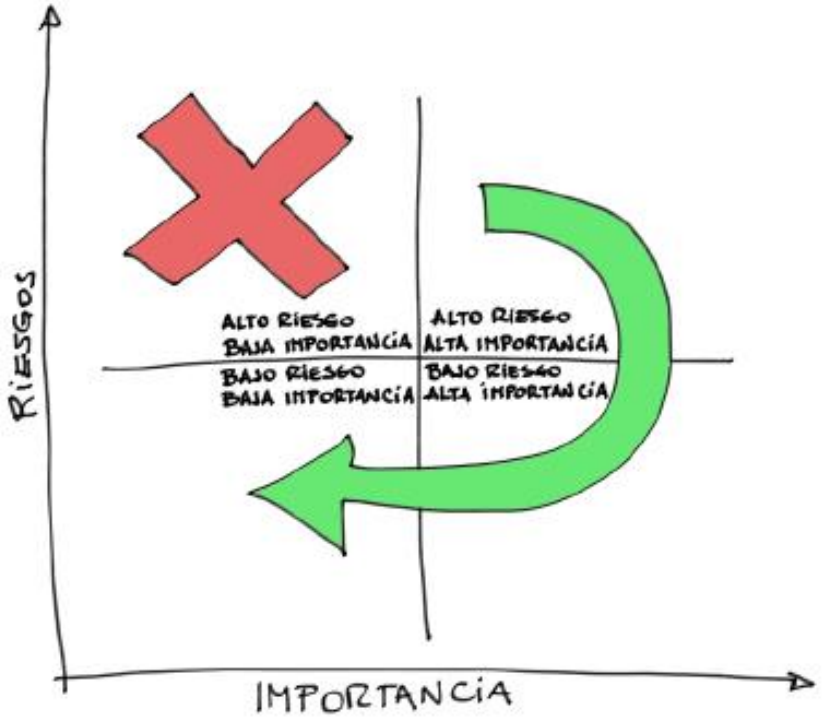
Product Backlog: Priorización

Prioridades según la importancia y el riesgo de cada PBI

Las anteriores formas de priorización determinan en base a algún factor, la importancia de cada PBI. Adicionalmente, estos pueden verse complementariamente afectados por el nivel de riesgo asociado a cada uno de ellos.

De esta manera, deberíamos aprovechar la construcción iterativa y evolutiva de Scrum para mitigar riesgos en forma implícita: construyendo primero aquellas características con mayor riesgo asociado y dejando las que poseen menor riesgo para etapas posteriores.

Se recomienda que los PBIs de baja importancia y alto riesgo sean evitados, por ejemplo, transfiriéndolos o eliminándolos del alcance.

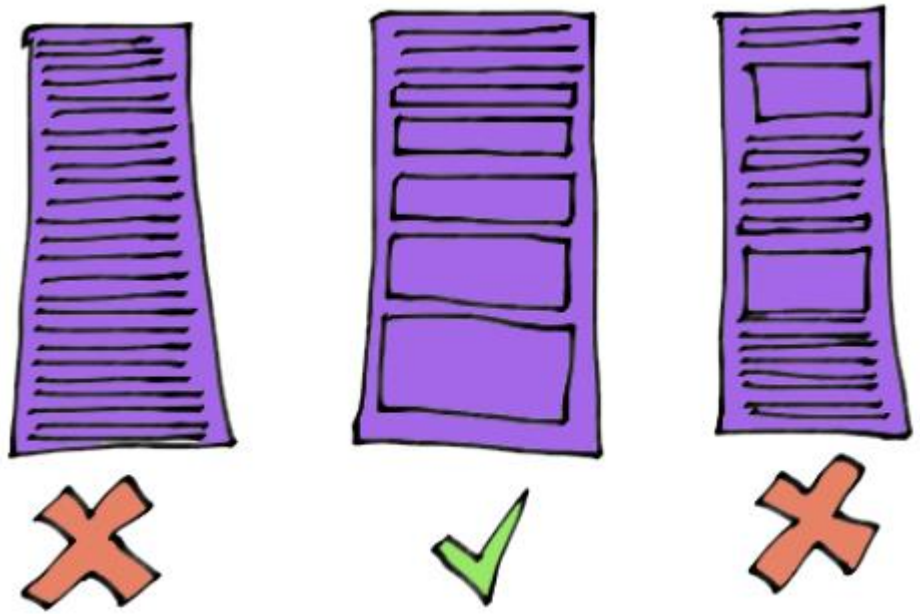


Elementos de Scrum:

Product Backlog: Eficiencia

Cuando hablamos de eficiencia, hablamos de obtener el mayor beneficio con el menor esfuerzo posible. Este concepto llevado al Product Backlog significa invertir el esfuerzo de exploración y especificación de la manera más inteligente posible para evitar re-trabajos y desperdicios.

Por esto, fomentamos un Product Backlog donde sus ítems más prioritarios están expresados con un nivel de detalle mucho mayor que los ítems de menor prioridad, los cuales están descriptos a un nivel más alto, ya que son los más susceptibles de ser alterados o reemplazados.

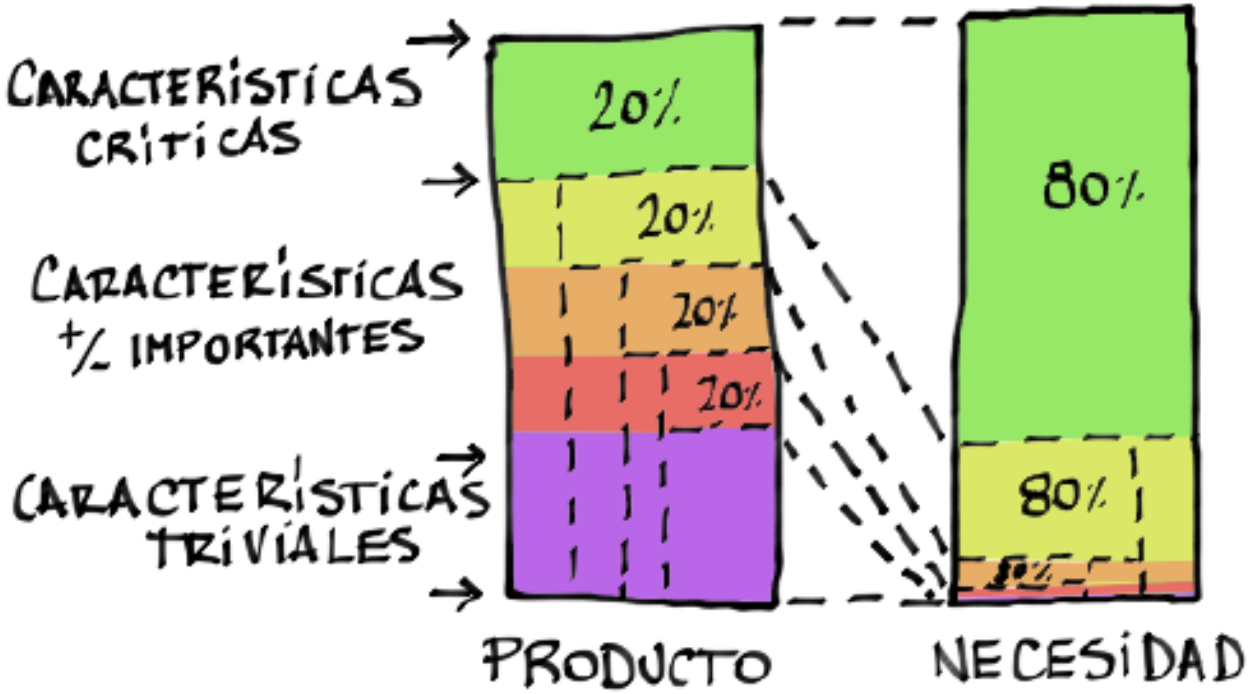


Elementos de Scrum:

Product Backlog: Eficiencia

Aplicando el Principio de Paretto al desarrollo de software, podemos decir que el 20% de las características de un sistema resuelven el 80% de la necesidad de negocio que le da origen.

Y de manera recursiva, el 20% del 80% restante de las características, resuelven el 80% del 20% restante de negocio.



Elementos de Scrum:

Product Backlog: Manejo de Contingencias

Aprovechando que el alcance es variable y que todo lo que debemos realizar está priorizado en el Product Backlog según su impacto en el negocio, vamos a utilizar las características menos prioritarias del producto como la contingencia del proyecto frente a imprevistos.

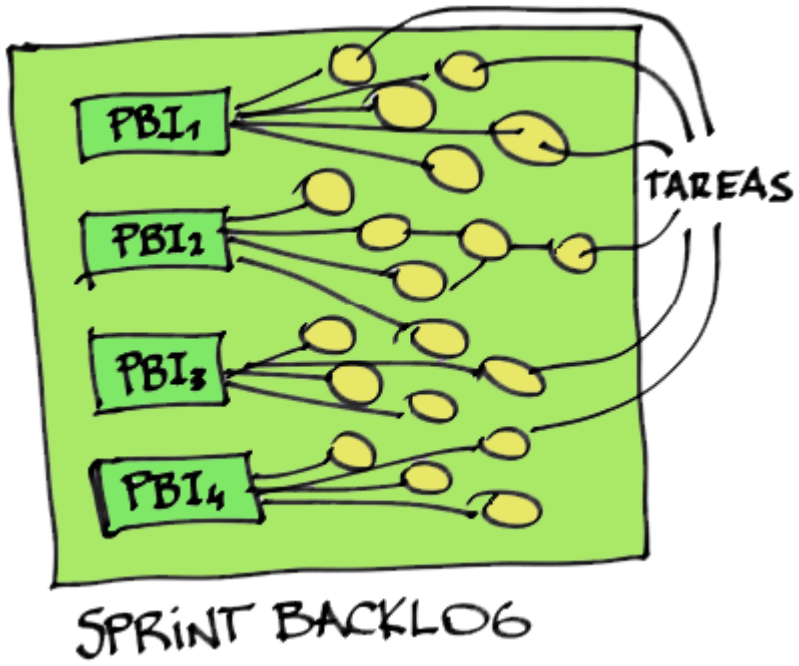
Esto quiere decir que, al respetar tiempo y costo, el alcance de menor prioridad sería el que pagaría el precio de retrasos o desvíos.

Para que este enfoque sea eficaz, es fundamental la labor del Product Owner y su habilidad para facilitar el descubrimiento de las prioridades por parte de todos los involucrados.

Elementos de Scrum:

Sprint Backlog

El Sprint Backlog es el conjunto de PBIs que fueron seleccionados para trabajar en ellos durante un cierto Sprint, conjuntamente con las tareas que el equipo de desarrollo ha identificado que debe realizar para poder crear un incremento funcional potencialmente entregable al finalizar el Sprint.



Elementos de Scrum:

Sprint Backlog

El resultado de cada Sprint debe ser un **Incremento Funcional Potencialmente Entregable**.

Incremento funcional porque es una característica funcional nueva (o modificada) de un producto que está siendo construido de manera evolutiva. El producto crece con cada Sprint.

Potencialmente entregable porque cada una de estas características se encuentra lo suficientemente validada y verificada como para poder ser desplegada en producción (o entregada a usuarios finales) si así el negocio lo permite o el cliente lo desea.

Dinámica de Scrum:

- Sprint
- Sprint Planning Meeting
- Daily Meetings
- Sprint Review
- Retrospectiva
- Refinamiento del Backlog

Dinámica de Scrum:

Sprint

- Las iteraciones en Scrum se conocen como Sprints.

- Una de las decisiones que debemos tomar al comenzar un proyecto o al adoptar Scrum es justamente la duración de los Sprints.

- Luego, el objetivo será mantener esta duración constante a lo largo del desarrollo del producto, lo que implicará que la duración de una iteración no cambie una vez que sea establecida.

- Se recomienda una duración de Sprint de entre 1 y 4 semanas, siendo 2 semanas lo más habitual.

- Podremos encontrar situaciones en donde el equipo de desarrollo se atrase o se adelante. En estos casos, la regla del *timeboxing* no nos permitirá modificar (adelantar o postergar) la fecha de entrega o finalización del Sprint. La variable de ajuste en estos casos será el alcance del Sprint, esto es, en el caso de adelantarnos deberemos incrementar el alcance del Sprint agregando nuevos PBIs y reducirlo en el caso de retrasarnos.

Dinámica de Scrum:

Sprint Planning Meeting

Al comienzo de cada Sprint se realiza una reunión de planificación del Sprint donde serán generados los acuerdos y compromisos entre el equipo de desarrollo y el Product Owner sobre el alcance del Sprint

Se divide en dos partes con finalidades diferentes: una primera parte estratégica y enfocada en el “qué”, y una segunda parte táctica cuyo hilo conductor principal es el “cómo”.

Dinámica de Scrum:

Sprint Planning Meeting: Parte 1 - ¿Qué trabajo será realizado?

Se trata de un taller donde el Product Owner expone todos y cada uno de los PBIs que podrían formar parte del Sprint, mientras que el equipo de desarrollo realiza todas las preguntas que crea necesarias para conocer sus detalles y así corroborar o ajustar sus estimaciones.

El objetivo es identificar “qué” es lo que el equipo de desarrollo va a realizar durante el Sprint, es decir, todos aquellos PBIs que el equipo se comprometerá a transformar en un producto funcionando y utilizable o en otras palabras: incremento funcional potencialmente entregable.

El equipo de desarrollo utiliza su capacidad productiva (también conocida como Velocidad o *Velocity*), obtenida de los Sprints pasados, para conocer hasta cuánto trabajo podría comprometerse a realizar.

Dinámica de Scrum:

Sprint Planning Meeting: Parte 2 - ¿Cómo será realizado el trabajo?

Durante este espacio de tiempo el equipo de desarrollo determinará la forma en la que llevará adelante el trabajo. Esto implica la definición inicial de un diseño de alto nivel, el cual será refinado durante el Sprint mismo y la identificación de las actividades que el equipo en su conjunto tendrá que llevar a cabo.

Si bien el Product Owner no participa de esta reunión, debería ser contactado en el caso de que el equipo de desarrollo necesite respuestas a nuevas preguntas con la finalidad de clarificar su entendimiento de las necesidades.

Al finalizar esta reunión, el equipo habrá arribado a un Sprint Backlog o Commited Backlog que representa el alcance del Sprint en cuestión. Este Sprint Backlog es el que se coloca en el taskboard (pizarra de actividades) del equipo. Se dará comienzo al desarrollo del producto para este Sprint.

Dinámica de Scrum:

Daily Meetings:

Estas reuniones tienen, como su nombre lo indica, una frecuencia diaria y no deberían llevar más de 15 minutos.

Tiene tres objetivos: 1) incrementar la comunicación 2) explicitar los compromisos y 3) dar visibilidad a los impedimentos

No se trata de una reunión de reporte de avance o status.

A estas reuniones acuden el ScrumMaster y el equipo de trabajo. En el caso de que sea necesario, se podrá requerir la presencia del Product Owner y de los stakeholders.

El ScrumMaster actúa como facilitador, y todos y cada uno de los miembros toman turnos para responder las siguientes tres preguntas, y de esa manera comunicarse entre ellos:

1. ¿Qué hice desde la última reunión diaria hasta ahora?
2. ¿En qué voy a estar trabajando desde ahora hasta la próxima reunión diaria?
3. ¿Qué problemas o impedimentos tengo?

Dinámica de Scrum:

Daily Meetings:

El objetivo de la primera pregunta (¿qué hice...?) es verificar el cumplimiento de los compromisos contraídos por los miembros del equipo en función del cumplimiento del objetivo del Sprint.

La finalidad de la segunda pregunta (¿qué voy a hacer...?) es generar nuevos compromisos hacia el futuro. Cuando hablamos de compromisos, hacemos referencia a aquéllos que los miembros del equipo asumen ante sus compañeros.

La última pregunta (¿qué problemas...?) apunta a detectar y dar visibilidad a los impedimentos. Estos impedimentos no se resuelven en esta reunión, sino en posteriores. Es responsabilidad del ScrumMaster que se resuelvan lo antes posible, generando las reuniones que sean necesarias e involucrando a las personas correctas.

Dinámica de Scrum:

Sprint Review:

Se realiza al finalizar el sprint.

Sirve para evaluar el incremento funcional potencialmente entregable construido por el equipo de desarrollo (el “qué”). En esta reunión el Equipo Scrum y los Stakeholders revisan el resultado del Sprint. Cuando decimos “resultado” hablamos de “producto utilizable” y “potencialmente entregable” que el los interesados utilizan y evalúan durante esta misma reunión, aceptando o rechazando así las funcionalidades construidas.

Los *Stakeholders* evalúan el producto construido y proveen feedback. Este feedback puede ser acerca de cambios en la funcionalidad construida o bien nuevas funcionalidades que surjan al ver el producto en acción

Toda la retroalimentación que los stakeholders aporten debe ser ingresada como PBIs en el Product Backlog, debiendo los mismos ser priorizados y estimados.

En el caso de que una funcionalidad sea rechazada, el PBI correspondiente reingresa al Product Backlog con máxima prioridad, para ser tratado en el siguiente Sprint.

Dinámica de Scrum:

Retrospectiva

En un método empírico como Scrum, la retrospección del equipo es el corazón de la mejora continua y las prácticas emergentes. Mediante el mecanismo de retrospección, el equipo reflexiona sobre la forma en la que realizó su trabajo y los acontecimientos que sucedieron en el Sprint que acaba de concluir para mejorar sus prácticas. Todo esto sucede durante la reunión de retrospectiva.

Tiene lugar inmediatamente después de la reunión de revisión. Mientras que la reunión de revisión se destina a revisar el producto (el “qué”), la retrospectiva se centra en el proceso (el “cómo”).

Este tipo de actividad necesita un ambiente seguro donde el Equipo Scrum pueda expresarse libremente, sin censura ni temores, por lo cual se restringe solo al Equipo de Desarrollo y al ScrumMaster.

Valiéndose de técnicas de facilitación y análisis de causas raíces, se buscan tanto fortalezas como oportunidades de mejora. Luego, el Equipo Scrum decide por consenso cuáles serán las acciones de mejora a llevar a cabo en el siguiente Sprint. Estas acciones y sus impactos se revisarán en la próxima reunión de retrospectiva.

Otro objetivo importante que se debe perseguir en esta reunión es la detección de riesgos implícitos en los PBIs que se estén analizando

Scrum: Desarrollo Evolutivo

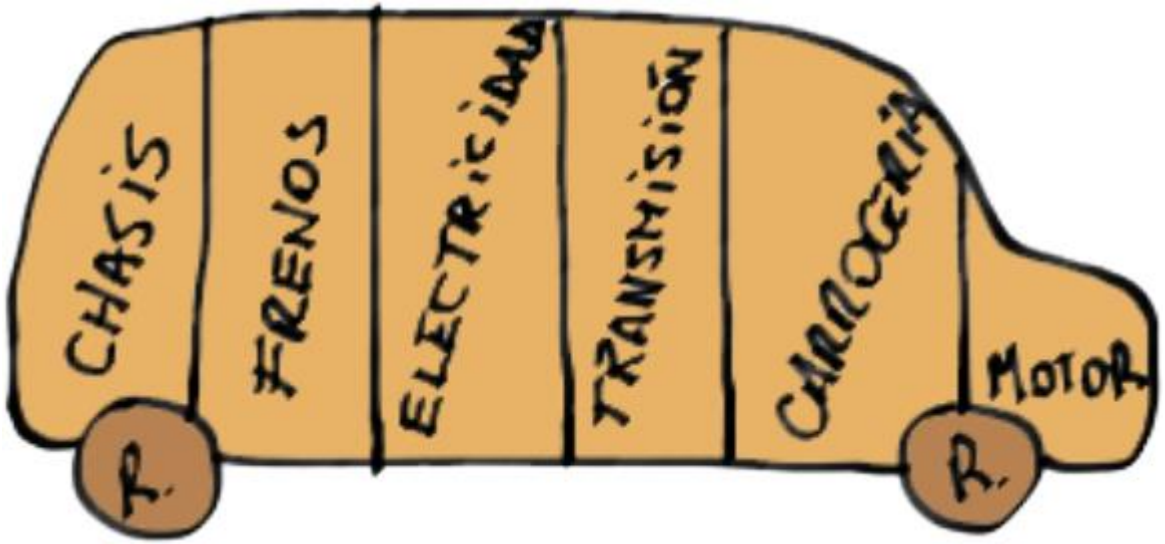
- Minimum Viable Product
- Minimum Marketable Feature
- Visual Story Mapping

Desarrollo Evolutivo

Creación Evolutiva:

- Supongamos que nos han contratado de una empresa de transporte para construir autobuses para el traslado de niños desde su casa a la escuela y desde la escuela a su casa.

- Luego de analizar las características o funcionalidades que el autobús debe tener, hemos dividido la problemática en:

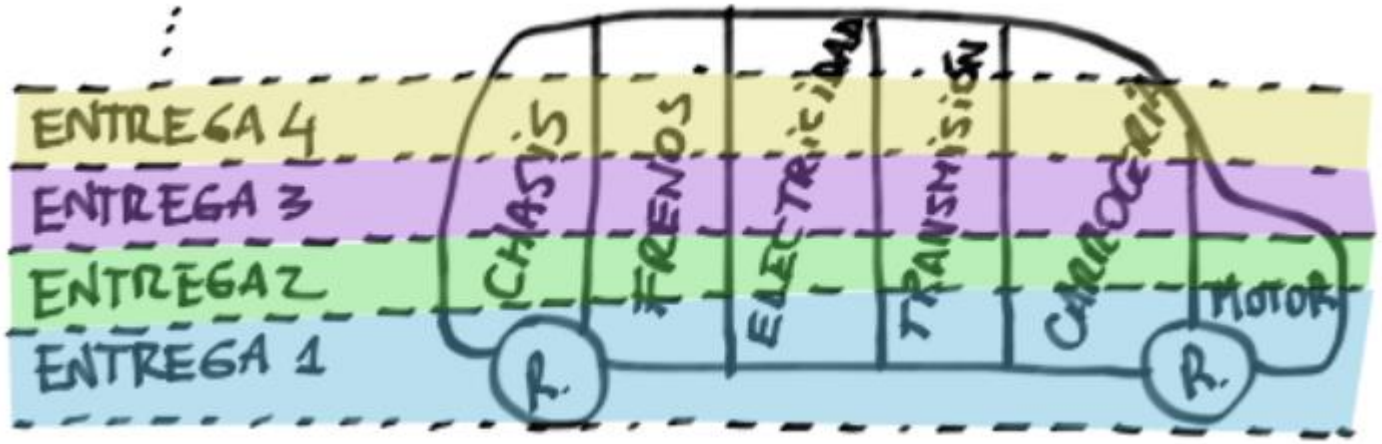


- Una alternativa para construir el autobús sería dividirlo en 4 iteraciones:
  - Iteración 1: Chasis y frenos
  - Iteración 2: Motor y Carrocería
  - Iteración 3: Electricidad y transmisión
  - Iteración 4: Ruedas

Desarrollo Evolutivo

Creación Evolutiva:

Sin embargo, si nosotros decidimos construir el vehículo utilizando un marco de desarrollo Scrum (que es iterativo e incremental), deberíamos tener una unidad funcionando al final de cada iteración (sprint), lo que significa segmentar el desarrollo de forma transversal a dichas funcionalidades con el fin de proveer una pequeña porción de cada una en cada entrega, formando un producto utilizable:



De esta forma, podríamos buscar los siguientes objetivos:

- Entrega 1: base del chasis, ruedas de motocicleta, dirección básica. Objetivo: verificar rodamiento, traslado seguro y maniobrabilidad
- Entrega 2: sumamos refuerzos del chasis, transmisión básica, suspensores, ruedas de segunda mano, motor básico, paredes de la carrocería. Objetivo: verificar dimensiones y suspensión.
- Entrega 3: electricidad, motor diesel, frenos, techo, asientos, ruedas nuevas, sistema ABS. Objetivo: Habilidad de circular públicamente.
- Entrega 4: etc.

Desarrollo Evolutivo

Minimum Viable Product:

El Minimum Viable Product (MVP) es la versión mínima de un producto, tal que nos permita recolectar la mayor cantidad de información de nuestro mercado y clientes con el menor esfuerzo posible.

Consiste en hacer foco en las características mínimas y necesarias para que el producto pueda lanzarse al mercado. Esto nos permitirá:

- Evitar crear productos que nadie necesita
- Maximizar el aprendizaje por unidad monetaria invertido

El MVP es una estrategia de *Lean Startup* que apunta a acercarnos a nuestros clientes con la menor inversión posible (tiempo/dinero) y con ello determinar si nuestro producto es o no es viable

Desarrollo Evolutivo

Minimum Marketable Features

Todas las metodologías ágiles coinciden en que un producto debe construirse de forma evolutiva en pequeñas entregas.

De todas formas no es suficiente, como vimos anteriormente, dividir el producto en tres o cuatro entregas sucesivas, sino que debemos hacerlo de forma criteriosa para que cada entrega pueda aportar valor suficiente a los usuarios finales.

Esos grupos de características se denominan MMF: Minimum Marketable Features, y pueden definirse como "*el conjunto más pequeño posible de funcionalidad que, por si misma, tiene valor en el mercado*"

Desarrollo Evolutivo

Visual Story Mapping

Conjugando el Desarrollo Evolutivo, la Priorización del Backlog y el concepto de Minimum Marketable Feature, Jeff Patton plantea una técnica de Análisis Ágil llamada **Mapeo Visual de Historias** o *Visual Story Mapping.*

La teoría del Visual Story Mapping comienza en un nivel "humano" identificando los **Objetivos** que toda persona persigue y dividiéndolos en **Actividades** para las cuales deben utilizarse **Herramientas,** resultando entonces en una jerarquía de:

- Objetivos → Actividades → Herramientas.

El último nivel denominado "herramientas", puede desagregarse a su vez en diferentes niveles de confort. Por este mismo principio una persona puede viajar de una ciudad a otra en un automóvil del año 1965 o en un último modelo siendo que la actividad "llegar de una ciudad a otra" seguirá cumpliéndose.

Desarrollo Evolutivo

Visual Story Mapping

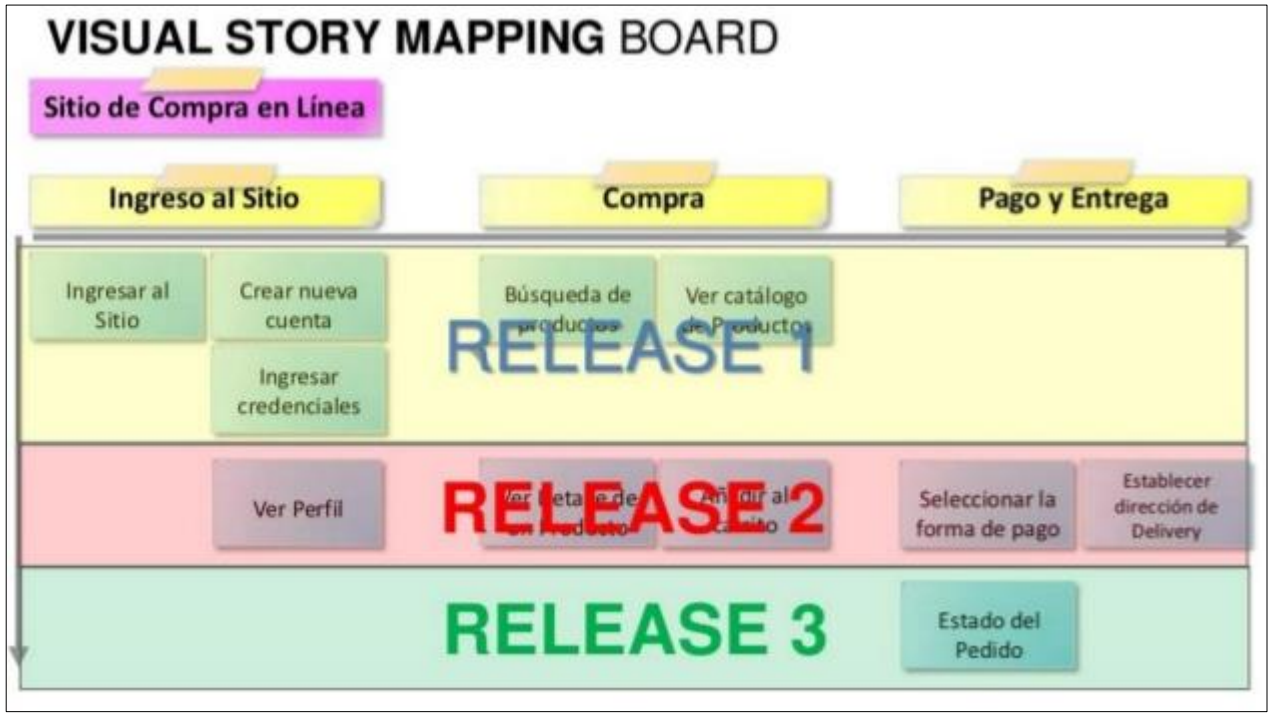
Haciendo una analogía con las organizaciones, esta jerarquía de Objetivo → Actividad → Herramienta, puede traducirse en:

- Proceso de Negocio → Actividad → Software.

El Software como herramienta también puede otorgarnos diferentes niveles de confort. Uniendo entonces el concepto de MMF y de nivel de confort, deberíamos pensar la construcción del software de forma evolutiva, naciendo desde lo mínimo posible (MMF) e ir escalando en los niveles de confort de las funcionalidades iteración tras iteración, tratando de abarcar tanta funcionalidad como sea posible en la extensión del proceso de negocio.

Teniendo en cuenta entonces que el software será construido evolutivamente, incrementando la funcionalidad entrega tras entrega y sumando elementos visuales, surge entonces una herramienta colaborativa para analizar el alcance del software A ser construido y para dividirlo en diferentes entregas.

Esta técnica visual utiliza elementos físicos como marcadores, notas autoadhesivas y papel afiche con el propósito de fomentar la colaboración entre las personas.



Desarrollo Evolutivo

Proceso de Análisis Ágil

Iremos describiendo esta técnica en forma práctica, basándonos en un ejemplo real: el análisis de un sistema de gestión de cursos de capacitación

En primer lugar identificaremos los diferentes Roles de Usuario presentes, y luego pasaremos a crear el Visual Story Mapping.

Desarrollo Evolutivo

Roles de Usuario:

Previo al análisis del sistema, es necesario identificar los, posibles usuarios que tendrá. Para esto utilizaremos una técnica colaborativa descripta por Mike Cohn basada en el trabajo de Constantine & Lockwood. Esta técnica se realiza en equipo, durante un taller donde el cliente y tantos desarrolladores como sea posible colaboran en la identificación de los roles. El taller se compone de cuatro actividades:

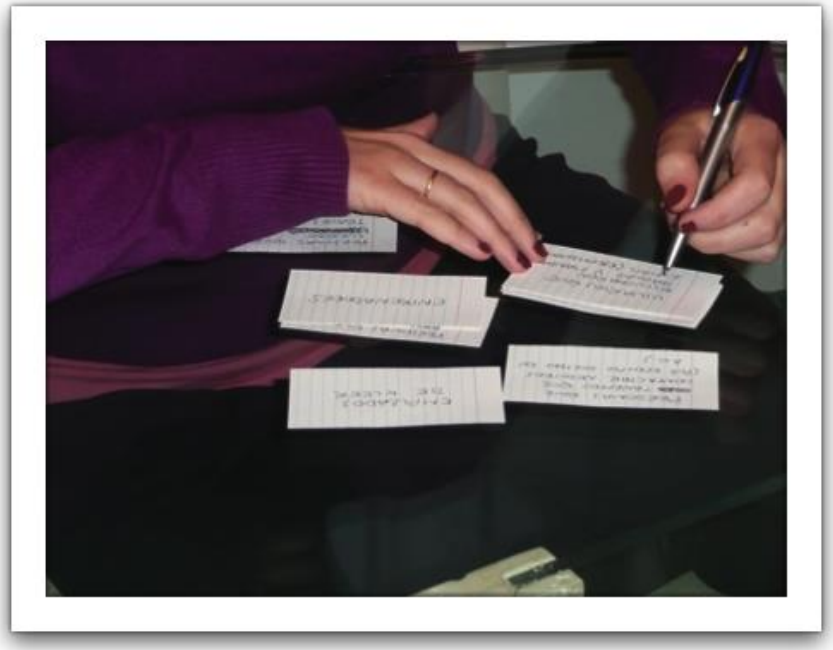
1. Brainstorming de un conjunto inicial de roles
2. Organización del conjunto inicial de roles
3. Consolidación de roles
4. Refinamiento de roles

Desarrollo Evolutivo

Roles de Usuario: Brainstorming de un conjunto inicial de roles

- La intención de esta actividad es que sea lo más colaborativa posible. Tanto el cliente como el Equipo completo deberían participar.

- Cada participante escribe tanto roles como sea posibles en fichas o posts, en silencio y sin compartirlos con el resto de las personas. Siendo que esta actividad es un *Brainstorming* no debe haber discusión ni censura para cada rol que alguien escribe.



Comercial

Interesado

Alumno

Gestor de Cobranza

Estudiante

Gestor de Logística

Cobrador

Facturador

Desarrollo Evolutivo

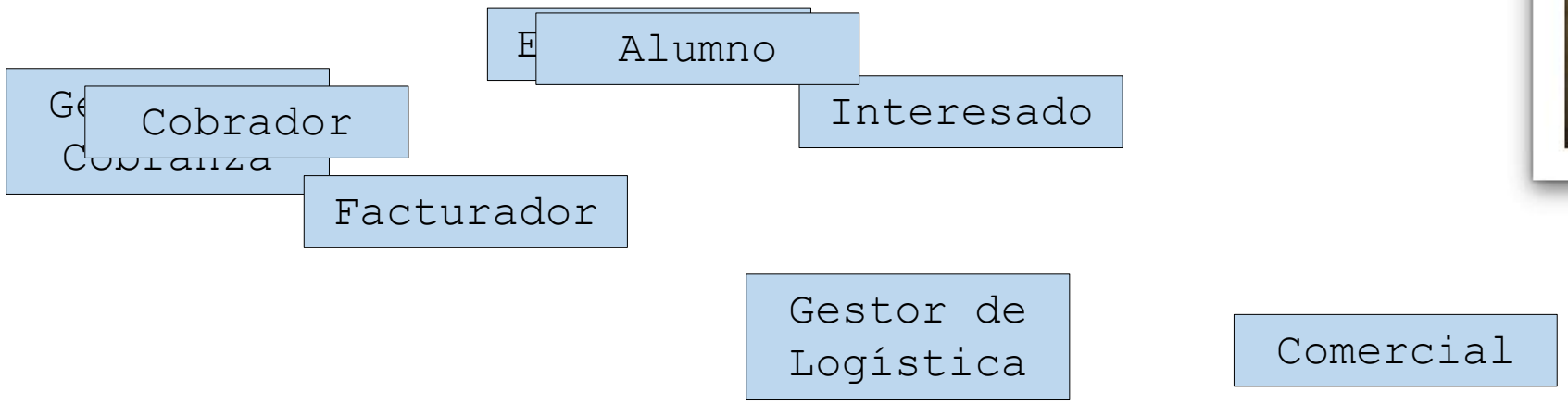
Roles de Usuario: Organización del conjunto inicial de roles

- Una vez que el grupo haya terminado de identificar los roles, el próximo paso es organizarlos. Para esto, se los dispondrá sobre de forma tal que las similitudes queden representadas de forma visual.

- Esto se logra solapando levemente aquellos roles que tienen pocas similitudes, solapando por completo aquellos que son iguales y separando los que no tienen relación.

- Los participantes deben compartir los roles con el resto del equipo y describir cada uno de ellos, discutiendo e indagando para poder entender las similitudes y diferencias.

- Esto a su vez ayudará a diseminar el conocimiento entre los integrantes del Equipo.



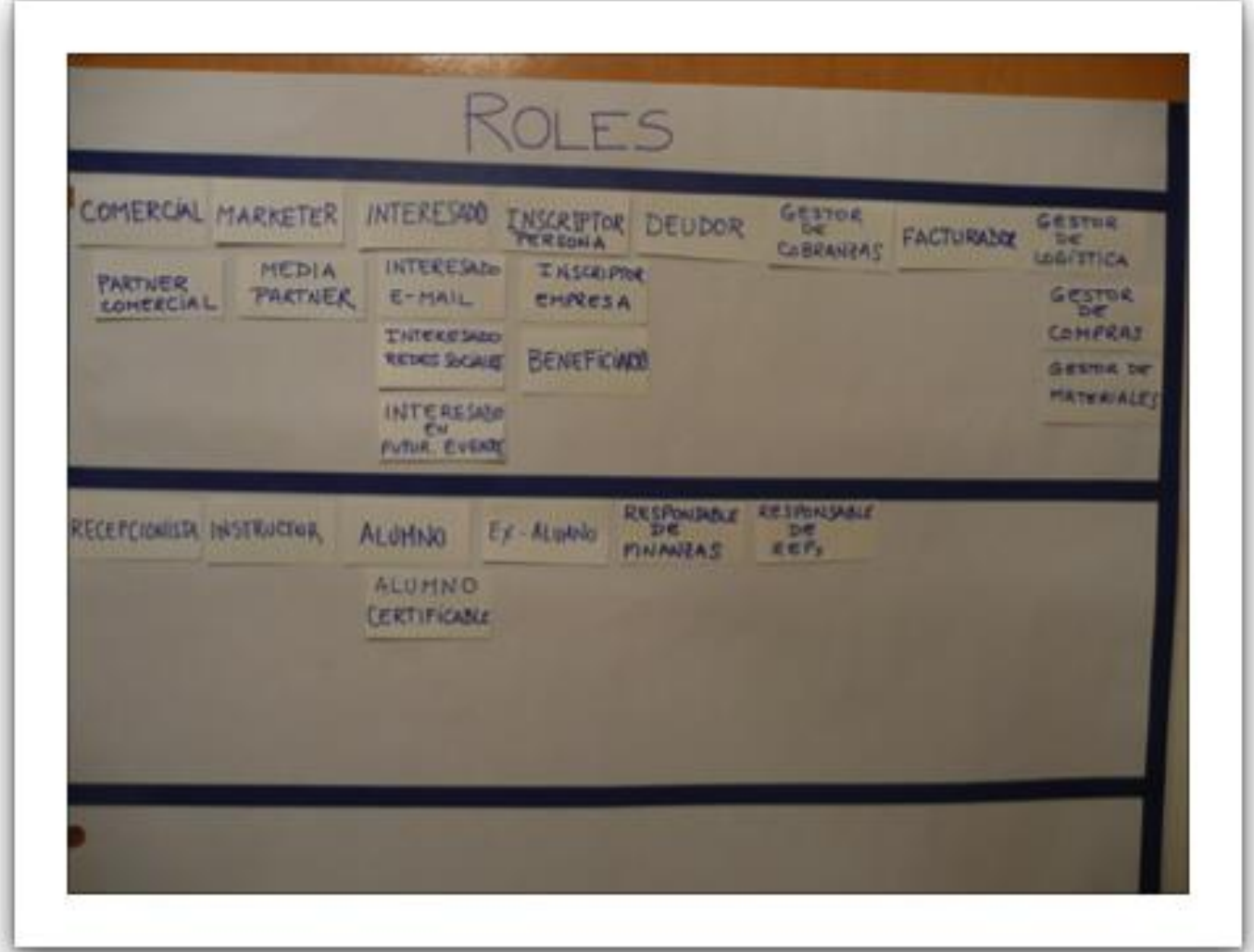
Desarrollo Evolutivo

Roles de Usuario: Consolidación de roles

- Luego de haber agrupado los roles, el siguiente paso será consolidar y condensar.

- Para esto se comienza por aquellas fichas que tienen el mayor solapamiento, se discuten para entender si podrían condensarse en un único rol y, en el caso de que sea posible hacerlo, se buscará un único nombre para que las represente.

- Luego de discutir los diferentes roles, se agruparon de forma tal de representar los roles principales en los niveles superiores, y los sub-roles o especializaciones en los niveles inferiores, trasladados a su vez, hacia la Derecha.



Desarrollo Evolutivo

Roles de Usuario: Refinamiento de roles

El cuarto y último paso de la identificación de roles consiste en lograr su refinamiento mediante la descripción de las siguientes características:

- Frecuencia de uso del sistema por parte del usuario
- Nivel de experiencia del usuario en el dominio del problema
- El nivel general de experiencia del usuario con el manejo de tecnologías.
- El nivel general de experiencia del usuario con el sistema
- Objetivo del usuario con la utilización del sistema

Desarrollo Evolutivo

Roles de Usuario: Refinamiento de roles

Ejemplos de una Descripción Refinada de Roles:

Rol: Interesado Uso infrecuente del sistema sin conocimiento del dominio del problema. Se asumirá un nivel avanzado de experiencia en la utilización de computadoras y bajo nivel de experiencia con el uso del sistema en particular. Su interés será consultar el calendario y contenidos de los eventos. **Interesado E-mail** Ídem interesado, pero su interés es recibir información vía e-mail. **Interesado Redes Sociales** Ídem interesado, pero su interés es recibir información vía redes sociales. **Interesado en Futuros Eventos** Un tipo particular de Interesado, cuyo foco está en futuros eventos en una ciudad o país en particular.	**Comercial** Uso intensivo del sistema con gran conocimiento del dominio del problema. Posee un nivel intermedio de experiencia en la utilización de computadoras y alto nivel de experiencia con el uso del sistema en particular. Su responsabilidad será la de proveer información sobre los diferentes cursos frente a las consultas de los interesados. Esto incluye programa, contenidos, fechas, precios y cantidad de vacantes. También debe conocer el estado de completitud de cada curso en el calendario y tener la posibilidad de crear nuevos eventos. **Partner Comercial** Ídem Comercial, con la particularidad que los eventos creados por un Partner Comercial se registran como “tentativos” hasta que un Comercial los confirma.

Desarrollo Evolutivo

Visual Story Mapping en la Práctica: Identificación de los Procesos de Negocio

A continuación se realizará una identificación de procesos de negocio que el sistema deberá resolver, independientemente de los roles encontrados en el ejercicio anterior.

En este caso se identificaron los siguientes:

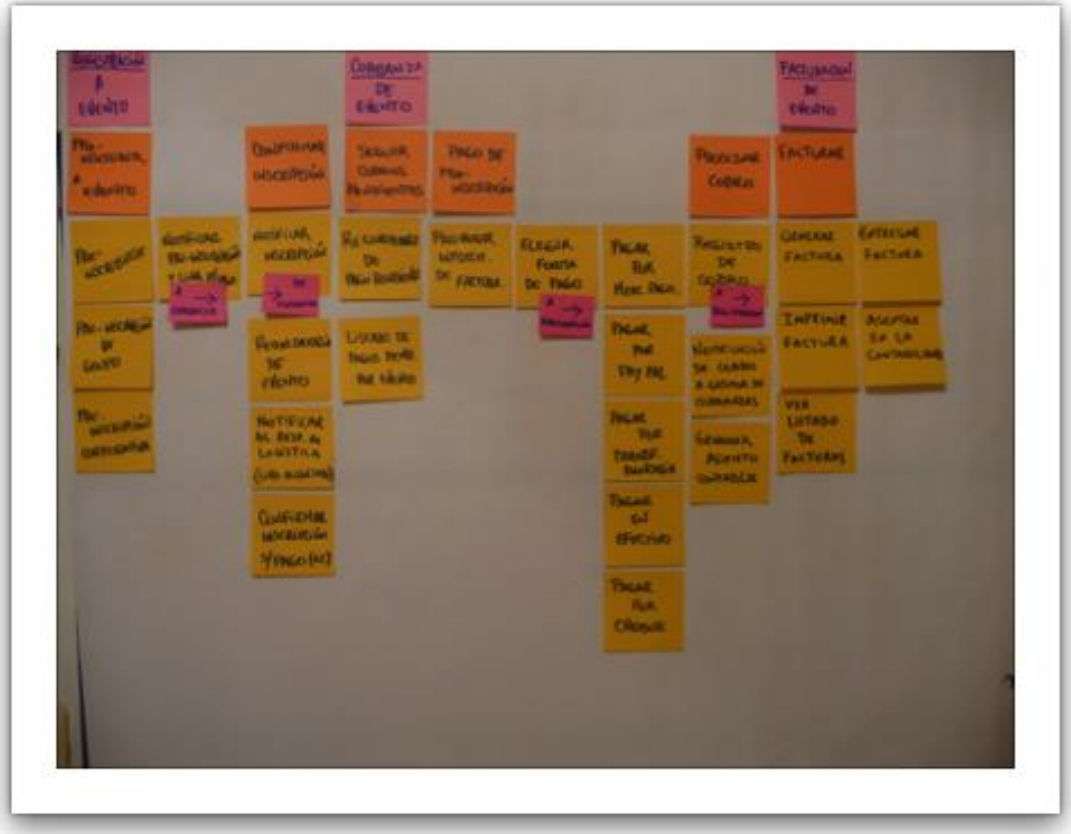
- Venta de Evento
- Registración a Evento
- Cobranza de Evento
- Facturación de Evento
- Evaluación de Evento
- Logística de Evento

Desarrollo Evolutivo

Visual Story Mapping en la Práctica: Identificación de Funcionalidades del Software (Herramientas)

El próximo paso consiste en la identificación de las funcionalidades con las que el sistema deberá contar. Esta actividad la realizamos teniendo en cuenta todos los roles identificados, efectuando sucesivas “pasadas” por todos los procesos de negocio y evaluando que cada uno de los roles involucrados en ellos cuenten con las funcionalidades requeridas para la realización de sus objetivos.

Al igual que la identificación de roles, esta actividad se realiza en forma colaborativa junto al Product Owner y la mayor cantidad de miembros del equipo posible. En las fotografías siguientes se podrán identificar los procesos de negocio en color rosa, las actividades en color naranja y las funcionalidades en color amarillo.



Desarrollo Evolutivo

Visual Story Mapping en la Práctica: Identificación de MVP y posteriores entregas

Como hemos indicado anteriormente, la construcción del sistema se realizará en forma orgánica o evolutiva, naciendo desde el MVP (producto mínimo viable) y produciendo incrementos funcionales (MMFs) potencialmente entregables en cada iteración.

Para el sistema en cuestión hemos identificado las siguientes funcionalidades por cada uno de los procesos de negocio:

Venta de Evento	Registración a Evento	Cobranza de Evento	Facturación de Evento	Evaluación de Evento	Logística de Evento
Sugerir Evento Crear Evento Difundir Evento Responder Consultas Visualizar Estado de Inscripciones	Pre-Inscripción a Evento Confirmación de Inscripción	Seguimiento de Cobros Pendientes Pagar Pre-inscripción Procesar Cobro	Facturar Evento	Responder Preguntas Corregir Evaluación Notificar Resultado Recuperar Evaluación	Gestionar Maestros de Logística Generar Checklist Actualizar Checklist Operar Checklist

De ser posible, conviene ordenar por prioridades las diferentes funcionalidades identificadas para cada uno de los procesos de negocios, colocando las más prioritarias al comienzo. De esta manera nos resultará más fácil definir el MVP y el MMF, así como asignar las funcionalidades a desarrollar en cada sprint.

Desarrollo Evolutivo

Visual Story Mapping en la Práctica: Identificación de MVP y posteriores entregas

Venta de Evento	Registración a Evento	Cobranza de Evento	Facturación de Evento	Evaluación de Evento	Logística de Evento
Sugerir Evento Crear Evento Difundir Evento Responder Consultas Visualizar Estado de Inscripciones	Pre-Inscripción a Evento Confirmación de Inscripción	Seguimiento de Cobros Pendientes Pagar Pre-inscripción Procesar Cobro	Facturar Evento	Responder Preguntas Corregir Evaluación Notificar Resultado Recuperar Evaluación	Gestionar Maestros de Logística Generar Checklist Actualizar Checklist Operar Checklist

Sprint 1: MVP – Comercializar Evento

Sprint 2: MMF – Comercialización de Evento con Facturación

Sprint 3

Sprint 4

Scrum: Historias de Usuario

- Son las especificaciones funcionales que utiliza Scrum
- Surgieron en *eXtremme Programming (XP)*
- Se basan en el siguiente principio: El método más eficiente y eficaz de transmitir información hacia y dentro de un equipo de desarrollo es mediante la comunicación cara a cara
- Las Historias de Usuario son especificaciones funcionales que invitan a la conversación para que el detalle sea consecuencia de esta última y no un remplazo

Historias de Usuario

Componentes de una Historia de Usuario

Una Historia de Usuario se compone de 3 elementos, también conocidos como “las tres Cs” de las Historias de Usuario:

Card (Ficha) – Toda historia de usuario debe poder describirse en una ficha de papel pequeña. Si una Historia de Usuario no puede describirse en ese tamaño, es una señal de que estamos traspasando las fronteras y comunicando demasiada información que debería compartirse cara a cara.

Conversación – Toda historia de usuario debe tener una conversación con el Product Owner. Una comunicación cara a cara que intercambia no solo información sino también pensamientos, opiniones y sentimientos.

Confirmación – Toda historia de usuario debe estar lo suficientemente explicada para que el equipo de desarrollo sepa qué es lo que debe construir y qué es lo que el Product Owner espera. Esto se conoce también como *Criterios de Aceptación*.

Historias de Usuario

Redacción de una Historia de Usuario:

Como [rol], Necesito [funcionalidad], Para [beneficio]

Los beneficios de este tipo de redacción son, principalmente:

- Primera Persona: La redacción en primera persona de la Historia de Usuario invita a quien la lee a ponerse en el lugar del usuario.

- Priorización: Tener esta estructura para redactar la Historia de Usuario ayuda al Product Owner a priorizar. Si el Product Backlog es un conjunto de ítems como “Permitir crear un evento tentativo”, “Confirmar un evento tentativo”, “Notificar al responsable de logística”, “Ver el estado de inscripciones”, etc. el Product Owner debe trabajar más para comprender cuál es la funcionalidad, quien se beneficia y cuál es el valor de la misma.

- Propósito: Conocer el propósito de una funcionalidad permite al equipo de desarrollo plantear alternativas que cumplan con el mismo propósito en el caso de que el costo de la funcionalidad solicitada sea alto o su construcción no sea viable.

Historias de Usuario

Características de una Historia de Usuario

Se recomienda que toda Historia de Usuario cumpla con 6 características que podemos recordar bajo la regla mnemotécnica “INVEST”:

Historias de Usuario

Características de una Historia de Usuario: Independientes (I)

- Las Historias de Usuario deben ser independientes de forma tal que no se superpongan en funcionalidades y que puedan planificarse y desarrollarse en cualquier orden.

- Muchas veces esta característica no puede cumplirse para el 100% de las Historias. El objetivo que debemos perseguir es preguntarnos y cuestionarnos en cada Historia de Usuario si hemos hecho todo lo posible para que ésta sea independiente del resto.

Historias de Usuario

Características de una Historia de Usuario: Negociable (N)

- Una buena Historia de Usuario es *Negociable*. No es un contrato explícito por el cual se debe entregar todo-o-nada. Por el contrario, el alcance de las Historias (sus criterios de aceptación) podrían ser variables: pueden incrementarse o eliminarse con el correr del desarrollo y en función del feedback del usuario y/o laperformance del Equipo.

- En el caso de que uno o varios criterios de aceptación se eliminen de una Historia de Usuario, estos se transformarán en una o varias Historias de Usuario nuevas.

- Esta es la herramienta que el Product Owner y el Equipo tienen para negociar el alcance de cada Sprint.

Historias de Usuario

Características de una Historia de Usuario: Valorable (V)

- Una Historia de Usuario debe ser Valorable por el Product Owner. Los Desarrolladores pueden tener actividades técnicas como parte del BackLog, pero para que puedan ser consideradas una Historia de Usuario, deben ser enmarcadas de forma tal que el Product Owner las considere importantes, caso contrario, no deberían formar parte del BackLog.

- En general, esta característica representa un desafío a la hora de dividir Historias de Usuario. Bill Wake propone pensar en una Historia de Usuario como si fuese una torta de múltiples capas, por ejemplo: una capa de persistencia, una capa de negocio, una capa de presentación, etc. Cuando dividamos esa Historia de Usuario, lo que vamos a estar sirviendo es una parte de esa “torta” y el objetivo debería ser darle al Product Owner la esencia de la “torta” completa, y la mejor manera de hacerlo es cortando una rodaja vertical de esta “torta” a través de todas las capas. Los Desarrolladores tenemos una inclinación especial de trabajar en una capa a la vez hasta completarla, pero una capa de persistencia de datos completa y terminada tiene muy poco o ningún valor para el Product Owner si no hay una capa de negocio y de presentación.

Historias de Usuario

Características de una Historia de Usuario: Estimable (E)

- Una Historia de Usuario debería ser estimable

- Mike Cohn, identifica tres razones principales por las cuales una Historia de Usuario no podría estimarse:

1. La Historia de Usuario es demasiado grande. En este caso la solución sería dividir la Historia de Usuario en historias más pequeñas que sean estimables.

2. Falta de conocimiento funcional. En este caso la Historia de Usuario vuelve al Product Owner para bajar en detalle la Historia o inclusive (y recomendable) tener una conversación con el Equipo de Desarrollo.

3. Falta de conocimiento técnico. Muchas veces el Equipo de Desarrollo no tiene el conocimiento técnico suficiente para realizar la estimación. En estos casos el Equipo de Desarrollo puede dividir la historia en 1) un time-box conocido como “spike” que le permita investigar la solución y proveer una estimación más certera y 2) la funcionalidad a desarrollar como parte de la Historia en si misma.

Historias de Usuario

Características de una Historia de Usuario: Pequeña (Small - S)

- Toda Historia de Usuario debe ser lo suficientemente pequeña de forma tal que permita ser estimada por el Equipo de Desarrollo. Algunos Equipos fijan el tamaño de una Historia de usuario como no más de dos semanas de una persona. Si bien no es una medida explícita, tener entre 4 y 6 Historias de Usuario por Sprint es una buena señal de tamaño.

- Las descripciones de las Historias de Usuario también deberían ser pequeñas, y escribirlas en fichas pequeñas ayuda a que eso suceda.

Historias de Usuario

Características de una Historia de Usuario: Verificable (Testable - T)

- Una buena Historia de Usuario es Verificable. Se espera que el Product Owner no solo pueda describir la funcionalidad que necesita, sino que también logre verificarla (probarla).

- Algunos Equipos acostumbran solicitar los criterios de aceptación antes de desarrollar la Historia de Usuario. Si el Product Owner no sabe cómo verificar una Historia de Usuario o no puede enumerar los criterios de aceptación, esto podría ser una señal de que la Historia en cuestión no está siendo lo suficientemente clara.

Historias de Usuario

Definición de Listo (Definition of Ready)

- Es el conjunto de características que una Historia de Usuario debe cumplir para que el Equipo de Desarrollo pueda comprometerse a su entrega, es decir, incluirla en un Sprint Backlog.

- Una típica definición de listo podría ser:

    - La Historia de Usuario debe ser INVEST
    - Todos sus pre-requisitos están resueltos (ej: dependencias con otros Equipos)

Historias de Usuario

Definición de Terminado (Definition of Done)

- Es el conjunto de características que una Historia de Usuario debe cumplir para que el equipo de desarrollo pueda determinar si ha terminado de trabajar en ella.

- Un típico criterio de “Terminado” podría ser:

    - Todos los criterios de aceptación funcionan correctamente
    - Todos los archivos fuentes están en el repositorio de código fuente y el *build* se ejecutó exitosamente
    - El Product Owner dio su visto bueno de la funcionalidad construida antes de llegar a la Review Meeting.

Historias de Usuario

Tener en cuenta que:

Dada la naturaleza evolutiva del alcance de un proyecto ágil, las Historias de Usuario de mayor prioridad estarán más detalladas que las Historias de Usuario de menor prioridad, las cuales inclusive podrían considerarse EPICS (agrupaciones de varias Historias de usuario)

Historias de Usuario

Ejemplos

Como ...	Necesito ...	Para ...	Criterios de Aceptación
Comercial	Crear un evento confirmado	Hacer el seguimiento del mismo	- Debe tener Nombre, Fecha, Descripción, Destinatarios, Programa, Instructor, Lugar, Ciudad, País, Capacidad, Precios y Promociones: SEB (Super Early Bird), EB (Early Bird), dto. en % para 2 personas y dto. en % para 3 o más personas. - Las promociones son opcionales. - Las fechas de SEB y EB deben ser anteriores a la fecha del evento - Por defecto SEB=30 días antes, EB=10 días antes, 2personas=-10%, 3+ personas=-15%. - Un evento puede ser público o privado
Comercial	Modificar evento confirmado	Corregir cualquier error o re programarlo	- Permite modificar todos los campos.
Comercial	Cancelar evento confirmado	Dejar de seguirlo	- Desaparece del listado de eventos confirmados.

Scrum: Estimaciones Ágiles

Cono de la Incertidumbre:

- Describe la evolución de la incertidumbre durante la ejecución de un proyecto.
- Al comienzo, hay poco conocimiento sobre el producto y el resultado del trabajo, por tanto las estimaciones están sujetas a una gran incertidumbre.
- A medida que avanzamos en el proyecto obtenemos mayor conocimiento sobre el entorno, la necesidad de negocio, el producto y el proyecto mismo. Esto causa que la incertidumbre tienda a reducirse progresivamente hasta desaparecer.
- Esto ocurre generalmente hacia el final del proyecto: no se alcanza una incertidumbre del 0% sino hasta haber finalizado.

Scrum: Estimaciones Ágiles

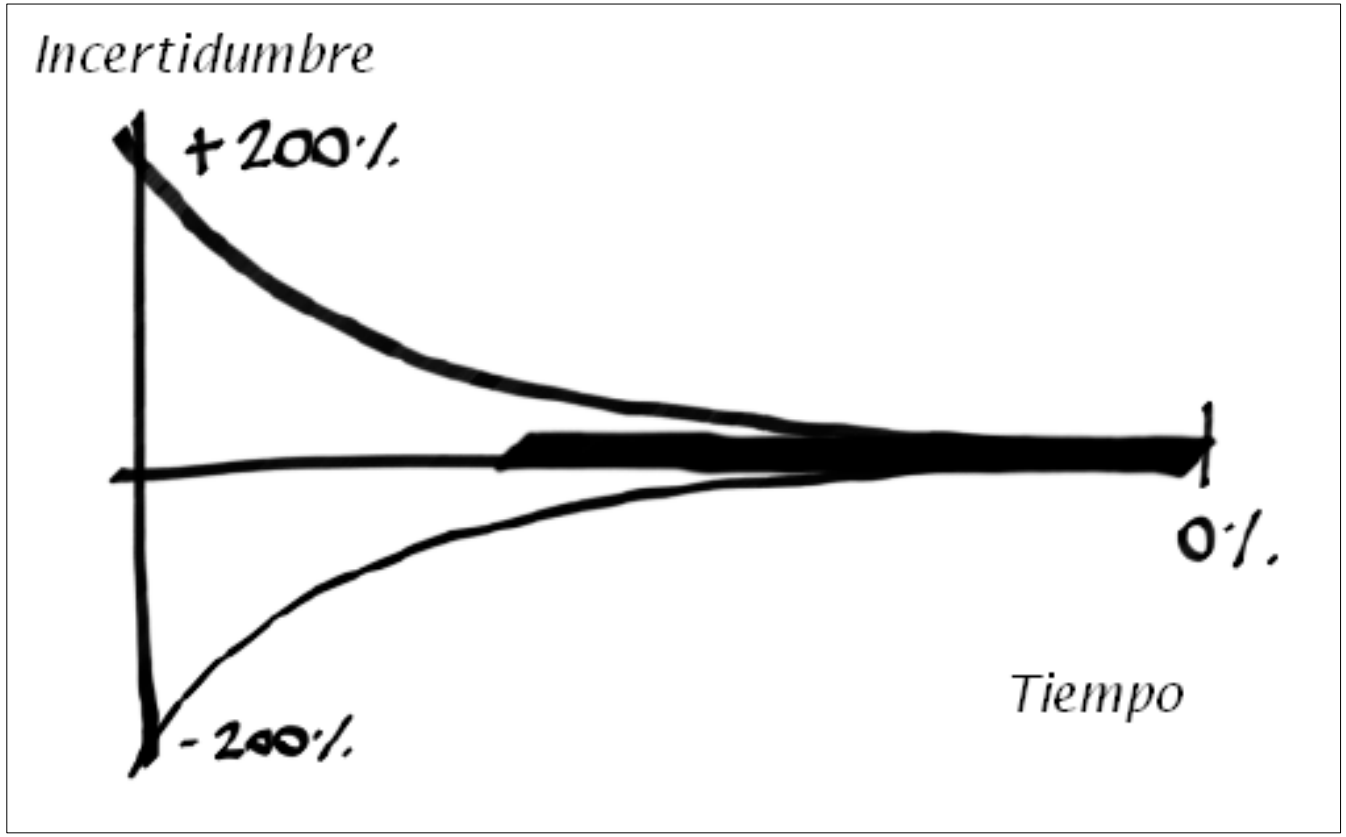
Cono de la Incertidumbre: Evolución de la incertidumbre

- Muchos ambientes cambian tan lentamente que la incertidumbre puede ser considerada constante durante la duración de un proyecto típico: Evolución Estática.

- En estos contextos la gestión tradicional de proyectos hace hincapié en lograr un entendimiento total mediante el análisis y la planificación detallada antes de comenzar a trabajar.

- De esta manera los riesgos son reducidos a un nivel en el que pueden ser gestionados cómodamente.

- En estas situaciones, el nivel de incertidumbre decrece rápidamente al comienzo y se mantiene prácticamente constante (y bajo) durante la ejecución del proyecto.



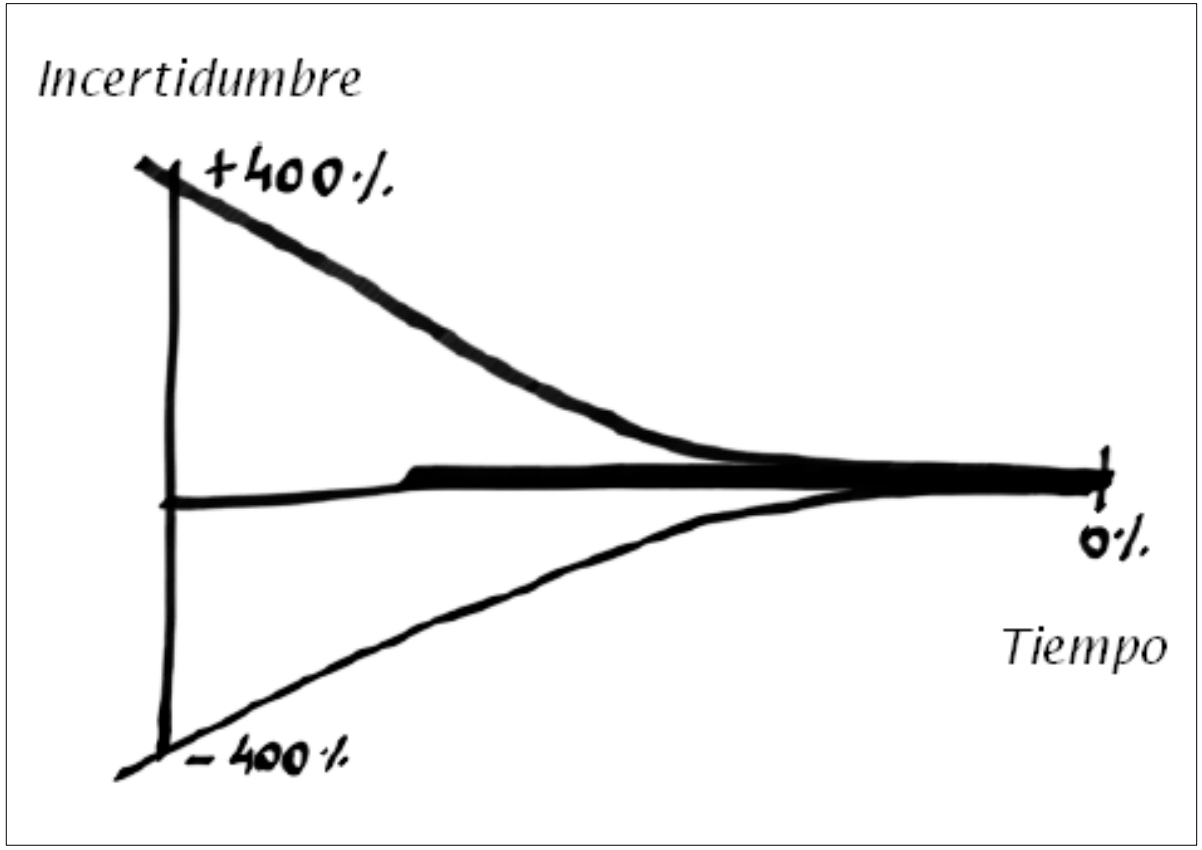
Scrum: Estimaciones Ágiles

Cono de la Incertidumbre: Evolución de la incertidumbre

El contexto del software, por el contrario, es un contexto altamente volátil donde hay muchas fuerzas externas actuando para incrementar el nivel de incertidumbre, como lo son los cambios producidos en el contexto de negocio, los cambios tecnológicos y aquellos surgidos por la mera existencia del producto construido que acontecen durante la ejecución del proyecto.

Debido a esta razón, se requiere trabajar activa y continuamente en reducir el nivel de incertidumbre.

Investigaciones han demostrado que en la industria del software, el nivel de incertidumbre al comienzo de un proyecto es del +/- 400%, esta incertidumbre tiende a decrementarse durante la evolución del proyecto, pero sin garantías de ello.



Scrum: Estimaciones Ágiles

Estimaciones en contextos inciertos

- La propia palabra “estimación” deja en claro que no calculamos un valor exacto, determinístico. De hecho, toda estimación tiene supuestos, y estos supuestos suman incertidumbres.

- Como consecuencia, las metodologías ágiles proponen comenzar a trabajar en un proyecto sin la necesidad de tener una estimación precisa basada en supuestos y siendo conscientes de que la estimación inicial es de un orden de magnitud probable, para poder ganar experiencia rápidamente y así estimar con mayor certeza prescindiendo de supuestos.

- Para mitigar el riesgo de proveer estimaciones incorrectas, en metodologías ágiles se opta por reducir la precisión de las estimaciones en función de cuánto conocimiento se tiene sobre el esfuerzo que se requiere estimar. De esta manera, los “requerimientos” y sus “estimaciones” se categorizan en diferentes niveles de precisión.

Scrum: Estimaciones Ágiles

Escalas de PBIs y Estimaciones

Podemos enumerar la siguiente escala de PBIs y estimaciones:

- **Alto Nivel**: EPIC (bloque funcional) estimada en Tamaño (XS, S, M, L, XL)
- **Nivel Medio**: Historia de Usuario (funcionalidad) estimada en Puntos de Historia (Sucesión de Fibonacci)
- **Bajo Nivel**: tareas o actividades estimadas en horas, preferiblemente menos de un día.

Scrum: Estimaciones Ágiles

Escalas de PBIs y Estimaciones

Al comenzar el proyecto, nuestro *Product Backlog* se compone de bloques funcionales que podemos estimar según sus tamaños:

- XS – Muy Pequeño
- S – Pequeño
- M – Medio
- L – Grande
- XL – Muy Grande

Esto nos permitirá tener una primera aproximación a la problemática de negocio y a las características del producto que se desea construir. Conociendo las prioridades de dichos bloques funcionales, se toman los de mayor prioridad y se descomponen en funcionalidades más específicas, logrando de esa manera PBIs de menor nivel, llamados **Historias de Usuario** o *User Stories*.

Scrum: Estimaciones Ágiles

Escalas de PBIs y Estimaciones

- A las Historias de Usuario las estimaremos utilizando la sucesión Fibonacci: 0, 1, 2, 3, 5, 8, 13, 21, 34…

- Para estimar las Historias de Usuario utilizamos una técnica comparativa llamada **Estimación Relativa**. Esto significa asignar uno de los números de la serie de Fibonacci a cada una de las Historias de Usuario.

- De esta manera, aquellas historias que tengan el número 2 requerirán aproximadamente el doble de esfuerzo que las que lleven el número 1, aquellas que lleven el número 3 requerirán aproximadamente el triple de esfuerzo de las que lleven el número 1, una vez y media el esfuerzo de las que lleven el número 2, etc.

Scrum: Estimaciones Ágiles

Escalas de PBIs y Estimaciones

Finalmente llegamos al nivel más bajo de estimación: la estimación en horas.

Solo aplica a las tareas o actividades de las Historias de Usuario que han sido seleccionadas para formar parte de un determinado Sprint.

En la reunión de planificación de dicho Sprint, estas Historias de Usuario son divididas por el Equipo en tareas o actividades y a su vez, las tareas o actividades, estimadas en horas. Lo importante es que la estimación en horas solo se realiza para un las actividades de un determinado Sprint.

Scrum: Estimaciones Ágiles

Métodos Delphi de Predicción y Estimación

- El Método Delphi es una técnica creada hacia fines de la década de los 40's para la elaboración de pronósticos y predicciones sobre el impacto de la tecnología en la Guerra Fría.

- Su objetivo es lograr un consenso basado en la discusión entre expertos. Este método se basa en la elaboración de un cuestionario que ha de ser contestado por una serie de expertos.

- Una vez recibida la información, se vuelve a realizar otro cuestionario basado en el anterior para ser contestado nuevamente.

- Al final, el responsable del estudio elaborará sus conclusiones a partir de la explotación estadística de los datos obtenidos en las iteraciones anteriores.

- El Método Delphi se basa en: El anonimato de los participantes, La repetición y retroalimentación controlada, La respuesta del grupo en forma estadística

Scrum: Estimaciones Ágiles

Métodos Delphi de Predicción y Estimación

Basado en el Método Delphi, en 1970 se elabora la variante conocida desde entonces como **Wideband Delphi**.

Se trata de una técnica basada en la obtención de consensos para la estimación de esfuerzos, llamada “wideband” porque a diferencia del conocido método Delphi, esta técnica requiere de un mayor grado de interacción y discusión entre los participantes.

Presenta los siguientes pasos para su ejecución:

- Un coordinador presenta a cada experto una especificación y un formulario de estimación.
- El coordinador convoca a una reunión de grupo en la que los expertos debaten temas de estimación.
- Los expertos llenan los formularios de forma anónima.
- El coordinador prepara y distribuye un resumen de las estimaciones.
- El coordinador convoca a una reunión de grupo, centrándose específicamente en aquellas estimaciones donde los expertos varían ampliamente.
- Los expertos completan los formularios una vez más de forma anónima, y los pasos 4 a 6 son repetidos para tantas rondas como sea necesario.

Scrum: Estimaciones Ágiles

Planning Poker

Presentado por James Greening en 2002, y popularizado en 2005 por Mike Cohn, se basa en el método Wideband Delphi para realizar la estimación de requerimientos (o User Stories) de forma colaborativa en un Equipo.

La técnica consiste en que cada integrante del Equipo posee en sus manos una baraja de cartas con los números correspondientes a la sucesión de Fibonacci y se siguen los siguientes pasos:

- El responsable del negocio presenta una historia de usuario para ser estimada.
- Todos los participantes proceden a realizar su estimación en forma secreta, sin influenciar al resto del Equipo, poniendo su carta elegida boca abajo sobre la mesa.
- Una vez que todos los integrantes han estimado, se dan vuelta las cartas y se discuten principalmente los extremos.
- Al finalizar la discusión se levantan las cartas y se vuelve a estimar, esta vez con mayor información que la que se tenía previamente.
- Las rondas siguen hasta que se logra consenso en el Equipo y luego se continúa desde el punto número uno con una nueva historia de usuario.

Scrum: Estimaciones Ágiles

La Sabiduría de las Multitudes (Wisdom of Crowds)

La tesis detrás de la Sabiduría de las Multitudes es simple: dadas las circunstancias requeridas, un grupo de personas puede tomar una decisión más acertada que la mejor de las decisiones de la mayoría (si no todos) los integrantes del grupo individualmente.

Para que esto pueda suceder, se recomiendan las siguientes condiciones:

- **Diversidad de opiniones**: cada persona debería tener información particular aún si es sólo una interpretación excéntrica de los hechos conocidos. El grupo debe tener diversidad de perfiles.

- **Independencia**: las opiniones de los participantes no deberían estar influenciadas por las opiniones de los que los rodean, con el objetivo de evitar el Pensamiento de Grupo.

- **Agregación**: El grupo debería tener la capacidad de sumar las opiniones individuales y no simplemente votar por la mejor opción.

Scrum: Estimaciones Ágiles

Conclusiones sobre estimaciones Ágiles

Muchas teorías y enfoques convergen en las siguientes características sobre estimaciones en proyectos ágiles:

- No tiene sentido presentar estimaciones certeras al comienzo de un proyecto ya que su probabilidad de ocurrencia es extremadamente baja por el alto nivel de incertidumbre.

- Intentar bajar dicha incertidumbre mediante el análisis puede llevarnos al “Análisis Parálisis”. Para evitar esto debemos estimar a alto nivel con un elevado grado de probabilidad, actuar rápidamente, aprender de nuestras acciones y refinar las estimaciones frecuentemente. Este enfoque se conoce también como “Rolling Wave Planning” o “Elaboración Progresiva”.

- La mejor estimación es la que provee el Equipo de trabajo. Esta estimación será mucho más realista que la estimación provista por un experto ajeno al Equipo.