Metodología Scrum

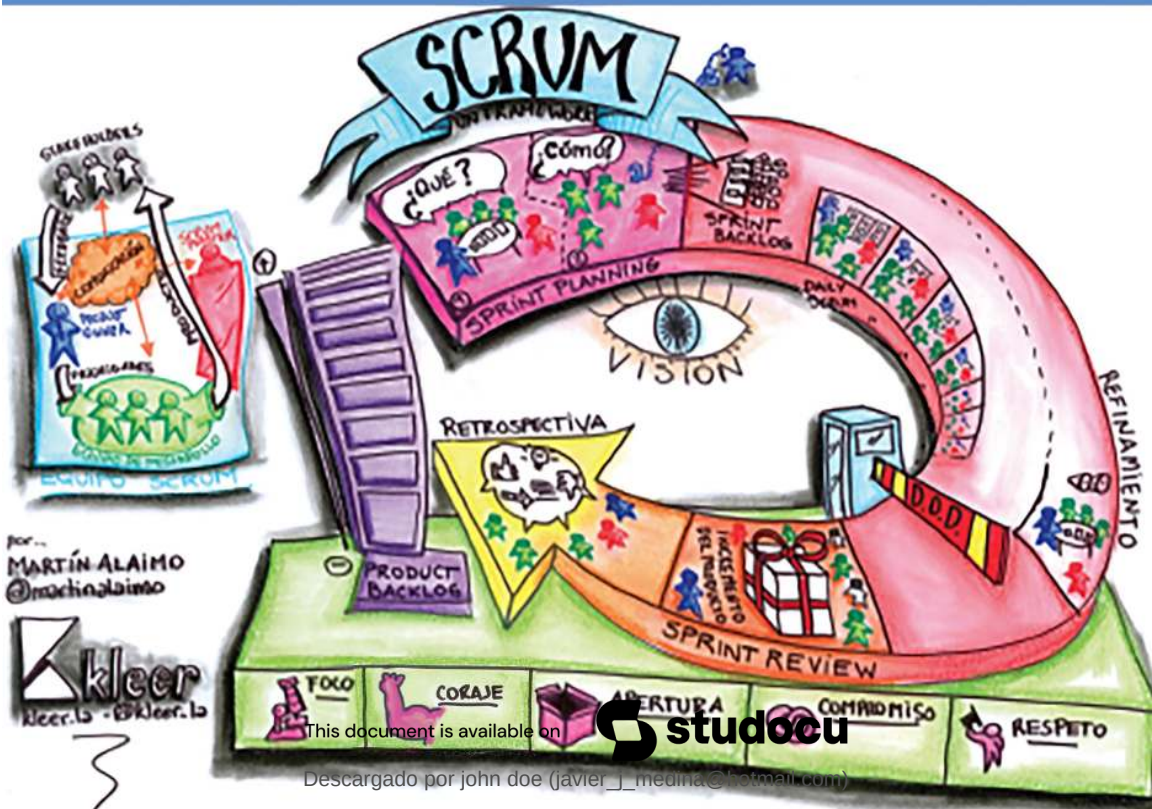
Technologias del aprendizaje (Universidad Tecnológica del Perú)

Martín Alaimo - Martín Salías

Proyectos Ágiles con #Scrum

2da EDICIÓN
Alineada con la Guía Oficial de Scrum

Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos



PROYECTOS ÁGILES CON SCRUM

MARTÍN ALAIMO & MARTÍN SALÍAS

PROYECTOS ÁGILES CON SCRUM

Flexibilidad, aprendizaje, innovación y colaboración en contextos complejos

Buenos Aires
2015

Alaimo, Diego Martín
Proyectos ágiles con Scrum : flexibilidad, aprendizaje, innovación y colaboración en contextos complejos / Diego Martín Alaimo y Martín Salías. - 2a ed. - Ciudad Autónoma de Buenos Aires : Kleer, 2015.

ISBN DIGITAL

1. Gestión. 2. Liderazgo. 3. Informática. I. Salías, Martín
II. Título
CDD 005.3

Hecho el depósito que prevé la ley 11723

A la comunidad ágil latinoamericana, por sus aportes, apoyo, revisiones y espíritu de camaradería. A mis colegas en Kleer, por permitirme aprender día a día. A Daniela, por el empuje y acompañamiento en este viaje.

Martín Alaimo

A mi viejo, que encendió la llama. A las comunidades literarias, técnicas y ágiles en las que participé, que me mostraron el valor de trabajar en red. A mis hermanitos kleerers en toda América Latina, que desafían día a día mis neuronas. Y sobre todo, a Ceci, Ana y Miki, por estar conmigo y mantenerme atento.

Martín Salías

Contenido

1. De la secuencialidad a la iteración

El fracaso del modelo en cascada

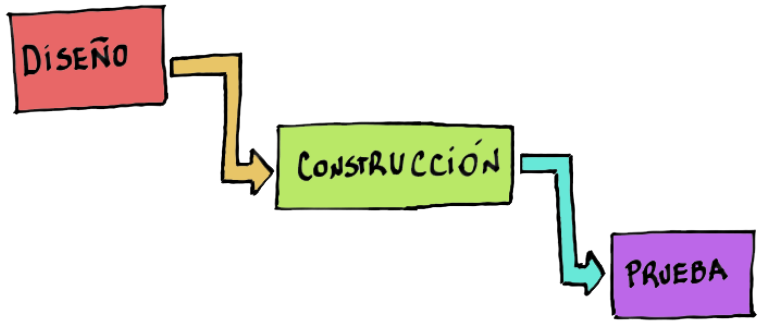


Ilustración 1: Modelo en cascada

La Agilidad, como grupo de metodologías, enfoques y marcos de trabajo, tiene origen en el desarrollo de software. El desarrollo de software no es una disciplina sencilla. En las últimas décadas los lenguajes estructurados modernos[1], de modelado (UML)[2] y posteriormente varias herramientas[3] intentaron sin éxito posicionarse como las "balas de plata"[4] para resolver algunos de sus problemas. Incluso se había llegado a contar con herramientas poderosas y procesos de modelado y diseño sin tener muy en claro cómo aplicarlos a la hora de construir el software.

[1] Edward Yourdon, *Análisis Estructurado Moderno*, Prentice-Hall, 1993

[2] Unified Modeling Language (UML), Information technology, ISO/IEC 19501:2005 http://www.uml.org

[3] Herramientas CASE (U-CASE, M-CASE, L-CASE): http://es.wikipedia.org/wiki/Herramienta_CASE

[4] El término ha sido adoptado como metáfora referida a cualquier solución sencilla que tiene una eficacia extrema. Suele aparecer con la expectativa de que algún nuevo desarrollo tecnológico o práctica fácil de implementar resuelva alguno de los problemas vigentes. Fuente: Wikipedia

No fue sino hasta la adopción amplia y consciente de un tercer elemento que injustamente había sido relegado a un papel secundario: las metodologías de desarrollo, que se encontraron soluciones adecuadas a muchos problemas. La mayoría de estas metodologías fueron introducidas desde la ingeniería civil, lo que resultó en un exhaustivo control sobre los procesos y las tareas.

El Modelo Secuencial de Procesos, también conocido como Waterfall Model o Modelo en Cascada, 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, y hasta prácticamente imposibles. Como no existía proceso alguno en la industria del software, esta condición no impidió su adopción.

La primera mención pública (reconocida) de este tipo de metodologías fue realizada en un artículo que data de 1970 donde el Dr. Winston W. Royce[5] presenta -sin mencionar la palabra "Waterfall"- un modelo secuencial para el desarrollo de software que comprendía las siguientes fases:

- Especificación de requerimientos
- Diseño
- Construcción (también conocida como implementación o codificación)
- Integración
- Verificación o prueba y debugging
- Instalación

[5] Winston Royce, *Managing the Development of Large Software Systems*, Proceedings of IEEE WESCON 26, 1970: 1–9

- Mantenimiento

El proceso Waterfall sugiere una evolución secuencial. Por ejemplo: primero se realiza la fase de especificación de requerimientos. Una vez que se encuentra completa se procede a un "sign-off" (firma/aprobación) que *congela* dichos requerimientos, y es recién aquí cuando se inicia la fase de diseño del software, fase donde se produce un plano o "blueprint" del mismo para que los codificadores/programadores lo implementen.

Hacia el final de la fase de implementación, diferentes componentes desarrollados son integrados con el fin de pulir las interfaces entre ellos. El siguiente paso es la fase de verificación en la que los testers someten el sistema a diferentes tipos de pruebas funcionales mientras los programadores corrigen el código donde sea necesario. Una vez que el sistema responde satisfactoriamente a la totalidad de las pruebas, se inicia una etapa de instalación y mantenimiento posterior.

Los problemas detectados en los modelos tradicionales o de tipo Waterfall se fundamentan, por un lado, en el entorno altamente cambiante propio de la industria, y por el otro, en el proceso mismo de desarrollo de software donde el resultado depende de la actividad cognitiva de las personas más que de las prácticas y controles empleados.

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 Reporte del CHAOS

En el año 1994 el Standish Group publicó un estudio conocido como el "CHAOS Report"[6] donde se encontró la siguiente tasa de éxito en los proyectos de desarrollo de software en general:

- 31.1% es cancelado en algún punto durante el desarrollo del mismo
- 52.7% es entregado con sobrecostos, en forma tardía o con menos funcionalidades de las inicialmente acordadas
- 16.2% es entregado en tiempo, dentro de los costos y con las funcionalidades comprometidas

Los datos publicados, entre otros, mostraron estos índices:

Sobrecostos	% de Respuestas
Menos del 20%	15.5%
21 - 50%	31.5%
51 - 100%	29.6%
101 - 200%	10.2%
201 - 400%	8.8%
Mayor al 400%	4.4%

Funcionalidad entregada	% de Respuestas
Menos del 25%	4.6%
25 - 49%	27.2%
50 - 74%	21.8%
75 - 99%	39.1%
100%	7.3%

[6] The CHAOS Report (1994), Standish Group - http://www.standishgroup.com/sample_research/chaos_1994_1.php

Factores mas importantes para el éxito de un proyecto	% de Respuestas
Involucramiento del usuario	15.9%
Apoyo de la gerencia	13.9%
Claridad en los requerimientos	13.0%
Planificación apropiada	9.6%
Expectativas realistas	8.2%
Hitos más acotados	7.7%
Personal competente	7.2%
Compromiso	5.3%
Objetivos y visión claros	2.9%
Staff enfocado y dedicado	2.4%
Otros	13.9%

Factores mas comunes de cancelación de proyectos	% de Respuestas
Requerimientos incompletos	13.1%
Falta de involucramiento del usuario	12.4%
Falta de recursos	10.6%
Expectativas irreales	9.9%
Falta de soporte gerencial	9.3%
Requerimientos y especificaciones cambiantes	8.7%
Falta de planificación	8.1%
No se necesitaba más	7.5%
Falta de gestión IT	6.2%
Analfabetismo técnico	4.3%
Otros	9.9%

Factores mas importantes de desafío para los proyectos	% de Respuestas
Falta de input del usuario	12.8%
Requerimientos y especificaciones incompletas	12.3%
Requerimientos y especificaciones cambiantes	11.8%
Falta de apoyo gerencial	7.5%
Falta de conocimientos técnicos	7.0%
Falta de recursos	6.4%
Expectativas irreales	5.9%
Objetivos poco claros	5.3%
Calendario poco realista	4.3%
Nuevas tecnologías	3.7%
Otros	23.0%

Demora	% de Respuestas
Menos del 20%	13.9%
21 - 50%	18.3%
51 - 100%	20.0%
101 - 200%	35.5%
201 - 400%	11.2%
Mayor al 400%	1.1%

Las conclusiones de la investigación sugieren que el involucramiento del usuario y el empleo de periodos de tiempo más cortos son claves para incrementar las tasas de proyectos exitosos.

Bajo este contexto surgieron nuevas metodologías, como por ejemplo:

- Metodologías en Espiral
- Metodologías Iterativas
- Metodologías Ágiles

Tanto las Metodologías en Espiral como las Metodologías Iterativas se encuentran fuera del alcance de este trabajo y pueden considerarse pasos intermedios hacia las Ágiles, por lo que pasaremos directamente a entender estas últimas.

El origen de las Metodologías Ágiles

En los ‘90 surgieron varios movimientos identificados con el nombre de Metodologías Livianas (*Lightweight Methodologies*). Entre estos se encuentran Extreme Programming (XP), Scrum, Pragmatic Programming, 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 que les permitirían a los equipos desarrollar software de forma más acertada con las necesidades del cliente y responder mejor a los 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.

En esta reunión se creó la Agile Alliance[7], una organización sin fines de lucro cuyo objetivo es el de promover los valores y principios de la filosofía ágil y ayudar a las organizaciones en su adopción. También se declaró la piedra angular del movimiento ágil, conocida como Manifiesto Ágil (*Agile Manifesto*[8]).

Manifiesto Ágil

Ilustración 2: Manifiesto Ágil, Utah, 2001

El Manifiesto Ágil se compone de 4 valores y 12 principios.

Valores

1. Individuos e interacciones sobre procesos y herramientas

Las personas son el principal factor de éxito de un proyecto para la creación de producto. Es más importante construir un buen equipo que construir el contexto. Muchas veces se comete el error de construir primero el entorno de trabajo y esperar que el equipo se adapte. Por el contrario, la agilidad propone crear el equipo y que éste construya su propio entorno y procesos en base a sus necesidades.

[7] http://www.agilealliance.org
[8] http://www.agilemanifesto.org

2. Software funcionando sobre documentación extensiva

La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Se prefieren documentos cortos y centrados en lo esencial. La documentación (diseño, especificación técnica de un producto) no es más que un resultado intermedio y su finalidad no es dar valor en forma directa al cliente. Medir avance en función de resultados intermedios se convierte en una simple "ilusión de progreso".

3. Colaboración con el cliente sobre negociación contractual

Se propone que exista una interacción constante entre el cliente y el equipo de proyecto. Esta mutua colaboración será la que dicte la marcha del proyecto y asegure su éxito.

4. Respuesta ante el cambio sobre seguir un plan

La habilidad de responder a los cambios que puedan surgir a lo largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también su éxito o fracaso. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Principios

Los valores anteriores son los pilares sobre los cuales se construyen los doce principios del Manifiesto Ágil. Los dos primeros son generales y resumen gran parte del espíritu ágil del desarrollo de software, mientras que los siguientes son más específicos y orientados al proceso o al equipo de desarrollo:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.

2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.

3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.

4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.

5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.

6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.

7. El software funcionando es la medida principal de progreso.

8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.

9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.

10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados.

12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Waterfall: La anécdota detrás del error

El Origen

En 1985, el Departamento de Defensa de los Estados Unidos publicó el Estándar 2167 (DoD-STS-2167)[9] que establecía un proceso estandarizado para el desarrollo de software: el modelo en cascada.

Este estándar influenciaría a varios otros estándares significativos de la industria, como el JSP-188 (Gran Bretaña), V-Model (Alemania), GAM-T-17 (Francia), entre otros.

El modelo en cascada proponía un enfoque secuencial para el desarrollo de software, donde las diferentes etapas eran seguidas en serie, una detrás de otra: "Requerimientos, Análisis, Diseño, Codificación, Prueba y Operación".

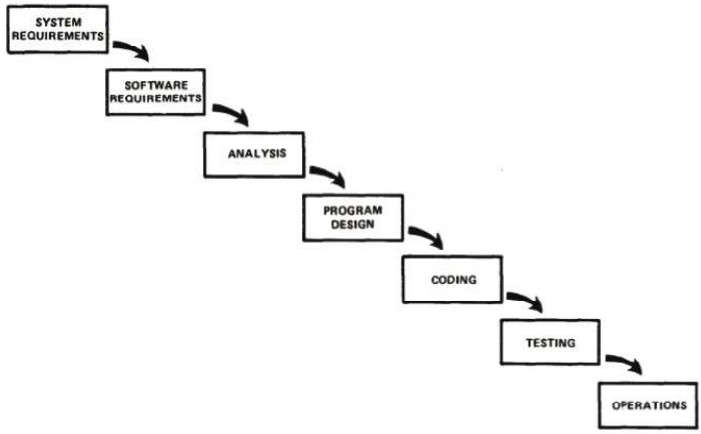
El Origen del Origen

Este estándar se basó en un paper llamado "Managing the Development of Large Software Systems"[10], escrito en 1970 por Winston Royce.

En la página 2 de dicho paper, se puede encontrar la clásica representación del modelo en cascada.

[9] DoD-STS-2167 - http://www.product-lifecycle-management.com/download/DOD-STD-2167A.pdf

[10] Managing the Development of Large Software Systems - http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf



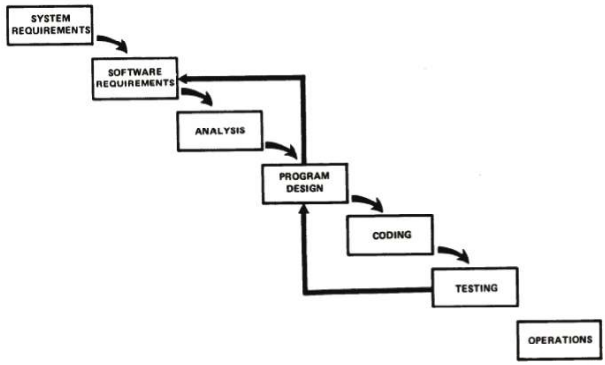
Lo interesante viene inmediatamente después. Winston Royce, conocido como "el padre de la metodología waterfall" escribe:

"*Creo en este concepto, pero la implementación descrita anteriormente es arriesgada e invita al fracaso*".

El problema se ilustra en la Figura 4 de dicho paper. En palabras de Royce: "*La fase de prueba que se produce al final del ciclo de desarrollo es el primer evento para el cual el tiempo, el almacenamiento, la transferencias de entrada/salida, etc., se experimentan en vez de analizarse. Estos fenómenos no son precisamente analizables. No son las soluciones a las ecuaciones de derivadas parciales estándares de la física matemática, por ejemplo.* ***Sin embargo, si estos fenómenos no satisfacen las diversas limitaciones externas, entonces se requiere invariablemente un importante rediseño****. Un simple parche o rehacer algo de código aislado no solucionará este tipo de dificultades. Los cambios de diseño requeridos son propensos a ser tan perturbadores que los requisitos de software sobre el que el diseño se basa y que permiten la justificación de todo el resto, son violados. O bien los*

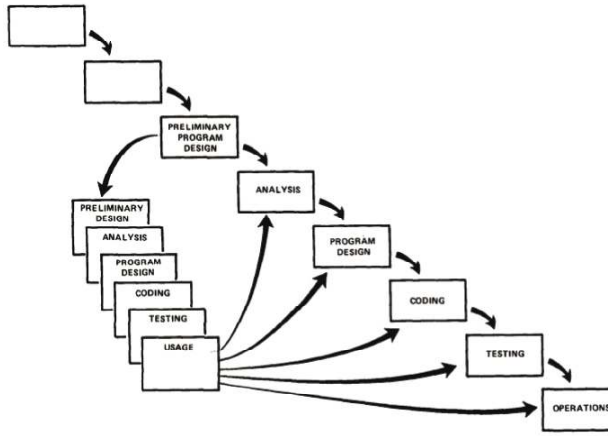
requisitos deben ser modificados, o se requiere un cambio sustancial en el diseño. ***En efecto, el proceso de desarrollo ha vuelto al origen y se puede esperar un exceso de hasta el 100% del tiempo y/o costo.***"

Y aquí la mencionada Figura 4 del paper:



En definitiva, la propuesta de Winston Royce con respecto a la validación del diseño y la incorporación de feedback temprano, en el mismo paper, es la ejecución de un piloto de aproximadamente el 30% del la duración total del proyecto: "*Si el esfuerzo de ejecución es de 30 meses, entonces este desarrollo temprano de un modelo piloto puede ser programado para durar 10 meses.*"

En la Figura 7 del paper mencionado, podemos ver la sugerencia de Winston Royce de realizar un prototipo utilizable para la obtención de feedback que alimentaría las fases posteriores:



El mal entendido

En 2003, Craig Larman y Victor Basili publican un artículo llamado "Iterative and Incremental Development: A Brief History"[11] en el IEEE Journal de Junio de ese año, donde citan un fragmento de una entrevista/conversación que mantuvieron con Walker Royce, hijo del ya difunto Winston Royce. En esta conversación, Walker Royce menciona acerca de su padre: "Él siempre fue un defensor del desarrollo iterativo, incremental, evolutivo. Su artículo describe la cascada como la descripción más simple, pero eso no funcionaría para todos los proyectos, excepto aquellos más sencillos. El resto de su trabajo describe las [prácticas iterativas] en el contexto del modelo de contratación del gobierno de los 60s/70s (un conjunto de serias restricciones)."

[11] Iterative and Incremental Development: A Brief History - http://www.craiglarman.com/wiki/downloads/misc/history-of-iterative-larman-and-basili-ieee-computer.pdf

Volviendo al Estándar del DoD

Ya de vuelta en el Departamento de Defensa (DoD) de los Estados Unidos, en 1987, varios informes del departamento comenzaron a advertir públicamente en contra del estandar 2167 basados en los pobres resultados alcanzados. En 1994 el proceso waterfall del Departamento de Defensa fue reemplazado por el estándar MIL-STD-498[12]. Este nuevo estándar del DoD sugiere evitar el uso del Modelo de Cascada en proyectos que no sean críticos para la seguridad nacional, ya que genera grandes excesos de presupuesto debido a su enfoque formal y burocrático. El MIL-STD-498 fomenta un enfoque más iterativo para el desarrollo de software y reconoce el hecho de que los requisitos cambian y el diseño es un proceso evolutivo.

Lamentablemente, no sucedió lo mismo con el resto de los estándares mundiales que se habían basado en el 2167.

Craig Larman, en uno de sus libros[13], menciona lo siguiente:

"*En 1996 visité el área de Boston y almorcé con el autor principal del estándar 2167. Él expresó su pesar por la creación de la norma de la cascada rígida de un solo paso. Dijo haber sido influenciado por el conocimiento común y la práctica de la época, además de otras normas. No estaba familiarizado en ese momento con la práctica de desarrollo iterativo realizado en periodos fijos de tiempo (timeboxes) y requerimientos evolutivos, y dijo que hubiera hecho una firme recomendación con respecto al desarrollo iterativo e incremental, en lugar de lo que había en el estándar 2167.*"

[12] MIL-STD-498 - http://en.wikipedia.org/wiki/MIL-STD-498
[13] Larman C., "Agile and Iterative Development: A Manager's Guide", 2003

El DoD Hoy

El 28 de Octubre de 2009, el congreso de los Estados Unidos publicó la Ley Pública 111-84[14] (Ley de Autorización de Defensa Nacional para el Año Fiscal 2010) en cuya sección 804 enuncia las condiciones que el DoD debe seguir al momento de efectuar contrataciones con respecto a los Sistemas de Información.

Los contratos deben diseñarse de forma tal de incluir:

- la participación temprana y continua del usuario;
- múltiples incrementos, o liberaciones de rápida ejecución, de capacidades funcionales;
- la creación temprana de prototipos sucesivos para apoyar un enfoque evolutivo; y
- un enfoque modular, de sistemas abiertos.

A partir de esta solicitud, el Departamento de Defensa emitió el reporte al congreso "A New Approach for Delivering Information Technology Capabilities in the Department of Defense"[15].

En este reporte, comunican los cambios realizados a su proceso de contrataciones de servicios de desarrollo de sistemas informáticos.

[14] National Defense Authorization Act For Fiscal Year 2010 - http://www.gpo.gov/fdsys/pkg/PLAW-111publ84/pdf/PLAW-111publ84.pdf

[15] A New Approach for Delivering Information Technology Capabilities in the Department of Defense - http://www.afei.org/WorkingGroups/section804tf/Documents/OSD_Sec_804_Report.pdf

Conclusiones

La creación del modelo en cascada (waterfall) fue institucionalizado por el Departamento de Defensa de los Estados Unidos como consecuencia de una mala interpretación del trabajo de Winston Royce, "Managing the Development of Large Software Systems", donde el autor sugiere que este tipo de enfoque para el desarrollo de software es "*arriesgado y que invita al fracaso"*.

Años más tarde, el DoD corrigió este error, pero varios estándares del Gobierno de los Estados Unidos como de Gobiernos Europeos (Gran Bretaña, Alemania, Francia) que habían derivado de este primero, no siguieron el mismo camino.

A partir del año 2010, el Departamento de Defensa se volcó explícitamente a los modelos ágiles, tanto para desarrollos internos como para contratación de proveedores.

Muchos otros organismos significativos están siguiendo sus pasos.

En el año 1994 el Standish Group publicó un estudio conocido como el "CHAOS Report"[16] donde se encontró la siguiente tasa de éxito en los proyectos de desarrollo de software en general:

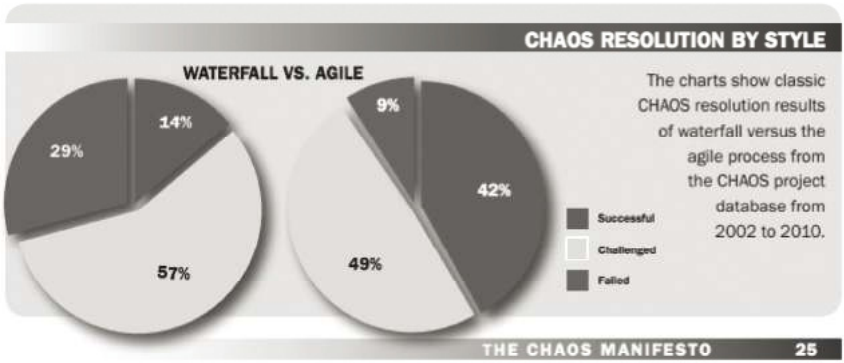
- 31.1% de los proyectos fracasaron, fueron cancelados
- 52.7% de los proyectos se excedieron en costos y/o tiempo
- 16.2% de los proyectos fueron exitosos

Recordemos que en 1994 la metodología utilizada era casi exclusivamente el Modelo en Cascada.

[16] The CHAOS Report (1994), Standish Group - http://www.standishgroup.com/sample_research/chaos_1994_1.php

Once años más tarde de la aparición del Agile Manifesto, en 2012, el Standish Group publicó su clásico análisis anual de gestión de proyectos de la industria del desarrollo de software, ahora llamado CHAOS Manifesto[17].

En este reporte incluye una comparación entre Waterfall y Agile:



En ese año, indicaría:

"*El proceso ágil es el remedio universal para el fracaso en los proyectos de desarrollo de software. Las aplicaciones de software desarrolladas a través del proceso ágil tienen tres veces la tasa de éxito del método en cascada tradicional y un porcentaje mucho menor de demoras y sobrecostos. [...] El software debería ser construido en pequeños pasos iterativos, con equipos pequeños y enfocados.*"

[17] Standish Group, CHAOS Manifesto, 2011

2. Scrum

Cynefin: la complejidad que nos rodea

Antes de hacer hincapié en Scrum en sí mismo, veamos el contexto para el cual Scrum es más eficiente. Podemos utilizar el marco Cynefin[18] 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.

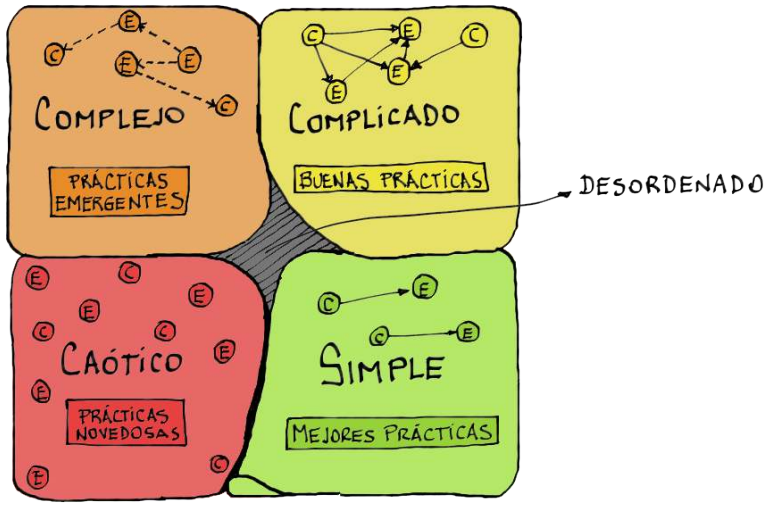


Ilustración 3: Marco Cynefin

El marco Cynefin compara las características de cinco dominios de complejidad diferentes: simple, complicado, complejo, caótico y desordenado, en el centro (ver Ilustración 3). Analicemos cada uno de ellos.

[18] Snowden & Boone, *A Leader's Framework for Decision Making*, BHR, 2007

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 y probadas 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.

Dominio Complicado

En este dominio encontramos problemas con mayor cantidad de variables, 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.

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.

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.

Dominio Desordenado

Nos movemos en el espacio desordenado cuando no sabemos en qué dominio estamos. Se 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.

Y entonces… ¿qué es Scrum?

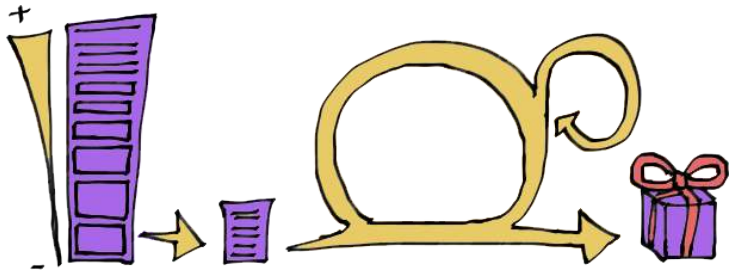


Ilustración 4: Scrum

> ***Scrum**: Un marco de trabajo mediante el cual las personas pueden hacer frente a problemas adaptativos complejos, mientras entregan, creativa y productivamente, productos del mayor valor posible. – La Guía Scrum – Julio 2013*

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 transparencia, inspección y adaptación constante para que los involucrados vayan ajustando y mejorando 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.

En Scrum hay un equipo identificado como el Equipo Scrum que se divide en tres partes: el Equipo de Desarrollo, el Scrum Master y el Product Owner. El Scrum Master es el responsable de que el Equipo Scrum comprenda y utilice Scrum, de liderar de forma servicial al Equipo Scrum y asistir a todos las personas que se relacionan con el Equipo Scrum a comprender cuáles de esas relaciones son benéficas y cuales no. Puede ser considerado un coach o facilitador encargado de acompañar al Equipo Scrum y a las personas externas para generar relaciones e interacciones que maximicen el valor entregado por el Equipo Scrum. El Product Owner es quien representa al negocio, *stakeholders*, cliente y usuarios finales. Tiene la responsabilidad de maximizar el valor del producto y del trabajo del Equipo de Desarrollo.

El progreso de los proyectos que utilizan Scrum se realiza, inspecciona y adapta 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, entregándose un incremento de producto. En este momento es cuando se realiza una reunión de revisión del incremento de producto construido durante el Sprint, donde el Equipo Scrum lo revisa junto a los stakeholders claves. El feedback obtenido en esta reunión puede ser incluido entre las funcionalidades a construir en futuros Sprints.

Principios de Scrum

Scrum es el modelo más utilizado dentro de las Metodologías Ágiles. Muchos de los valores y principios del Manifiesto Ágil tienen su origen en Scrum. Revisemos, ahora desde la perspectiva de Scrum, los valores del Manifiesto Ágil:

1. **Individuos e interacciones sobre procesos y herramientas.** Scrum se apoya en la confianza hacia las personas, sus interacciones y los equipos. El Equipo Scrum identifica lo que hay que hacer y toma la responsabilidad de hacerlo, removiendo todos los impedimentos que encuentre en su camino y estén a su alcance. El Equipo Scrum trabaja en conjunto con otras partes de la organización cuando los impedimentos están fuera de su ámbito de control.

2. **Software funcionando sobre documentación extensiva.** 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. El Equipo Scrum puede documentar tanto como crea 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 en forma iterativa y evolutiva.

3. **Colaboración con el cliente sobre la negociación contractual.** El Product Owner es el responsable de la relación que existe con los clientes 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 sobre seguir un plan.** Scrum, por diseño, se asegura que todo el mundo dentro del Equipo Scrum 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 el producto funcionando y la lista de características pendientes está visible continuamente y para todos los miembros. Esto permite que el alcance del proyecto cambie constantemente en función de la retroalimentación provista por los *stakeholders*. Fomentar el cambio es una ventaja competitiva.

Valores de Scrum

Además de los 4 valores del Manifiesto Ágil, Scrum se construye sobre 5 valores própios:

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-to-market*.

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.

Pilares de Scrum

Scrum es un proceso empírico de control. El empirismo afirma que el conocimiento surge de la experiencia y que las decisiones se deben tomar sobre la base de los hechos conocidos.

Como todo proceso empírico de control, Scrum se apoya en 3 pilares:

1. **Transparencia.** Los aspectos significativos del proceso deben ser visibles y entendibles por todos los responsables del resultado. Por ejemplo: lograr un lenguaje común con respecto al proceso, estar de acuerdo en las expectativas entre quienes construyen y quienes aceptan el producto.

2. **Inspección.** Los responsables del resultado deben inspeccionar frecuentemente los artefactos y el progreso hacia el objetivo para detectar variaciones indeseadas.

3. **Adaptación.** Si en una inspección se detectan oportunidades o ideas que agregan mayor valor que aquello ya planificado, el proceso o el producto pueden ajustarse para maximizar el valor entregado.

Roles de Scrum

Un Equipo Scrum se compone de tres roles: Product Owner, Equipo de Desarrollo y Scrum Master.

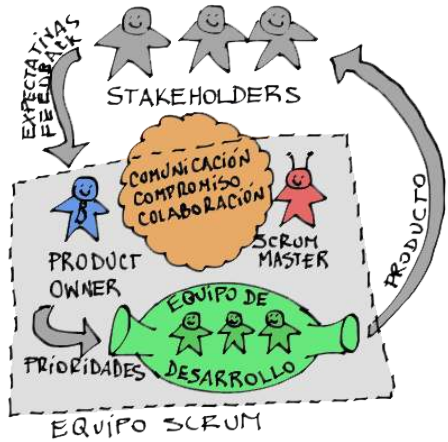


Ilustración 5: Equipo Scrum e interacciones

Product Owner

El Product Owner es la persona responsable de maximizar el valor, tanto del producto como del trabajo realizado por el Equipo de Desarrollo.

Su trabajo se refleja, principalmente, en la gestión del Product Backlog. Esto incluye:

- Asegurar la claridad de los ítems del Product Backlog
- Asegurar la visibilidad, transparencia y comprensión del mismo por parte de todos los involucrados

- Ordenar los Ítems del Product Backlog para lograr los objetivos de la mejor manera posible
- Asegurar que los miembros del Equipo de Desarrollo comprenden los Ítems del Product Backlog al nivel que sea requerido

El Product Owner puede delegar este trabajo en el Equipo de Desarrollo, sin embargo, el Product Owner sigue siendo el responsable.

Las actividades más habituales de un Product Owner son:

- Facilitar la co-creación de la **visión** del producto
- Gestionar las **expectativas de los stakeholders**
- Determinar y conocer en detalle las **características funcionales** de alto y de bajo nivel
- Generar y mantener el **plan de entregas** (*release plan*): fechas de **entrega** y **contenidos** de cada una
- Maximizar la **rentabilidad** del producto
- Determinar las **prioridades** de cada una de las características
- Cambiar las prioridades de las características según avanza el proyecto, acompañando así los cambios en el negocio y la tecnología

Ilustración 6: 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.

El Product Owner puede representar los deseos y decisiones de un comité, pero sigue siendo una única persona responsable, no un comité.

La organización debe respetar las decisiones del Product Owner.

Equipo de Desarrollo

El Equipo de Desarrollo está formado por todos los individuos necesarios para la construcción de un incremento "terminado" de producto al final de cada Sprint. Es el único responsable por la construcción del producto.

Solo el Product Owner tiene la autoridad para establecer el conjunto de requerimientos para el Equipo de Desarrollo. Es el Equipo de Desarrollo quien descompondrá esos requerimientos en tareas, como parte de su auto-organización.

El Equipo de Desarrollo 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. Nadie, ni siquiera el Scrum Master, tiene autoridad para decirle al Equipo de Desarrollo la forma en la que debe hacer su trabajo.

Es el mismo equipo quien determina la forma en que realizará el trabajo y cómo resolverá cada problemática que se presente. La contención de esta auto-organización está dada por el objetivo a cumplir: entregar un incremento de producto "terminado" al final de cada Sprint, que represente los requerimientos comprometidos al principio del Sprint.

Ilustración 7: Equipo de desarrollo

El Equipo de Desarrollo es multifuncional. El equipo, representado por conjunto de sus miembros, posee todas las habilidades necesarias para construir un incremento de producto "terminado" al finalizar cada Sprint. Dentro del equipo de desarrollo no existen especialistas exclusivos, sino más bien individuos generalistas con capacidades especiales. Lo que se espera de un miembro del 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.

En el Equipo de Desarrollo no existen títulos ni rangos jerárquicos: todos son desarrolladores, sin importar el tipo de trabajo que la persona realice. Tampoco hay sub-equipos, al margen de la cantidad de dominios de actividades que sea necesario realizar (ejemplo: prueba, análisis, arquitectura). La Guía de Scrum es tan enfática al respecto, que menciona, para ambos casos que "no hay excepciones a esta regla".

Aunque los miembros del Equipo de Desarrollo tengan habilidades enfocadas en diferentes aspectos de la construcción del producto, la responsabilidad corresponde al equipo como un todo.

El tamaño del Equipo de Desarrollo debe ser lo suficientemente pequeño para mantenerse ágil y lo suficientemente grande para construir un incremento de producto significativo en cada Sprint.

El número buscado es de no menos de tres personas y no más de nueve personas. Con menos de tres personas dentro del equipo de desarrollo, son altas las posibilidades de no contar con todas las habilidades necesarias. Con más de nueve personas el esfuerzo de coordinación es demasiado grande y complejo para un proceso de control empírico como Scrum.

El equipo de desarrollo tiene tres responsabilidades tan fundamentales como indelegables. 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 determinado incremento de producto para cumplir con el objetivo buscado. Y finalmente, también es responsable por la entrega del producto terminado al finalizar cada Sprint.

Scrum Master

La principal función del Scrum Master es asegurar el entendimiento y seguimiento de Scrum tanto por el Equipo Scrum como por los profesionales con los que este equipo se relaciona.

Al mismo tiempo, el Scrum Master es el Coach del equipo y es quien lo ayuda a alcanzar su máximo nivel de productividad posible.

Tomando algunas referencias de Leonardo Wolk podemos decir que el Scrum Master, en tanto que coach, es un *líder, facilitador, provocador, detective y soplador de brasas*.

Líder por ser un ejemplo a seguir, **facilitador** por fomentar contextos de apertura y discusión donde todos pueden expresar sus opiniones y lograr consensos comunes, **provocador** por desafiar las estructuras

rígidas y las antiguas concepciones sobre cómo deben hacerse las cosas, **detective** por involucrarse activamente en la búsqueda e identificación de indicios y pistas en la narrativa del equipo y los individuos y finalmente, **soplador de brasas**, "*un socio facilitador del aprendizaje, que acompaña al otro en una búsqueda de su capacidad de aprender para generar nuevas respuestas*"[19]. Soplar brasas para re-conectar a las personas con sus pasiones, con sus fuegos, muchas veces apagados.

Se espera, además, que el Scrum Master 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.

Las responsabilidades principales del Scrum Master son:

- Velar por el **correcto empleo** y **evolución** de Scrum
- Facilitar el **uso de Scrum**.
- Asegurar que el equipo de desarrollo sea **multi-funcional** y **auto-organizado**.
- **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[20]
- Asegurar la **cooperación** y **comunicación** dentro del equipo
- Acompañar al Product Owner de forma que éste aprenda y pueda:
    - Gestionar del Product Backlog de una forma eficiente

[19] Leonardo Wolk, *Coaching – El arte de soplar brasas*, 2003, p. 22-23

[20] Estos impedimentos podrán ser resueltos dentro del equipo de desarrollo (usualmente), entre diferentes equipos (*Scrum de Scrums*) o con la intervención de la gerencia

- Maximizar el valor mediante la priorización
- Entender y practicar la agilidad
- Comprender la planificación del producto en un contexto empírico

- Acompañar al Equipo de Desarrollo de forma que éste aprenda y pueda:
  - Crear productos de alto valor para el negocio
  - Remover impedimentos que estén a su alcance
  - Enfocarse en su trabajo durante el Sprint

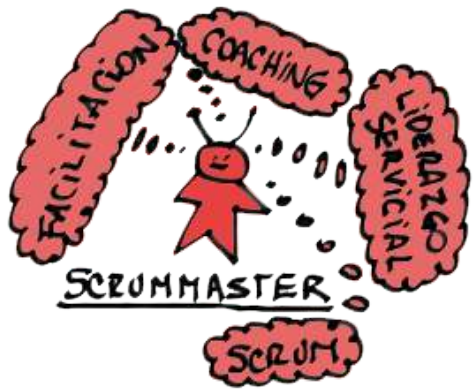


Ilustración 8: Scrum Master

Además de estas cuestiones, el Scrum Master debe detectar **problemas y conflictos interpersonales** dentro del equipo de trabajo. Para respetar la filosofía auto-organizativa del equipo, lo ideal es que el equipo mismo sea quien resuelva estas cuestiones. En el caso de no poder hacerlo, deberá involucrarse al Scrum Master y eventualmente a niveles más altos de la gerencia.

El Scrum Master es un Líder Facilitador

No es casualidad la aparición de un nuevo nombre o rol. Este nuevo concepto del enfoque ágil representa el cambio respecto de las responsabilidades y el modelo de gestión de los gerentes de proyectos tradicionales en relación al equipo de trabajo.

El Scrum Master puede ser visto como un Facilitador o Coach, incluso muchas veces se lo referencia así en lugar de Scrum Master. Su responsabilidad es asegurar que se cumpla con el proceso de Scrum sin interferir directamente en el desarrollo del producto final. Es importante establecer que el equipo de Scrum elige la forma de trabajo que más prefiera, siempre que se cumplan las pautas básicas de Scrum, por ello mientras lo hagan no existe una forma "errónea" de trabajar.

El rol del Scrum Master también incluye asegurar que el desarrollo del producto tenga la mayor probabilidad de ser completado de forma exitosa. Para lograr este cometido, trabaja de cerca con el Product Owner asegurando una correcta priorización de los requerimientos, por un lado, y con el equipo de desarrollo para convertir los requerimientos en un producto funcionando, por el otro.

Por lo que hemos visto, el Scrum Master tiene un rol más indirecto que un Gerente de Proyectos tradicional. A pesar de esto es un rol vital para el éxito de Scrum. Para todo Gerente de Proyectos tradicional, el cambio hacia esta nueva filosofía de gestión es desafiante. Se dice que "Scrum es fácil, hacer Scrum es difícil[21]". Esta afirmación tiene sus fundamentos en la idea de que una cosa es aprender Scrum y otra muy diferente es aplicar Scrum exitosamente. Emprender este camino significa adoptar una filosofía de liderazgo servil por sobre el comando y control.

[21] "Scrum is simple, doing Scrum is hard" - Jim York, CST

Finalmente, cuando un Scrum Master logra cubrir exitosamente su rol, la implementación de Scrum sucede sin sobresaltos. Las responsabilidades del Scrum Master deberían cubrir la totalidad de su tiempo. Si bien hay casos en los que el Scrum Master cumple, además de su rol, el rol de desarrollador, no siempre es la mejor de las situaciones ya que ambas responsabilidades podrían llegar a exceder la disponibilidad de una sola persona, y así alguno de ambos roles no estaría siendo cubierto satisfactoriamente.

Artefactos de Scrum

El proceso de Scrum posee una mínima cantidad necesaria de artefactos formales para poder llevar adelante la construcción de un producto. A continuación describiremos cada uno de ellos.

Product Backlog

El primero de los artefactos, y principal de Scrum, es el Product Backlog o Pila del Producto.

Ilustración 9: Product Backlog

El Product Backlog es básicamente un listado ordenado de todo aquello que es necesario que forme parte del producto y es la única fuente de requerimientos o cambios a realizar sobre el producto. El Product Backlog se forma de ítems (Product Backlog Ítems, PBIs) o características del producto a construir, mantenido y ordenado 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.

Esta prioridad es responsabilidad exclusiva del Product Owner y, aunque el equipo de desarrollo pueda hacer sugerencias o recomendaciones, es el Product Owner quien tiene la última palabra sobre la prioridad final de los ítems del Product Backlog, teniendo en cuenta el contexto de negocio, el producto mismo y el mercado en el que está inserto.

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.

Si planteáramos un ejemplo que ilustre el valor de negocio de los PBIs podríamos decir: en un proyecto cuyo objetivo es aumentar la afluencia de alumnos y facilitar la comunicación de los contenidos de las diferentes carreras de una universidad, se ha decidido crear un sitio web con diferentes características que se encuentran listadas en el Product Backlog. Dos de ellas son 1) que el alumno pueda acceder a los programas de estudios de las diferentes carreras y sus contenidos y 2) que el alumno pueda efectuar el pago en línea de su matrícula y cuotas utilizando una tarjeta de crédito.

En esta situación, muchos podríamos pensar que el requerimiento que implica el pago online con tarjeta de crédito representará un mayor valor de negocio que darle acceso a los alumnos a los contenidos de los programas de estudio, cuando la realidad es a la inversa: 1) el hecho de que un alumno pueda acceder a los contenidos de los programas de las diferentes carreras aporta un mayor valor hacia el cumplimiento del objetivo del producto (aumentar la afluencia de alumnos e incrementar la comunicación de los programas) que lo que el pago online podría hacer y 2) un alumno podría seguir abonando con tarjeta de crédito telefónicamente.

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, si bien es una simple fórmula matemática, tiene implícita la problemática de 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.

Prioridades según la importancia y el riesgo de cada PBI

Ya sea que los ítems del Backlog se prioricen por valor de negocio o por ROI, en cualquier caso llamémosle "priorizar por importancia", éstos pueden verse complementariamente afectados por el nivel de riesgo asociado a cada uno de ellos.

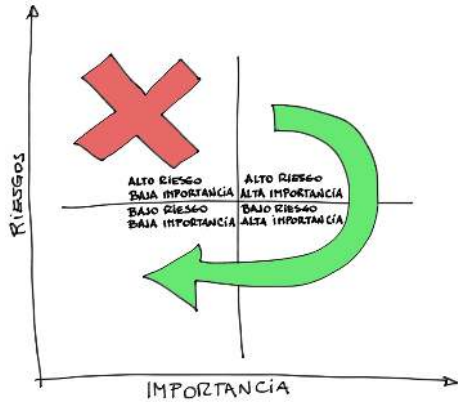


Ilustración 10: Mitigación de riesgos

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.

Alcance Variable

Debido a que asumimos que no nos es posible conocer de manera anticipada y con un nivel muy fino de detalle todas las características del producto que pretendemos construir, sino que es un viaje de descubrimiento que emprendemos junto con el cliente, este Backlog es un elemento vivo que muta a través del tiempo, al ritmo que vamos aprendiendo sobre el producto y su contexto, con las entregas iterativas y el *feedback* frecuente.

El Product Backlog nunca está finalizado. La confección inicial del mismo se basa únicamente en los requerimientos más conocidos y mejor entendidos. El Product Backlog evoluciona y va mutando a la par del producto y el contexto en el cual este opera. El product Backlog es dinámico, cambia constantemente.

Si bien, tradicionalmente, el alcance se ha intentado fijar desde el comienzo de un proyecto, y así manejar el costo y el tiempo como los elementos variables, desde la agilidad, esta ecuación se invierte: el tiempo y el costo son los componentes fijos del proyecto, mientras que el alcance es el componente variable.

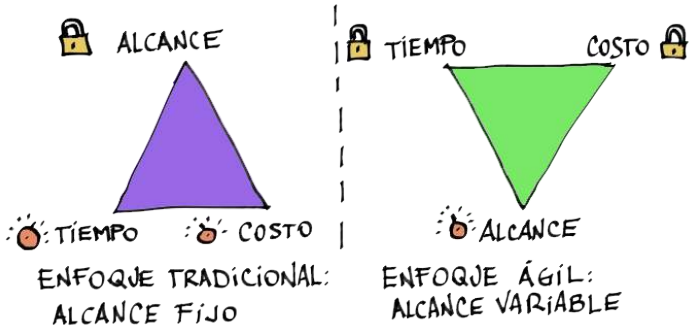


Ilustración 11: Triángulos de hierro

Un Backlog Eficiente

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.

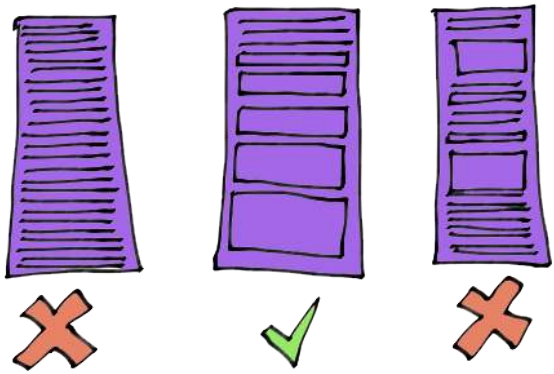


Ilustración 12: Niveles de detalle de PBIs

El Principio de Pareto

Wilfredo Pareto nació en 1848 en Italia, donde creció formando parte de la clase media alta. Fue un reconocido ingeniero, sociólogo, economista, político y filósofo. Uno de sus estudios más reveladores, en aquella época, dejó al descubierto que el 80% de las tierras de Italia pertenecían el 20% de la población. A partir de ese descubrimiento, varios matemáticos y economistas derivaron esas observaciones y las verificaron en otros ámbitos. Uno de ellos fue Joseph Juran, quien en 1941 planteó el Principio de Pareto (o regla del 80/20) aplicado a la calidad: el 80% de los efectos son producidos por el 20% de las causas. Esta ley también se conoció como el principio de "los pocos vitales (el 20% principal que genera el 80% importante) y los muchos triviales (el 80% restante que genera el 20% remanente)".

Aplicando este principio al diseño y construcción de productos, podemos decir que el 20% de las características de un producto 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. Podemos representar esta relación, recursiva, mediante el siguiente gráfico:

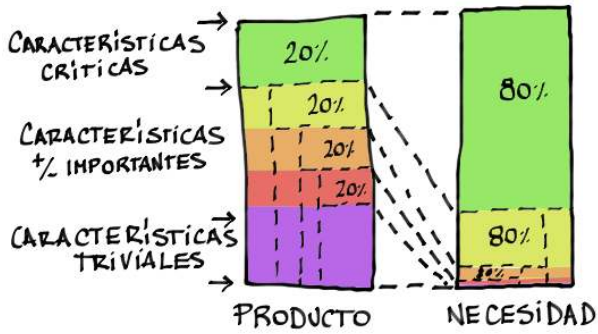


Ilustración 13: Regla del 80/20

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.

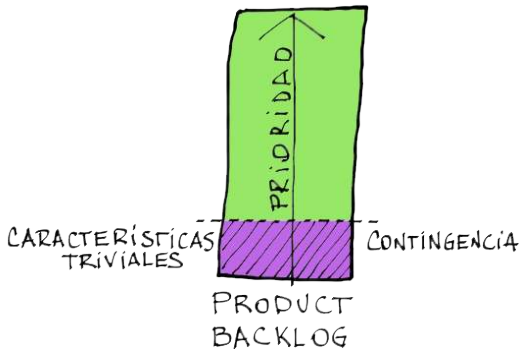


Ilustración 14: Alcance contingente

Indicación de progreso estratégico

En cualquier momento durante la construcción del producto, puede ser identificada la cantidad de trabajo restante para alcanzar cierto objetivo. El Product Owner es quien lleva el registro constante del total del trabajo remanente, como mínimo al finalizar cada Sprint.

Al llevar registro constante del trabajo remanente en el Product Backlog al finalizar cada Sprint, el Product Owner puede realizar una proyección de trabajo a realizar con respecto a alguna fecha objetivo.

Como toda la información en Scrum, esta información es transparente y visible para todos los Stakeholders.

En la Guía Scrum, Jeff Sutherland y Ken Schwaber, hacen una mención explícita sobre la utilización de herramientas y gráficos para llevar este registro de progreso: "*Se han utilizado diversas prácticas predictivas para proyectar tendencias referentes al progreso del trabajo, como burn-downs, burn-ups o diagramas de flujo acumulado. Estas han demostrado ser útiles. Sin embargo, estas no deberían reemplazar la importancia del empirismo. En contextos complejos, lo que pasará es desconocido. Solo lo que ha sucedido en el pasado es lo que puede utilizarse para tomar decisiones con respecto al futuro.*"

Sprint Backlog

El Sprint Backlog es el conjunto de Ítems del Product Backlog (PBIs) que fueron seleccionados para trabajar durante un determinado Sprint, conjuntamente con el plan para lograr la entrega del Incremento de Producto al finalizar el Sprint y alcanzar el objetivo del Sprint.

Este plan, y las tareas que su ejecución requiera, es elaborado por el Equipo de Desarrollo. El Sprint Backlog hace visible y transparente todo el trabajo que el Equipo de Desarrollo identifica como necesario para alcanzar el objetivo del Sprint.

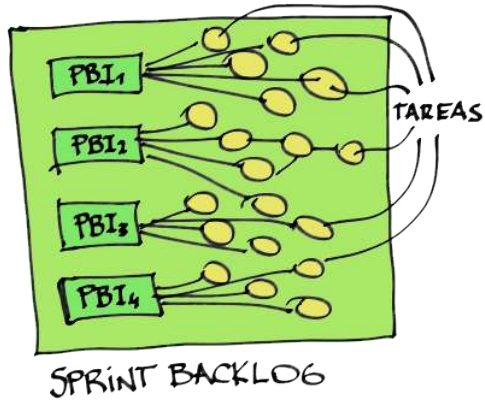


Ilustración 15: Sprint Backlog

El Sprint Backlog es un artefacto vivo. El Equipo de Desarrollo lo modifica durante el Sprint y el Sprint Backlog emerge durante este tiempo, a medida que el Equipo de Desarrollo ejecuta el plan y aprende sobre el trabajo necesario para alcanzar el objetivo del Sprint: si se descubre que es necesario realizar nuevo trabajo, se agrega al Sprint Backlog; si alguno de los elementos del plan del Sprint se tornan innecesarios, se remueven del Sprint Backlog.

Solo el Equipo de Desarrollo tiene la potestad de alterar el Sprint Backlog durante la ejecución del Sprint.

Indicador de progreso táctico

En cualquier momento durante la ejecución del Sprint, puede ser identificada la cantidad de trabajo restante del Sprint Backlog. El Equipo de Desarrollo es quien lleva el registro constante del total del trabajo remanente del Sprint, como mínimo al finalizar cada Daily Scrum.

Al llevar registro constante del trabajo remanente en el Sprint Backlog, el Equipo de Desarrollo puede realizar una proyección de trabajo a realizar para lograr el objetivo del Sprint, y gestionar el progreso.

Como toda la información en Scrum, esta información es transparente y visible para todos los Stakeholders y miembros del Equipo Scrum.

Incremento de Producto

El resultado de cada Sprint es 1) la sumatoria de todos los Ítems del Product Backlog (PBIs) completados, y 2) el valor total de este conjunto de PBIs, sumados a todos los incrementos de Sprints anteriores.

Ilustración 16: Incremento Funcional

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

Terminado: al finalizar el Sprint, el incremento del producto debe estar "Terminado". Esto significa que está en una condición tal que lo hace utilizable y que, además, cumple con la "Definición de Terminado" del Equipo Scrum.

Eventos de Scrum

Antes de describir en detalle los eventos de Scrum, recordemos el mecanismo de time-box[22] promovido por Scrum y los principios de ritmo sostenible, entrega frecuente de software valioso y adaptación constante que encontramos en el Manifiesto Ágil[23]. La razón es que en conjunto constituyen la piedra angular de la dinámica de Scrum, como de cualquier proceso empírico de control: transparencia, inspección y adaptación.

Una característica común de todos lo eventos de Scrum es que están comprendidos en un time-box, esto quiere decir que cada evento tiene una duración máxima de tiempo que no debe ser superada. Todos los eventos de Scrum tienen un objetivo, y pueden ser terminados anticipadamente una vez alcanzado ese objetivo. La única excepción es el Sprint, el cual no se debe prolongar o acortar.

Cada evento en Scrum es una oportunidad de inspeccionar y adaptar algo. La omisión de algún evento de Scrum resulta en una reducción de la transparencia, de la visibilidad y de la posibilidad de inspección y adaptación.

[22] Ver principios de Scrum
[23] Ver valores del Manifiesto Ágil

Sprint

Las iteraciones en Scrum se conocen como Sprints. Scrum, como todos los enfoques ágiles, es un proceso de construcción incremental e iterativo. Esto significa que el producto se construye en incrementos funcionales entregados en periodos cortos para obtener feedback frecuente.

En general, Scrum recomienda una duración de Sprint de no más de un mes, siendo 2 o 3 semanas lo más habitual que encontraremos en la industria. 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.

Como excepción podemos mencionar aquellas situaciones donde el equipo mismo decida probar con iteraciones más largas o más cortas. Esta decisión se basa principalmente en la volatilidad del contexto: mientras más volátil sea (negocio cambiante, requerimientos desconocidos, tecnología nueva, etc.) más corta será la duración del Sprint. Lo importante es recordar que se logra mayor ritmo y previsibilidad teniendo Sprints de duración constante.

Durante el Sprint, no se deben hacer cambios que pongan en peligro el objetivo del Sprint, la expectativa de calidad no debe reducirse y el alcance debe ser clarificado y negociado entre el Product Owner y el Equipo de Desarrollo a medida que se va dando el aprendizaje

Retrasos y adelantos en un Sprint

Muchas veces 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.

Cancelación de un Sprint

Un Sprint puede ser cancelado antes del cumplimiento del time-box. Solo el Product Owner tiene la autoridad para cancelar un Sprint. Tanto el Equipo de Desarrollo, como el Scrum Master o los Stakeholders pueden recomendar la cancelación.

Cuando un Sprint se cancela, todos los PBIs que cumplen con el criterio de "Terminado" son revisados, mientras que los remanentes PBIs son reestimados y devueltos al Product Backlog.

La cancelación de un Sprint es, habitualmente, un hecho traumático para el Equipo Scrum.

Sprint Planning (Planificación de Sprint)

Al comienzo de cada Sprint se planifica el trabajo que se hará durante el mismo. Este es un trabajo colaborativo de todo el Equipo Scrum.

La planificación del Sprint tiene un timebox de un máximo de ocho horas para un Sprint de un mes. Para Sprints de menos de un mes, el timebox de este evento suele ser más corto.

La responsabilidad del Scrum Master es asegurar que este evento se realice, que los participantes comprendan el propósito del evento y que el timebox sea respetado.

El propósito de este evento de planificación es responder dos preguntas:

1. ¿**Qué** es lo que se puede entregar como parte del Incremento resultante de este Sprint?

2. ¿**Cómo** se realizará el trabajo necesario para entregar este Incremento?

Habitualmente, el Sprint Planning se divide en dos partes, cada parte destinada a responder una de las preguntas anteriores.

Parte uno: ¿**Qué** es lo que se puede entregar en este Sprint?

Ilustración 17: Sprint Planning (Parte 1)

El Equipo Scrum completo colabora para identificar cuáles son los Ítems del Product Backlog que pueden ser realizados durante este Sprint, y que conformarían el Incremento de Producto necesario para alcanzar el Objetivo del Sprint.

Podríamos decir que, en la práctica, 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.

Aún asumiendo que los PBIs ya han sido estimados con anterioridad[24], debido al principio de "aceptar los cambios aun en etapas avanzadas del proyecto", es posible que en esta reunión aparezcan PBIs que no habían sido estimados anteriormente. Frente a esta situación, el equipo de desarrollo indagará y estimará esos PBIs de inmediato.

El objetivo buscado durante esta parte de la reunión es identificar "qué" es lo que el equipo de desarrollo va a realizar durante el Sprint, es decir, todos aquellos PBIs que el equipo cree que podrá entregar. Seleccionar la cantidad de PBIs del Product Backlog es algo que corresponde exclusivamente al Equipo de Desarrollo, no pudiendo ni el Product owner ni el Scrum Master forzar esta decisión.

El Product Owner y el equipo de desarrollo deben participar de esta parte de la reunión como protagonistas principales. El Scrum Master, al tiempo que facilita la reunión, también debe asegurar que cualquier *stakeholder* del proyecto que sea requerido para profundizar en detalles esté presente o sea contactado.

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. Esto determinaría en un principio cuáles son los PBIs comprometidos en este Sprint.

Como se ha visto, hemos hablado en potencial*:* el equipo de desarrollo "podría", esto "determinaría". La razón es que cada uno de los ítems del Product Backlog debe ser discutido para entender cuáles son sus criterios de aceptación y así conocer en detalle qué se esperará de cada uno. De esta manera, el equipo de desarrollo discutirá con el Product Owner sobre cada PBI y generará un compromiso de entrega para aquellos que considera suficientemente claros como para comenzar a

[24] La mecánica de esta estimación será explicada más adelante

trabajar y que además podrían formar parte del alcance del Sprint que está comenzado. A esto se lo conoce como planificación basada en compromisos o Commitment-based Planning.

Objetivo del Sprint

Una vez identificados los PBIs que formarán parte del alcance de este Sprint, el Equipo Scrum determina un Objetivo del Sprint. El Objetivo del Sprint es una meta que será alcanzada durante el Sprint por medio de la implementación del Product Backlog, y provee una guía para que el Equipo de Desarrollo comprenda la razón por la cual está construyendo este Incremento de Producto.

Al finalizar esta primera parte de la reunión, tanto el Product Owner como los stakeholders involucrados (si los hubiese) podrían retirarse, dejando así al Scrum Master y al equipo de desarrollo para que den comienzo a la segunda parte de esta reunión, que se describe a continuación.

Parte dos: ¿**Cómo** se realizará el trabajo de este Sprint?

Ilustración 18: Sprint Planning (Parte 2)

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.

Se espera que el diseño sea emergente, es decir, que surja de la necesidad del equipo de desarrollo a medida que éste avance en el conocimiento del negocio. Por esta misma razón es que no se recomienda realizar un diseño completo y acabado de lo que será realizado durante el Sprint. En cambio, se buscará un acuerdo de alto nivel que será bajado a detalle durante la ejecución de la iteración.

Esto mismo sucede con las actividades del Sprint, es decir que no es estrictamente necesario enumerar por completo todas las actividades que serán realizadas durante la iteración ya que muchas aparecerán a medida que avancemos. Es recomendable que las actividades duren idealmente menos de un día. Esto permitirá detectar bloqueos o retrasos durante las reuniones diarias (ver Scrum Diario más adelante).

Si bien el Product Owner no participa de esta reunión, debería ser contactado en el caso de que

a) el equipo de desarrollo necesite respuestas a nuevas preguntas con la finalidad de clarificar su entendimiento de las necesidades y/o

b) sea necesario renegociar el alcance del Sprint debido a nuevos descubrimientos del Equipo de Desarrollo

El Equipo de Desarrollo puede invitar a otros participantes con el objetivo de que provean asesoramiento técnico o de negocio.

Al finalizar esta reunión, el equipo habrá arribado a un Sprint Backlog, que representa el alcance del Sprint en cuestión. Muchos Equipos Scrum colocan este Sprint Backlog en una pizarra de tareas, taskboard, visible por todos para garantizar transparencia.

Aún más importante que el alcance del Sprint, es su objetivo. El objetivo del Sprint se determina, también, en la reunión de planificación del Sprint. Que el Sprint tenga un Objetivo, le da al Equipo de Desarrollo cierta flexibilidad con respecto al alcance. Mientras el Equipo de Desarrollo trabaja durante el Sprint, mantiene siempre presente el objetivo perseguido.

Así se dará comienzo a la construcción del Incremento de Producto para este Sprint.

Scrum Diario

Uno de los beneficios de Scrum está dado por el **incremento de la comunicación** dentro del equipo de proyecto. Esto facilita la coordinación de acciones entre los miembros del equipo de desarrollo y el conocimiento "en vivo" de las dependencias de las actividades que realizan.

Por otro lado, se requiere además **aumentar y explicitar los compromisos** asumidos entre los miembros del equipo de desarrollo y **dar visibilidad a los impedimentos** que surjan del trabajo que está siendo realizando y que muchas veces nos impiden lograr los objetivos.

Adicionalmente, un equipo auto-organizado se beneficia mucho teniendo **instancias de sincronización frecuentes** para evaluar el progreso y tomar decisiones de replanificación.

Estos cuatro objetivos: 1) incrementar la comunicación, 2) explicitar los compromisos, 3) dar visibilidad a los impedimentos y 4) sincronizarse frecuentemente, son logrados mediante las reuniones diarias de Scrum (*Daily Scrums*). Estas reuniones tienen, como su nombre lo indica, una frecuencia diaria y un timebox de 15 minutos.

Cada día, en estos 15 minutos, los miembros del Equipo de Desarrollo sincronizan sus actividades y re planifican lo que crean necesario hasta el siguiente Scrun Diário.

Con el objetivo de reducir la complejidad, esta reunión se realiza todos los días, a la misma hora y en el mismo lugar.

A la reunión diaria acude el Scrum Master y el equipo de trabajo. En el caso de que sea necesario, se podrá requerir la presencia del Product Owner y de los stakeholders. De todas maneras, se intenta que sea una reunión abierta donde cualquier interesado en escuchar lo que sucede pueda participar en calidad de observador. Se recomienda que los observadores no participen activamente en la reunión, y mucho menos, que soliciten a los miembros del equipo justificación del progreso y explicación de los problemas.

El Scrum Master es el responsable de que la reunión se lleve a cabo, pero es el Equipo de Desarrollo quien la conduce.

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 que nos ayuda a alcanzar el objetivo del Sprint?

2. ¿En qué voy a estar trabajando desde ahora hasta la próxima reunión diaria para ayudarnos a alcanzar el objetivo del Sprint?

3. ¿Qué problemas o impedimentos me impiden, o nos impiden, alcanzar el objetivo del Sprint?

Es importante destacar que en ningún momento se trata de una reunión de reporte de avance o status al Scrum Master ni a otras personas. Por el contrario, es un espacio de estricta comunicación entre los miembros del equipo de desarrollo.

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, que contribuyan a alcanzar el objetivo del Sprint. Cuando hablamos de compromisos, hacemos referencia a aquéllos que los miembros del equipo asumen ante sus compañeros.

Ilustración 19: Scrum Diario

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

En el caso de que los PBIs del Sprint se hubiesen podido dividir en actividades de menos de un día: si una de estas actividades se encuentra en progreso durante dos reuniones diarias seguidas (con 24hs de separación) claramente se advierte un retraso.

Revisión de Sprint

Al finalizar cada Sprint se realiza una reunión de revisión del Sprint (*Sprint Review*), donde se evalúa el incremento funcional potencialmente entregable construido por el equipo de desarrollo (el "qué").

El timebox de esta reunión es de cuatro horas para un Sprint de un mes de duración. Para Sprints más cortos este evento habitualmente dura menos tiempo.

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 los interesados utilizan y evalúan durante esta misma reunión, aceptando o rechazando así las funcionalidades construidas.

Ilustración 20: Revisión del Sprint

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.

El grupo de personas completo colabora en la identificación de qué hacer luego, para que estas decisiones sean un input importante y valioso para la próxima Sprint Planning.

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. La única excepción a esta regla es que el Product Owner, por decisión propia, prefiera dar mayor prioridad a otros. En este caso, nada debe salir del Backlog ya que esto no sería considerado como un incremento en el alcance.

La responsabilidad del Scrum Master es asegurar que el evento tenga lugar y que los participantes entiendan su objetivo. El Scrum Master ayuda a que la reunión se mantenga dentro de los límites del timebox.

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.

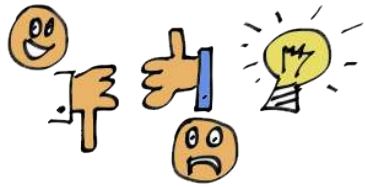


Ilustración 21: Retrospectiva

Esta reunión tiene lugar inmediatamente después de la reunión de revisión y antes de la planificación del próximo Sprint. Mientras que la reunión de revisión se destina a revisar el producto (el "qué"), la retrospectiva se centra en el proceso (el "cómo").

El timebox de esta reunión es de tres horas para un Sprint de un mes de duración. Para Sprints más cortos, este evento, habitualmente dura menos tiempo.

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 Scrum Master. Personalmente yo también recomiendo la participación del Product Owner. En el caso de que se requiera la participación de stakeholders o gerentes, estos podrán ser convocados.

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.

Refinamiento del Product Backlog

El refinamiento del Backlog es una actividad constante a lo largo de todo el Sprint, aunque algunos equipos prefieren concentrarla en una reunión que se realiza durante el Sprint y en función de las necesidades. Es el Equipo Scrum quien decide cuándo y cómo se realiza esta actividad. Su objetivo es profundizar en el entendimiento de los PBIs que se encuentran más allá del Sprint actual y así dividirlos en PBIs más pequeños, si lo requieren, y estimarlos. Idealmente se revisan y detallan aquellos que potencialmente se encuentren involucrados en los próximos dos o tres Sprints.

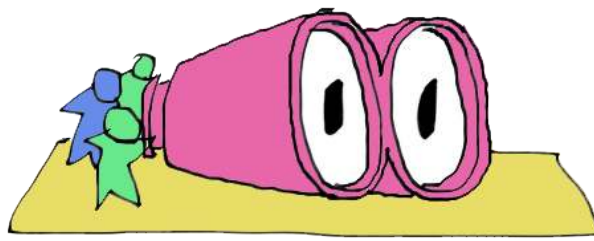


Ilustración 22: Refinamiento del Product Backlog

Otro objetivo importante que se debe perseguir es la detección de riesgos implícitos en los PBIs que se estén analizando, y en función de ellos revisar y ajustar las prioridades del Product Backlog.

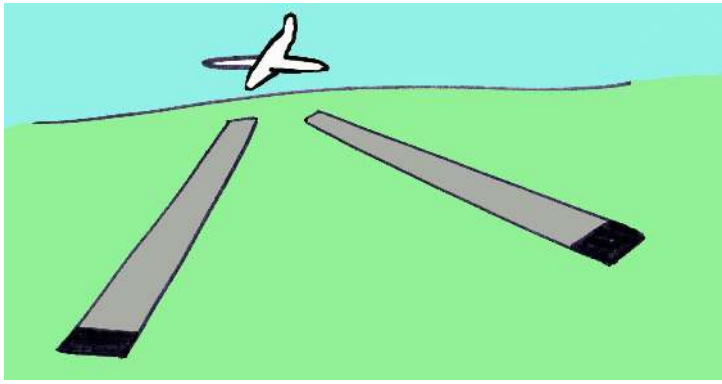
La participación de todo el Equipo Scrum es esencial para el éxito de esta reunión. Sin ellos esta actividad no tendría sentido.

En el caso de que se realice como una reunión, la responsabilidad de convocarla es del Product Owner, entre una y dos veces por Sprint, facilitadas por el Scrum Master.

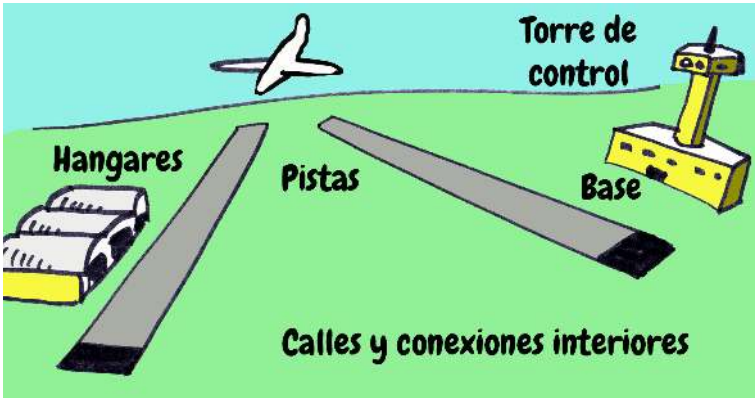
3. Desarrollo Evolutivo

Creación Evolutiva

Supongamos que nos han contratado el municipio de un pueblo para asistirlo en la construcción de un aeropuerto. El objetivo del proyecto es potenciar la incipiente actividad turística de la zona.



Luego de analizar las características y funciones que el aeropuerto debe tener, hemos dividido la problemática en:



Una alternativa para construir el aeropuerto sería dedicar la primera entrega a las pistas, la segunda a la torre de control, la tercera a los hangares, etc., tal como se muestra a continuación.

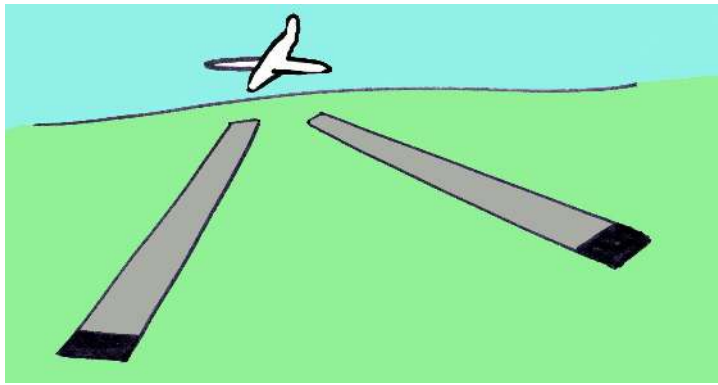


Ilustración 23: Desarrollo Secuencial

Sin embargo, si nosotros decidimos construir el aeropuerto de forma evolutiva e incremental deberíamos tener una unidad funcionando al final de cada iteración, 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:

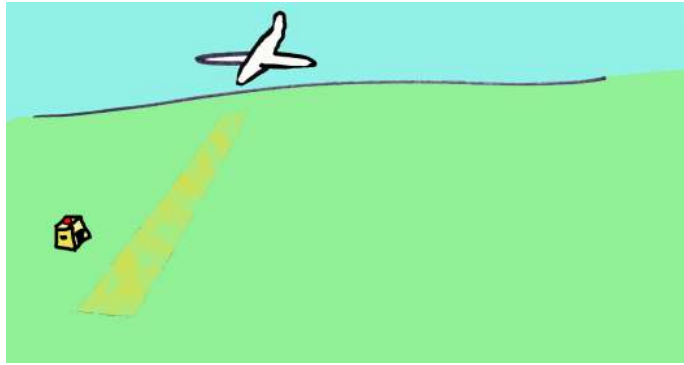


Ilustración 24: Desarrollo Evolutivo

De esta forma, podríamos buscar los siguientes objetivos:

- Entrega 1: una única pista de tierra, una plataforma de tierra y una casilla con una pequeña baliza. Objetivo: permitir rápidamente el tráfico de aviones pequeños y validar los supuestos sobre condiciones de despegue y aterrizaje.

- Entrega 2: mejorar el material de la pista, expandir la cabina para almacenar materiales de mantenimiento y guardia. Objetivo: soportar aviones mayores y garantizar el mantenimiento en base al uso real.

- Entrega 3: pista alternativa de tierra, señales lumínicas de la pista principal, primer hangar. Objetivo: tener alternativa durante tareas de mantenimiento de la pista principal, permitir mayor rango de operación (nocturno, con niebla y otras condiciones de baja visibilidad), proteger un avión chico aparcado en el aeropuerto.

- Entrega 4: etc.

- Entrega 5: etc.

Partiendo de esta base, vamos a introducir tres conceptos complementarios entre sí: ***Minimum Viable Product***, ***Minimum Marketable Feature*** y ***User Story Mapping***.

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 dólar 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.

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*"[25]

User 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 de Historias de Usuario** o *User Story Mapping*[26].

[25] "Phased Releases", James Shore, 2004
[26] User Story Mapping, Jeff Patton, 2009

La teoría del User 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.

Este nivel de confort está dado por 1) la necesidad de negocio (necesito que el viaje se haga en menos de 30 minutos) y 2) cuánto estemos dispuestos a invertir (precio del auto).

Haciendo una analogía con las organizaciones, esta jerarquía de Objetivo → Actividad → Herramienta puede traducirse en Proceso de Negocio → Actividad → Software.

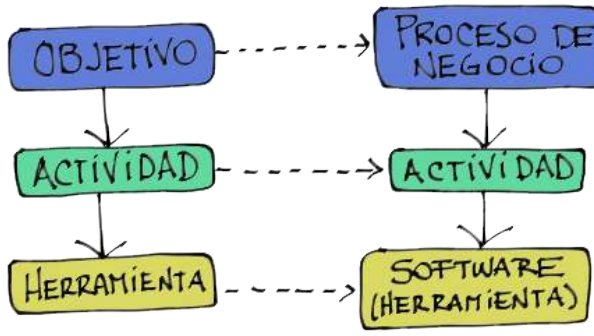


Ilustración 25: Jerarquía operativa

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 y no tanto en profundidad.

Teniendo en cuenta entonces que el software (o cualquier producto basado en conocimiento e innovación aplicados) 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.

Proceso de Análisis Ágil

Para el desarrollo de esta técnica en forma práctica, nos basaremos en un ejemplo real: el análisis de un sistema de gestión de cursos de capacitación. Este ejemplo servirá como columna vertebral en la que vamos a ir aplicando las técnicas ágiles descriptas.

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[27] basada en el trabajo de Constantine & Lockwood[28]. Esta técnica se realiza en equipo, durante un taller donde tantos clientes y miembros del equipo como sea posible colaboran en la identificación de los roles. El taller se compone de cuatro actividades:

[27] Agile Estimating and Planning, Mike Cohn, 2005
[28] Software for Use, Constantine & Lockwood, 1999

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

Brainstorming de un conjunto inicial de roles

Como se ha mencionado anteriormente, la intención de esta actividad es que sea lo más colaborativa posible. Tanto el cliente como el Equipo completo deberían participar, aunque muchas veces será suficiente con la participación de un conjunto representativo del Equipo de desarrollo.

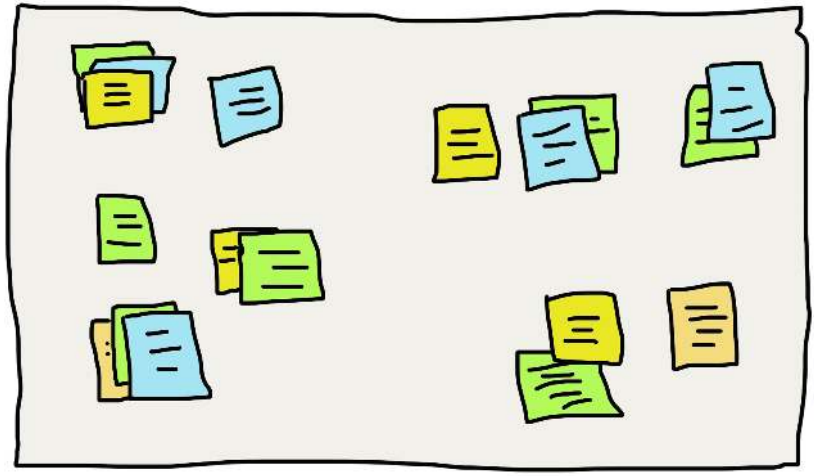
La reunión se lleva a cabo sobre una mesa lo suficientemente grande para todos los participantes. Cada uno toma varias fichas de una pila dispuesta en el centro de la mesa y escribe un rol en la misma. Siendo que esta actividad es un *Brainstorming* no debe haber discusión ni censura para cada rol que alguien escribe.

Una opción que suele funcionar muy bien es que los participantes anoten tantos roles como sea posible en sus fichas, en silencio y sin compartirlos con el resto de las personas.

Organización del conjunto inicial de roles

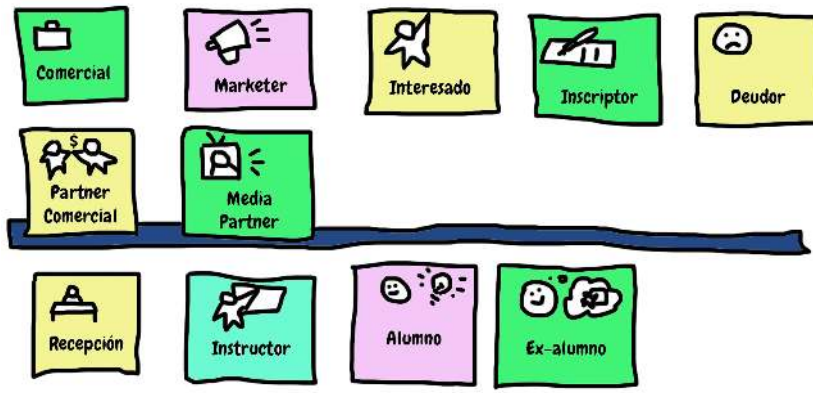
Una vez que el grupo haya terminado de identificar los roles, el próximo paso es organizarlos. Para esto, los dispondrá sobre la mesa de forma tal que las similitudes queden representadas de forma visual. Esto se logra solapando levemente aquellos roles que tienen pocas similitudes, solapando por completo aquellos que son iguales y separando los que no tienen relación.

Para poder llegar a ese resultado, 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. Para nuestro sistema bajo análisis, el resultado de este ejercicio fue algo como lo que se ve en el dibujo:



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. En nuestro ejemplo, luego de esta dinámica se llegó al siguiente resultado:



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, pero no hay una única manera de representar estas relaciones, y podemos experimentar con diferentes disposiciones.

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 uso de computadoras
- El nivel general de experiencia del usuario con el sistema
- Objetivo del usuario con la utilización del sistema

Descripción refinada de *algunos* roles:

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.

Marketer

Uso frecuente del sistema con conocimiento limitado del dominio del problema. Posee un nivel avanzado de experiencia en la utilización de computadoras y nivel intermedio de experiencia con el uso del sistema en particular y alto nivel de experiencia en el uso de redes sociales como Twitter y Facebook. Su responsabilidad será la de promover y difundir los eventos en internet.

Media Partner

Uso eventual del sistema con bajo conocimiento del dominio del problema. Posee un nivel avanzado de experiencia en la utilización de computadoras y nivel bajo de experiencia con el uso del sistema en particular y alto nivel de experiencia en

su sitio web. Su responsabilidad será la de difundir los eventos entre los usuarios de sus sitios web, realizar sorteos y proveer códigos de descuento. También debe conocer el estado de su cuenta en el caso de obtener beneficios económicos en base a referidos.

Interesado

Uso infrecuente del sistema sin conocimiento del dominio del problema. Se asumirá un nivel medio 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.

Persona a Inscribirse

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 es inscribirse a un determinado evento.

Empresa con Personas a Inscribir

Uso infrecuente del sistema sin conocimiento del dominio del problema. Se asumirá un nivel intermedio de experiencia en la utilización de computadoras y bajo nivel de experiencia con el uso del sistema en particular. Su interés es inscribirse a un determinado grupo de personas, todas de una misma empresa a un evento en particular.

Beneficiario de Empresa

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 es recibir información sobre los eventos a los que fue inscripto por una tercera persona.

Deudor

Uso infrecuente del sistema sin conocimiento del dominio del problema. Se asumirá un nivel medio de experiencia en la utilización de computadoras y bajo nivel de experiencia con el uso del sistema en particular. Su interés es la realización de los pagos pendientes para poder asistir al evento al cual está inscripto.

Recepcionista

Uso frecuente 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 recepción de los asistentes, la toma de asistencia y la autorización de participación a los mismos.

Instructor

Uso frecuente del sistema con gran conocimiento del dominio del problema. Posee un nivel intermedio a avanzado de experiencia en la utilización de computadoras y alto nivel de experiencia con el uso del sistema en particular. Su objetivo será la creación de tipos de eventos y la provisión de los contenidos, programas, lecturas y material asociado a cada uno de ellos. También será su responsabilidad la evaluación de los exámenes rendidos por los alumnos.

Alumno

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. Está interesado en acceder a los contenidos de los diferentes cursos o eventos a los cuales asiste, como así también en poder rendir los exámenes que cada curso requiera y obtener los correspondientes certificados de examen y asistencia.

Ex-Alumno

Ídem Alumno. Su interés es recibir información sobre nuevos eventos o cursos relacionados o correlativos a los cursos o eventos a los que ha asistido.

User 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.

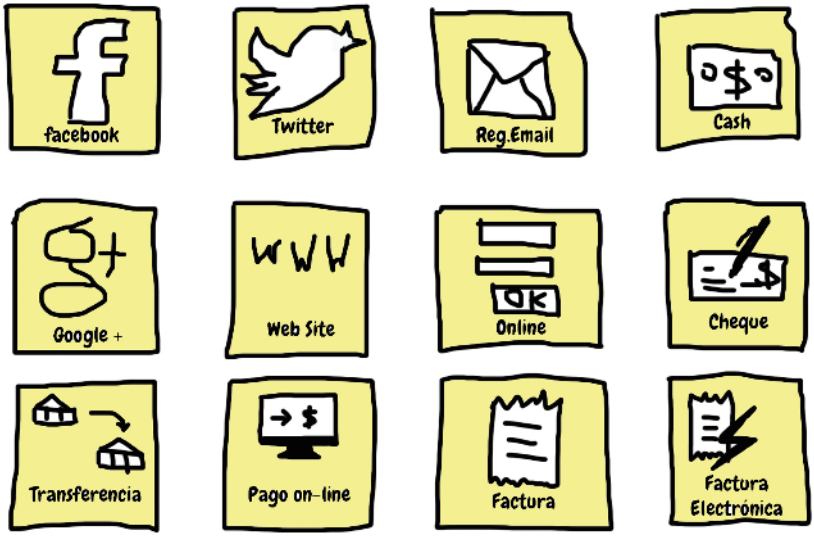
Los procesos de negocio identificados como parte de este taller fueron relativos a eventos de entrenamiento:

- Venta (Promoción y Registración)
- Cobranza y Facturación
- Logística y Evaluación

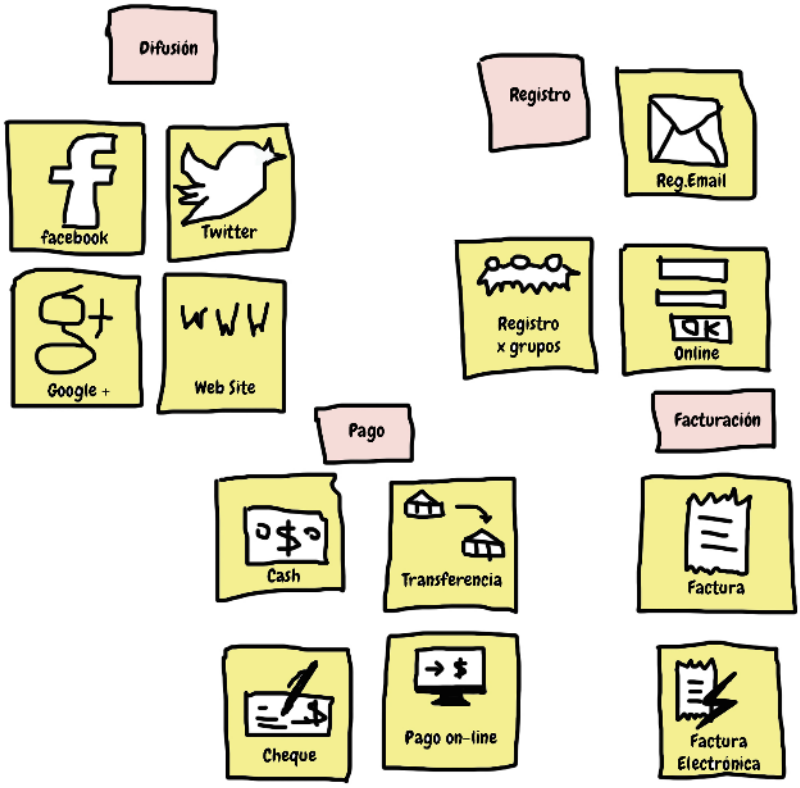
Identificación de Funcionalidades

Continuando con la práctica de User Story Mapping, 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 la ilustración vemos nuevamente *algunas* de las funcionalidades que se hallaron:

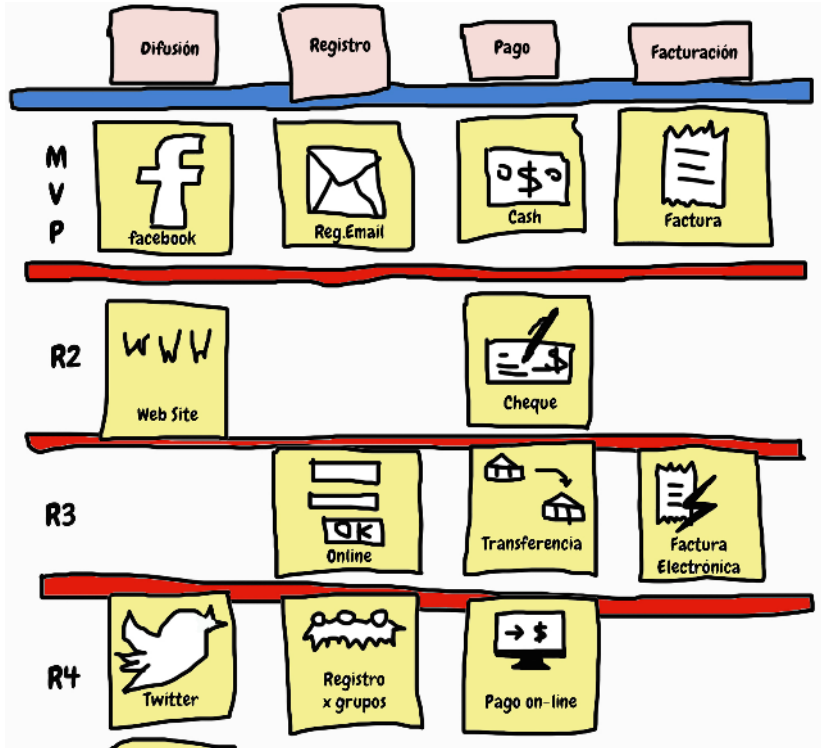


Una vez que se encontraron las funcionalidades, se agrupan según el proceso, identificando las actividades principales, por ejemplo:



Como en la imagen anterior, en cualquier momento puede aparecer una funcionalidad nueva que no habíamos tenido en cuenta (en el ejemplo, Registro por grupos).

El siguiente paso es usar esas actividades como títulos de columnas, y colocar debajo las funcionalidades, priorizándolas por lo que mayor valor de negocio aporta cada una en el corto plazo.



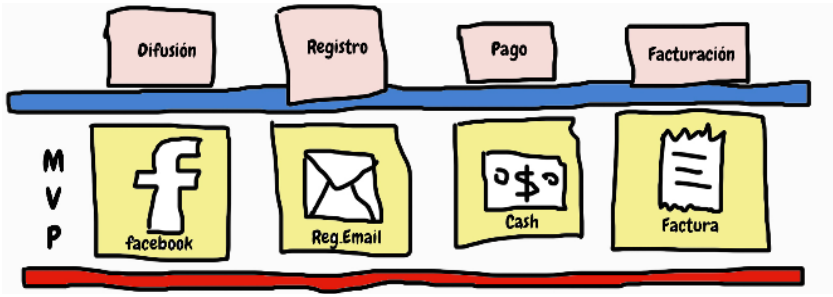
Por supuesto, la valoración tiene que ver con el escenario de negocio, y puede ser muy diferente dependiendo de la situación inicial.

En el caso de ejemplo, suponemos que partimos sin ninguna solución previa, así que buscaremos un plan que nos permita operar cuanto antes.

Identificación de MVP y posteriores entregas

Como hemos indicado anteriormente[29], 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 hallar el MVP, se discutirá la mínima funcionalidad que nos permite salir al mercado. En este caso, nuestro MVP ni siquiera implica desarrollar software:

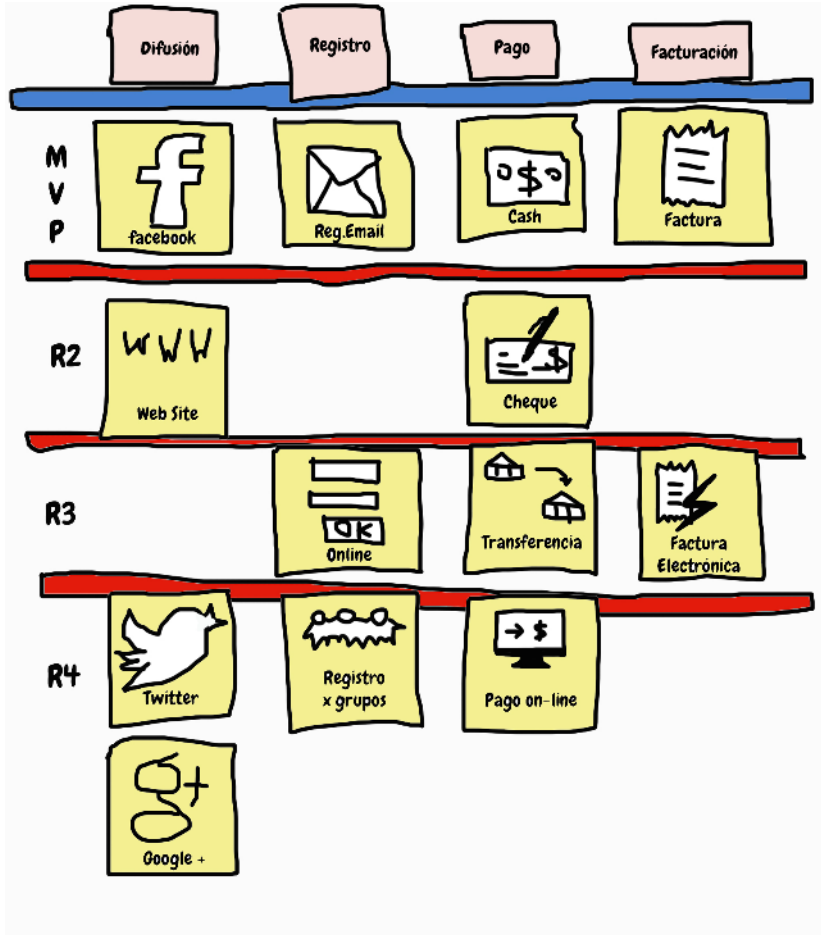


Como vemos, el MVP plantea publicar los eventos en Facebook, el registro de los asistentes se realizará por email, y el pago será inicialmente al contado, entregándose una factura física en el momento.

Esto no es una plataforma muy sofisticada, pero nos permite empezar a operar en un período muy corto, lo que implica aprender más temprano sobre el problema, antes de seguir tomando decisiones.

De todas maneras, tenemos un plan de más largo plazo, el que validaremos o ajustaremos en base al feedback de cada entrega. A continuación vemos cada una de las que se preveen en el Story Map final:

[29] Ver MVP, MMF y User Story Mapping



El plan implica que en la segunda entrega (R2 o Release 2), se construirá el sitio básico de promoción de los cursos, manteniendo aún el registro por e-mail, y agregando el pago con cheque como opción.

La siguiente entrega agregará al sitio un formulario para registro en línea, sumará el pago por transferencia bancaria, y la emisión de facturas electrónicas.

La última entrega de este plan sumará difusión por otras redes sociales, registro de grupos (más de una persona), y el pago en línea (a través de una plataforma de pagos externa).

4. Historias de Usuario

Valor: Software funcionando por sobre la documentación extensiva

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.

Agile Manifesto – 2001

Las **Historias de Usuario** surgieron en *eXtremme Programming (XP)* como una respuesta a una situación habitual en los proyectos de desarrollo do software: los clientes o especialistas de negocio se comunican con los equipos de desarrollo a través de extensos documentos conocidos como especificaciones funcionales. A su vez, las especificaciones funcionales son la documentación de supuestos y están sujetas a interpretaciones, lo que causa malos entendidos y que finalmente el software construido no se corresponda con la realidad esperada.

Una de las principales razones por las cuales la utilización de especificaciones detalladas como medio de comunicación no conduce a resultados satisfactorios es porque solo cubre una porción mínima (7%) del espectro de la comunicación humana: el contenido. Según Albert Mehrabian, la comunicación humana se compone de tres partes[30]:

1. En un 7%: El contenido (las palabras, lo dicho)
2. En un 38%: El tono de la voz
3. En un 55%: Las expresiones faciales

[30] "Silent messages: Implicit communication of emotions and attitudes.", Albert Mehrabian, 1981

Por esto se concluye que para tener una comunicación sólida, completa, es necesario el contacto cara-a-cara entre los interlocutores. En un esfuerzo orientado a que esas conversaciones existan, podemos decir que 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.

Componentes de una Historia de Usuario

Una Historia de Usuario se compone de 3 elementos, también conocidos como "las tres Cs"[31] 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*.

[31] "Essential XP: Card, Conversation, Confirmation", Ron Jeffries, 2001

Ilustración 26: Componentes de una historia de usuario

Redacción de una Historia de Usuario

Mike Cohn sugiere una determinada forma de redactar Historias de Usuario bajo el siguiente formato:

Como (rol) Necesito (funcionalidad) Para (beneficio)[1]

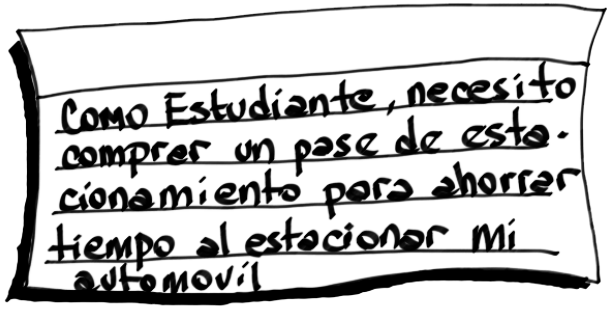


Ilustración 27: Redacción típica de una historia de usuario

Los beneficios de este tipo e 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.

INVEST - 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"[32]:

[32] "INVEST in Good Stories, and SMART Tasks", Bill Wake, 2003

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.

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 la performance 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.

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.

Estimable (E)

Una Historia de Usuario debería ser estimable. Mike Cohn[33], identifica tres razones principales por las cuales una Historia de Usuario no podría estimarse:

- **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.

- **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.

- **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

[33] "User Stories Applied", Mike Cohn, 2003

"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.

Pequeña (Small)

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.

Verificable (Testable)

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.

Ilustración 28: Características de una buena historia de usuario

Definición de Listo

También conocido como *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)

Ilustración 29: Definición de Listo (*Definition of Ready*)

Definición de Terminado

También conocido como *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
- En el caso de Product Owners muy exigentes: El Product Owner dio su visto bueno de la funcionalidad construida antes de llegar a la Review Meeting.

Ilustración 30: Definición de Terminado (*Definition of Done*)

Las Historias de Usuario de Nuestro Producto

A continuación redactamos las Historias de Usuario que comprenden el Product Backlog de nuestro producto. 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):

Entrega 1 - Comercializar Eventos				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
1	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

2	Comercial	Ver listado de eventos confirmados	No superponer eventos	- Mostrar Nombre, Ciudad y País - Muestra solo los futuros - Ordenado por fecha ascendente
3	Comercial	Modificar evento confirmado	Corregir cualquier error o re programarlo	- Permite modificar todos los campos.
4	Comercial	Cancelar evento confirmado	Dejar de seguirlo	- Desaparece del listado de eventos confirmados.
5	Comercial	Listar los eventos en un sitio web	Que los interesados puedan verlos	- Solo se listan los eventos públicos - Listado por fechas (a futuro) - Agrupados por Ciudad
6	Comercial	Publicar los detalles de cada evento	Que los interesados puedan verlos	- Accesible desde el listado de eventos - Muestra los detalles de cada evento: Nombre, Fecha, Descripción, Destinatarios, Programa, Instructor, Lugar, Ciudad, País, Precios y Promociones
7	Comercial	Generar un texto con fechas y valores	Pegarlo en los e-mail de respuesta	- Debe generarlo agrupando por ciudad, cursos, fechas y precios de cada uno.

8	Comercial	Dashboard de inscripciones a cursos	Conocer el estado de completitud de cada curso	- Muestra los eventos con colores: Rojo, Naranja, Amarillo y Verde. Los criterios son: → Un evento debe estar al 50% al menos 15 días antes → Un evento debe estar al 75% al menos una semana antes → Un evento debe estar al 100% dos días antes - La varianza sobre esos números alteran los colores: → Menos del 50% (Rojo) → Del 50% al 75% (Naranja) → Del 75% al 90% (Amarillo) → Del 90% al 100% (Verde)

9	Interesado	Pre-Inscribirme	Iniciar la reserva de mi vacante	Debe solicitar Nombre*, Apellido*, Teléfono de Contacto*, Email*, Empresa/ Carrera, Rol y Solicitar confirmación de que el asistente llevará notebook* si el curso lo requiere. * = obligatorio, el resto, opcional.
10	Comercial	Ser notificado de cada inscripción	Poder reaccionar en tiempo real frente a cada una	El email debe ser enviado a una dirección de correo configurable indicando los datos de contacto de la persona que realizó la inscripción.
11	Comercial	Confirmar la inscripción sin pago (pago a cuenta)	Financiar ciertas vacantes	Un pre-inscripto puede convertirse en inscripto sin haber realizado el pago.
12	Comercial	Conocer los Pagos Pendientes por evento	Realizar el seguimiento de los pagos	Listar los eventos con pagos pendientes y un detalle de las pre-inscripciones pendientes de pago por cada evento.
13	Interesado	Pagar en efectivo	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar en efectivo.
14	Interesado	Pagar con Cheque	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar con cheque.

15	Interesado	Pagar por Transferencia Bancaria	Confirmar mi vacante	Una vez que un interesado se pre-inscribe debe proporcionarle los datos para poder pagar por transferencia bancaria.
16	Comercial	Registrar los Pagos	Realizar el seguimiento de los pagos	Una pre-inscripción puede convertirse en inscripción registrando el pago realizado (fecha, monto y forma de pago).
17	Gestor de Cobranzas	Ser notificado del cobro de un evento	Realizar el seguimiento de los pagos	Cada vez que una pre-inscripción se convierte en inscripción, se debe enviar un e-mail a una casilla configurable.

Entrega 2 - Toma de evaluaciones on-line				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
18	Alumno	Responder Preguntas Multiple-Choice	Rendir el examen final	- Si el evento requiere un examen final, el alumno debería poder responder las preguntas on-line (solo multiple-choice) - Se deben poder administrar exámenes, preguntas y sus posibles respuestas. - Se asigna un tipo de examen a los alumnos seleccionados de un determinado evento.
19	Alumno	Un aviso de Finalización de Evaluación	Para saber que he finalizado	Se avisará en pantalla

20	Instructor	Que se realice la corrección automática de preguntas multiple-choice	Reducir mi carga de trabajo post-evento	- Todo examen debe tener un puntaje mínimo requerido (expresado en porcentaje) - Siendo preguntas multiple-choice, las correctas suman un punto, las incorrectas no suman ni restan.
21	Alumno	Recibir una notificación del resultado por e-mail	Para conocer el resultado de mi examen	- Se notifica por e-mail al alumno tan pronto finalice el examen.
22	Alumno	Generar mi certificado de evaluación aprobada	Presentarlo donde sea necesario	- El alumno podrá bajar un PDF son la constancia de su aprobación de examen
23	Instructor	Conocer los recuperatorios pendientes	Hacer seguimiento con los alumnos	- Para un determinado evento, se listarán los exámenes y recuperatorios pendientes.
24	Alumno	Conocer las preguntas erradas	Con el fin de saber dónde he fallado mi evaluación	- Se listarán las preguntas correctas con su explicación y las preguntas erradas, sin explicación.
25	Alumno	Recuperar las preguntas erradas	Con el fin de aprobar el examen	- Solo se realizará la respuesta de las preguntas erradas - Al finalizar el recuperatorio, aplican las mismas acciones que para un examen estándar: aviso de finalización, corrección automática y aviso de resultado

Entrega 3 - Pre-Inscripción Individual y Corrección de Exámenes a Desarrollar				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación

26	Interesado	Recibir un aviso de Pre-Inscripción y pasos siguientes	Poder confirmar mi pre-inscripción	- Al inscribirse, el alumno recibe por e-mail un instructivo sobre los pasos a seguir para efectivizar su inscripción.
27	Alumno	Responder Preguntas a Desarrollar	Poder rendir el examen	- Ciertas preguntas deberán solicitar un texto libre como respuesta.
28	Instructor	Listado de evaluaciones a corregir	Poder corregir evaluaciones	- Se deberán listar las evaluaciones con preguntas a desarrollar que estén pendientes de corrección
29	Instructor	Corrección manual de preguntas a desarrollar	Poder calificar a los alumnos	- Las respuestas de texto libre deberán ser corregidas manualmente por el instructor
30	Instructor	Feedback de corrección	Poder recomendar o sugerir acciones a los alumnos	- Al finalizar la corrección, el instructor podrá dar feedback de la evaluación por medio de un campo de texto libre.

Entrega 4 - Pre-Inscripción Corporativa y Seguimiento de Pagos				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
31	Empresa	Realizar una Pre-Inscripción corporativa	Inscribir varios empleados de una sola vez	Al inscribir se deberá solicitar responsable (mismos campos que pre-inscripción) y cantidad de inscriptos (hasta 10). - Luego, cada inscripto deberá tener Nombre, Apellido, E-mail y Número de Contacto.

32	Interesado	Recibir un Recordatorio de Evento	Alertarme sobre la proximidad del evento e informarme sobre los pormenores	- Enviado por e-mail al e-mail de contacto de cada inscripto (copiando a los responsables de inscripciones corporativas una sola vez). - Recordar horario, lugar y requisitos. Solicitar aviso en el caso de que el evento tenga almuerzo y el interesado tenga restricciones alimenticias. - Se debe enviar dos días antes del evento.
33	Responsable Logístico	Ser notificado sobre cupo alcanzado	A definir	A definir
34	Interesado	Ser notificado sobre el pago pendiente	Poder confirmar mi vacante a tiempo	- Se enviará un recordatorio de pago pendiente 48hs luego de la pre-inscripción, avisando que la misma vence en 24hs hábiles.
35	Administrativo	Obtener la información de facturación	Poder emitir las facturas correctamente	- Con cada Pre-Inscripción se solicitará información de facturación: Razón Social, Domicilio Fiscal, CUIT, Situación frente al IVA.
36	Interesado	Pagar por PayPal	Confirmar mi Vacante	- El sistema deberá proveer un link de pago de PayPal.
37	Interesado	Pagar por MercadoPago	Confirmar mi Vacante	- El sistema deberá proveer un link de pago de MercadoPago.

Entrega 5 - Logística de Eventos				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación

38	Responsable de Logística	Gestionar diferentes Tipos de Eventos	Crear checklists de cada tipo	- ABM de Tipos de Evento. Solo Nombre y Descripción.
39	Responsable de Logística	Gestionar diferentes modelos de Checklist	Que cada evento pueda instancias su checklist en base a un modelo pre armado	- Uno por cada tipo de evento
40	Responsable de Logística	Gestionar diferentes listados de Materiales	Saber qué se debe comprar por cada evento	- Un listado de materiales por cada tipo de evento.
41	Responsable de Logística	Hacer el seguimiento de cada Checklist de Evento	Que el mismo se realice de forma eficiente	- Poder marcar como “cumplido” los hitos de un checklist y dejar anotaciones (opcionales).
42	Responsable de Logística	Modificar los datos de un Checklist	Tener flexibilidad a la hora de gestionar un evento	- se podrán agregar o eliminar hitos de un checklist de evento particular.
43	Responsable de Logística	Ser notificado al modificar un checklist	Para estar al tanto de las modificaciones	- Se enviará un e-mail al responsable de logística con cada modificación de checklist (no incluye al avance del checklist de evento).
44	Responsable de Logística	Conocer los eventos y el progreso de checklist de cada uno	Para asegurar el correcto seguimiento de los checklists	- Listar los eventos y el porcentaje de avance de cada checklist
45	Responsable de Logística	Detalle de checklist de evento	Para asegurar el correcto seguimiento de los checklists	- Detallar el estado de cada checklist con hitos cumplidos y pendientes y fechas esperadas por cada uno.

Entrega 6 - Eventos Tentativos

Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
46	Partner Comercial	Crear evento tentativo	Proponer la realización del mismo	A definir
47	Comercial	Ser notificado de un nuevo evento tentativo	Realizar las acciones necesarias para la confirmación del mismo	A definir
48	Partner Comercial	Consultar agenda de eventos	Conocer las fechas y disponibilidad para crear eventos tentativos	A definir
49	Partner Comercial	Modificar evento tentativo	Realizar correcciones o reprogramar eventos tentativos	A definir
50	Partner Comercial	Cancelar evento tentativo	Dejar de seguirlo	A definir
51	Comercial	Ver listado de eventos tentativos	Tener un panorama de la planificación futura de eventos	A definir
52	Comercial	Confirmar evento tentativo	Transformarlo en un evento agendado y publicarlo.	A definir
53	Partner Comercial / Comercial / Instructor	Ser notificado sobre la confirmación de evento	Comenzar a comercializarlo	A definir
54	Comercial	Ver listado de eventos tentativos agrupados por Partner Comercial y/o Región	Tener un panorama de la planificación futura de eventos	A definir
55	Interesado	Obtener un brochure de cada evento	Evaluar la información con mayor detalle	A definir
56	Interesado	Pre-Inscribir un grupo de personas	Asistir varios a un mismo evento sin ser una organización	A definir

Entrega 7 – Integración con sistemas Externos				
Prio.	Como ...	Necesito ...	Para ...	Criterios de Aceptación
57	Partner Comercial	Consultar agenda de instructores	Conocer su disponibilidad	A definir
58	Comercial	Registrar evento tentativo en Google Calendar	Publicar su existencia a todos los suscriptos a dicho calendario	A definir
59	Comercial	Registrar evento confirmado en Google Calendar	Publicar su existencia a todos los suscriptos a dicho calendario	A definir
60	Responsable Financiero	Crear balance contable del evento en Google Docs	Comenzar a hacer el seguimiento financiero de un evento	A definir
61	Comercial	Publicar Evento en Twitter, Facebook & LinkedIn	Dar a conocer su existencia	A definir
62	Comercial	Difundir vía Mailchimp	Dar a conocer su existencia	A definir
63	Comercial	Difundir en forma masiva	Dar a conocer su existencia	A definir
64	Comercial	Difundir a leads comerciales	Dar a conocer su existencia	A definir
65	Administrativo	La generación de asiento contable de Cobro	Registrar el cobro con menor esfuerzo	A definir
66	Administrativo	Generar e Imprimir Factura	Reducir mi esfuerzo y probabilidad de error	A definir
67	Administrativo	Ver listado de Facturas	Conocer las facturas generadas	A definir
68	Administrativo	Entregar Factura	Realizar el cobro de un evento	A definir
69	Administrativo	Asentar Factura en Contabilidad	Reducir mi esfuerzo y probabilidad de error	A definir

5. Estimaciones Ágiles

Cono de la Incertidumbre

En gestión de proyectos, el cono de la incertidumbre describe la evolución de la incertidumbre durante la ejecución de un proyecto. Al comienzo, poco es conocido 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.

Muchos ambientes cambian tan lentamente que la incertidumbre reinante puede ser considerada constante (evolución estática) durante la duración de un proyecto típico. 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.

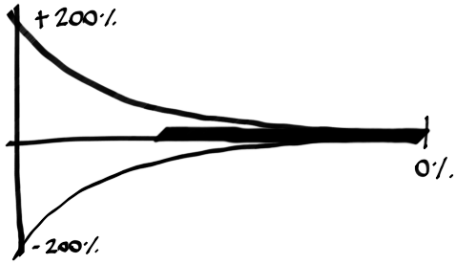


Ilustración 31: Cono de la Incertidumbre en un contexto estable

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%[34], esta incertidumbre tiende a decrementarse durante la evolución del proyecto, pero sin garantías de ello.

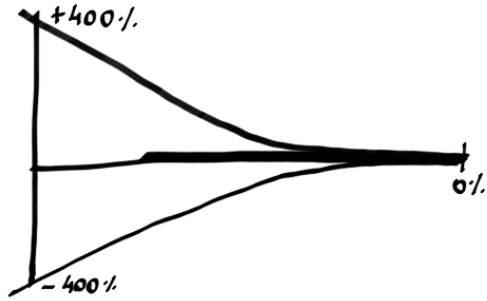


Ilustración 32: Cono de la Incertidumbre en un contexto inestable

Estimaciones en contextos inciertos

Como es de esperar según los gráficos previos, proveer una estimación precisa en etapas tempranas de un proyecto tiene como consecuencia un compromiso poco probable de ser cumplido.

[34] McConnell, S (2006) *Software Estimation: Demystifying the Black Art*, Microsoft Press.

A medida que adquiramos conocimiento, nuestras estimaciones se harán cada vez más precisas. El problema aparece a la hora de estimar cuando muchas de las decisiones se toman en base a supuestos que probablemente no sucedan o no sean los correctos.

"*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.*"[35]

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.

Escalas de PBIs y Estimaciones

Podemos enumerar la siguiente escala de PBIs y estimaciones:

- **Alto Nivel**: EPIC estimada en Tamaño (XS, S, M, L, XL)
- **Nivel Medio**: Historia de Usuario estimada en Puntos de Historia (Sucesión de Fibonacci[36])

[35] Fontela, Carlos (2007) *Estimaciones y Estadística*, Blog CyS Ingeniería de Software

[36] http://es.wikipedia.org/wiki/Sucesi%C3%B3n_de_Fibonacci

- **Bajo Nivel**: tareas o actividades estimadas en horas, preferiblemente menos de un día.

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*. A las Historias de Usuario las estimaremos utilizando la sucesión Fibonacci:

- 0, 1, 2, 3, 5, 8, 13, 21, 40, 100[37]

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.

[37] Con una leve deformación ya que se interrumpe la sucesión en el número 21 y se agregan luego los números 40 y 100

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.

Autores como Jeff Sutherland[38] expresan no estar de acuerdo con estimar a este bajo nivel, mientras otros como Mike Cohn[39] promueven su utilización. Esta situación da origen a dos modelos de planificación de Sprint: la planificación basada en velocidad (*velocity-based planning*) donde el Equipo se compromete a realizar tantos User Stories de modo que sumen una estimación en Puntos de Historia igual a la velocidad del Equipo, por un lado, y la planificación basada en compromisos (*commitment-based planning*) donde sin importar la velocidad, el Equipo revalúa las Historias de Usuario y se compromete en función de la estimación de cada una de ellas, por el otro. En este último caso, las Historias de Usuario involucradas deberían sumar una cantidad de Puntos de Historia aproximada a la Velocidad del Equipo. Dice Mike Cohn con respecto a la utilización de la Velocidad del Equipo como herramienta de planificación:

"*Supongamos que un equipo de básquetbol está en la mitad de su temporada. Han anotado un promedio de 98 puntos por cada partido de los 41 partidos jugados hasta el momento. Sería conveniente para ellos decir "Nosotros anotaremos un promedio de 98 puntos por partido por el resto de la temporada." Pero no deberían nunca decir antes de cualquier partido "Nuestro promedio es de 98 puntos, por lo*

[38] Jeff Sutherland (2010),"*Story Points: Why are they better than hours?*", http://scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html

[39] Mike Cohn (2007), "*Why I Don't use Story Points for Sprint Planning*", http://blog.mountaingoatsoftware.com

que se anotarán 98 puntos esta noche." Es por eso que afirmo que la velocidad es un predictor útil para el largo plazo, pero no lo es para el corto plazo."[40]

Métodos Delphi de Predicción y Estimación

El **Método Delphi** es una técnica creada por la Corporación RAND[41] 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

Basados en el Método Delphi, Barry Boehm y John Farquhar elaboraron en 1970 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

[40] *Ibid*

[41] La Corporación RAND (Research And Development) es un laboratorio de ideas (think tank) norteamericano formado, en un primer momento, para ofrecer investigación y análisis a las fuerzas armadas norteamericanas. (fuente: Wikipedia)

conocido método Delphi, esta técnica requiere de un mayor grado de interacción y discusión entre los participantes. Wideband Delphi fue popularizado en 1981 por Boehm en su libro "*Software Engineering Economics*" donde 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.

Planning Poker

James Greening presentó en su paper en 2002 llamado "*Planning Poker (o cómo evitar análisis parálisis en la planificación de liberaciones)*"[42] donde 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

[42] http://renaissancesoftware.net/files/articles/PlanningPoker-v1.1.pdf

del Equipo posee en sus manos una baraja de cartas con los números correspondientes a la sucesión de Fibonacci[43] 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.

Este método fue popularizado en 2005 por Mike Cohn en su libro "*Agile Estimating and Planning*".

La Sabiduría de las Multitudes (Wisdom of Crowds)

James Surowieki explica en su libro "La Sabiduría de las Multitudes" (2004): "*Normalmente solemos favorecer la opinión de los expertos, pues consideramos que sólo una persona con experiencia y conocimientos suficientes es capaz de emitir juicios correctos en un área o materia en particular. Sin embargo, hay evidencias de que*

[43] Se puede repasar en la sección de Escalas de PBIs y Estimaciones

las decisiones tomadas colectivamente por un grupo de personas suelen ser más atinadas que las decisiones tomadas sobre la base del conocimiento de un experto".

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, Surowieki recomienda en su tesis 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[44].

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

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.

[44] http://es.wikipedia.org/wiki/Pensamiento_de_grupo

- Intentar bajar dicha incertidumbre mediante el análisis puede llevarnos al "Análisis Parálisis"[45]. 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.

[45] http://en.wikipedia.org/wiki/Analysis_paralysis

6. Plan de Entregas (*Release Plan*)

A continuación se presentan las Historias de Usuario estimadas por el Equipo de Desarrollo, utilizando Planning Poker con Fibonacci y estimando una velocidad de Iteración de 15 puntos de historia con una duración de dos semanas:

Entrega 1 - Comercializar Eventos				
Prio.	Como ...	Necesito ...	Para ...	Estim.
Sprint 1 – Velocidad: 15 puntos				
1	Comercial	Crear un evento confirmado	Hacer el seguimiento del mismo	3
2	Comercial	Ver listado de eventos confirmados	No superponer eventos	2
3	Comercial	Modificar evento confirmado	Corregir cualquier error o re programarlo	2
4	Comercial	Cancelar evento confirmado	Dejar de seguirlo	1
5	Comercial	Listar los eventos en un sitio web	Que los interesados puedan verlos	2
6	Comercial	Publicar los detalles de cada evento	Que los interesados puedan verlos	5
Sprint 2 – Velocidad: 15 puntos				
7	Comercial	Generar un texto con fechas y valores	Pegarlo en los e-mail de respuesta	5
9	Interesado	Pre-Inscribirme	Iniciar la reserva de mi vacante	2
8	Comercial	Dashboard de inscripciones a cursos	Conocer el estado de completitud de cada curso	8
Sprint 3 – Velocidad: 14 puntos				

10	Comercial	Ser notificado de cada inscripción	Poder reaccionar en tiempo real frente a cada una	2
11	Comercial	Confirmar la inscripción sin pago (pago a cuenta)	Financiar ciertas vacantes	2
12	Comercial	Conocer los Pagos Pendientes por evento	Realizar el seguimiento de los pagos	3
13	Interesado	Pagar en efectivo	Confirmar mi vacante	1
14	Interesado	Pagar con Cheque	Confirmar mi vacante	1
15	Interesado	Pagar por Transferencia Bancaria	Confirmar mi vacante	1
16	Comercial	Registrar los Pagos	Realizar el seguimiento de los pagos	2
17	Gestor de Cobranzas	Ser notificado del cobro de un evento	Realizar el seguimiento de los pagos	2
Entrega 2 - Toma de evaluaciones on-line				
Prio.	Como ...	Necesito ...	Para ...	Estim.
Sprint 4 – Velocidad: 15 puntos				
18	Alumno	Responder Preguntas Multiple-Choice	Rendir el examen final	8
19	Alumno	Un aviso de Finalización de Evaluación	Para saber que he finalizado	2
20	Instructor	Que se realice la corrección automática de preguntas multiple-choice	Reducir mi carga de trabajo post-evento	5
Sprint 5 – Velocidad: 15 puntos				

21	Alumno	Recibir una notificación del resultado por e-mail	Para conocer el resultado de mi examen	2
22	Alumno	Generar mi certificado de evaluación aprobada	Presentarlo donde sea necesario	5
23	Instructor	Conocer los recuperatorios pendientes	Hacer seguimiento con los alumnos	3
25	Alumno	Recuperar las preguntas erradas	Con el fin de aprobar el examen	5
Sprint 6 – Velocidad: 15 puntos				
24	Alumno	Conocer las preguntas erradas	Con el fin de saber dónde he fallado mi evaluación	5
Entrega 3 - Pre-Inscripción Individual y Corrección de Exámenes a Desarrollar				
Prio.	Como ...	Necesito ...	Para ...	Estim.
26	Interesado	Recibir un aviso de Pre-Inscripción y pasos siguientes	Poder confirmar mi pre-inscripción	2
27	Alumno	Responder Preguntas a Desarrollar	Poder rendir el examen	8
Sprint 7 – Velocidad: 16 puntos				
28	Instructor	Listado de evaluaciones a corregir	Poder corregir evaluaciones	3
29	Instructor	Corrección manual de preguntas a desarrollar	Poder calificar a los alumnos	3
30	Instructor	Feedback de corrección	Poder recomendar o sugerir acciones a los alumnos	2
Entrega 4 - Pre-Inscripción Corporativa y Seguimiento de Pagos				
Prio.	Como ...	Necesito ...	Para ...	Estim.

31	Empresa	Realizar una Pre-Inscripción corporativa	Inscribir varios empleados de una sola vez	8
Sprint 8 – Velocidad: 15 puntos				
32	Interesado	Recibir un Recordatorio de Evento	Alertarme sobre la proximidad del evento e informarme sobre los pormenores	3
33	Responsable Logístico	Ser notificado sobre cupo alcanzado	A definir	2
34	Interesado	Ser notificado sobre el pago pendiente	Poder confirmar mi vacante a tiempo	3
35	Administrativo	Obtener la información de facturación	Poder emitir las facturas correctamente	2
36	Interesado	Pagar por PayPal	Confirmar mi Vacante	2
37	Interesado	Pagar por MercadoPago	Confirmar mi Vacante	3
Entrega 5 - Logística de Eventos				
Prio.	Como ...	Necesito ...	Para ...	Estim.
Sprint 9 – Velocidad: 15 puntos				
38	Responsable de Logística	Gestionar diferentes Tipos de Eventos	Crear checklists de cada tipo	2
39	Responsable de Logística	Gestionar diferentes modelos de Checklist	Que cada evento pueda instancias su checklist en base a un modelo pre armado	8
40	Responsable de Logística	Gestionar diferentes listados de Materiales	Saber qué se debe comprar por cada evento	5
Sprint 10 – Velocidad: 15 puntos				

41	Responsable de Logística	Hacer el seguimiento de cada Checklist de Evento	Que el mismo se realice de forma eficiente	5
42	Responsable de Logística	Modificar los datos de un Checklist	Tener flexibilidad a la hora de gestionar un evento	8
43	Responsable de Logística	Ser notificado al modificar un checklist	Para estar al tanto de las modificaciones	2
Sprint 11 – Velocidad: 15 puntos				
44	Responsable de Logística	Conocer los eventos y el progreso de checklist de cada uno	Para asegurar el correcto seguimiento de los checklists	3
45	Responsable de Logística	Detalle de checklist de evento	Para asegurar el correcto seguimiento de los checklists	3
Entrega 6 - Eventos Tentativos				
Prio.	Como ...	Necesito ...	Para ...	Estim.
46	Partner Comercial	Crear evento tentativo	Proponer la realización del mismo	3
47	Comercial	Ser notificado de un nuevo evento tentativo	Realizar las acciones necesarias para la confirmación del mismo	2
48	Partner Comercial	Consultar agenda de eventos	Conocer las fechas y disponibilidad para crear eventos tentativos	3
49	Partner Comercial	Modificar evento tentativo	Realizar correcciones o reprogramar eventos tentativos	1

Sprint 12 – Velocidad: 16 puntos				
50	Partner Comercial	Cancelar evento tentativo	Dejar de seguirlo	2
51	Comercial	Ver listado de eventos tentativos	Tener un panorama de la planificación futura de eventos	2
52	Comercial	Confirmar evento tentativo	Transformarlo en un evento agendado y publicarlo.	2
53	Partner Comercial / Comercial / Instructor	Ser notificado sobre la confirmación de evento	Comenzar a comercializarlo	2
54	Comercial	Ver listado de eventos tentativos agrupados por Partner Comercial y/o Región	Tener un panorama de la planificación futura de eventos	3
55	Interesado	Obtener un brochure de cada evento	Evaluar la información con mayor detalle	5
Sprint 13 – Velocidad: 16 puntos				
56	Interesado	Pre-Inscribir un grupo de personas	Asistir varios a un mismo evento sin ser una organización	8
Entrega 7 – Integración con sistemas Externos				
Prio.	Como ...	Necesito ...	Para ...	Estim.
57	Partner Comercial	Consultar agenda de instructores	Conocer su disponibilidad	3
58	Comercial	Registrar evento tentativo en Google Calendar	Publicar su existencia a todos los suscriptos a dicho calendario	5
Sprint 14 – Velocidad: 15 puntos				

59	Comercial	Registrar evento confirmado en Google Calendar	Publicar su existencia a todos los suscriptos a dicho calendario	5
60	Responsable Financiero	Crear balance contable del evento en Google Docs	Comenzar a hacer el seguimiento financiero de un evento	5
61	Comercial	Publicar Evento en Twitter, Facebook & LinkedIn	Dar a conocer su existencia	5
Sprint 15 – Velocidad: 15 puntos				
62	Comercial	Difundir vía Mailchimp	Dar a conocer su existencia	5
63	Comercial	Difundir en forma masiva	Dar a conocer su existencia	5
64	Comercial	Difundir a leads comerciales	Dar a conocer su existencia	5
Sprint 16 – Velocidad: 15 puntos				
65	Administrativo	La generación de asiento contable de Cobro	Registrar el cobro con menor esfuerzo	8
66	Administrativo	Generar e Imprimir Factura	Reducir mi esfuerzo y probabilidad de error	5
67	Administrativo	Ver listado de Facturas	Conocer las facturas generadas	3
Sprint 17 – Velocidad: 11 puntos				
68	Administrativo	Entregar Factura	Realizar el cobro de un evento	3
69	Administrativo	Asentar Factura en Contabilidad	Reducir mi esfuerzo y probabilidad de error	8

Duración del Proyecto

Etapa	Duración	Desde	Hasta
Incepción	2 semanas	3-Oct-2011	14-Oct-2011
Entrega 1 - Comercializar Eventos			
Sprint 1	2 semanas	17-Oct-2011	28-Oct-2011
Sprint 2	2 semanas	31-Oct-2011	11-Nov-2011
Sprint 3	2 semanas	14-Nov-2011	25-Nov-2011
Entrega 2 - Toma de evaluaciones on-line			
Sprint 4	2 semanas	28-Nov-2011	9-Dic-2011
Sprint 5	2 semanas	12-Dic-2011	23-Dic-2011
Sprint 6	2 semanas	26-Dic-2011	6-Ene-2012
Entrega 3 - Pre-Inscripción Individual y Corrección de Exámenes a Desarrollar			
Sprint 7	2 semanas	9-Ene-2012	20-Ene-2012
Entrega 4 - Pre-Inscripción Corporativa y Seguimiento de Pagos			
Sprint 8	2 semanas	23-Ene-2012	3-Feb-2012
Entrega 5 - Logística de Eventos			
Sprint 9	2 semanas	6-Feb-2012	17-Feb-2012
Sprint 10	2 semanas	20-Feb-2012	2-Mar-2012
Sprint 11	2 semanas	5-Mar-2012	16-mar-2012
Entrega 6 - Eventos Tentativos			
Sprint 12	2 semanas	19-Mar-2012	30-mar-2012
Sprint 13	2 semanas	2-Abr-2012	13-Abr-2012
Entrega 7 – Integración con sistemas Externos			
Sprint 14	2 semanas	16-Abr-2012	27-Abr-2012
Sprint 15	2 semanas	30-Abr-2012	11-May-2012
Sprint 16	2 semanas	14-May-2012	25-May-2012
Sprint 17	2 semanas	28-May-2012	8-Jun-2012

7. Incepción Ágil

¿Cómo se inicia un proyecto ágil?

Como vimos en el Capítulo 2, la construcción de un producto usando Scrum se inicia con el Product Backlog, con sus ítems priorizados, que irá refinándose y evolucionando en el tiempo. Scrum no aclara de dónde o cómo surge esa primera versión del backlog. Hemos visto en el capítulo 3 la técnica de User Story Mapping, pero incluso previo a eso necesitamos consensuar la visión general y las restricciones básicas con las que queremos trabajar.

Una de las prácticas más populares en los últimos años en la comunidad ágil para generar esta visión común, previa al Story Mapping, es la **Incepción Ágil**[46].

A continuación vamos a recorrer cada una de las 10 actividades usuales al facilitar esta actividad, describiendo nuestro estilo de facilitarlas, que puede diferir de otras personas en la comunidad.

¿Cuándo y cómo se realiza una Incepción Ágil?

Para empezar, el formato que recomendamos es el de un taller colaborativo, y el foco es que estén presentes (en un mismo lugar físico) todas las personas fuertemente involucradas en el problema a discutir (ya que tal vez no esté claro si se convertirá en un proyecto, varios, o nada).

Lo ideal es lograr un buen nivel de compromiso desde los patrocinadores del proyecto, o las personas que tienen responsabilidad sobre el problema que queremos resolver. Esto es importante para poder convocar también a la gente que conoce el problema (tal vez desde un

[46] The Agile Samurai, Jonathan Rasmusson, 2010

punto más operativo), otros (sectores o áreas) relacionados fuertemente o quienes estarán potencialmente involucrados en la implementación o soporte (por ejemplo, gente de tecnología, desarrollo, diseño).

Por otro lado, es clave el rol del facilitador del taller, que debe entender bien el formato general y las actividades individuales, poder moderar discusiones que se desborden, mantener el ritmo de la reunión, asistir las necesidades de los equipos, y otras tareas generales. En organizaciones donde ya hay gente que actúa como Scrum Master o Coach, ellos suelen ser los más indicados. Pero también se puede contar con algún entusiasta que idealmente haya participado en una Incepción previa.

La duración también depende del nivel de importancia del proyecto/problema que vamos a tratar. Puede variar entre medio día y dos días completos.

En la práctica se ven Incepciones desde medio día con menos de 10 personas, hasta un día y medio con casi 20. No hay una regla muy específica a aplicar, y cada organización y contexto debe encontrar su punto. Es preferible reservar más tiempo del esperado, y terminar antes a quedarse "corto" y que los participantes se dispersen sin haber terminado.

El estilo de la reunión

Algo que creemos fundamental es el estilo que le damos a este evento. Preferimos contar con un espacio abierto para poder moverse, algunas mesas para trabajar en grupos de 4 o 5 personas, y sobre todo muchas paredes para poder ir pegando resultados de las actividades. La mayor parte del tiempo lo que se va generando son láminas con diferentes visiones del problema/proyecto, usando colores, tijeras, materiales varios, y en general un tipo entrañables livianos, poco formales, y que requieren trabajo manual.

Parte del secreto es que al trabajar con las manos, haciendo dibujos o armando cosas con las manos, generamos un ambiente en el que las jerarquías tienden a borrarse, impulsando mayor participación y una discusión más abierta, que al contrario de la formalidad excesiva, tiende a sacar a la luz mucho más fácil, pero sin tanto riesgo, montones de temas críticos.

Las 10 actividades de una Incepción Ágil

El motivo de la reunión

La primer actividad en un taller de Incepción es una ronda en la cada uno se presenta y comenta para qué cree que está en el taller.

En todas las actividades el facilitador debe mantener el timebox, es decir, aclarar que cada uno tiene un tiempo acotado (en este caso podría ser uno o dos minutos por persona), y tratar de ser claro en lo que esperamos de cada participante, por ejemplo:

- Nuestro nombre y perfil (rol, área ó especialidad)
- Quién nos convocó
- Lo que creemos que podemos aportar en esta reunión

Al terminar la ronda, podemos hacer que cada uno escriba brevemente en un post-it cuál cree que es el objetivo principal de la Incepción: ¿qué problema queremos resolver?

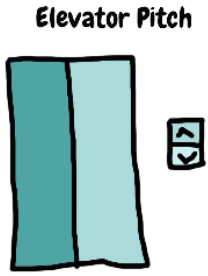
Nuevamente ponemos un timebox de 2 a 3 minutos, y después pegamos todo en una hoja, agrupamos los que son iguales o similares, y discutimos brevemente las ideas que son muy disimiles.

El objetivo central es que lleguemos a tener claro entre todos el problema que queremos resolver, sin que nadie "se lo comunique" al resto. Queremos que surjan las inconsistencias o diferencias de nivel de abstracción que haya entre los asistentes.

Otro tema importante: si entre la ronda y la definición final del foco descubrimos que alguno de los participantes probablemente no tenga mucho que aportar, ni le podamos aportar el resto, lo liberamos, agradeciéndole haber venido, y manteniendo un contacto por si descubriésemos que hay algún tema puntual para consultarle. Esto es algo que debe manejar el facilitador para que suceda sin conflictos. No queremos que nadie sienta que su participación "no tiene valor" per se, pero tampoco queremos que haya asistentes que no estén realmente involucrados con el foco de esta Incepción.

Visión de Alto Nivel

En el segundo paso de nuestra Incepción de Proyectos vamos a tratar de condensar la visión de cómo resolver el problema que nos convocó a muy alto nivel, pero entrando en algunos primeros detalles que podamos discutir abiertamente, y nos sirvan para seguir la conversación.



El “Elevator Pitch” es un término que viene del mundo del marketing de los años 60~70, y la idea es tener un argumento tan bien preparado y condensado, que pueda usarse para “vender” una idea a alguien al encontrarlo en el ascensor, aprovechando el escaso tiempo de un piso al otro.

Aunque es probable que en el caso de nuestro proyecto no necesitemos “vender” la idea más allá del grupo de la Incepción (o si, aún más tarde, cuando queremos enrolar a alguien más en el proyecto), utilizamos su estructura porque destaca una serie de elementos que nos resultan útiles discutir.

Nuevamente, es tarea del facilitador dividir a la audiencia en grupos de tres o cuatro personas, con papel y lápiz, para crear diferentes versiones del “Elevator Pitch” dentro de un time box de 5~10 minutos, que después presentaremos y discutiremos entre todos.

Una ayuda es mostrar un modelo posible, sobre todo para resaltar los componentes que buscamos incluir:

Para [cliente | público]

que tiene [necesidad | oportunidad]

[nombre producto] es un [tipo de producto]

que [beneficio | razón de compra]

A diferencia de [principal competidor | alternativa]

nuestro producto [diferencial competitivo]

Como se ve, todavía tiene muchos elementos de marketing, pero los elementos están ahí. Si se trata de un proyecto de desarrollo, podemos pensar quién es nuestra audiencia principal, su necesidad, que categoría de solución vamos a darle, cuál será el principal beneficio, cómo se diferencia de lo que se utiliza actualmente o alternativas que ya existan en el mercado, y así.

Suelo resaltar que no queremos listas de características. Es importante que podamos leer el resultado en 20 a 30 segundos. Los detalles vendrán después. Queremos por ahora sólo lo más importante.

Esta actividad finaliza con la discusión abierta y una reelaboración de la que obtenemos una sola frase consolidada, o unas pocas alternativas principales.

Manos a la obra

En esta actividad vamos a ir más allá de la visión preliminar del paso anterior usando una metáfora un poco más arriesgada, pero también mucho más divertida.

Usualmente para esta actividad consigo cajas en blanco, o cajas de un tamaño razonable (desde cereales hasta cajas de zapatos) y las cubro con papel blanco.

Nuevamente generamos grupos de trabajo (que idealmente pueden ir variando entre actividades para maximizar los cruces de opiniones) y le damos a cada uno una caja en blanco, marcadores y lápices de colores, papeles o post-its de colores, cintas, pegamento y otros materiales que sirvan para trabajar.

¿Cuál es el desafío?

Diseñar una caja que represente el proyecto que estamos encarando, como si fuese un producto de supermercado. La caja deberá ser atractiva, destacar las características principales sin apabullarnos, detallar en algún lado los componentes o características en más detalle, y más.

Entre otras cosas, usualmente este ejercicio hace que los equipos pongan un nombre al proyecto/producto, si no lo tiene.

Y es importante destacar que hacemos este ejercicio con portales web, sistemas de seguros, de salud, campañas de marketing, proyectos urbanos y montones de cosas que **nada** tienen que ver con un producto de consumo de masivo que se vende en caja. Estamos jugando con una metáfora.

¿Para que sirve, entonces?

Desde mi punto de vista, como muchos de estos ejercicios, nos sacan de nuestra zona de confort y nos exigen conceptualizar a un nivel diferente del que estamos acostumbrados, desatando más nuestra creatividad. A veces, la exageración aporta más visibilidad a ciertos temas y facilita la discusión.

Por otro lado, el hecho de trabajar en esta etapa temprana en una actividad muy manual, donde todos dibujan, recortan y pegan papeles, lleva a los grupos a un nivel de diversión y colaboración que cambia el tono de la reunión, reduciendo fronteras jerárquicas y de especialidades.

Al terminar sus cajas, como siempre dentro de un timebox (de 15 a 20 minutos), los equipos hacen una recorrida mirando las de los demás, y pueden votar por la más significativa, o dedicarle un rato a producir una con una versión conjunta.

Uno de los secretos de esta actividad es la cantidad de risas y entusiasmo que se genera. Es común que algunos grupos terminen trabajando en el suelo, o que se atrevan a utilizar el humor mucho más allá de otros espacios más formales.

Resultados a largo plazo

Aunque esta actividad parece tan inocente y lúdica, suele ser una de las que tiene enormes efectos a largo plazo.

Como facilitador, muchas veces no lo comento en el momento, pero lo verifico y aprovecho si tengo participación en el proyecto a largo plazo. Alguno efectos interesantes son:

- El nombre que inventan a veces para la caja se convierte en el nombre de código del proyecto. Así, por ejemplo, proyectos como el "nuevo servicio de atención para reclamos de siniestros", termina llamándose "Venecia" (por ejemplo).

- Una o varias de las cajas perduran y terminan colgadas o pegadas en el espacio del equipo. He escuchado comentarios de miembros del equipo explicándole a otra persona, con entusiasmo, por ejemplo: "¡y este dibujo de aquí lo hizo González, el Gerente de Canales!". Conexión del equipo con el negocio: altísima.

- Es común que los miembros de los equipos, al estar mezclados, hayan generado un lazo diferente en ese lapso tan breve, que les permite comunicarse entre ellos mucho más fácilmente durante el proyecto, porque comparten desde entonces un objetivo común que quedó plasmado en ensuciase juntos los dedos.

Limitando el alcance

La siguiente actividad es relativamente sencilla de realizar, pero suele generar discusiones fundamentales, y tiene que ver con limitar ciertas decisiones respecto al alcance.

Como siempre, dividimos en grupos a los participantes, y en cada grupo generamos una lista de características ordenadas en tres grupos:

- Las cosas que definitivamente queremos **dentro** del alcance
- Las que estamos de acuerdo que quedan **fuera**
- Las cosas que no podemos decidir (al menos por ahora)

Como siempre, al presentar los resultados parciales de cada equipo aparecerán inconsistencias que resolver.

En general prefiero ir anotando los puntos de conflicto, en lugar de discutirlos en el momento, y seguir avanzando en la comparación. Una vez que comparamos todos los resultados, podemos recorrer la lista de los conflictos y tratar de zanjarlos.

Usualmente algunos de los puntos "indefinidos" para un grupo se logran definir por si o por no entre el resto, pero siempre pueden quedar algunos que específicamente quedarán a profundizar una vez que el proyecto arranque realmente. Lo bueno es que están identificados.

La lista resultante nos sirve de input para poder comenzar a construir el backlog cuando el proyecto comience, y la lista de las cosas que están fuera de alcance nos servirá a lo largo del proyecto como filtro de alerta cuando se quieran agregar cosas al backlog. Por supuesto, no implica un rechazo automático, pero si debería plantearse siempre una discusión al respecto.

Dime con quien andas…

El siguiente paso es trabajar sobre los diferentes involucrados en el proyecto, y su nivel de participación.

Para esto volvemos a dividir en grupos a la audiencia y dejamos que trabajen en establecer su propio mapa. Suelo usar círculos concéntricos y post-its, o pueden utilizar un vector u otra representación.

La idea es que cada grupo piense en los diferentes "stakeholders", ya sean individuos con nombre y apellido, roles, sectores o áreas de una organización, otras organizaciones, instituciones o dependencias gubernamentales.

De alguna manera lo que tratamos de manera es que tanto impacto tienen sobre el proyecto. Este mapa de influencias permitirá que pensemos cada cuánto tiempo el equipo de proyecto tendrá que tener contacto con estos actores, e incluso si necesita convocar a algunos de ellos para alguna actividad posterior a Incepción, pero también fundacional del proyecto, como un taller de User Story Mapping (como mencionamos en el capítulo 3).

Como en los otros casos, al terminar revisamos los diferentes mapas y tratamos de consensuar el o los más representativos.

¿Y entonces qué hacemos?

Esta actividad de la Incepción se puede realizar en grupos si tenemos gente con buena visión técnica como para ayudar en conceptualizar la solución a desarrollar.

Cuando digo “técnica” me refiero al punto de vista de lo que vamos a construir (si es un sistema, un arquitecto o líder técnico; si es de marketing o publicidad, un diseñador o experto en campañas; si es un proceso, alguien experto en ese área).

Si en la audiencia contamos con una sola persona con esas características, podemos realizar esta actividad todos juntos. La idea es que en grupo, y con la ayuda del experto, analizamos el tipo de solución a construir, a muy alto nivel.

Por ejemplo, supongamos que el equipo elabora un diagrama de componentes para un sistema, indicando que va a haber un servidor, que hablará con una aplicación existente, que conecta con tales bases de datos e integra los servicios X, Y y Z...

Tal vez aparezcan detalles de seguridad, o se sugiera mover algo a una nube, o utilizar algún tipo de tecnología, y lo importante de exponerlo ante todos es que surjan dudas como:

- ¿Y eso funciona en un Iphone?
- No se si conviene conectar con Siebel... porque que si el proyecto del CRM se demora, nosotros también.
- Antes de poner esos datos en la nube, deberíamos consultar con legales ¿Podemos dejar eso en una base local y mover lo demás?

- Creo que en el proyecto ZYX ya resolvieron esa interconexión ¿podemos pedirles ayuda?

Como antes, no buscamos que este diagrama o visión sea definitiva. Queremos explicar algunos detalles de cómo sería el proyecto para que la gente de otras áreas, técnicas, de negocio o administrativas, vean de qué se trata y comenten cualquier problema o relación que no hayamos tenido en cuenta.

¿Qué nos quita el sueño?

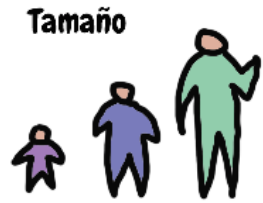
Llegamos a la actividad donde nos dividimos en grupos y nos ponemos todos el sombrero negro, pensando en todo lo que puede salir mal.

Trataremos de discutir todo lo que potencialmente podría quitarnos el sueño durante el proyecto, desde todos los puntos de vista. La gente de tecnología aportará riesgos de conocimiento del equipo, de complejidad tecnológica, de requisitos difíciles por distintos temas; la gente de seguridad pensará en las vulnerabilidades, potenciales brechas, criticidad de datos; la gente de negocio analizará costos, oportunidades, dependencias con otras áreas, necesidad de flexibilidad; y así diferentes actores expondrán miedos respecto al inicio, desarrollo, puesta en marcha, ejecución, mantenimiento y evolución del proyecto.

Al finalizar, cada grupo comparte los resultados y volvemos a tener una discusión abierta en la que tratamos de consensuar cuáles son los principales problema potenciales, y anotamos algunas de las medidas con las que podríamos mitigarlos.

Estimación Global

Ahora que ya tenemos una idea general del alcance, el tipo de solución a construir, y los riesgos principales que anticipamos, podemos empezar a pensar qué tan grande será el proyecto.



Nuevamente dividimos equipos (recordemos que lo ideal es combinar gente diferente a lo largo de la Incepción) y cada grupo hace una estimación de alto nivel, teniendo en cuenta:

- ¿Cuánta gente necesitamos en el equipo? ¿Cómo debería estar compuesto?
- Si vamos a ejecutar con Scrum, podemos pensar quienes serían el Scrum Master y el Product Owner
- Dado ese equipo que imaginamos ¿Cuánto tiempo duraría el proyecto?
- Podemos pensar alternativas de equipos y fases, e incluso alternativas de solución, para diferentes extensiones de proyecto.

Aquí cabe una aclaración: el equipo final **deberá revisar** la estimación, pero como estamos pensando un proyecto ágil, lo importante es que sabemos que en el tiempo que pensemos no esperamos resolver necesariamente todo el alcance, si no lo más valioso del alcance, por lo que aceptamos que no sabemos el detalle total. Es importante que todos los participantes comprendan que **no están tomando una decisión,** sino realizando un ejercicio para analizar lo que el proyecto implica.

Al finalizar, y una vez que cada grupo expuso su estimación, se discuten y seleccionan las mejores alternativas, para usarlas como referencia, y probablemente tomemos un márgen de unas a otras.

El objetivo que buscamos es tener un rango general:

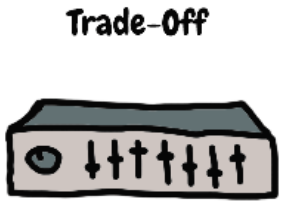
- ¿Será un solo equipo de 5~6 personas?
- ¿Serán 4 equipos de 8~9?
- ¿Podemos tener resultados en 6 semanas, 6 meses, 2 años?
- ¿Tenemos una idea al menos general de qué es lo más valioso resolver en cada etapa?

Aunque suene insistente, esto nos sirve para discutir si el proyecto es viable o no según estos parámetros, y eventualmente revisar algunos de los ejercicios anteriores. Como esto es una Incepción **Ágil**, sabemos que todo va a tener variaciones cuando el proyecto comience, y sobre todo cuando se entreguen los primeros resultados y el proceso de aprendizaje se potencie.

Hablemos de prioridades

Como mencionamos antes, sabemos que nuestra Incepción va a darnos una idea general, pero que el proyecto va a ir mutando y adaptándose en el tiempo para lograr el mejor resultado posible.

Si embargo, podemos tratar de seleccionar algunas restricciones tempranas que nos ayuden a pensar si los cambios o novedades que aparezcan nos están alejando o no de la idea inicial. Y podremos en ese momento decidir que no es un problema, o incluso re-evaluar estas prioridades, pero lo bueno es tener más elementos para evaluar esas decisiones.



Para esta actividad los equipos se dividen en grupos nuevamente y piensan en una lista de requisitos no-funcionales o preocupaciones transversales al proyecto, como por ejemplo:

- Costo de operación del producto final
- Mantenibilidad
- Time-to-market (cuanto antes tengamos algo, mejor)
- Interoperabilidad
- Equipo propio (o local)
- Seguridad o Confidencialidad
- Tiempo de respuesta
- Cumplimiento de regulaciones
- Cobertura desde múltiples dispositivos

...o muchos otros, desde características de calidad, temas de mercado, cuestiones internas o lo que nos preocupe.

Lo importante es que cada grupo quede con los 5 a 10 temas que les parecen más importantes tener en cuenta.

El siguiente paso es priorizarlos. Para eso prefiero una variante que aprendí de Alan y Ariel, que es seleccionar entre todos los que queremos priorizar, y a continuación elegir un participante por cada uno de los atributos a priorizar, que se pone de pie con una hoja de papel en la mano con el nombre de ese atributo. Esas personas se ponen en fila, y el resto, mirándolos de lado, de manera de verlos a todos, va intercalando con ellos, haciendo que se muevan hacia adelante o atrás en la fila, hasta que llegamos a un acuerdo.

Idealmente, al llegar al orden final, tomamos una foto de todos sosteniendo su cartel bien visible.

Esta actividad es más divertida con ese componente físico, y tiene también el valor testimonial de la foto con mucha gente involucrada, que perdura y queda para el equipo de proyecto. Muchas veces al ver las personas que participaron en esa priorización el mensaje es reforzado.

Insisto: esas prioridades **no son inamovibles**, pero tenerlas como filtro nos permite tener las discusiones necesarias a tiempo, y nos da contexto.

La pregunta del millón

Y finalmente llegamos a la última actividad.

¿Parece largo? En realidad muchas de las actividades pueden regularse para que avancen rápido ó podemos llegar a saltear algunas si tenemos menos tiempo. Eso es parte de la decisión del facilitador.

Mi recomendación es tratar de recorrer las 10 actividades, aunque sea dedicándole 20 a 30 minutos en promedio a cada una, para que entre en una mañana o una tarde.

En esta última actividad vamos a tratar de tener una estimación de **muy** alto nivel sobre el costo del proyecto. Casi nunca llegamos a un número, sino a un nivel de inversión, como para responder dudas como (dependiendo del tipo de proyecto y organización):

- ¿Podemos encarar esto con el presupuesto que ya teníamos?
- ¿Necesitamos conseguir un inversor?
- ¿Qué chances hay de que el proyecto se auto-financie a partir de X entrega?
- ¿Hace falta conseguir aprobación presupuestaria del Gerente del Área? ¿Del Directorio?

Para poder llegar a alguna conclusión de ese tipo, lo que hacemos en grupos es una "lista de compras", en la que incluimos todos los costos importantes que pueden incidir, como:

- Costo del equipo (si son internos y tenemos que costearlos, o si debemos contratar)
- Equipamiento, espacio físico, licencias de software, suscripciones

- Homologaciones, certificación o auditorías necesarias
- Espacio físico o instalaciones necesarias durante el proyecto

Nuevamente, cada grupo presenta su lista, y se discute en conjunto hasta lograr una versión unificada. No es importante ponerle valores, más allá de una estimación muy gruesa para llegar a entender el nivel de presupuesto, pero esa lista será el punto de partida para empezar a elaborar el presupuesto definitivo cuando el proyecto arranque realmente.

Bibliografía Recomendada

1. ADZIC, G., 2012, *Impact Mapping: Making a big impact with software products and projects*, Provoking Thoughts

2. COHN, M., 2004, *User Stories Applied: For Agile Software Development*, Addison-Wesley Professional

3. COHN, M., 2005, *Agile Estimating and Planning*, Prentice Hall

4. COHN, M., 2009, *Succeeding with Agile: Software Development Using Scrum*, Addison-Wesley Professional

5. PATTON, J., 2008, *User Story Mapping, http://www.agileproductdesign.com*

6. PICHLER, R., 2010, *Agile Product Management with Scrum: Creating Products that Customers Love*, Addison-Wesley Professional

7. PINK, D., 2010, *Drive: The Surprising Truth About What Motivates Us*, Canongate Books

8. RASMUSSON, J., 2013, *The Agile Samurai: How Agile Masters Deliver Great Software*, Pragmatic Bookshelf

9. RIES, E., 2011, *The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses*, Crown Business

10. SCHWABER, K.; BEEDLE, M., 2001, *Agile Software Development with Scrum*, Prentice Hall

Acerca de los Autores

Martín Alaimo

Me desempeño como coach organizacional y entrenador profesional. Como Certified Scrum Coach (CSC) y Certified Scrum Trainer (CST), mi principal área de intervención es el trabajo en equipo bajo un marco colaborativo y relacional conocido como Scrum dentro del ámbito de del desarrollo de productos tecnológicos. Mi principal inquietud hoy en día pasa por asistir a otros rubros, más allá del tecnológico, a percibir y capitalizar los beneficios de esta nueva propuesta de trabajo, haciendo énfasis en las relaciones interpersonales y las competencias de los miembros de las organizaciones que pretenden adoptarla.

Crecí como profesional en el rubro tecnológico, tuve la oportunidad de formar parte de empresas pequeñas, medianas y organizaciones multinacionales con más de 200.000 empleados. En el 2009 fundé Kleer, empresa de la que hoy formo parte. Junto a profesionales a los que admiro, día a día, llevamos adelante nuestra organización con el objetivo de promover y asistir a otras organizaciones a adoptar nuevas y más efectivas formas de trabajar y técnicas de creación de productos y servicios innovadores.

"Mi compromiso es asistir a los equipos y las organizaciones a alcanzar esos resultados que aun no alcanzaron"

http://www.martinalaimo.com
martin.alaimo@kleer.la

Martín Salías

Llevo más de 30 años diseñando y desarrollando productos, o ayudando a otros a hacerlo, principalmente en el mundo del software, pero también en el ambiente editorial y de contenidos.

Desde el inicio de mi carrera participé en comunidades culturales, técnicas o metodológicas, y siempre me interesó compartir mis experiencias y conocimientos como una manera extra de aprendizaje, ya que explicar un tema a una audiencia variada exige un nivel de conceptualización que va más allá de la teoría y la práctica.

Mi interés por lo métodos ágiles se inició a mediados de los ’90, junto a muchos otros en las nacientes comunidades virtuales, e inicialmente me sumé a las prácticas de XP. Entendí lo que agregaba Scrum años después, y lo usé en múltiples proyectos en América Latina, USA y Canadá.

Actualmente me dedico al entrenamiento y acompañamiento de organizaciones, equipos e individuos, sobre todo dentro de ese segmento cada vez más numeroso que llamamos “trabajadores del conocimiento”.

“Me gusta trabajar con equipos que buscan innovar y mejorar constantemente para disfrutar el trabajo que hacen, lo que siempre genera mejores resultados.”

http://www.CodeAndBeyond.org
martin.salias@kleer.la

Acerca de Kleer

Somos una empresa de capacitación y coaching. Creemos en una forma de trabajo clara, centrada en las personas y orientada hacia las necesidades específicas de cada contexto.

Nuestras premisas son:

- **Mantenerlo simple**. Las metodologías y prácticas de trabajo en las que confiamos aportan claridad a los proyectos. Los proyectos claros se vuelven más previsibles y con menor incidencia de errores evitables.

- **Las personas son todo**. No acompañamos proyectos sino equipos y personas. Nuestro propósito es transmitirles nuestro conocimiento y experiencia para que logren resultados de los que se sientan orgullosos.

- **Cada contexto es un mundo**. Estudiamos las características de cada proyecto y organización para entender cuáles son las prácticas y metodologías que mejor responden a sus necesidades y objetivos de negocio.

"En Kleer disfrutamos co-creando ambientes humanos más conscientes y asombrosos*"*

Kleer.la
http://www.kleer.la
entrenamos@kleer.la