# El espíritu de Scrum

## *El arte de amar los lunes*

*Alan Cyment, CST*

---

*v0.2*

# Introducción

## Mi objetivo

Este libro surge de una necesidad concreta: mi principal actividad hoy en día consiste en dictar cursos introductorios a Scrum. Dado que no utilizo slides ni nada similar durante la clase, surgió la necesidad de poder entregar a los alumnos un texto que resuma la ideas transmitidas y compartidas durante la capacitación. A este curso asisten, entre otros, profesionales con años de Scrum sobre sus espaldas. Curiosamente muchos de ellos salen sorprendidos del curso: acaban de entender Scrum. Estaban haciendo otra cosa. Estaban haciendo lo mismo de siempre.

Este texto tiene un claro objetivo: describir Scrum. Tan simple, pero tan difícil. Tan útil, pero tan bastardeado. Tan desconocido fuera del software[1]. Tanto potencial desperdiciado. Y la razón es una: Scrum representa un nuevo paradigma, una nueva forma de entender el mundo del trabajo. Y cambiar nuestros modelos mentales es, digamos, al menos incómodo.

## Por qué una visión

Frida y Hans vivían su tierna adolescencia en un bucólico pueblito bávaro a comienzos del siglo XX. Se habían criado juntos, corriendo por las praderas y cazando liebres en el bosque. Hasta que un día las hormonas comenzaron a bullir de pasión. Frida comenzó a ver a Hans con un cariño distinto, novedoso. Hans comenzó a ver a Frida con un cariño distinto, novedoso. Estaban enamorados. Nervios y más nervios, hasta que un día Hans llevó a Frida a un oscuro rincón del mercado del pueblo, le robó un beso y declaró su amor eterno. Frida respondió entre lágrimas que el amor de Hans era correspondido.

Pero los amantes sabían que algo se interpondría entre ellos: sus familias. Campesinos ellos, apostaban a lograr un buen matrimonio para sus hijos. Un

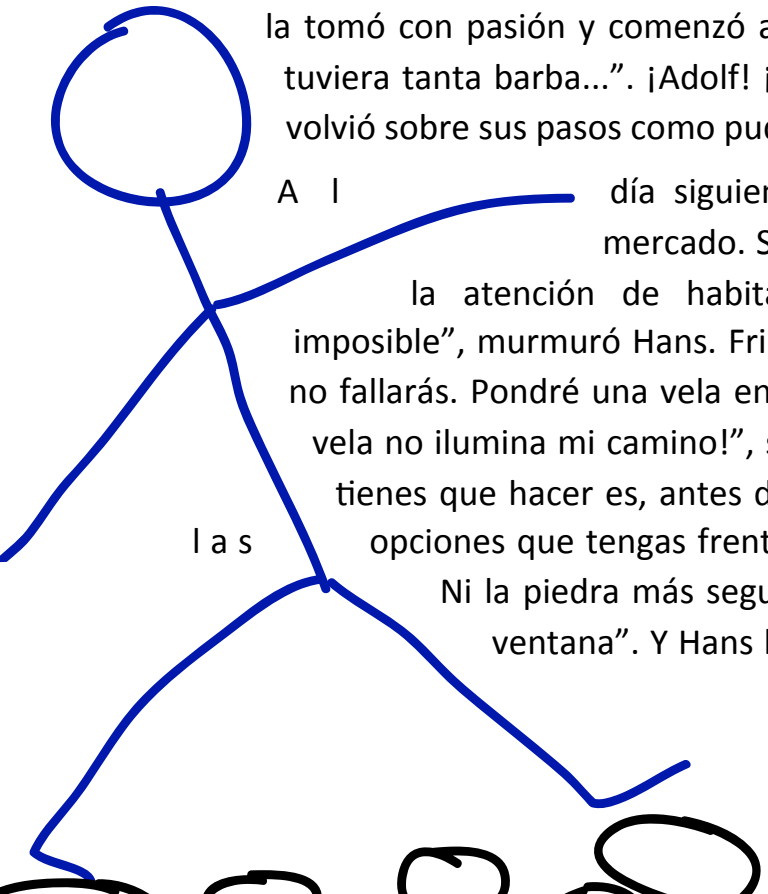
---

[1] Éste no es un libro técnico. Ojalá ni siquiera sea un libro sobre el mundo del software. Sin embargo, Scrum ha sido utilizado sobre todo en ese contexto, por lo que algunos ejemplos los daré desde esa perspectiva.

escaloncito más en la escala social al menos, pensaban sus padres. Hans lo meditó sesudamente y finalmente anunció en un alemán seco pero firme: “Nuestro amor es imposible, hermosa mía”. “Hans, cariño”, dijo Frida, “hay una posibilidad: que nuestro amor sea secreto. Vendrás todas las noches a visitarme a casa de mis padres. Hasta que un día el destino sabrá cómo seguir. Esta noche, amor mío, esta noche vendrás a verme”.

Hans, emocionado, se vistió esa noche con sus ropas más elegantes y se dirigió rumbo a la casa de los padres de Frida. De pronto lo sorprendió la angustia: entre su casa y la de su amada corría un caudaloso arroyo y él nunca, en tantos años, lo había cruzado de noche. El arroyo, transparente y con lecho de piedras, se veía amenazante a esas horas. Y lo peor de la situación: en Baviera de noche hay niebla…mucha niebla.

Hans acostumbró su vista a la oscuridad y comprobó que algo podía vislumbrar: una que otra piedra podía divisarse delante suyo. Eligió entonces su primer salto: esa roca enorme y bien firme. Imposible trastabillar. Avanzó y pudo ver sus nuevas opciones: momento de tomar un gran riesgo y saltar hasta ese grupito de guijarros de un gran envión. Lo logró y, feliz por la velocidad de su avance, siguió y siguió, hasta escuchar el arrullo del agua sobre la orilla. “Ramas que se quiebran, pasos, respiración entrecortada…¡mi amada está cerca!” pensó emocionado Hans. Vislumbró una sombra que se acercaba a pasos agigantados, la tomó con pasión y comenzó a besarla. “Curioso, no recordaba que Frida tuviera tanta barba...”. ¡Adolf! ¡El suegro! Insultos, ballestas, piedras. Hans volvió sobre sus pasos como pudo, tragando incluso más orgullo que agua.



A l día siguiente Frida lo llamó subrepticiamente en el mercado. Se cubría con una capa, no queriendo llamar la atención de habitante alguno. “Frida mía, lo nuestro es imposible”, murmuró Hans. Frida lo calmó: “Mi Hans, escucha, esta noche no fallarás. Pondré una vela en mi ventana y llegarás a destino”. “¡Pero la vela no ilumina mi camino!”, se quejó Hans. “No, amor mío, lo único que tienes que hacer es, antes de dar un salto, levantar la vista y elegir, de l a s opciones que tengas frente a ti, la que más te acerque a mi ventana. Ni la piedra más segura ni la más lejana: la que te acerque a mi ventana”. Y Hans llegó y llegó y llegó. Y Boris, nieto de Hans y

Frida y entrenador de Scrum que me relató esta anécdota allá lejos y hace tiempo, nació...unas décadas más tarde.

## La visión y Scrum

Si Hans tuviese memoria fotográfica, ¿podría haber seguido el camino recorrido la última vez cada día? No, no y no. El río corre y el contexto, por más que a simple vista el proyecto sea el mismo, cambia y con ello la debida la marcha del nuevo proyecto. Stanislavsky dijo que una gran interpretación de un actor es una rosa marchita. La siguiente vez el actor pierde tiempo y energía tratando de revivir esa rosa: lo único que puede hacer es plantar una semilla y mantener su mirada con claridad en el objetivo, mas no en las formas, que serán únicas e irrepetibles cada vez.

## La visión para este libro

Una visión es un cuadro que pinta un escenario feliz al finalizar un proyecto. ¿Cómo se ve el mundo si este proyecto llega a ser un éxito? La visión debe servirnos para poder decidir con claridad hacia dónde saltar: ¿Por qué hacer este proyecto? ¿Por qué nosotros? ¿Cuál es la medida de éxito?

Aprender Scrum es un proyecto en su mismo. La lectura de este texto lo es. Definamos entonces una visión para este libro:

"Comprender por qué Scrum es muy simple, muy difícil y funciona en el contexto adecuado. Por funcionar entendemos que ayuda a mejorar la productividad, la calidad y la felicidad de quienes toman parte de un proyecto"

¿Que pasará si se cumple esa visión? Pues será hora de tomar una decisión:

Scrum no es para mí: mejor no seguir malgastando mi tiempo aquí

Sí lo es: entiendo que siempre voy a estar transitando el arduo e interminable camino que significa utilizar Scrum

# Ese maldito engranaje

Desde hace ya décadas que venimos leyendo sobre el advenimiento de la era de la información. Hemos escuchado a decenas de gerentes decir frases rimbombantes como *"nuestro principal capital es el humano"*. Cada vez es mayor la proporción de los llamados *trabajadores del conocimiento* en la empresa moderna. Año tras año valor agregado significa producción de conocimiento, desarrollo de productos novedosos. Y sin embargo la mirada predominante del mundo laboral no ha cambiado. Dueños, jefes y empleados siguen con el mismo modelo mental de antaño: el paradigma industrial. Órdenes, control, procedimientos, cumplimiento, predictibilidad, líneas de producción, uniformidad, premios y castigos. La persona es un *recurso*, como también lo son las sillas y las maquinarias. Si se rompe o se pierde, se cambia. ¿No sabe cómo reemplazar ese recurso? Pues mire el procedimiento, siga los pasos y ya.

Esta disfunción es real y perjudica a las empresas, pero sobre todo a las personas que trabajan en ellas. Tal vez sea hora de cambiar el paradigma hacia uno que explique mejor la realidad que vivimos día a día. Todo parece indicar que la tierra es redonda, pero cuesta tanto tanto sacarnos de la cabeza esa tabla haciendo equilibrio sobre una tortuga.

## TRABAJADORES DEL CONOCIMIENTO: TU Y YO

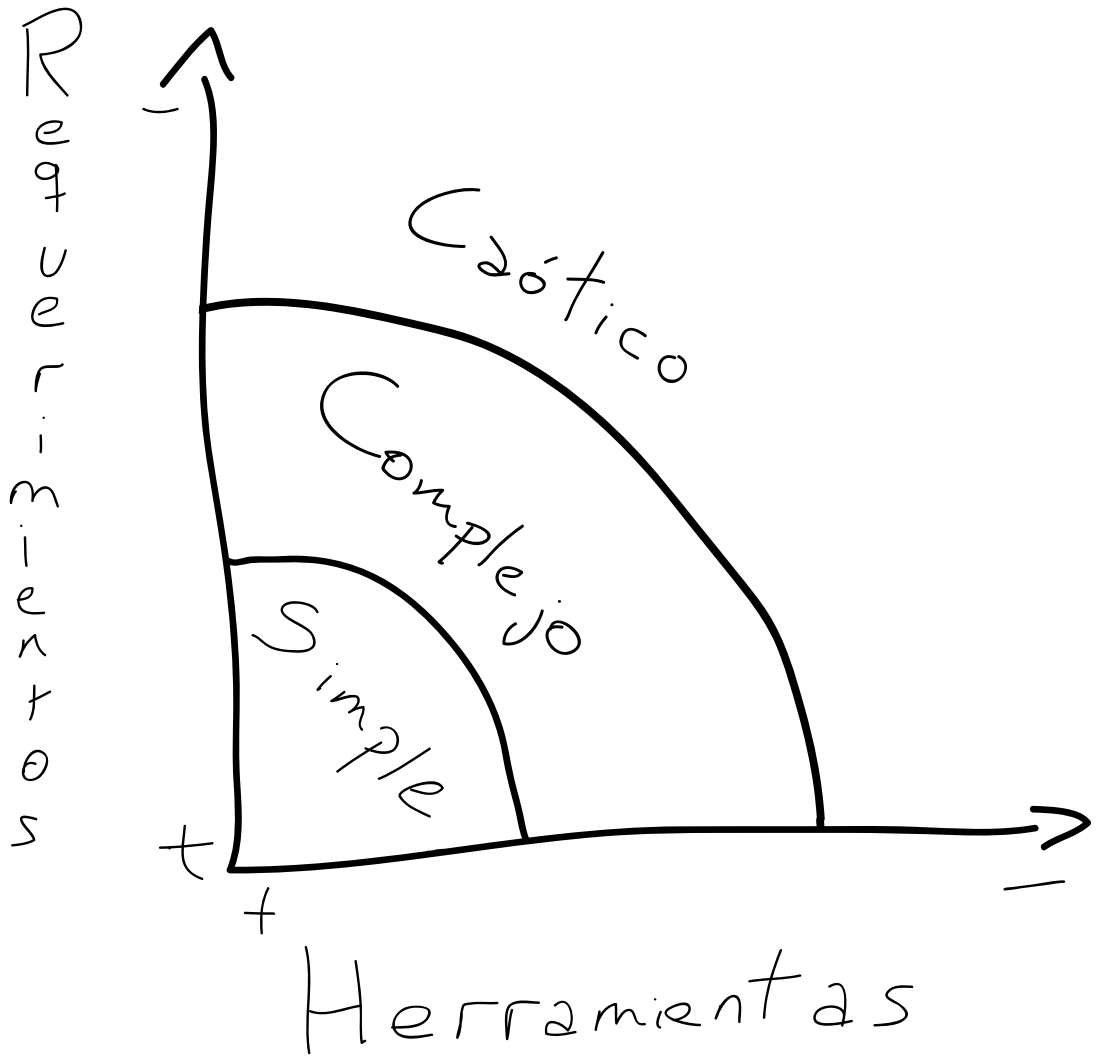
Eres un trabajador del conocimiento porque vales por lo que sabes, por aquello que puedes comprender y, por sobre todas las cosas, por tu capacidad de innovar. Eres útil porque generas información trabajando en colaboración con tus pares. Ayudas a construir lo que no existe trabajando en un equipo que está alineado con una estrategia. Eres un trabajador del conocimiento porque puedes establecer relaciones entre conceptos, comprender causas y efectos, eres capaz de crear una estrategia y balancear el pensamiento convergente con el divergente. Eres un trabajador del conocimiento porque eres desarrollador de software, gerente de proyecto, médico, abogado, arquitecto, publicista, ingeniero, científico, bibliotecario. Trabajas produciendo conocimiento y por ello tu trabajo no es ni mecánico ni predecible y tu capacidad se verá cercenada al ser

regida por completo por procedimientos rígidos definidos por un tercero. Y tal vez lo más importante: el conocimiento es tuyo y puedes ir con él a donde te plazca.

## MARTILLANDO CLAVOS

¿De qué hablamos cuando hablamos del paradigma industrial? Hablamos del Ford T. De Henry Ford anunciando con una sonrisa envidiable que sus clientes pueden elegir el color de su auto mientras sea negro. De una *línea* de producción, a lo largo de la cual distintos trabajadores cumplen de forma rutinaria un trabajo detallado en un procedimiento, redactado por su superior, quien posee más educación y/o experiencia que el obrero raso. El procedimiento indica claramente la forma de trabajar: llega la carrocería, se ubican tres clavos en cada eje y se martilla a gran velocidad. Si el obrero se equivoca, se enferma, muere o se sindicaliza será cuestión de horas, tal vez minutos, tener un reemplazante que iguala, o incluso mejora, la productividad del anterior. El conocimiento en esta función es explícito: no está en la mente del trabajador, sino que se encuentra descrito de principio a fin en el procedimiento. El obrero solamente aporta su fuerza, no su intelecto, al menos desde la visión industrial más clásica. Es así como, a medida que pasaron las décadas, los obreros fueron fácilmente reemplazados por máquinas. ¿Quién continúa siendo imprescindible? El equipo de ingenieros que diseñan y mejoran la línea.

## TODO ES RELATIVO




Para poner en perspectiva el trabajo del conocimiento y el industrial veamos este gráfico. Su objetivo es describir lo que vamos a llamar *complejidad relativa* de un proyecto. En el eje horizontal figura nuestra experiencia, nuestro conocimiento de las herramientas con las que trabajaremos en este proyecto. ¿A qué me refiero cuando hablo de herramientas? Técnicas de dibujo, lenguajes de programación, brainstorming, buenas prácticas de manejo de proyectos: lo que sea que utilizamos para desarrollar el producto asociado al proyecto. Cuanto más a la derecha nos ubiquemos en este eje, mayor será nuestro desconocimiento de la herramienta. No solamente es novedosa, sino que, por sus características, es

difícil de utilizar. Es importante resaltar que en este caso la herramienta es novedosa y difícil *para nosotros*. El proyecto es complejo en términos *relativos*.

¿Qué ocurre en el eje vertical? En este caso se plasmará la complejidad de los requerimientos *para nosotros*. Si nos ubicamos en la zona inferior estamos en una situación cómoda: tanto el cliente como nosotros entendemos qué es lo que hay que hacer de manera precisa, sin duda alguna. Si nos vamos hacia arriba en el eje aparecen las dudas, la falta de certezas, la probabilidad de que haya modificaciones en los requerimientos a mitad de proyecto.

Vamos a dividir el gráfico en tres tipos de proyectos. Tres clases de proyectos radicalmente distintos. Distintos enfoques para distintas realidades:

*Los proyectos simples:* conocemos las herramientas y los requerimientos. Podemos planificar sin temor a equivocarnos. Los resultados son previsibles aunque no hay demasiado lugar para los cambios. Bajo riesgo y, por ende, baja posibilidad de innovación. ¿Cuál es la manera óptima de trabajar en estos proyectos? La línea de producción: cada etapa de la línea se encuentra trabajando en un producto distinto, lo que maximiza la productividad. En software tenemos un equivalente más que claro: el desarrollo en cascada.

*Los proyectos caóticos:* comienza el proyecto y parece como si hubiéramos despertado en medio de un mar encabritado. No sabemos para donde ir ni cómo se nada. Puede ser que terminemos el proyecto…algún día. La investigación científica vive en el caos. Solamente hay una manera de sacar esto adelante: *prueba y error, prueba y error*.

*Los proyectos complejos:* entre la simpleza y el caos se ubica la complejidad. Aquí residen los proyectos típicos que encaramos los trabajadores del conocimiento. Riesgosos, creativos, pero con grandes posibilidades de éxito. ¿Qué hacer en este caso? Probemos con Scrum…

## POR QUÉ DEBERÍAS AMAR LOS LÍMITES

Según la mitología griega Caos era lo único que existía antes de que apareciese el mundo, el cosmos. A partir del caos todo es posible. Innovar, descubrir, inventar, crear. Queremos crear, pero debemos ser productivos. Los trabajadores del conocimiento debemos ser predecibles. Para aprovechar lo mejor de ambos

mundos vamos a crear una forma de trabajo que nos permita vivir en un *equilibrio inestable*. La propuesta: surfear el borde del caos.

Para pensar el cómo hagamos un pequeño ejercicio mental. Empecemos. Levanta la vista y sé creativo. Ya. Ya mismo. ¿Difícil, no? ¿Y qué tal si te pido que dibujes? Tal vez la mente la tengas menos en blanco ahora, pero seguramente cuesta empezar. ¿Y si te pido que dibujes con lápiz en una hoja A4? Cuesta, pero menos…¿Y si agrego que solamente puedes dibujar círculos y tienes un minuto? Ahora sí, estás listo para la acción. Tu mente no está segura del resultado final, pero tiene el impulso necesario para comenzar y ser creativa. Logramos fijar la cantidad mínima de límites a ese mar embravecido que es la abrumadora infinitud de posibilidades de un universo caótico para caer en un entorno más controlable: la complejidad.

Saber hacer Scrum va a ser similar a canalizar el rumbo del agua de mar tierra adentro: dentro del canal seguimos teniendo aguas turbulentas que nos permiten probar, innovar, equivocarnos. Pero serán las paredes del canal las que nos permitirán llevar el agua turbulenta hacia el destino al que nos interesa arribar. Scrum es el arte de balancear límites con libertad, para poder ser creativos y productivos a la vez. Como un periodista, que tiene lista esa nota de opinión polémica y elegante en dos horas, justo antes del cierre de la edición de hoy.

Pero sigamos: ahora solamente puedes dibujar tres círculos, del mismo tamaño y que éstos se toquen entre si. ¿Aburrido, no? Demasiados límites nos llevan de lo complejo a lo simple. Antes de empezar ya te imaginas el resultado. Puedes hacer el ejercicio decenas de veces y seguirá siendo bastante predecible el tipo de dibujo que obtendrás. En los proyectos *simples en términos relativos* es fácil predecir el producto y cómo hacerlo. Pero para un alma inquieta como la de un trabajador del conocimiento los proyectos simples son, simplemente, aburridos.

## SCRUM NO HACE NADA

Vamos a plantear otro tipo de complejidad: la absoluta. Para eso recurrimos a Aristóteles y un análisis muy simple que hizo de los problemas. En primer término

un problema tiene una complejidad *esencial*. Esta complejidad es inherente al problema. Es irrompible, nadie ni nada podrán jamás simplificarlo.

A ver, vamos con una anécdota para tratar de tomar el concepto por las astas. En la Patagonia sur, al sur del sur de la Argentina, que queda muy al sur, hay una serie de pequeñas cadenas montañosas hechas de granito entre medio de los Andes. El granito, como todos sabemos, es *muy* duro. Entre estas montañas de granito se encuentra el Cerro Torre. Muy poco vistoso pero bastante vertical, es considerado por muchos como uno de los mayores desafíos que puede llegar a enfrentar un alpinista. El Cerro Torre no solamente está compuesto de granito, sino que además es prácticamente vertical y se ve azotado de forma casi permanente por fuertes vientos.  En 1959 un italiano llamado Maestri subió con un colega austríaco pero bajó solo: el pobre amigo teutón había fallecido en el camino. Maestri contó al mundo, champaña en mano, que había llegado a la cima. Pero, oh fortuna injusta eres, el austríaco llevaba la cámara de fotos y una avalancha se los había llevado a ambos al averno. Dudas, suspicacias, años de orgullo herido y Maestri, harto de tanto comentario cizañero, decidió hacia 1970 mostrarle al mundo que quien escala el Cerro Torre una vez, lo hace dos. Esta vez no iban a quedar dudas. Ni una sola.

Maestri emprendió camino junto a una verdadera troupe que pudiera no solo comprobar sus hazañas sino sobre todo llevar el *taladro neumático*, con su correspondiente compresor, con el que terminó agujereando la temible pared de granito que lleva a la cima. ¿Cómo logró Maestri llegar a la cima? No llegó en helicóptero, pero casi. Cambió el juego: dejó de hacer alpinismo y se puso el casco de ingeniero civil. O, puesto en términos aristotélicos, disminuyó la complejidad esencial de un problema…¡cambiando el problema! Tentación en la cual caemos muchos trabajadores del conocimiento cuando nos damos cuenta de que somos incapaces de resolver un problema que se nos plantea. La omnipotencia intelectual nos traiciona de a ratos y nos vuelve un poco Maestri.

¿La lección? Scrum podrá ser maravilloso o podrá no serlo, pero seguro no resuelve el problema en si. ¿Tenemos que estimar el costo de un proyecto de dos años de duración del que participarán quince personas que no se conocen entre si? Eso *es muy complejo* y no hay Scrum que lo resuelva. La pregunta es entonces: ¿Para qué sirve Scrum entonces? Sigamos con Aristóteles, que siempre suena

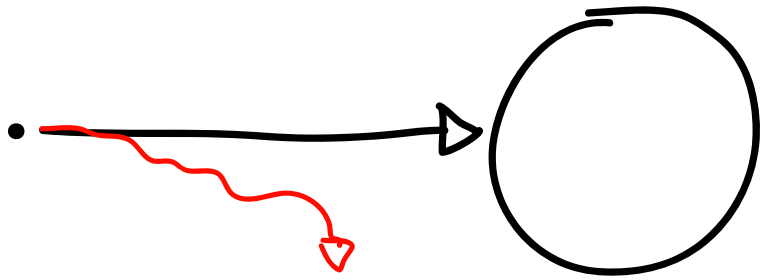
elegante citarlo. Como se ve en el dibujo, siempre que nos enfrentemos a un problema deberemos lidiar no solamente con la complejidad esencial, sino también con la complejidad *accidental*. Esa complejidad no es propia del problema, sino que viene de regalo. En la filosofía de gestión Lean, de la cual no importa si has escuchado antes o no, se habla del *desperdicio* o esfuerzo que no aporta para resolver el problema en si mismo. Lean significa “magro”: con poco o nada de grasa. Pensemos entonces en la complejidad esencial como la carne y los huesos y en la accidental como la grasa innecesaria. Hay grasa cuando no confiamos en nuestro jefe ni él en nosotros. Hay grasa si la silla es incómoda y el baño de la oficina huele mal. El proyecto se hace innecesariamente más complejo.


Para esto sirve Scrum: para detectar la capa de grasa más voluminosa y enfrentarnos cara a cara con ella. Scrum es una pequeña maquinita que hace que emerjan las

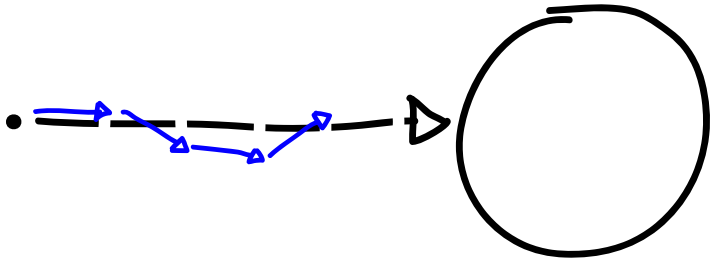
disfuncionalidades organizacionales. ¿Cómo se resuelven estas disfuncionalidades? Hay tantas respuestas como situaciones. Scrum no se inmiscuye: es solamente el mensajero de la malas nuevas. Matarlo o escucharlo es decisión de quien lo utiliza.

## SIMPLEMENTE PLANTA UNA SEMILLA

¿Y cómo funciona la maquinita? Presentemos primero un concepto: el *desarrollo iterativo incremental*. Iterar significa literalmente repetir un mismo



proceso. Entendámoslo comenzando por su opuesto, el *desarrollo lineal*. El desarrollo lineal consiste en armar sesudamente un plan para luego ajustarse a él e ir revisando periódicamente su cumplimiento. Esto se denomina usualmente *project tracking*. Durante el transcurso del proyecto la idea no es cambiar el plan, sino meramente saber si venimos cumpliéndolo en tiempo y forma. Este enfoque inflexible no suena muy conveniente para un proyecto complejo, pero sin embargo es el más utilizado a la hora de firmar un contrato. Armamos un plan y firmamos con nuestra sangre que lo cumpliremos. Luego solo resta ir comprobando que todo va sobre rieles. Muchas veces esto se convierte en una mera *ilusión de control*.

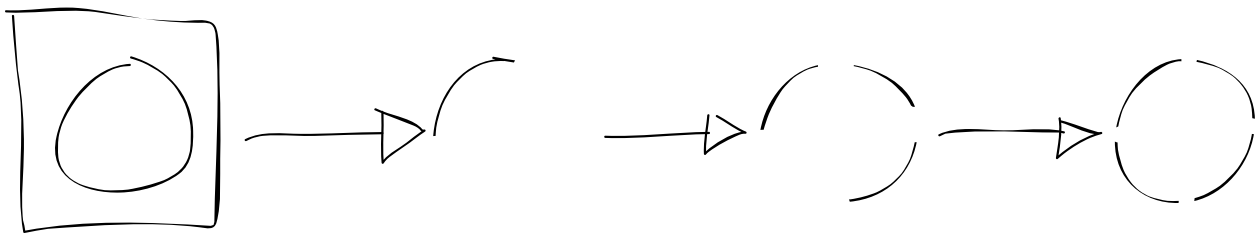


La alternativa que proponemos es el desarrollo *iterativo*. Éste consiste en apenas esbozar un plan a mediano plazo, para luego saltar de piedra en piedra,

reajustando el próximo salto en función de la información que obtuvimos luego del último. Esta alternativa promete menos exactitud el día del comienzo del proyecto, pero sin dudas tiene muchísimas más chances de llegar a destino a la hora de cruzar un arroyo caudaloso en una neblinosa noche bávara. En este caso hablaremos de llevar el proyecto haciendo *project steering*. Es decir que, de antemano, nos estamos dando espacio para poder maniobrar a medida que pasa el tiempo y avanza nuestro trabajo. El principio será entonces confiar en el terreno antes que en el mapa.

Espero que estén de acuerdo conmigo: para un proyecto complejo la mejor elección es el desarrollo iterativo. Ya tenemos la base sobre la que nos pararemos para pensar en cómo podemos desarrollar el producto. Vamos a presentar ahora dos maneras de hacer las cosas. La primera la llamaremos desarrollo *modular*.

Supongamos que tenemos un cliente que nos pide que le construyamos una

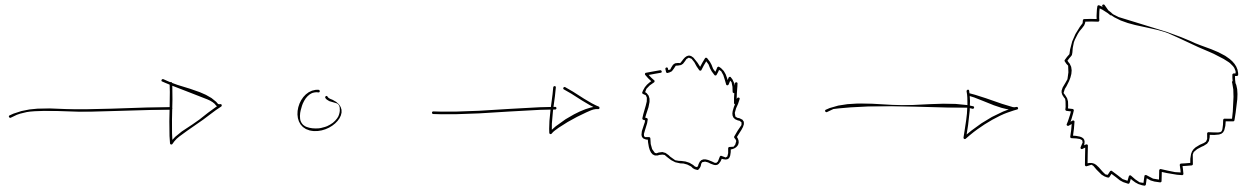


pelota grande. Digamos de unos 60cm de diámetro. Si vamos a construir esta pelota de manera modular, querremos que como producto de nuestro primer salto ya haya un diseño detallado de cómo será nuestra pelota. Sus materiales, su comportamiento ante distintas superficies, cómo vamos a probar que funciona. Luego vendrá el segundo salto, en el que construiremos un arco, claro está. En el siguiente salto un segundo y luego un tercero. Pero, este es un proyecto complejo, hemos estimado mal en nuestra planificación inicial, y se nos acabó tanto dinero como tiempo. No nos queda otra alternativa que recibir al cliente y llevarle el producto que tenemos: tres arcos. El pobre hombre nos mira y nos pregunta qué demonios es eso que traemos entre manos. Si nuestro cliente esperaba obtener 100 (pesos, duraznos, sonrisas, da igual) acaba de obtener un rotundo 0.

La alternativa al desarrollo modular la vamos a denominar *incremental* o, mejor aún, *orgánica*. ¿Cómo se construiría la pelota en cuestión de manera incremental? Antes de comenzar realizamos un rápido bosquejo del diseño del producto que vamos a construir. Al igual que los planes iterativos, los diseños incrementales comienzan como una idea vaga, falta de detalles. Lo mínimo indispensable para marcar el rumbo y comenzar a saltar hasta la próxima piedra.

Al concluir la primer iteración obtenemos no un arco, sino una pelota chiquitita. El dibujo de la pelotita tiene la misma cantidad de tinta que el primer arco construido de la manera modular. ¿La diferencia? Al final de esta primer iteración podemos llamar al cliente, darle la pelotita y pedirle que la use. El hombre la toma, la lanza cual bola de boliche por su jardín y, dado que en la casa de este señor hay un gran problema de filtraciones y el pasto está constantemente embarrado, la pelotita avanza unos centímetros y se frena. El cliente nos mira con ceño fruncido y nosotros lo tranquilizamos diciéndole que ambos acabamos de aprender algo: no sirve que la pelota tenga una superficie tan lisa. "Yo no vine a aprender nada acá. Pagué y quiero una pelota que ruede por mi jardín". Lógico, el cliente quiere valor. Le explicamos que acabamos de ahorrarnos dinero, dado que la próxima versión de la pelota será *rugosa*.

Luego de la segunda iteración obtenemos una pelota que ya parece más bien una piedra. ¿De qué tamaño es la nueva pelota? Si queremos hacer desarrollo incremental, pues deberá ser *más grande* que la anterior. No estaríamos practicándolo si hubiésemos desechado la primer pelota y hubiéramos



comenzado de nuevo. Debemos construir nuestro producto de forma tal que pueda *crecer*. Pues entregamos la versión rugosa al cliente, quien la arroja por su jardín. Rueda que te rueda, la nueva pelota llega hasta la casa del vecino. Nos miramos a los ojos con el cliente y decimos al unísono: "demasiado". El cliente quiere que este objeto (que cada vez se parece menos a una pelota) ruede unas cinco o seis veces, no más. Es caro, le explicamos, y ya casi se nos acaba tiempo y dinero, pero podemos ofrecerle un producto que ruede una sola vez. Asiente con

la cabeza y nos lanzamos a trabajar en la última iteración. Se nos acabó el presupuesto, pero el cliente tiene ya una pelota que rueda exactamente una vez. De los 100 que había pensado obtener de este proyecto tiene en mano un producto que vale, digamos, 70. No es el ideal, pero es bastante más que el 0 que habríamos obtenido si hubiéramos cumplido con el plan, construyendo una pelota perfectamente esférica, que no habría hecho más que enterrarse en el barro.

La diferencia en el modo de construcción nos permitió obtener *feedback temprano*.¿Por qué no podríamos haberlo obtenido construyendo de forma modular? Pues simplemente porque una serie de arcos no puede ser utilizada por el cliente. ¿Qué ganamos con el feedback? Entendemos, cliente y desarrollador, qué es lo que se necesita. En un proyecto complejo el producto a construir es complejo, por lo que es imposible definirlo con precisión a priori, intelectualmente, sin, literalmente, embarrarnos en el proceso de aprendizaje. Pero no solamente entendimos mejor *qué* se necesitaba, sino que además tuvimos un producto y un plan *maleables*, lo que nos permitió adaptarnos a los cambios que implicó el entender mejor de qué se trata el producto. ¡Y todavía hay más! Construir incrementalmente nos permitió entregar *valor* aún habiendo estimado de manera inexacta en un comienzo duración y costo del proyecto.

Vamos con un ejemplo más: esta vez nuestro cliente nos pide que construyamos en su jardín un árbol que le dé sombra a 30 personas. Decidimos construirlo de manera orgánica. Hacemos un pequeño pozo en un rincón, plantamos una semilla y regamos una vez al día. Luego de nuestra primera iteración tenemos con nosotros…¡un árbol! Solemos llamar a esto un brote, pero en términos del desarrollo iterativo incremental ya tenemos un producto que entrega valor. ¿Por qué? Porque cumple con la definición de un árbol: tiene un diminuto hilo que toma nutrientes y agua del suelo (que vamos a llamar “raíz”), se sostiene apenas con un palito (que vamos a denominar “tronco”), que es a su vez verde, dado que tiene clorofila, y que por ende va a realizar fotosíntesis (a esto llamaremos “hojas”). Y seguimos iterando y nuestro arbolito crece y crece. Hasta que un día el vecino decide construir una casa al lado de nuestro árbol. La naturaleza es sabia y el árbol, previa poda, continúa su crecimiento hacia el lado

opuesto a la pared. Y tanta mala suerte tenemos que el alcalde decide construir una ruta al otro lado de nuestro árbol. Previa poda, nuestro árbol seguirá creciendo, pero a partir de la altura del camión más grande que suela pasar por la ruta. Se nos acaba el tiempo y dinero presupuestados. El árbol da sombra a unas 18 personas. El cliente frunce el ceño. Esperaba 30, obtuvo 18. Fue lo mejor que se pudo.

Tal vez si hubiésemos construido el árbol de manera modular nos habría ido mejor. Probemos: invertimos una semana diseñando por completo el producto, de forma tal que resulte económico tener un árbol que dé sombra a 30 personas. Una vez obtenido el plano alquilamos una excavadora. Según nuestros cálculos, necesitamos raíces que alcancen los 10 metros de profundidad. Mientras la topadora avanza nosotros comenzamos a amasar una serie de robustas raíces, que soporten todo el peso de un árbol del porte que necesitamos. Comenzamos a cubrir con tierra nuestras primeras grandes raíces cuando sentimos que alguien nos toca el hombro: es el vecino, para contarnos que va a construir una casa al lado de nuestro árbol. ¡Llamen a la topadora de nuevo! Y de vuelta al tablero de diseño. Y a serruchar las raíces que con tanto amor y expectativas habíamos tallado. Luego de semanas de trabajo estamos listo para comenzar con el tronco. Pero de repente…alguien nos toca el hombro: es el alcalde, para avisarnos que va a construir una ruta al lado del árbol. El tablero de diseño, la topadora y nuestros nervios vuelven a pura furia. Perdemos noción del tiempo y tallamos las nuevas raíces con dedicación, pasión y bastantes nervios. Hasta que nos vuelven a tocar el hombro: es el cliente. Se terminó el tiempo presupuestado y le preocupa un poco ver solamente un poco de tierra revuelta. Lo miramos con desconcierto y pedimos un poco más de tiempo, plano en mano. Valor entregado: ni un poco de sombra. Al construir modularmente no logramos *adaptarnos a los cambios en el contexto*.

Resumiendo, el desarrollo orgánico permite:

•Entender mejor, mediante el feedback obtenido del cliente, qué producto es necesario

•Adaptar plan y producto a los cambios producidos por este mejor entendimiento

•Adaptar plan y producto a cambios en el contexto

•Entregar valor aún habiendo estimado la duración de forma imprecisa al comienzo del mismo

Nada mal para un proyecto que surfea el borde del caos…

# Las reglas del juego

## MI PROCESO, TU PROCESO, CUÁL PROCESO

Definamos una palabra, al menos a lo largo de este texto: *proceso*. Proceso será para nosotros "la forma en la que trabajamos". El proceso puede ser caótico, empírico, definido. Es parte del proceso que sepamos trabajar en equipo, que haya goteras en la oficina, que no le dirija la palabra al gerente de recursos humanos, que al final de cada día tenga que reportar hora por hora en qué trabajé.

*Metodología* será un proceso definido, en el que existen respuestas para las preguntas ¿*cómo?*, ¿*cuándo?*, ¿*quién?*, ¿*se puede?*. Scrum *no es una metodología*. Scrum *ni siquiera es un proceso*. A lo sumo podríamos definirlo como un metaproceso: una maquinita que nos ayuda a *construir iterativa e incrementalmente nuestro propio proceso*. ¿Por qué? Porque al decidir utilizar Scrum estamos tomando una decisión, una postura frente al proyecto que estamos por llevar a cabo: *definir el proceso ideal para un proyecto que tiene como objetivo construir un producto complejo es complejo en si mismo*. Por ende vamos a repetir el enfoque: el proceso crecerá y se adaptará a nuestro aprendizaje y a los cambios de contexto orgánicamente.

Scrum es por ende un framework o marco de trabajo, que será el andamiaje que nos va a ayudar a encontrar, iteración a iteración, el mejor proceso posible dada nuestra realidad y nuestros potenciales.

## TAN SIMPLE COMO UN PASO A LA VEZ

La base de Scrum será el desarrollo iterativo e incremental de producto y proceso. En concreto se plantea una dinámica de pequeños saltos. Cada salto va a consistir en

•Planificar hacia dónde saltar (teniendo en cuenta la visión)

•Ejecutar el salto (¡Saltar, qué tanto!)

•Inspeccionar tanto el avance producido por el salto como la manera de saltar (producto y proceso respectivamente)

•Adaptar la dirección del salto (producto) y la manera de saltar (proceso), para acercarnos más y mejor al objetivo final

Saltaré y saltaré hasta que ocurra alguno de los siguientes eventos

•¡Llegué a destino! El proyecto fue un éxito: alcancé la visión.

•Se nos acabaron tiempo y/o dinero: llegamos hasta donde pudimos llevando al límite nuestras posibilidades

•Decidimos que no vale la pena seguir saltando. Cancelamos el proyecto. Tan horrible y decepcionante como suena, nunca hay que olvidar el dicho turco que dice "no importa cuánto hayas caminado, si es la ruta incorrecta entonces vuelve". Aunque a nuestro orgullo le duela en el alma, a veces la decisión más sabia simplemente es volver.

## QUIÉN

Existen tres roles en Scrum, que fueron elegidos para dividir de manera clara y simple *perspectivas y responsabilidades* entre ellos. Para comprender mejor esta división vamos a tomar una vieja definición de táctica y estrategia: *táctica* es la mejor manera que encontramos de ganar una batalla y *estrategia* es la mejor elección de batallas que decidimos en pos de ganar la guerra. Es decir, en términos de un proyecto, la estrategia estará dada por *qué características* tendrá el producto y la táctica por *cómo se desarrollarán* dichas características.

### *Equipo de desarrollo*

Y ahora sí, a por el primer rol: el equipo de desarrollo (o muchas veces "equipo" a secas). El equipo de desarrollo es un grupo, usualmente entre 5 y 9

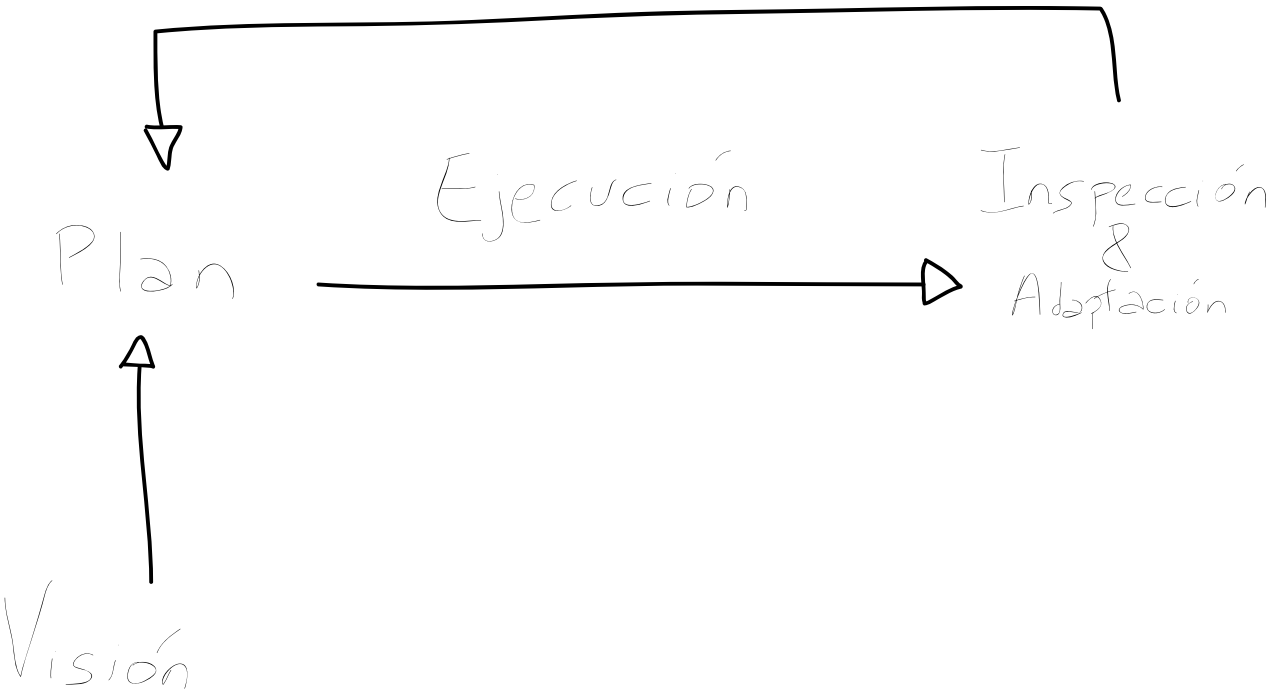
personas, que *poseen todos los skills necesarios para construir un producto que cumpla con la visión*. Por ejemplo, para un desarrollo tradicional de software, necesitaremos miembros de equipo que tengan conocimientos sobre programación, testing e incluso sobre diseño gráfico y usabilidad. Se deduce de esta definición que el equipo debe ser *multidisciplinario*.

Es importante la distinción entre *un miembro de equipo que sepa sobre testing* con *un tester*. Definir roles dentro del equipo suele minar el espíritu de compromiso: si hay un problema en la definición de las pruebas de aceptación la responsabilidad de corregirlas es del equipo, no del tester.

El equipo tiene claramente la responsabilidad y la perspectiva de la *táctica/el cómo* de la construcción del *producto*.

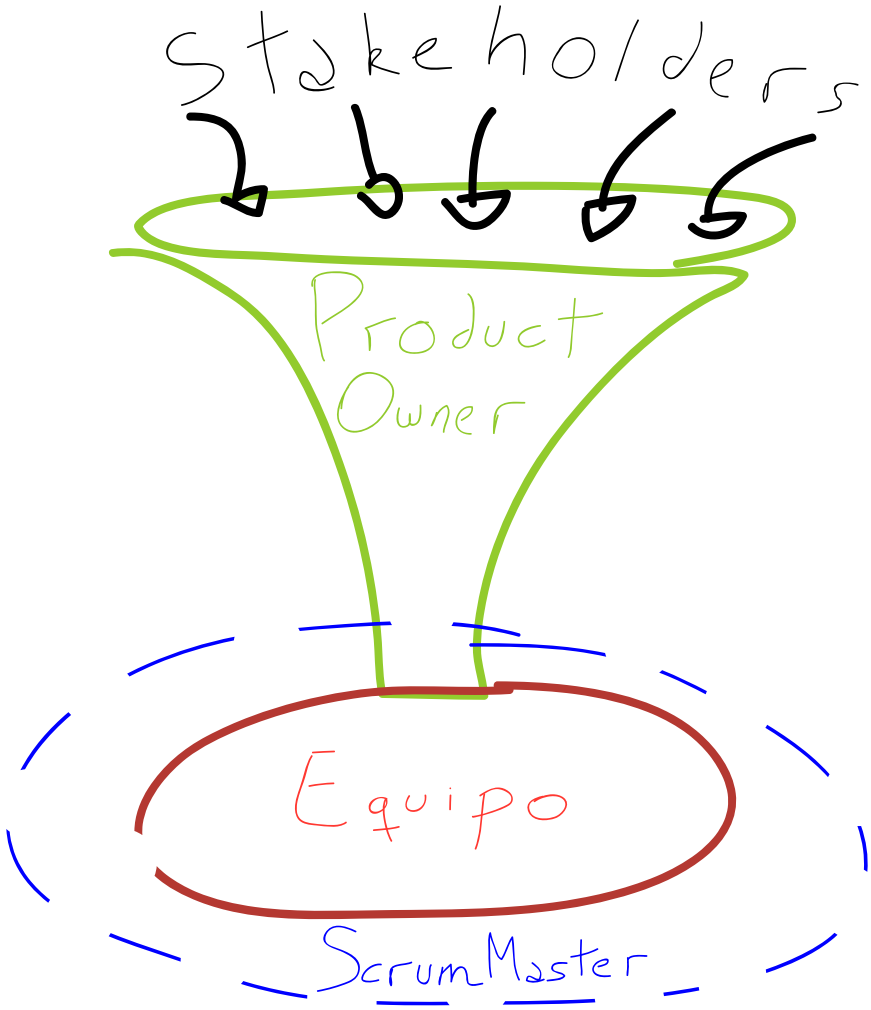
## *Product Owner*

El Product Owner es un individuo que posee como responsabilidad *maximizar el retorno de inversión del proyecto*. Es decir que, desde una perspectiva




*estratégica*, debe indicar el *qué* del *producto*, enarbolando cual faro la visión del proyecto.

¿Por qué una sola persona y no un equipo? Porque la experiencia nos muestra que en los proyectos en los que participan los trabajadores del conocimiento la principal causa de fracaso suele ser la mala comunicación a la hora de transmitir los requerimientos al equipo. Por eso nombramos a uno y sólo un *embajador* de los *stakeholders*, que son todos aquellos que tienen intereses en el proyecto y la potestad para imponer esos intereses. Cualquier proyecto tiene variopintos stakeholders, con intereses y perspectivas contrapuestas. El Product Owner es responsable de representarlos de la mejor manera posible ante el equipo, evitando de este modo potenciales ambigüedades a la hora de definir las características del producto.




*Scrum Master*

Product Owner y equipo de desarrollo tienen sus ojos y su energía concentrados en una sola cosa: el producto. Y dado que no hemos dicho nada sobre cómo deben trabajar podemos afirmar que conforman un gran equipo *auto-gestionado*. Dado que ellos son quienes están día a día enfrentándose cara a cara con la realidad del proyecto, quién mejor para decidir el *proceso* que ellos mismos. Si aceptamos que el proceso que necesita un equipo para desarrollar de manera óptima un producto complejo es complejo en si mismo, no podemos quedarnos en la noción industrial (¿Se acuerdan del viejo paradigma?) en la que un supervisor *decide* desde fuera cómo debe trabajar el empleado. Eso está muy bien para martillar clavos, pero en el caso de un proyecto complejo, en el que no nos queda otra alternativa que apelar al lado más humano de los trabajadores, perderíamos potencial creativo y, sobre todo, compromiso con el resultado final si impusiésemos proceso.

Pero la auto-gestión no es ninguna panacea per se. Es simplemente un llamado a gritos al caos. Y ya vimos lo interesante, pero también lo riesgoso que puede ser el caos. Necesitamos límites, pero queremos que esos límites sean auto-impuestos. Traigamos a escena a una tercer figura, que actúe de espejo tanto para el equipo de desarrollo como para el Product Owner, para recordarles los límites que nos impone Scrum, a fin ayudarlos de canalizar el caos y llevar a buen puerto un proyecto que necesita de altas dosis de creatividad. Traigamos a un ScrumMaster.

El ScrumMaster, además de tener un nombre grandilocuente, es una figura poco tradicional en las organizaciones. El ScrumMaster es ni más ni menos que un espejo: es *responsable, desde una perspectiva de proceso, de que Product Owner y equipo de desarrollo optimicen y utilicen una manera de trabajar cada día mejor*. Para ello un ScrumMaster suele llevar a cabo tres labores básicas:

•Facilitador: alguien que ayuda a un grupo de personas a tomar una decisión no trivial desde un punto de vista neutral

•Coach: un evocador de excelencia - soplador de brasas, que supieron ser fuego y necesitan ser avivadas

•Mentor: maestro que instruye desde una posición de igualdad y procura que el mentoreado siga su propio camino lo antes posible

El ScrumMaster suele llevar una sola herramienta en su bolso: la pregunta. Puede ser insidiosa, básica, generadora de silencios incómodos, cándida, retórica. Las hay de todo tipo y color y, a diferencia de las respuestas, son ellas las que mueven al mundo.

## QUÉ

Vimos el predicado, luego el sujeto: le llegó el turno al objeto, a lo que en software se denominan artefactos. ¿Documentos? No necesariamente, sino meras herramientas de trabajo que son utilizadas constantemente por los tres roles durante un proyecto que utilice el framework. Cada una de ellas tendrá un *responsable*.

### *Visión*

La visión, como vimos anteriormente, es esa vela ubicada en la ventana. Esa descripción somera del norte que debe llevar el proyecto. El *responsable* de que la visión sea lo más representativa posible de las necesidades y potestades de los stakeholders es obviamente el Product Owner. El *responsable* de que el equipo comprenda la visión y desarrolle el mejor producto posible en términos de la misma es el Product Owner. Esto *no* significa que sea él mismo quien la redacta. Solamente es *responsable* de que eso suceda. ¿Quién la escribe entonces? Depende. Es decir, el framework deja, adrede, esa pregunta, como tantas otras, sin responder. ¿Por qué? Porque ofrecer una respuesta universal en este caso sería cercenar las posibilidades que nos brinda dejar este aspecto sin límite alguno. Scrum es, recordemos, equilibrio inestable. O lo que es igual, el balance entre límites y libertad.

### *Product Backlog*

La palabra "backlog" podría traducirse como "lista de lo que me falta para…". El Product Backlog será entonces la lista de características de las que carece hoy en día el producto. En él se ve plasmada la *estrategia* del proyecto. El responsable de tener el backlog que marque el mejor recorrido posible hacia la visión será obviamente el Product Owner. Por eso decimos que es el *responsable* de este artefacto. Nuevamente, esto *no* significa que es el Product Owner quien construirá el Product Backlog. Esto, nuevamente, dependerá del contexto. Cada vez que mencionemos a los stakeholders, usualmente querrá decir que es el

Product Owner quien dialoga cara a cara con el equipo, dado que su principal atributo es el poder conjugar las distintas necesidades de todos los interesados en el proyecto en un único mensaje.

El Product Backlog es simplemente una lista de lo que llamaremos, aunque sea confuso para nosotros los hispanohablantes, PBIs: Product Backlog Items. El framework no prescribe cómo se materializan los PBIs. Alguno será lo que en software llamamos Caso de Uso, algún otro en la lista será una Historia de Usuario y unos cuantos tal vez sean meras servilletas garabateadas. El framework solamente describe dos características de los PBIs:

•Cada vez que el equipo desarrolle un PBI el producto habrá aumentado el valor percibido por los stakeholders (crecieron raíces, tronco y rama hasta darle sombra a una persona más)

•Los PBIs se encuentran ordenados según el criterio que decida el Product Owner. Usualmente se utiliza como criterio para esto la secuencia que maximice la relación costo/beneficie, siempre que la misma respete dependencias y considere riesgos, tanto técnicos como funcionales.

Y ahora la pregunta de rigor: ¿Para qué pide esto el framework?

El primer punto es la manera de poner en práctica el desarrollo o crecimiento orgánico del producto. Cada salto entre piedra y piedra nos acerca de manera consolidada (o no nos acerca, en caso que el equipo haya construido un PBI de manera incorrecta) a la visión. Su aceptación por parte del Product Owner será binaria: brinda valor o no lo brinda. *No* es un PBI un arco de la pelota o solamente una rama. Que el PBI entregue valor a los stakeholders nos asegura que se ha crecido de manera tal que nada a quedado a medias: si hemos estimado mal el proyecto quedaremos bien parados (tendremos producto para entregar - le habremos dado sombra a un número de personas) y, más importante aún, sólo así recibiremos feedback útil de los stakeholders, dado que ellos tienen la perspectiva de *negocio*, de *valor*. Un PBI incompleto es aquel batalla que aún no se ha decidido y por lo tanto sería contraproducente tomar decisiones estratégicas basadas en puras especulaciones.

El segundo punto (que los PBIs tengan un orden, una prioridad) es simplemente la manera de hacer que el crecimiento orgánico juegue a nuestro

favor. En el viejo paradigma, el Ford T se hace completo o no se hace. En concreto, se hace todo lo que los stakeholders piden al comienzo del proyecto o hemos fracasamos. En Scrum, como ya lo vimos en los ejemplos de desarrollo iterativo incremental, hacemos algo *bastante bueno* que, según el proverbio chino, es *enemigo de lo perfecto*. La lógica es simple: dado que admitimos que existe la posibilidad de que no lleguemos a desarrollar todo lo prometido, construiremos aquellas características prioritarias primero, de forma de *maximizar el valor entregado a los stakeholders*.

Por último, nos adelantamos a la presentación del flujo completo del framework, presentando la relación existente entre el Product Backlog y el equipo. Al comienzo de cada iteración *el equipo se comprometerá* a entregar una porción del Backlog al final de la misma. Luego entraremos en mayor detalle en este punto, pero aprovechemos para bautizar a este subconjunto de PBIs como *Backlog Comprometido* o *Pronóstico*.

## *Sprint Backlog*

El Sprint Backlog materializa la *táctica* utilizada por el equipo para desarrollar PBIs durante una iteración (que en Scrum llamaremos *Sprint*). El equipo será obviamente el *dueño*, el *responsable* de esta herramienta. En ella se verán plasmadas las *tareas* que consideren necesarias para poder entregar al final del Sprint los PBIs a los que se han comprometido. Veremos en breve en qué momento del Sprint se produce este compromiso.

## *Incremento orgánico del producto*

Llegamos al artefacto para el que, al fin y al cabo, decidimos usar Scrum: el producto. Al finalizar el Sprint el equipo presenta qué PBIs ha desarrollado, lo que brinda a los stakeholders, dado que cada PBI de manera individual hace crecer orgánicamente el producto, un resultado que a priori entrega valor.

Se dice que el incremento es orgánico porque cumple con lo que daremos en llamar el *"criterio de hecho"*. Vamos a definir que algo está hecho cuando "nadie debe preocuparse más por eso". Por ejemplo, imaginemos una pareja que acaba de volver del trabajo. El marido decide cocinar fideos, mientras la mujer ordena la

ropa. Gente muy hacendosa ellos dos. La mujer pregunta desde la habitación si la comida está lista. El marido responde que no, que en cinco minutos. Al cabo de un rato la mujer insiste y el marido, orgulloso, responde que sí, que la cena está lista. La mujer se dirige raudamente a la mesa y encuentra…papeles desordenados y un vaso medio vacío. Le pregunta al esposo qué pasa: ¡ni siquiera está hecha la mesa! El marido, sorprendido, le contesta que los fideos están listos, lo que significa que la cena está lista. Ahora resta servirlos en platos, limpiar la mesa, etc, etc. ¿Quién tiene razón? ¡Ambos! ¿Cuál es el problema? Que ambos tienen distintos *criterios de hecho*. ¿La consecuencia? No solamente el enojo de ambos, sino que las decisiones (minúsculas en este caso, millonarias en muchos proyectos) se tomaron según el propio criterio de hecho. Si un equipo entrega a un Product Owner un PBI *hecho*, a menos que ambos se hayan puesto de acuerdo en lo que eso significa, puede producir un gran problema tiempo después de la entrega.

Vamos con otro ejemplo del software, que es la disciplina en la que más se utiliza Scrum: un equipo entrega funcionalidad tras funcionalidad durante varios Sprints. Luego, por un cambio en la situación del mercado, el Product Owner comunica que hay que variar la estrategia y realizar algunos cambios sobre el software existente. El equipo explica que eso es muy riesgoso, dado que no se hicieron pruebas sobre lo ya entregado. El Product Owner tiene, digamos, palpitaciones. ¿Cómo se previene? Simplemente explicitando cuál será el criterio. ¿Quién es responsable de que esté definido de la mejor manera posible? Claramente el Product Owner, pues es el responsable de definir qué condiciones debe cumplir un PBI para entregar valor de negocio.

## *Process/Impediment Backlog*

Explicamos hace ya varias páginas que en Scrum se desarrollan orgánicamente tanto producto como proceso, dado que se considera a priori que ambos son complejos. El mejoramiento o construcción del proceso puede ser visto con dos tipos de lentes:

- Mitad del vaso lleno: la organización posee un conjunto de virtudes que pueden ser potenciadas

- Mitad del vaso vacío: la organización posee un conjunto de problemas que pueden ser resueltos

Usualmente combinaremos ambas perspectivas, con mayor foco en los problemas. Cada ítem puede ser entonces un problema a resolver o una virtud a potenciar. En el primer caso cada Process Backlog Item será una situación, una *complejidad accidental* que nos impide ser más productivos, desarrollar productos con más calidad y trabajar con mayor felicidad. En el segundo caso nos enfocaremos en aspectos como ser las razonnes detrás de un aumento en la productividad, buena colaboración entre miembros del equipo o un acercamiento del Product Owner al equipo.

Para que nuestro proceso crezca de manera orgánica vamos a tratar a este backlog de la misma manera que al de producto: sus ítems suelen estar ordenados con el objetivo de maximizar la relación costo/beneficio. Esto es, emismo estará dada por el *retorno de inversión* de cada uno de sus ítems. En este caso la valoración es mucho más abstracta que en el producto: el *valor* está dado por cuánta productividad, calidad y felicidad estimamos obtener al remover el impedimento, mientras que su costo será una amalgama del costo monetario (ej: el servidor es lento, hay que comprar uno nuevo), humano (ej: la reuniones son monopolizadas siempre por la misma persona) y/o políticos (ej: convencer a un gerente que deje de dar órdenes de manera directa a los miembros del equipo).

## CÓMO

Hemos visto hasta ahora al sujeto y al objeto: es hora de describir el predicado. En esta sección vamos a presentar la dinámica, el flujo que tiene un proyecto que utiliza el framework.

### *Ciclos de feedback*

La dinámica de un proyecto Scrum puede resumirse a grandes rasgos como una serie de iteraciones durante las cuales se irán desarrollando orgánicamente *tanto producto como proceso*. Cada iteración se asemejará a los saltos que realizó Hans para poder cruzar el río. Decido hacia dónde saltar, salto, levanto la cabeza y decido: ¿estoy saltando rumbo hacia la visión o debo virar? ¿la manera de saltar ha sido la óptima o debo cambiarla (ej: pruebo saltar con ambos pies en vez de dar un paso largo)? Para poder tomar la decisión de virar y/o modificar la manera

de saltar necesito información, que provendrá de la revisión de lo hecho recientemente. La duración de los distintos ciclos de feedback representará los límites que procurarán encauzar el caos sin cercenar la creatividad.

## *Producto*

Como vimos anteriormente, el desarrollo del producto se dividirá en dos perspectivas complementarias: la estrategia y la táctica.

## *Estrategia*

El ciclo de feedback estratégico será el Sprint o iteración. Durante el mismo el Equipo de Desarrollo procurará convertir el Backlog Comprometido en un incremento del producto que refleje los PBIs comprometidos. El Backlog Comprometido quedará *sellado* durante la duración del Sprint. Esto es, no se podrán agregar, quitar o modificar PBIs del Backlog Comprometido durante la iteración ¿Cuál es la idea detrás del sellado? Sencillamente poder encauzar el caos estratégico. De esto se deduce que la duración del Sprint, si bien podrá variar de iteración a iteración, no podrá variar durante la ejecución del mismo. ¿Cuánto dura un Sprint? No se encuentra especificado en Scrum, por lo que cada equipo encontrará su propia cadencia, seguramente mediante prueba y error.

El ciclo estratégico comenzará en la reunión de Planificación Estratégica y concluirá en el Review o Revisión, prácticamente sobre el final de la iteración.

## *Planificación Estratégica*

La reunión de planificación estratégica tiene como principal objetivo que el equipo de desarrollo se comprometa a la porción del backlog más prioritaria que quepa dentro de su capacidad estimada de trabajo para este Sprint.

## *Revisión*

La Reunión de Revisión tiene lugar sobre el final del Sprint. El objetivo principal de la misma será que el Product Owner decida si acepta o rechaza cada uno de los distintos PBIs que el equipo haya desarrollado.

## *Táctica*

El ciclo de feedback táctico será mucho más corto que el estratégico: ningún plan resiste el contacto con el enemigo. En la táctica se verá reflejado el *cómo*: las tareas que realizará el equipo de desarrollo durante el Sprint para construir el incremento del producto correspondiente al backlog comprometido.

## *Planificación Táctica*

Inmediatamente después de la reunión de planificación estratégica el equipo de desarrollo se reunirá para elaborar su plan inicial. El objetivo aquí es lograr un primer esbozo de la serie de tareas que serán necesarias para desarrollar los PBIs comprometidos.

## *Reunión Diaria de Replanificación Táctica*

La táctica nunca se sella: en cualquier momento del día el equipo tiene la potestad de actualizarla. Sin embargo, existe un momento bien definido en el cual el equipo de desarrollo se reúne con el único objetivo de inspeccionar y adaptar la táctica: durante el mismo se procederá a la asignación, definición y actualización del estado de las tareas que conforman el Sprint Backlog. Esta reunión suele llamarse Daily Meeting, Scrum diario, Standup Meeting entre otras variaciones y es la única que tiene una duración máxima ya definida en el framework: solamente 15 minutos.

## *Proceso*

En Scrum partimos de una premisa fundamental: encontrar el mejor proceso posible para que un equipo auto-organizado desarrolle un producto complejo es un proyecto complejo en si mismo. Por ende en Scrum aplicaremos las mismas ideas que utilizamos para desarrollar un producto para construir el mejor proceso posible.

## *Retrospectiva*

La retrospectiva es el corazón que le da vida a un proyecto Scrum. Es el motor que nos empuja a vivir un proyecto persiguiendo lo que podemos llamar una útopia útil: *la perfección existe, es imposible de alcanzar y, sin embargo, todos los días intento estar más cerca*. Tal vez en este concepto esté la principal diferencia entre Scrum y muchas otras formas de trabajo. Es importante, por ende, entender que será en la retrospectiva cuando se decidirá si hemos podido o no poner en marcha esta filosofía de trabajo. La retrospectiva es no solo la reunión más importante del framework, sino que suele ser también la más díficil.

Una mala retrospectiva nos deja parados en el viejo paradigma. Lamentablemente la retrospectiva más común es la que no se hace. Si salteamos la retrospectiva estamos diciendo en voz baja que el proceso tiene una importancia secundaria, que nuestro paradigma actual es el mejor posible.

Una buena retrospectiva consiste en inspeccionar y adaptar nuestra forma de trabajo. No bastará con llevar a cabo, por más sesudo que sea, un mero análisis de la situación actual. La retrospectiva debe ser generadora de propuestas concretas de mejora. Debe abrir y cerrar un ciclo de feedback sobre el proceso. Al comenzar la misma revisaremos los elementos del Process Backlog a los que los miembros del equipo Scrum se han comprometido y evaluaremos, entre todos, si el problema se ha eliminado o el potencial se ha multiplicado. Luego de esta revisión realizaremos la planificación del siguiente ciclo de feedback: previa priorización del Process Backlog, el equipo desglosará el Process Backlog Item en tareas que, al ser llevadas a cabo, mejorarán la forma de trabajo de alguna forma u otra.

¿Quiénes participan de la retrospectiva? Sobre el equipo de desarrollo y el ScrumMaster no hay muchas dudas, pero tal vez si las haya con el Product Owner. El mismo es por definición parte integrante del equipo Scrum y, por ende, partícipe del proceso. Sin embargo, es cierto que una buena porción de los proyectos en los que recién se comienza a utilizar Scrum el Product Owner es una figura de poder. Esto traerá dos consecuencias: su presencia inhibirá a los miembros del equipo de desarrollo y su ausencia lo coronará como gran chivo expiatorio. Recordemos aquí que Scrum es balance entre pragmatismo e idealismo. Si el equipo de desarrollo así lo decide, el Product Owner quedará excluído de las retrospectivas. Pero el ScrumMaster bien sabe que uno de sus principales objetivos a mediano plazo será trabajar con ambas partes para que el Product Owner pueda participar de esta reunión como un mero colega que, sencillamente, tiene otra perspectiva y responsabilidad en el proyecto.

# De brújulas, árboles y dolores

Ya vimos las reglas y el por qué utilizar Scrum. Ahora adentrémonos en lo que tal vez sea lo más interesante: el resto. Comencemos por el “qué hago mañana si decido utilizarlo”.

## TANTOS PEROS...

Escucho, leo y hasta tengo pesadillas con la misma pregunta: "¿Estoy haciendo Scrum o no?" En términos generales los coaches, entrenadores y viejos lobos de mar tienen una respuesta clara y concisa: "Si sigues todas las reglas del framework la respuesta es 'sí'. Si no, lo siento pero será un rotundo 'no'”. La lógica detrás de este razonamiento tiene mucho sentido: Scrum es una pequeño pero poderoso motorcito que no alcanzará su objetivo si lo customizamos.

A partir de esta definición harto simple, Eric Gunnerson acuñó hace ya años en su blog una palabra que resume la *malas* implementaciones de Scrum: ScrumBut (ScrumPero).

“Sí, claro, estamos haciendo Scrum, pero tenemos tres Product Owners"

"¡Por supuesto que estamos usando Scrum! Eso sí, los sprints varían entre una semana y tres meses, según lo decida el Product Owner"

“¡Esto de hacer Scrum es genial! Es una verdadera lástima que los daily meetings duren casi una hora"

Etc, etc, etc

Siguiendo con esta línea de pensamiento existe una clara forma de comenzar tu camino con Scrum: sigue todas las reglas desde el primer día sin cuestionarlas. El motorcito va a hacer su trabajo, iluminando el camino que tenemos por delante. Sin motor no hay luz.

## EL ARCOIRIS QUE GOTEA

Scrum es una excelente manera de tratar con una organización disfuncional, pero no tiene sentido plantearse como escenario de trabajo a una organización que *no funciona* en absoluto. Al comenzar su utilización de Scrum las compañías tienen un cierto modo de trabajo que, mal que mal, funciona. Si no, claramente no habría organización. ¿Cómo hacemos entonces para utilizar Scrum sin despreciar ni desperdiciar un proceso que, aunque sea a duras penas, entrega resultados? Los cambios radicales raramente funcionan: las dietas son un excelente ejemplo. De gordo a flaco a gordo en, digamos, semanas. Tal vez poner en duda el enfoque del ScrumPero sirva de algo...

En lugar de considerar que estamos en un terreno absolutamente oscuro, imaginemos un cuadro distinto: si bien el status quo no puede ser descrito en términos de un cielo despejado y colinas alfombradas de césped, no estamos completamente a ciegas. Imaginemos una escena montañosa, gris, nublada. Deprimente pero transitable ¿Qué significa Scrum dentro de esta topografía apocalíptica? Un hermoso y brillante arco iris. Al final del arco iris, por supuesto, se encuentra el caldero repleto de monedas de oro. Nunca nadie ha llegado ni llegará al caldero, pero este arco iris es muy especial: acaba de ser pintado y todavía gotea monedas doradas. Existe un camino repleto de oro. Repleto. Eso sí, el sendero no está ni marcado ni es sencillo. Cuestas que trepar, desfiladeros riesgosos y precipicios que saltar. Scrum es, ni más ni menos, tu brújula. El norte es, claro está, el final del arco iris. Nunca llegarás al final pero sabes que vale la pena caminar. Eso sí, será una ardua caminata. Impedimentos. Muchos, muchos impedimentos.

## SIÉNTETE ORGULLOSO DE ESE BROTE DE SCRUM

Tu adopción de Scrum es un viaje único, irrepetible. Si la vemos como proyecto es sin dudas uno enormemente complejo. El enfoque que tomaremos copiará a la naturaleza: crecimiento orgánico. Nuevamente plantaremos y cuidaremos de un árbol, nuestro árbol de Scrum. El ScrumMaster (coach Scrum o como querramos llarmarlo) es la semilla de este árbol y las retrospectivas serán el agua y el sol. El terreno en el que este árbol crecerá es único, distinto de cualquier otro, irrepetible. ¿Con solamente tener un coach y hacer retrospectivas estoy haciendo

Scrum? Yo creo que sí. En mi opinión Scrum no es lo mismo que el framework Scrum. Scrum es el árbol y hay árbol en cuanto haya un brote que ve la luz del sol.

¿Cómo se verán sus primeras raíces, ramas y hojas? Recordemos en qué consiste una retrospectiva: el ScrumMaster, actuando como facilitador de la reunión, ayuda al equipo para que reconozca sus dolores más agudos. La gran mayoría de esos dolores han estado allí por tanto tiempo que ya nadie los recuerde. La inercia es como la anestesia: la mente nos permite sobrevivir simplemente negando la realidad.

El equipo ha aceptado, en voz alta, que tiene problemas. Graves. Los síntomas emergieron. Ahora es momento de llegar a la causa raíz. A la enfermedad. Como lo haría un médico clínico, el coach ayuda al equipo, preguntando con el fin de generar un diálogo exploratorio, a que ellos mismos lleguen al problema que causa el dolor. Una vez que alguien admite tener un problema su perspectiva del mismo cambia indefectiblemente. Este es el momento justo para cambiar el proceso: el equipo está sediento de curas. Está listo para probar lo que el médico recomiende. Sobre todo si la propuesta no solo parece tener sentido, sino que tampoco representa tomar un gran riesgo.

Supongamos que un equipo ha estado trabajando en un proyecto durante meses. Tal vez trabajan en una consultora. Tal vez es un equipo pequeño. Tal vez su jefe es comprensivo o despótico. Tal vez tengan un sueldo altísimo. No importa el escenario: existen dolores. Tal vez el dolor más agudo hoy es la ambigüedad y contradicciones que poseen los requerimientos. La causa parece ser que el equipo actúa en respuesta al grito más histérico que reciban desde el exterior, sea del gerente de marketing o del auditor externo. El problema está sobre la mesa y el coach propone un pequeñísimo cambio en el proceso: “Un miembro del equipo intentará convertirse en el único punto de entrada para cualquier nuevo requerimiento”. Probemos esto durante dos semanas. Si no funciona descartamos la propuesta, al menos por ahora. La nueva estrategia está bien clara y ha sido abrazada por el equipo. Lo intentarán por unas semanas y seguramente funcionará. La plantita de Scrum ha crecido un poco: ahora posee raíces, hojas y un tronquito que posee una pequeñísima porción del framework llamada Product Owner.

Luego de dos semanas el equipo vuelve a reunirse con el coach. La propuesta funcionó. Las cosas están un poco, solamente un poco, mejor que antes. No fue sencillo: varios stakeholders montaron en cólera al escuchar una pregunta en vez de un "sí". Pero funcionó. Ahora es tiempo de reflexionar, explorar, diagnosticar. Otra vez. El dolor más agudo es ahora que algunas tareas nunca se hacen porque todos creen que alguien ya las hizo. En la raíz del dolor hay un problema de comunicación. El coach propone: “juntarse durante 10 minutos dos veces a la semana, con el objetivo de enterarse qué están haciendo el resto de los integrantes del equipo”. Suena razonable. Lo prueban y funciona. El árbol sigue creciendo. Esta pequeña rama ahora tiene una reunión que se parece, solamente un poco, a algo llamado Daily Meeting, que forma parte del framework Scrum. Y el árbol seguirá creciendo, siempre y cuando el coach siga regando la planta.

El árbol crece como resultado de la paulatina facilitación llevada a cabo por el coach. Semana a semana el facilitador ayuda a que el equipo responda una pregunta simple: qué cambiar y por qué. A este proceso lo llamamos facilitación guiada por el dolor (PDF: Pain-driven facilitation). El dolor no es infligido sino detectado y expuesto a la luz. De eso se trata Scrum al fin y al cabo.

## EL DESAFÍO DEL TRANSPLANTE

¿Y qué hacemos entonces con el dedo acusador del ScrumPero? Si frenamos para reconsiderar a cada implementación de Scrum como un proyecto complejísimo en si mismo, la política de utilizar el framework desde el día cero es equivalente a transplantar un arbolito del invernadero de Scrum a nuestra realidad. Transplantar a veces funciona…y a veces simplemente no. Y cuando no funciona, el joven árbol parece florecer durante cierto tiempo mientras por dentro no hace más que resecarse. El árbol muere. Lentamente. Y cuando un árbol muere sólo queda la corteza, o lo que es lo mismo en este caso, las ceremonias que forman la mecánica del framework. Roles y reuniones, simples nombres sin sentido. Hasta que un día sopla un fuerte viento y el árbol cae. Tal vez porque las métricas a fin de año muestran una baja en la productividad, tal vez porque alguien se harta por fin de tener reuniones diarias de horas y horas. Se abandona Scrum. Fue una decepción más. Oportunidad perdida.

# Esa nebulosa llamada espíritu

¿De qué estará hecho el árbol cuando nuestro Scrum está sano? Si un Scrum muerto es solamente reuniones, artefactos y roles, es evidente que algo más debe haber. Tratemos entonces de destilar su savia.

## DE LEYES Y LIBROS DE ARENA

¿Cómo toma decisiones un juez? La ley escrita es general, abstracta. Los casos particulares, el contexto y otros elementos subjetivos no suelen estar considerados en el texto de la misma. ¿Por qué no es entonces trivial ser un buen juez? Porque es necesario *interpretar* la ley. Simplemente porque la misma no contempla todos los posibles escenarios, dado que son infinitos. La humanidad decidió hace ya tiempo que la mejor forma de regular un mundo tan complejo como el real es con reglas por definición incompletas. ¿Qué han usado entonces los jueces durante siglos como base de sus interpretaciones? Jurisprudencia y espíritu. Ejemplos e intuición. ¿Y Scrum qué?

Scrum es simple pero difícil. ¿Por qué? Porque su definición es clara y concisa, pero a la hora de utilizar esa definición es necesario interpretarlo dependiendo del contexto en el que nos encontremos. La definición de Scrum es incompleta a propósito. Tenemos a su vez el equivalente a la jurisprudencia en el mundo de Scrum: los casos de estudio y las buenas prácticas (ej: User Stories) nos han ayudado durante años a perfeccionar nuestra práctica. Pero el espíritu ha quedado sistemáticamente apartado de la discusión. Al menos no ha tenido el rol eminente y explícito que estimo merece tener en la comunidad. La pregunta que resta hacer es entonces: ¿Puede ser descrito el espíritu?

El espíritu de la ley es la intención que los legisladores tenían en mente al momento de redactarla. ¿Por qué no está allí mismo en el texto, expresada en palabras? Simplemente porque a menudo las palabras nublan el entendimiento con su inevitable ambigüedad. Los legisladores prefieren un lenguaje simple, abstracto al momento de la redacción, para así permitir una posterior interpretación por parte de los magistrados. Los creadores de Scrum tenían muchas ideas, valores, principios, prácticas, nociones, preferencias cuando

esbozaron el framework. Esos conceptos han evolucionado con el correr de los años y la propia experiencia de literalmente decenas de miles de practicantes. El espíritu de la ley evoluciona con el tiempo y esto mismo ha ocurrido con el espíritu de Scrum. Los más experimentados con Scrum dividen a sus usuarios entre aquellos que "lo entienden" y los que no. Limpiemos la bruma elitista que está detrás del "lo". Hablemos de los conceptos y sus relaciones. Discutamos, aunque más no sea por escrito.

$$Scrum = reglas + espíritu + buenas\ prácticas$$

## COMPLEJIDAD Y EMPIRISMO

No hay un orden para recorrer estas ideas. O tal vez sí lo haya. Volemos a través de la nube un poco por aquí y otro poco por allá. Tal vez un buen comienzo para la travesía sea pensar en la **complejidad**. El software, como esbozó hace tiempo Brooks , es esencialmente más complejo que tantísimas otras disciplinas, incluyendo al hardware, ese primo cercano que duplica su potencia cada 18 meses. Repasemos entonces qué hace que el software, como tanto otro trabajo del conocimiento, sea tan, pero tan complejo.

La **maleabilidad** del software es un arma de doble filo. Cambiar software, como cambiar las ideas, es prácticamente gratuito. Eso es genial, dado que tenemos en nuestras manos una herramienta que nos ayuda a resolver situaciones en las que **el cambio es la única constante**. Pero cuidado, cambiar software es gratis, pero hacer que este cambio logre formar un todo homogéneo no lo es. Es genial tener una masilla infinitamente flexible a mano, pero no es nada fácil amasarla y esculpirla de manera constante con el objetivo de poder representar un cuerpo en movimiento.

Sigamos adelante. Desarrollar software que ya existe, así como proponer ideas que ya fueron planteadas, no agregará gran valor, al menos desde el punto de vista comercial o académico. Esto sucede porque el software, como las ideas, se duplican de forma gratuita. El desafío que se plantea entonces es, minuto a minuto, proyecto a proyecto, *diseñar lo inexistente*, crear lo que no hay ni hubo. Sólo el software nuevo aporta valor. Resumiendo, dado que tenemos que **construir de forma maleable y creativa**, podemos afirmar que tenemos entre manos un trabajo harto complejo.

[Maleabilidad + Creatividad => Complejidad]

¿Cómo solemos relacionarnos los trabajadores del conocimiento con la complejidad? Con menos **humildad intelectual** que la requerida para poder admitir que tanto el encontrar el mejor **producto** a construir como tallar el **proceso** ideal serán tareas complejas, muy complejas. El legado académico suele imprimir un dejo omnipotente en nuestra manera de encarar los desafíos laborales. Nada mejor que refrescarse un poco nadando de vez en cuanto en el mar de la vulnerabilidad. Es por ello que Scrum propone dejar de lado tanto racionalismo y abrazar el **empirismo** como filosofía de trabajo. Generación de

conocimiento mediante la experiencia. Prueba y error, prueba y error, prueba y error.

## EL ERROR COMO INVERSIÓN

El error, piedra angular del empirismo, es, como podemos aprender de las artes, la única manera de llegar a un resultado creativo. El error como tal siempre tendrá un costo asociado: en dinero, en amor, en orgullo. Lo más intuitivo sería considerar ese gasto como una pérdida. Tomando un enfoque distinto de la misma situación, podríamos comenzar a considerar al **error como inversión**. Al equivocarnos aprendemos qué no debe hacerse. Pero también nos dimos la suficiente libertad como para encontrar un diamante entre tanto carbón.

Echemos un vistazo a cómo otros llevan adelante el trabajo creativo. Preguntémonos cómo se pone en marcha una obra de teatro. Por ejemplo Romeo y Julieta. Los ensayos comienzan la semana entrante. El actor pasó los últimos días cepillando sus dientes como lo haría Romeo. La actriz se ha duchado cada mañana como lo haría Julieta. Cada uno ha construido de forma individual la versión inicial de sus respectivos personajes. Ahora es el turno de encontrarse y llevar adelante la primera improvisación. "Romeo flirtea con Julieta en la parada del bus". La escena dura solo unos minutos. ¿Cómo resulta? Seguramente mal, muy mal. Hace literalmente miles de años que las primeras improvisaciones salen mal. Nadie está satisfecho con el resultado. Romeo no transmitió romanticismo. Julieta se movió casi sin gracia por el escenario. Los actores no lograron establecer una conexión entre ellos. Sin embargo, todos celebran el error. Nadie está preocupado, reina el optimismo.

"Gente irresponsable la del teatro. Y claro, total lo suyo es pura diversión, pura irresponsabilidad. Mi trabajo, en cambio, es serio. Poco tengo para aprender yo de ellos." Basta una pequeña dosis de humildad intelectual para abrir la mente y repensar que, desde hace siglos, incontables compañías teatrales han entregado productos de altísima calidad, con gran satisfacción del cliente, cumpliendo con una fecha de entrega inamovible y utilizando un presupuesto bajísimo. Estas compañías comienzan sus proyectos con requerimientos muy vagos; del guión a la obra teatral hay una distancia sideral: creación de personajes, coreografías, vestuario, iluminación. Proyectos complejos, clientes exigentes, entregas exitosas. Seguramente algo podemos aprender de ellos. Tal vez parte del secreto esté en el tratamiento del error.

El error permite simplificar un desafío aparentemente impenetrable. Nos equivocamos y ahora el universo de posibles soluciones se ve reducido de un hachazo. Todo ese conjunto de soluciones *no* sirve. Pero hay algo más, algo más profundo. Gracias a la equivocación, acaba de nacer un trozo pequeñito del producto final. Tal vez la forma de caminar de Romeo comunicó la virilidad buscada en el personaje. Tal vez la voz de Julieta fue tan sensual como la que esperaba el director. Se ha *creado* parte del personaje. Se está un poco, solo un poco más cerca de tener la obra completa.

*[El error es una ~~pérdida~~ inversión]*

## LO BUENO, SI BREVE, DOS VECES BUENO

Excelente, el error considerado una inversión. ¿Pero qué riesgo acarrean las inversiones? Pueden, simplemente, salir mal. Para ello será necesario edificar una dinámica de trabajo que resulte en un **bajo costo del error**. Si vamos a equivocarnos seguido, pues que sea barato. Demos un paso atrás para mirar desde lejos cuál es andamiaje sobre el que se monta Scrum: planifico, ejecuto, inspecciono y adapto. El Ciclo de Deming, para los que lo conozcan. Si queremos costo bajo debemos iterar, dar saltos, a gran velocidad. O lo que es lo mismo, necesitamos que nuestra manera de trabajo consista de **ciclos de feedback cortos**. *Si me equivoco, quiero que sea lo antes posible.*

En el paradigma de Scrum **lo pequeño es bello**. Los proyectos o releases cortos me permiten detectar rápido si tengo el producto correcto. Los sprints cortos son breves ciclos de feedback estratégico, así como el Daily Meeting marca un brevísimo ciclo de feedback táctico.

## CRECIMIENTO ORGÁNICO

Recapitulemos. En Scrum somos empíricos porque el producto y el proceso para construirlo son complejos. Son complejos porque necesitamos modificarlos permanentemente, hacia destinos muchas veces desconocidos. No cambiamos porque nos gusta el cambio, sino porque simplemente sucede. No nos queda otra opción que aceptar que, al menos desde esta visión del mundo, **el cambio es la única constante**.

Cambiarán el entorno en el que se mueve el proyecto, así como también lo hará nuestra comprensión del mismo. Necesitamos entonces que nuestro producto y nuestro proceso puedan ser modificados con bajo costo. Es imprescindible que esos cambios, que serán tan frecuentes, no pongan en riesgo el proyecto entero, lo que se haya construido hasta el momento. En otras palabras, necesitamos que **producto y proceso** sean **maleables**.

Necesitamos, en ese mismo orden, un producto al que se le pueden agregar y quitar características a un costo razonable, equipos dispuestos a mejorar su forma de trabajo y, por último, un plan que resista la corrección a mitad de camino. La propuesta de Scrum para lidiar con este desafío la llamaremos **crecimiento orgánico**. Los organismos vivos son el ejemplo más básico y ubicuo de

maleabilidad, de adaptabilidad. Definiremos crecimiento orgánico como la posibilidad de agregar valor al producto o proceso a bajo costo, en pequeños intervalos de tiempo y con la seguridad de que puedo, en cualquier momento, quedarme con la versión actual y obtener todo el valor acumulado hasta el momento. Nada se ha roto, el árbol no se ha secado.

## Pareto juega de nuestro lado

La complejidad tiene una nueva consecuencia, esta vez sobre nuestra capacidad de estimar a fin de poder planificar. Tan difícil es comprender las necesidades del usuario que en los proyectos tradicionales se suele cumplir la ubicua ley de Pareto: el 20% de la funcionalidad entrega el 80% del valor de negocio. Basta un poco de sentido común para concluir que, sabiendo que gracias al desarrollo orgánico podemos detener el proyecto en cualquier momento, lo que más nos conviene es **priorizar** nuestro trabajo. Si nos vamos a equivocar, equivoquémonos lo menos posible. Intentemos, al menos, enfocarnos en construir *primero* ese 20%.

Más vale poco y valioso en mano, que mucho y perfecto volando.

## Lo perfecto es el enemigo de lo bueno

Y he aquí la salsa mágica de Scrum: el Product Backlog o, lo que es lo mismo, la mezcla justa de crecimiento orgánico y priorización. Con esta definición del Product Backlog reafirmamos nuevamente el paradigma, la visión del mundo desde la cual utilizamos Scrum. En un entorno complejo **el error va a suceder**. Las predicciones, en mayor o menor medida, serán erróneas. En esta tierra creemos que **lo perfecto es lo enemigo de lo bueno**. En el mundo de Scrum creemos que hemos tenido éxito si hemos logrado **maximizar el retorno de inversión** de los stakeholders. Solo un necio podría afirmar, en este contexto, que el éxito del proyecto estará dado por el grado de completitud del plan.

## Seres Humanos, No Engranajes

Por suerte o por desgracia, el trabajo creativo, el trabajo complejo, debe ser realizado por personas. Más aún, una gran parte de ese trabajo debe ser llevado a cabo por un *grupo* de personas con capacidades complementarias. Esta sea tal vez la principal diferencia que existe entre los modos de producción industriales y del conocimiento. Resulta difícil admitir, desde un punto de vista de management

industrial, que estas personas tienen alegrías, enojos, rencores, empatía, concentración, motivación o, muy a menudo, su perfecto opuesto.