Scrum

Por Félix Avendaño Scarabicchi

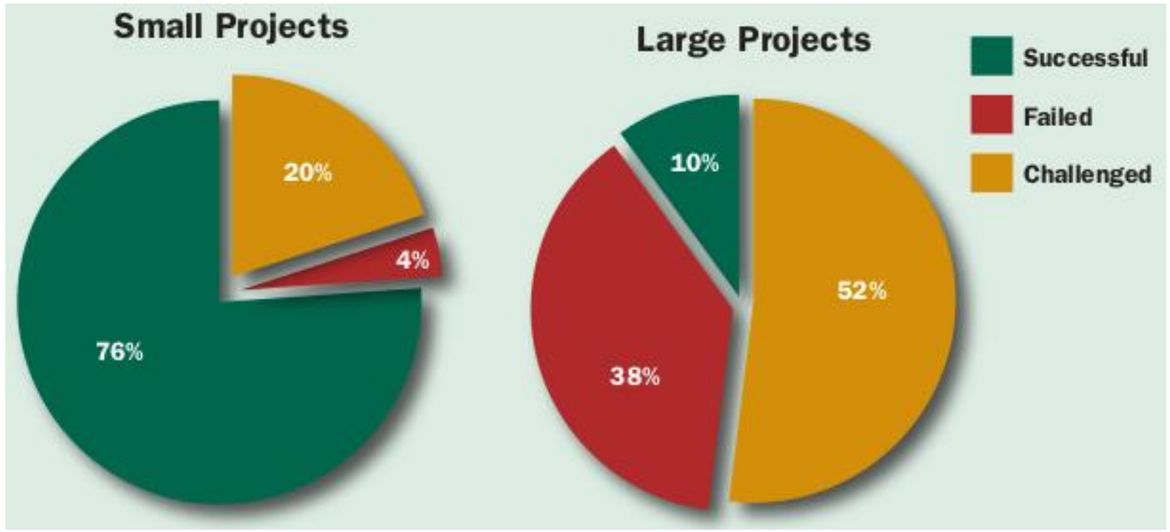
Índice

Índice

La crisis del software

Fracaso en el desarrollo de software
La ingeniería del software
Qué es el desarrollo de software
Los ciclos del desarrollo de software
- Spagetti Programming
- Cascada
- Espiral
- Iterativo

El problema del Gap semántico y como resolverlo



El principio de la solución:

La teoría de objetos (Booch, Rumbaugh, Jakobson)
Los casos de uso
El uso de Patrones de diseño (Erick Gamma)
UML y UP

Ejercicio: Elaborar un caso de uso en base al trabajo ya realizado

Scrum y Agile

Scrum

es un marco de trabajo por el cual las personas pueden abordar problemas complejos adaptativos a la vez que entregar productos del máximo valor posible productiva y creativamente
Es liviano, fácil de entender y difícil de dominar
No es un proceso para construir productos, es un marco de trabajo.

Las reglas de scrum relacionan
los eventos roles y artefactos gobernando las relaciones e interacciones entre ellos.

Metodologías agile con 2 enfoques:

- Lightweight
  - Scrum
  - Crystal
  - XP
- Fuller
  - DSDM
  - Proceso Unificado Agile (AUP)
  - Desarrollo impulsado por funciones

Proyectos Crystal

desarrollada por Alistair Cockburn

Entrega frecuente
Mejora reflexiva
comunicación cercana (osmótica)
Seguridad personal (Poder emitir desacuerdos)
Concentración (pocas tareas)
Fácil acceso a usuarios expertos
Entorno técnico (Manejo de configuraciones, planes de pruebas, etc…)

Muestras de Crystal

Orange

30 a 50 personas
cada persona tiene una clara descripción de su trabajo

Clear

Equipo de 3 a 10 personas colocalizadas

Yellow u Orange web para desarrollo web

Cualquier metodología es buena siempre que se vaya mejorando en forma sucesiva

Desarrollo impulsado por funciones desarrollado por Jeff De Luca y Peter Coad

• Se requiere un sistema para construir sistemas si se pretende escalar a proyectos
grandes.
• Un proceso simple y bien definido trabaja mejor.
• Los pasos de un proceso deben ser lógicos y su mérito inmediatamente obvio para cada miembro del equipo.
• Vanagloriarse del proceso puede impedir el trabajo real.
• Los buenos procesos van hasta el fondo del asunto, de modo que los miembros del equipo se puedan concentrar en los resultados.
• Los ciclos cortos, iterativos, orientados por rasgos (features) son mejores.

Hay tres categorías de rol en FDD:
•roles claves
•roles de soporte
•roles adicionales.

Los seis roles claves de un proyecto son:
(1) administrador del proyecto, quien tiene la última palabra en materia de visión, cronograma y asignación del personal;
(2) arquitecto jefe(puede dividirse en arquitecto de dominio y arquitecto técnico);
(3) manager de desarrollo, que puede combinarse con arquitecto jefe o manager de proyecto;
(4)programador jefe, que participa en el análisis del requerimiento y selecciona rasgos del conjunto a desarrollar en la siguiente iteración;
(5) propietarios de clases, que trabajan bajo la guía del programador jefe en diseño, codificación, prueba y documentación, repartidos por rasgos y
(6) experto de dominio, que puede ser un cliente, patrocinador, analista de negocios o una mezcla de todo eso

Los cinco roles de soporte comprenden

(1) administrador de entrega, que controla el
progreso del proceso revisando los reportes del programador jefe y manteniendo reuniones breves con él; reporta al manager del proyecto;
(2) abogado/guru de lenguaje,
que conoce a la perfección el lenguaje y la tecnología;
(3) ingeniero de construcción, que
se encarga del control de versiones de los builds y publica la documentación;
(4) herramientista (toolsmith), que construye herramientas ad hoc o mantiene bases de datos y sitios Web
(5) administrador del sistema, que controla el ambiente de trabajo o productiza el sistema cuando se lo entrega.

Los tres roles adicionales son los de verificadores, encargados del despliegue y escritores técnicos.

Un miembro de un equipo puede tener otros roles a cargo, y un solo rol puede ser compartido por varias personas.

Las fases son:

1) Desarrollo de un modelo general.
Cuando comienza este desarrollo, los expertos de
dominio ya están al tanto de la visión, el contexto y los requerimientos del sistema a construir. A esta altura se espera que existan requerimientos tales como casos de uso o especificaciones funcionales. FDD, sin embargo, no cubre este aspecto. Los expertos de dominio presentan un ensayo (walkthrough) en el que los miembros del equipo y el arquitecto principal se informan de la descripción de alto nivel del sistema. El dominio general se subdivide en áreas más específicas y se define un ensayo más detallado para cada uno de los miembros del dominio. Luego de cada ensayo, un equipo de desarrollo trabaja en pequeños grupos para producir modelos de objeto de cada área de dominio. Simultáneamente, se construye un gran modelo general para todo el sistema.
2) Construcción de la lista de rasgos.
Los ensayos, modelos de objeto y
documentación de requerimientos proporcionan la base para construir una amplia lista de rasgos. Los rasgos son pequeños ítems útiles a los ojos del cliente. Son similares a las tarjetas de historias de XP y se escriben en un lenguaje que todas las partes puedan entender. Las funciones se agrupan conforme a diversas actividades en áreas de dominio específicas. La lista de rasgos es revisada por los usuarios y patrocinadores para asegurar su validez y exhaustividad. Los rasgos que requieran más de diez días se descomponen en otros más pequeños.
3) Planeamiento por rasgo.
Incluye la creación de un plan de alto nivel, en el que los
conjuntos de rasgos se ponen en secuencia conforme a su prioridad y dependencia, y se asigna a los programadores jefes. Las listas se priorizan en secciones que se llaman paquetes de diseño. Luego se asignan las clases definidas en la selección del modelo general a programadores individuales, o sea propietarios de clases. Se pone fecha para los conjuntos de rasgos.
4) Diseño por rasgo y Construcción por rasgo.
Se selecciona un pequeño conjunto de
rasgos del conjunto y los propietarios de clases seleccionan los correspondientes equipos dispuestos por rasgos. Se procede luego iterativamente hasta que se producen los rasgos seleccionados. Una iteración puede tomar de unos pocos días a un máximo de dos semanas. Puede haber varios grupos trabajando en paralelo. El proceso iterativo incluye inspección de diseño, codificación, prueba de unidad, integración e inspección de código. Luego de una iteración exitosa, los rasgos completos se promueven al build principal. Este proceso puede demorar una o dos semanas en implementarse.

FDD consiste en un conjunto de “mejores prácticas” que distan de ser nuevas pero se combinan de manera original. Las prácticas canónicas son:

• Modelado de objetos del dominio, resultante en un framework cuando se agregan los rasgos. Esta forma de modelado descompone un problema mayor en otros menores; el diseño y la implementación de cada clase u objeto es un problema pequeño a resolver. Cuando se combinan las clases completas, constituyen la solución al problema mayor. Una forma particular de la técnica es el modelado en colores [CLD00], que agrega una dimensión adicional de visualización. Si bien se puede modelar en blanco y negro, en FDD el modelado basado en objetos es imperativo.

• Desarrollo por rasgo. Hacer simplemente que las clases y objetos funcionen no refleja lo que el cliente pide. El seguimiento del progreso se realiza mediante examen de pequeñas funcionalidades descompuestas y funciones valoradas por el cliente. Un rasgo en FDD es una función pequeña expresada en la forma <acción> <resultado> <por | para | de | a> <objeto> con los operadores adecuados entre los términos. Por ejemplo, calcular el importe total de una venta; determinar la última operación de un cajero; validar la contraseña de un usuario.

• Propiedad individual de clases (código). Cada clase tiene una sola persona nominada como responsable por su consistencia, performance e integridad conceptual.

• Equipos de Rasgos, pequeños y dinámicamente formados. La existencia de un equipo garantiza que un conjunto de mentes se apliquen a cada decisión y se tomen en cuenta múltiples alternativas.

• Inspección. Se refiere al uso de los mejores mecanismos de detección conocidos. FDD es tan escrupuloso en materia de inspección como lo es Evo.

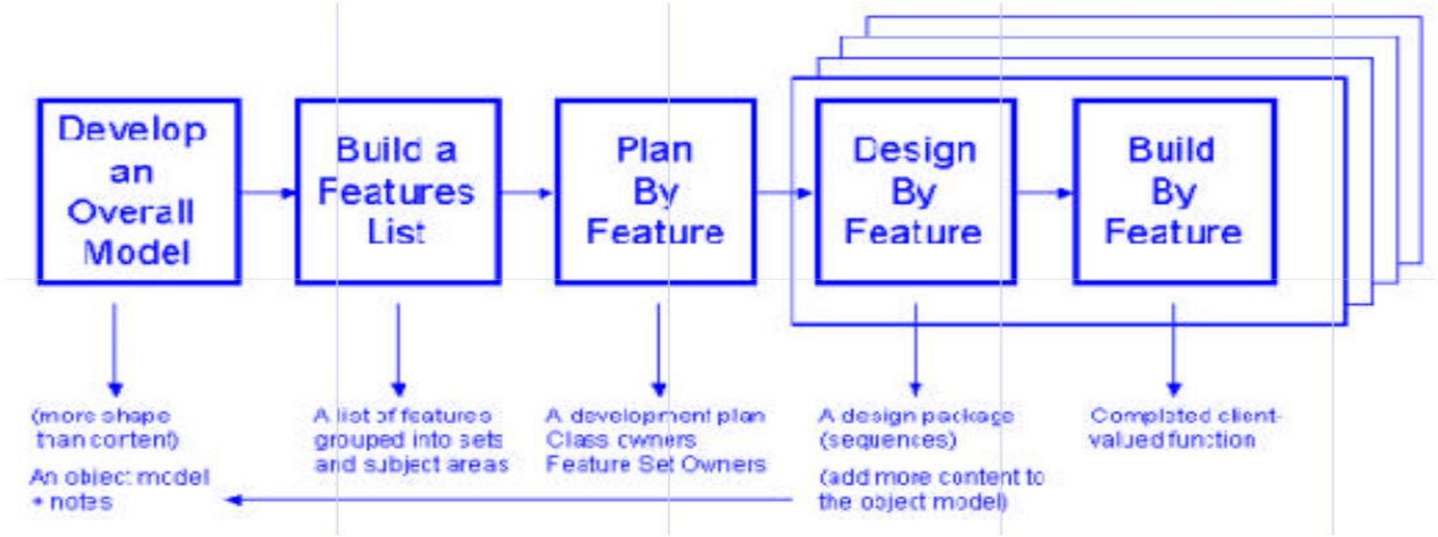
• Builds regulares. Siempre se tiene un sistema disponible. Los builds forman la base a partir de la cual se van agregando nuevos rasgos.

• Administración de configuración. Permite realizar seguimiento histórico de las últimas versiones completas de código fuente.

• Reporte de progreso. Se comunica a todos los niveles organizacionales necesarios.

http://www.nebulon.com/articles/fdd/latestprocesses.html

Feature Driven Development



10% Inicio	4% Inicio	2% Inicio	77%
4% Continuo	1% Continuo	2% Continuo	

Metodología dinámica de desarrollo de sistemas (Publicada en 1994)

DSDM consiste en 3 fases:

fase del pre-proyecto,
fase del ciclo de vida del proyecto, y
fase del post-proyecto.

La fase del ciclo de vida del proyecto se subdivide en 5 etapas:

estudio de viabilidad,
estudio de la empresa,
iteración del modelo funcional,
diseño e iteración de la estructura
implementación.

Hay 9 principios subyacentes al DSDM consistentes en cuatro fundamentos y cinco puntos de partida para la estructura del método. Estos principios forman los pilares del desarrollo mediante DSDM.

Involucrar al cliente es la clave para llevar un proyecto eficiente y efectivo, donde ambos, cliente y desarrolladores, comparten un entorno de trabajo para que las decisiones puedan ser tomadas con precisión.

El equipo del proyecto debe tener el poder para tomar decisiones que son importantes para el progreso del proyecto, sin esperar aprobación de niveles superiores.

DSDM se centra en la entrega frecuente de productos, asumiendo que entregar algo temprano es siempre mejor que entregar todo al final. Al entregar el producto frecuentemente desde una etapa temprana del proyecto, el producto puede ser verificado y revisado allí donde la documentación de registro y revisión puede ser tenida en cuenta en la siguiente fase o iteración.

El principal criterio de aceptación de entregables en DSDM reside en entregar un sistema que satisface las actuales necesidades de negocio. No está dirigida tanto a proporcionar un sistema perfecto que resuelva todas las necesidades posibles del negocio, si no que centra sus esfuerzos en aquellas funcionalidades críticas para alcanzar las metas establecidas en el proyecto/negocio.

El desarrollo es iterativo e incremental, guiado por la realimentación de los usuarios para converger en una solución de negocio precisa.

Todos los cambios durante el desarrollo son reversibles.

El alcance de alto nivel y los requerimientos deberían ser base-lined antes de que comience el proyecto.

Las pruebas son realizadas durante todo el ciclo vital del proyecto. Esto tiene que hacerse para evitar un caro coste extraordinario en arreglos y mantenimiento del sistema después de la entrega.

La comunicación y cooperación entre todas las partes interesadas en el proyecto es un prerrequisito importante para llevar un proyecto efectivo y eficiente.

DSDM también se apoya en otros principios (también llamadas asunciones).

Ningún sistema es construido a la perfección en el primer intento (El principio de pareto - regla 80/20). En el proceso de desarrollar un sistema de información, el 80% del beneficio de la empresa proviene del 20% de los requisitos del sistema, así DSDM comienza implementando primero este 20% de requisitos para cumplir con el 80% de las necesidades de la empresa, lo que es suficientemente bueno tanto en cuanto los usuarios estén íntimamente involucrados en el proceso de desarrollo y en una posición de asegurar que el 20% restante no causará serias consecuencias al negocio. Implementar la totalidad de requerimientos a menudo causa que un proyecto supere plazos y presupuestos, así la mayoría de las veces es innecesario construir la solución perfecta.

La entrega del proyecto debería ser a tiempo, respetando presupuestos y con buena calidad.

DSDM solo requiere que cada paso del desarrollo se complete lo suficiente como para que empiece el siguiente paso. De este modo una nueva iteración del proyecto puede comenzar sin tener que esperar a que la previa se complete enteramente. Y con cada nueva iteración el sistema se mejora incrementalmente. Recuérdese que las necesidades del negocio cambian constantemente y a cualquier ritmo con el tiempo.

Ambas técnicas de Desarrollo y Gestión del proyectos están incluidas en DSDM.

Además de desarrollar nuevos SI, DSDM puede ser usado también en proyectos de ampliación de sistemas TI actuales o incluso en proyectos de cambio no relacionados con las TI.

La Evaluación de riesgos debiera centrarse en entregar función de negocio, no en el proceso de construcción.

La gestión recompensa la entrega de productos más que la consecución de tareas.

La Estimación debería estar basada en la funcionalidad del negocio en lugar de líneas de código.

XP

formulada por Kent Beck

Las características fundamentales del método son:

- **Desarrollo iterativo e incremental**: pequeñas mejoras, unas tras otras.
- **Pruebas unitarias continuas**, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación. Véase, por ejemplo, las herramientas de prueba JUnit orientada a Java, DUnit orientada a Delphi, NUnit para la plataforma.NET o PHPUnit para PHP. Estas tres últimas inspiradas en JUnit, la cual, a su vez, se insipiró en SUnit, el primer framework orientado a realizar tests, realizado para el lenguaje de programación Smalltalk.
- **Programación en parejas**: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. La mayor calidad del código escrito de esta manera -el código es revisado y discutido mientras se escribe- es más importante que la posible pérdida de productividad inmediata.
- Frecuente **integración del equipo de programación con el cliente** o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- **Corrección de todos los errores** antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- **Refactorización del código**, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- **Propiedad del código compartida**: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.
- **Simplicidad en el código**: es la mejor manera de que las cosas funcionen. Cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Wikipedia

Roles:

Programador
Cliente
Tester
Tracker
Entrenador
Consultor
Gestor

Ventajas de XP

El equipo

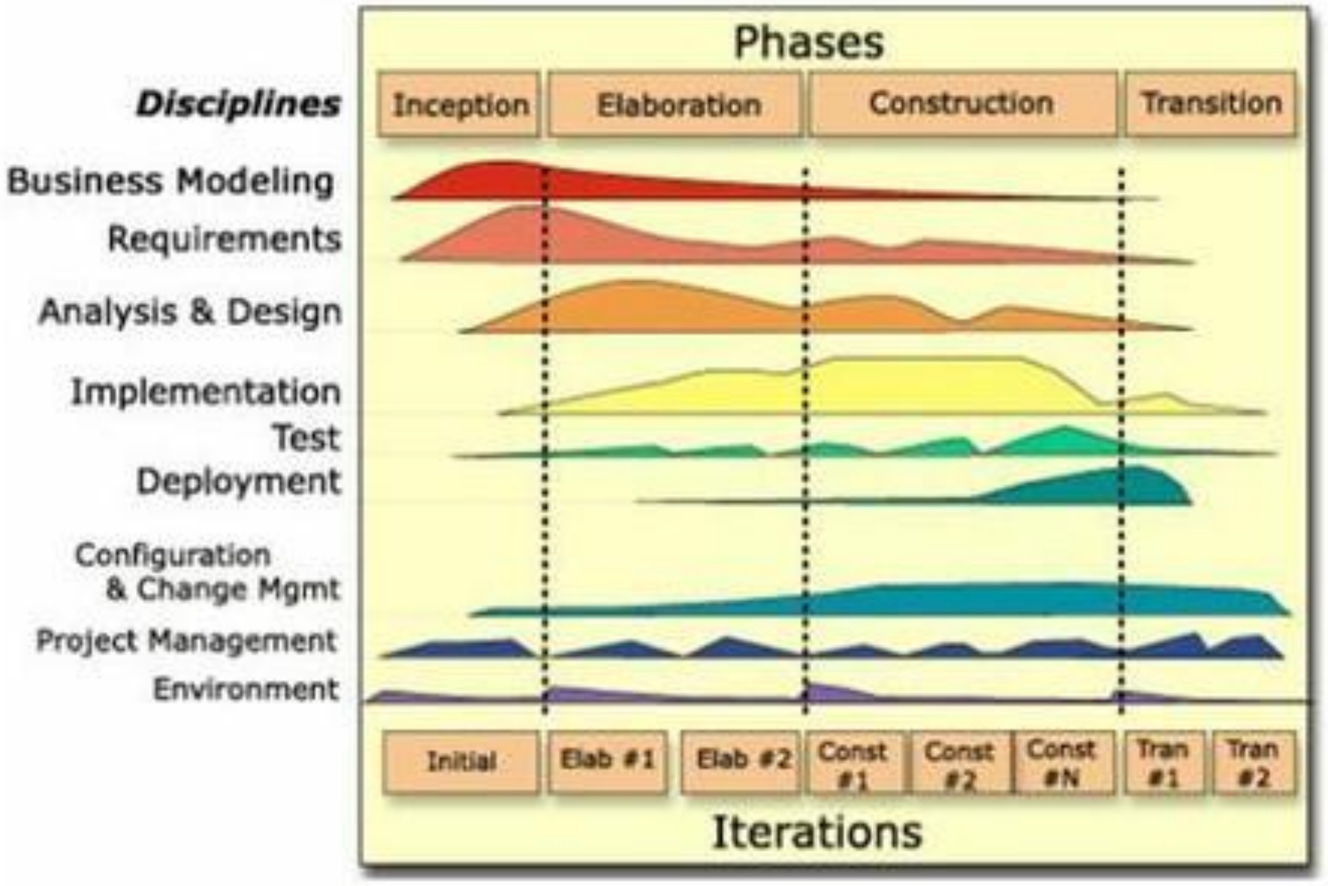
- Tiene requerimientos claros
- Realiza decisiones técnicas y sigue las mejores prácticas
- Tiene una alta motivación
- No trabaja mas de lo necesario

El cliente

- Obtiene un producto en forma rápida
- Tiene una retroalimentación constante y precisa
- Puede tomar decisiones o cambiar cosas en forma rápida

Se supone un ambiente de cooperación y Divertido!!!

Proceso Unificado de Agile (UPA)



La AUP es ágil, porque está basada en los siguientes principios:

1. **Iterativo e Incremental**
2. **Dirigido por los casos de uso**
3. **Emplea arquitectura basada en componentes**
4. **Agilidad**. Utiliza los valores y principios de Agile.
5. **Se centra en actividades de alto valor**. La atención se centra en las actividades que se ve que son esenciales para el de desarrollo, no todas las actividades que suceden forman parte del proyecto.
6. **Independencia en las herramientas**. Se puede usar cualquier conjunto de herramientas que se desee. Lo aconsejable es utilizar las herramientas que son las más adecuadas para el trabajo, que a menudo son las herramientas simples o incluso herramientas de código abierto.
7. **Enfocado en los riesgos**

Scrum

Scrum se origina en el Rugby. El origen es del año 1986

Scrum es el nombre con el que se denomina a los marcos de desarrollo ágiles.

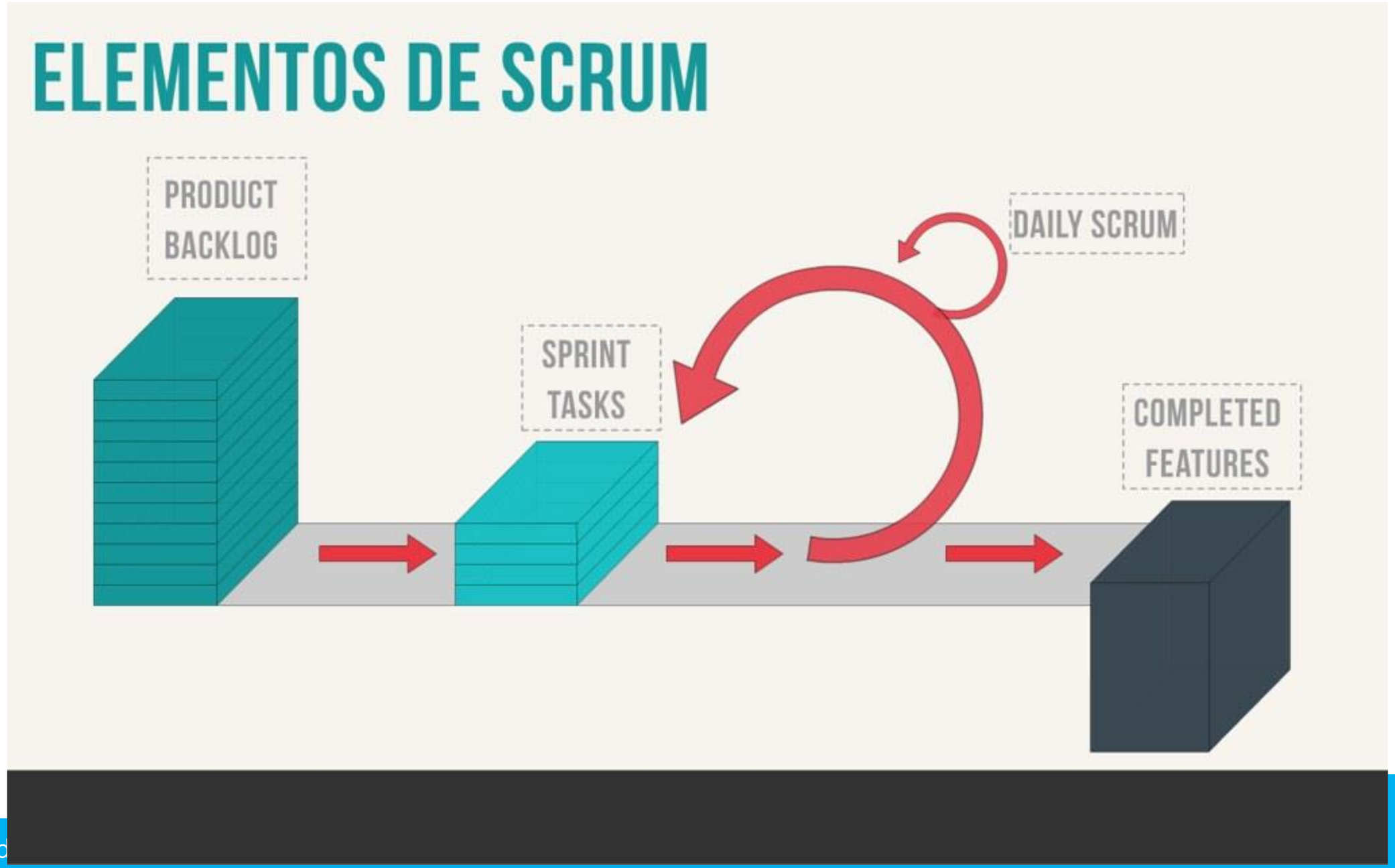
Es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo y obtener el mejor resultado posible de proyectos, caracterizado por:[1]

Adopta una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa del producto.

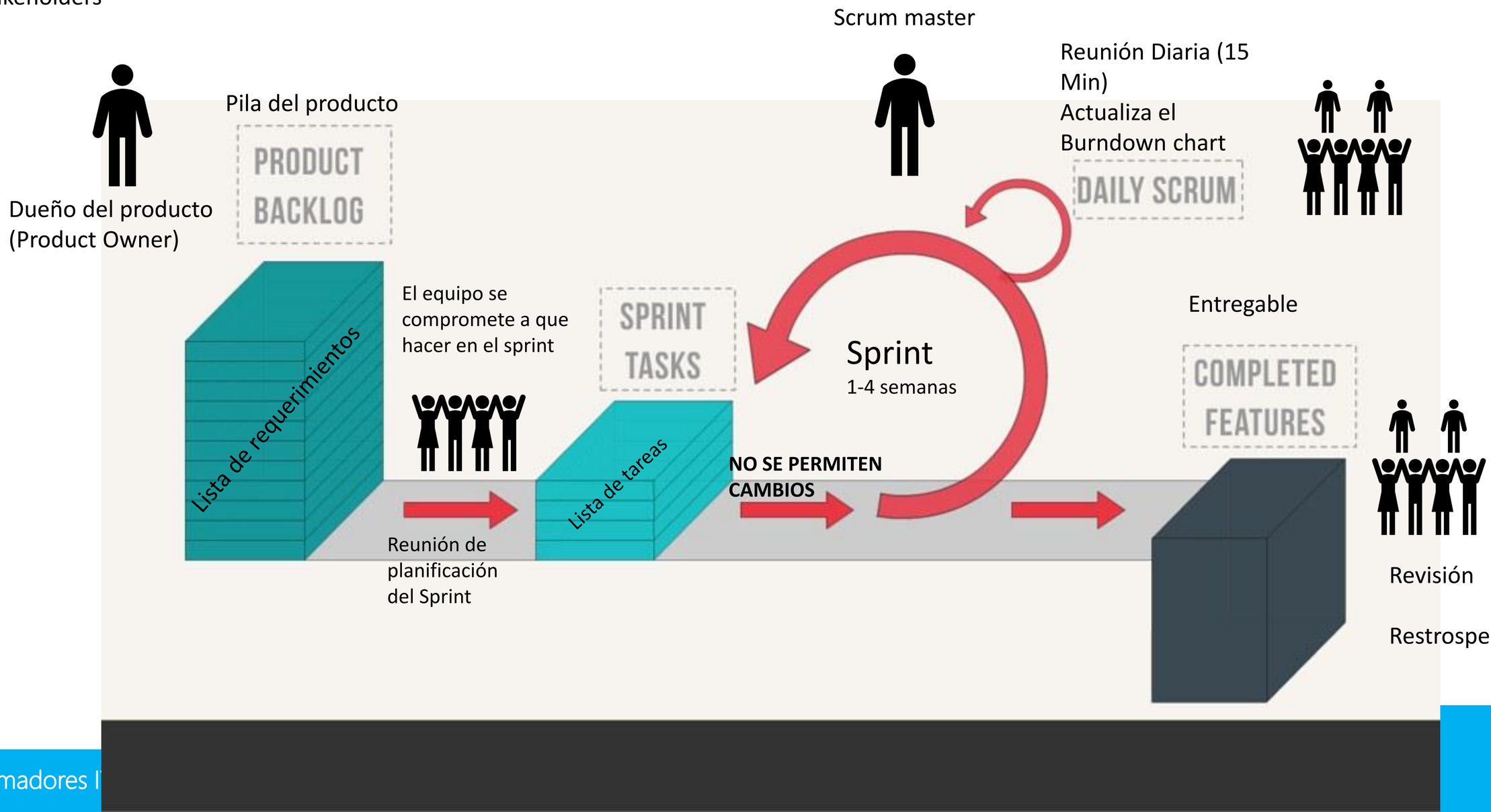
Basar la calidad del resultado más en el conocimiento tácito de las personas en equipos auto organizados, que en la calidad de los procesos empleados.

Solapamiento de las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada.

(Wikipedia)


ELEMENTOS DE SCRUM
PRODUCT BACKLOG
SPRINT TASKS
DAILY SCRUM
COMPLETED FEATURES

Entradas de usuarios finales, cliente, equipo y otros stakeholders

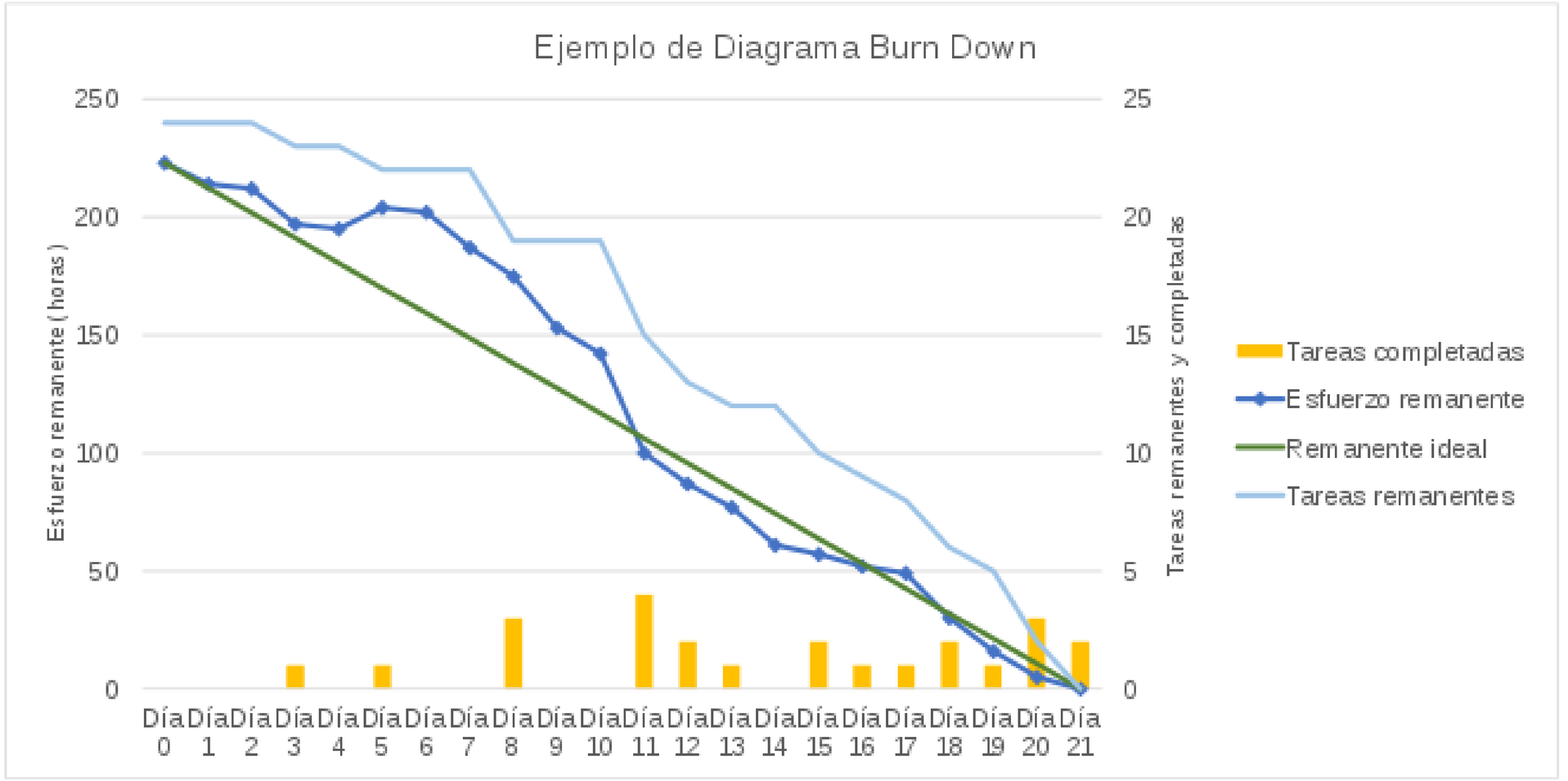


Burndown Chart

Un diagrama de Gantt no sirve para aterrizar un avión
En los proyectos pasan cosas que se desconocen por anticipado y se deben adaptar las cosas

Un **diagrama burn down** o *diagrama de quemado* es una representación gráfica del trabajo por hacer en un proyecto en el tiempo. Usualmente el trabajo remanente (o *backlog*) se muestra en el eje vertical y el tiempo en el eje horizontal. Es decir, el diagrama representa una serie temporaldel trabajo pendiente. Este diagrama es útil para predecir cuándo se completará todo el trabajo. Usualmente se usa en el desarrollo ágil de software, especialmente con Scrum. (Wikipedia)

El burndown chart puede verse para 1 sprint o en sprints para todo el proyecto


Ejemplo de Diagrama Burn Down
Esfuerzo remanente (horas)
Tareas remanentes y completadas
250
200
150
100
50
0
25
20
15
10
5
0
Día 0
Día 1
Día 2
Día 3
Día 4
Día 5
Día 6
Día 7
Día 8
Día 9
Día 10
Día 11
Día 12
Día 13
Día 14
Día 15
Día 16
Día 17
Día 18
Día 19
Día 20
Día 21
Tareas completadas
Esfuerzo remanente
Remanente ideal
Tareas remanentes

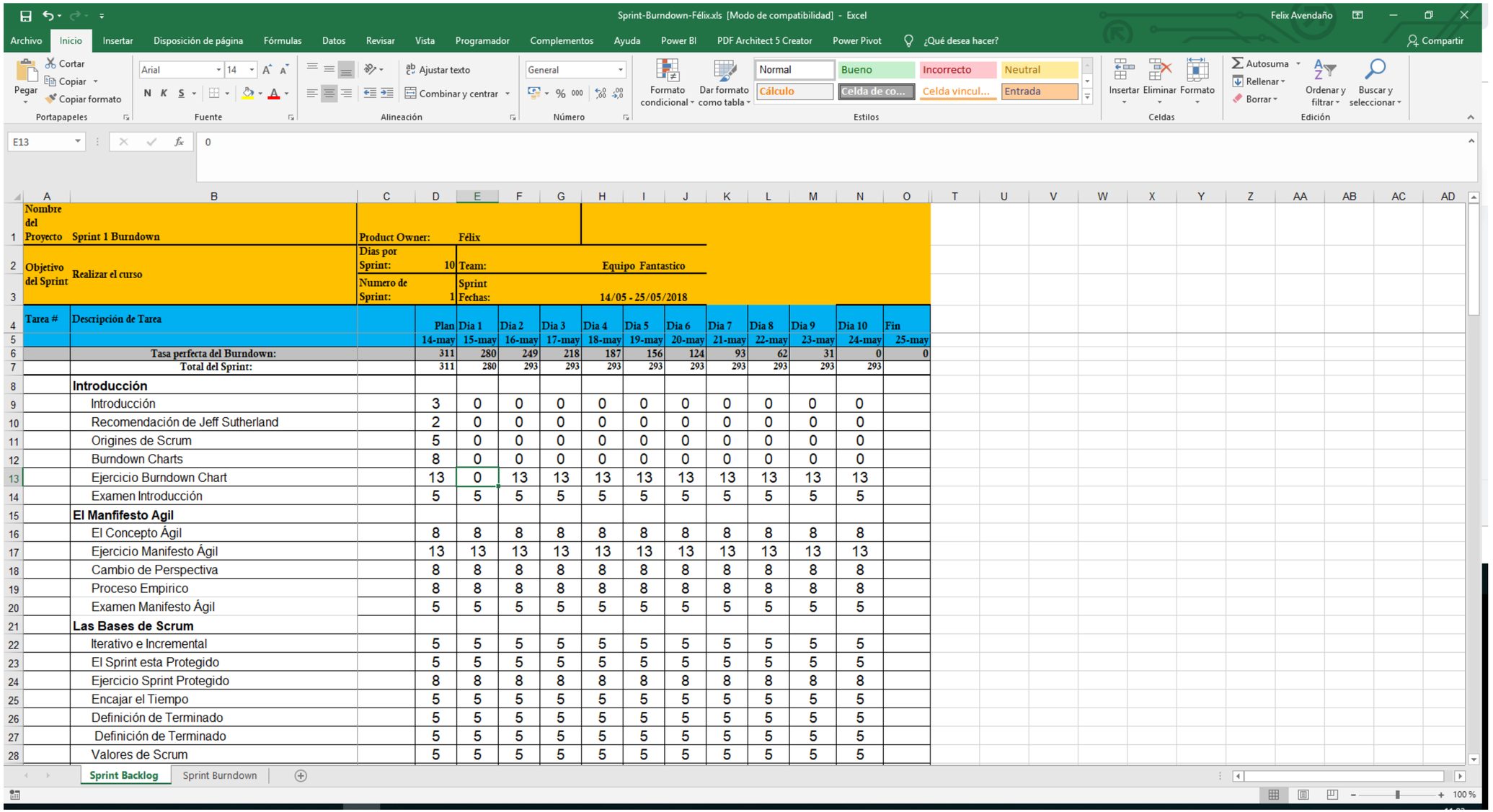
Ejercicio:

En función de los trabajos de las webs que han montado con HTML y Css elaborar El Burndown chart de ese trabajo

Para ellos deberá suponer que cuenta con el siguiente equipo de trabajo

El cliente
Un especialista en diseño web
Un desarrollador

Como material se le entregará un Excel para completar.



Sprint-Burndown-Félix.xls [Modo de compatibilidad] - Excel

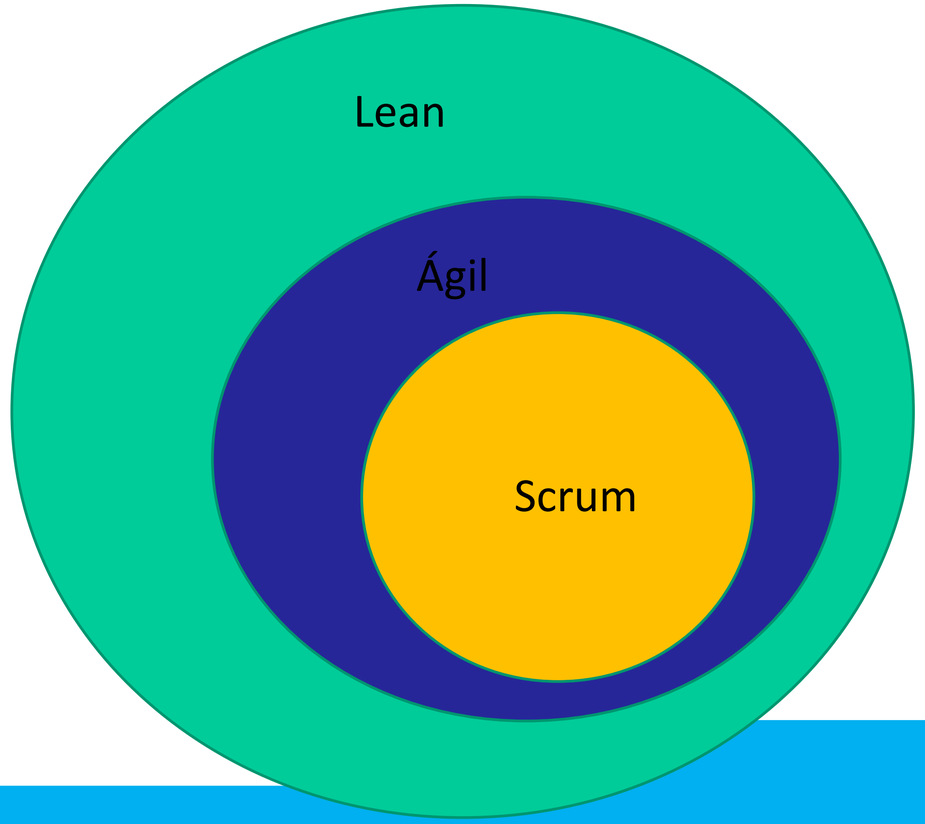
1	Nombre del Proyecto	Sprint 1 Burndown		Product Owner:	Félix									
2	Objetivo del Sprint	Realizar el curso		Dias por Sprint:	10	Team:	Equipo Fantastico							
3				Numero de Sprint:	1	Sprint Fechas:	14/05 - 25/05/2018							
4	Tarea #	Descripción de Tarea		Plan	Dia 1	Dia 2	Dia 3	Dia 4	Dia 5	Dia 6	Dia 7	Dia 8	Dia 9	Dia 10	Fin
5				14-may	15-may	16-may	17-may	18-may	19-may	20-may	21-may	22-may	23-may	24-may	25-may
6		Tasa perfecta del Burndown:		311	280	249	218	187	156	124	93	62	31	0	0
7		Total del Sprint:		311	280	293	293	293	293	293	293	293	293	293	
8		**Introducción**													
9		Introducción		3	0	0	0	0	0	0	0	0	0	0	
10		Recomendación de Jeff Sutherland		2	0	0	0	0	0	0	0	0	0	0	
11		Origines de Scrum		5	0	0	0	0	0	0	0	0	0	0	
12		Burndown Charts		8	0	0	0	0	0	0	0	0	0	0	
13		Ejercicio Burndown Chart		13	0	13	13	13	13	13	13	13	13	13	
14		Examen Introducción		5	5	5	5	5	5	5	5	5	5	5	
15		**El Manfifesto Agil**													
16		El Concepto Ágil		8	8	8	8	8	8	8	8	8	8	8	
17		Ejercicio Manifesto Ágil		13	13	13	13	13	13	13	13	13	13	13	
18		Cambio de Perspectiva		8	8	8	8	8	8	8	8	8	8	8	
19		Proceso Empirico		8	8	8	8	8	8	8	8	8	8	8	
20		Examen Manifesto Ágil		5	5	5	5	5	5	5	5	5	5	5	
21		**Las Bases de Scrum**													
22		Iterativo e Incremental		5	5	5	5	5	5	5	5	5	5	5	
23		El Sprint esta Protegido		5	5	5	5	5	5	5	5	5	5	5	
24		Ejercicio Sprint Protegido		8	8	8	8	8	8	8	8	8	8	8	
25		Encajar el Tiempo		5	5	5	5	5	5	5	5	5	5	5	
26		Definición de Terminado		5	5	5	5	5	5	5	5	5	5	5	
27		Definición de Terminado		5	5	5	5	5	5	5	5	5	5	5	
28		Valores de Scrum		5	5	5	5	5	5	5	5	5	5	5	

LEAN: Maximizar el valor para el cliente mientras se reducen los desperdicios

La producción lean es un modelo de gestión que se enfoca en minimizar las pérdidas de los sistemas de manufactura al mismo tiempo que maximiza la creación de valor para el cliente final. Para ello utiliza la mínima cantidad de recursos, es decir, los estrictamente necesarios para el crecimiento. (Wikipedia)

AGIL: No es un método con valores y principios

SCRUM: Marco



Las 3 M de Lean

La discusión sobre la aplicación de los principios lean al desarrollo de software se ha centrado en gran medida en identificar y eliminar el desperdicio (en japonés: muda).

Lean Thinking apunta igualmente a eliminar la sobrecarga (japonés: muri) y la variación innecesaria (japonés: mura).

Muda, muri y mura se llaman "las tres M".

Juntos forman una tríada disonante. Las tres M deben eliminarse para crear un proceso lean sostenible.

El manifiesto ágil

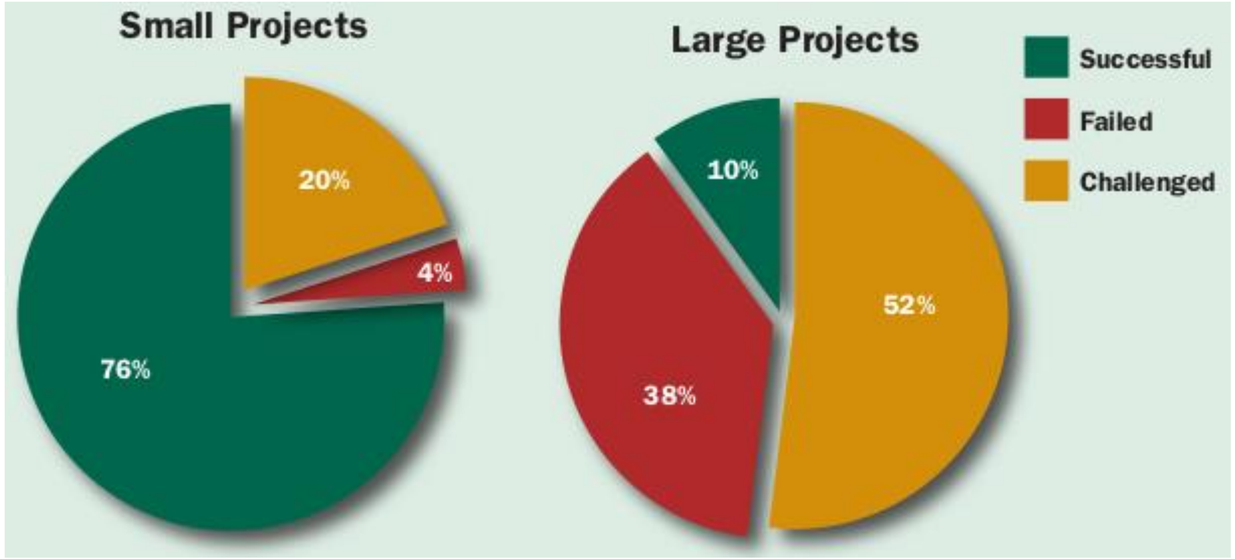
http://agilemanifesto.org/iso/es/manifesto.html

1) Dar importancia a las personas y las interacciones sobre los procesos y herramientas
2) Es mas importante que un un software (Producto) sea funcional antes que su documentación sea comprensible
3) La colaboración del cliente es mas importante que las negociaciones en el contrato
4) Responder a los cambios antes de seguir un plan

Cómo implementar Scrum:

Aplicamos la regla tal como esta escrita
Cuando tenemos experiencia podemos experimentar con las reglas y modificarlas
Cuando tenemos mucha experiencia podemos re inventar un nuevo método ágil

Fuente: https://pmworldlibrary.net/wp-content/uploads/2018/05/pmwj70-May2018-Rosato-go-small-for-project-success-student-paper.pdf



¿Y si flexibilizamos el alcance?

Hay que priorizar los requerimientos

El proyecto es exitosos si cumple con Alcance, tiempo y presupuesto

Solo un 20% de la funcionalidad del software se utiliza!!!

Un 36% de cambios de alcance en proyectos

Las bases de Scrum

Tipos de trabajo:

Trabajo Sencillo = es predecible (Cocinar patatas)

Trabajo Complicado (una línea de producción) = Es predecible y se pueden aplicar métodos tradicionales

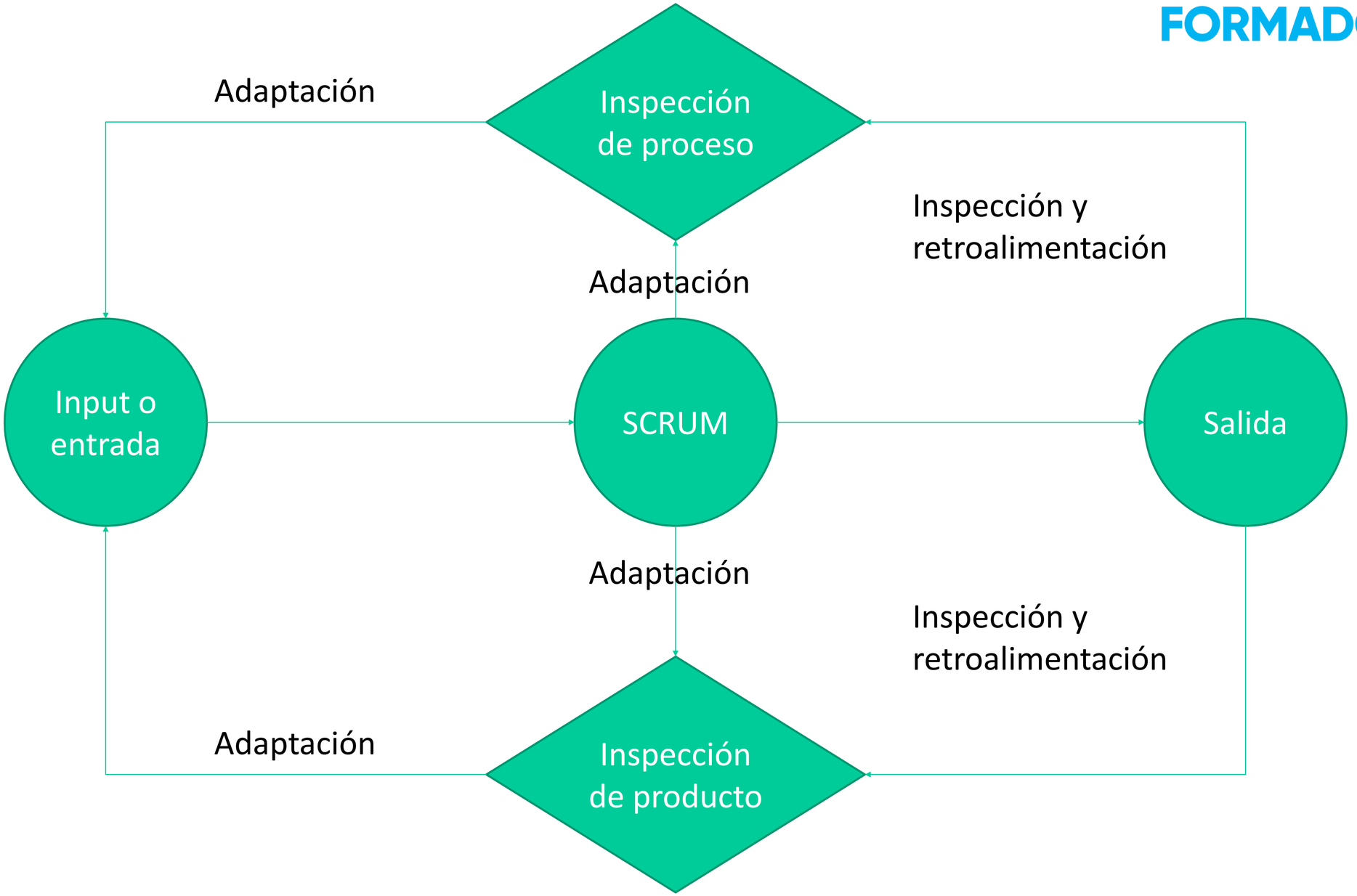
Trabajo Complejo = No es predecible, no se puede estar seguro de que puede pasar (el tiempo, la economía)

Un proyecto de desarrollo de software

Debemos probar, percibir y responder

Trabajo caótico = primero actuar, luego percibir y finalmente responder (Hay un desorden)

Scrum es bueno para Trabajo Complejo


Adaptación
Inspección de proceso
Inspección y retroalimentación
Adaptación
Input o entrada
SCRUM
Salida
Adaptación
Inspección y retroalimentación
Adaptación
Inspección de producto

Sprint

Un sprint es iterativo e incremental

Para una web,

iterativo significaría:

Programar la web completa, pero con faltantes. Se puede realizar todo el proceso pero no está “Bonito”

Luego agregamos lo que falta para que se vea bien

Incremental significaría:

Programamos una parte de la funcionalidad, en forma completa y exhaustiva

Luego agregamos mas funcionalidad, etc…

Finalmente obtenemos toda la funcionalidad

Sprint

El equipo se compromete para lograr un objetivo

Un sprint no admite cambios que afecten al objetivo

Nunca se disminuye la calidad del producto

El alcance se puede re-negociar a medida que se aprende

EN EL SIGUIENTE SPRINT

Excepcionalmente un sprint se puede terminar antes de tiempo por el producto owner
Si lo que se esta desarrollando no va a ser necesario

Sprint

Tiene una duración: 1-4 semanas.

Cuando se termina el tiempo, si no se terminaron las tareas, se pasa al siguiente sprint

La tendencia de los sprint es a ir a sprint mas cortos

Cuanto mas cortos, mas retroalimentación tendremos

Al cliente solo le sirven los productos terminados.
O algo esta finalizado o no lo esta

Algo esta terminado si el cliente lo puede utilizar.

5 Valores de Scrum:

- Compromiso
- Respeto
- Apertura (puede existir un tema cultural)
- Foco (con un objetivo claro)
- Coraje (Para aplicar Scrum)

Roles:

Product Owner

- Define y prioriza los elementos del producto backlog
- Toma decisiones acerca de entregas y fechas de lanzamiento del produto
- Es responsable del ROI (Retorno de la inversión)

Scrum Master:

- Es un facilitador del equipo y lo organiza
- Ayuda con los problemas al equipo
- Es responsable de la efectividad del equipo

Equipo de desarrollo

- Responsable de entregar el sprint y es autónomo
- de 3 a 9 personas

¡Todos son el equipo de Scrum!

Product Owner

Necesita autoridad para tomar decisiones
Conocimiento del producto y del negocio, necesita entender las necesidades del cliente
Necesita disponibilidad de tiempo

Hacer el producto exitoso
Crea la visión del producto
Mantiene el producto Backlog
Colabora con equipo
Colabora con stakeholders
Participa en las reuniones de sprints

Pasa un 50% del tiempo con el cliente
Tener visión atractiva de producto
Construir mapa para desarrollar visión
Construir el producto Backlog

Autoridad sobre el Product Backlog:

Claramente expresado, Ordenado, Valor optimizado (Máximo beneficio al cliente), El PB debe ser visible y transparente (todos pueden acceder a él), debe poder ser entendido por el equipo

Scrum Master

Actúa como agente de cambio (ayuda a la organización a cambiar a scrum)
Debe ayudar al Product Owner
Tiene que eliminar los impedimentos al equipo
Entrena al equipo
Protege al equipo contra interrupciones, entre otras cosas
Guía al equipo

Facilita la reunión diaria
Facilita la planeación de un sprint
Facilita la restrospectiva
Protege al equipo
Remueve impedimentos

NO DA ORDENES DEBE SER CREIBLE:

Confiable, valorar a las personas, trabajar en equipo

Debe ser un ejemplo, tener experiencia y tener conocimiento

Equipo de desarrollo

Se auto-organiza y es responsable como tal
Debe entregar un incremento de producto al finalizar unsprint
Administra el sprint backlog y controla el progreso del sprint
Participa de las reuniones

Tiene autoridad para tomar decisiones que le permitan alcanzar el éxito
Puede solicitar recursos (incluyendo formación)

Cómo lograr el objetivo es problema del equipo

3 a 9 personas
Es colaborativo
Autoadministrado
Muti Funcional (tiene todas la habilidades juntas)

Artefactos de Scrum:

Artefactos de Scrum:

Product Backlog

Única fuente de requerimientos que define el alcance
Tiene todo lo necesario para cumplir con la visión de producto
Es una lista ordenada de funciones, requerimientos, mejoras, errores, etc…
No es completo
Es cambiante- Algo que tiene valor hoy, mañana puede no tenerlo-
La guía de scrum solo habla de Product Backlog Items (PBI) que son los elementos del backlog
Los PBI son ordenados
El Product Owner es el que toma la última decisión sobre el PB
Actualmente los equipos de Scrum utilizan User Stories (Historias de usuarios de XP)

El tamaño de un ítem debe estar organizado y no debe ser muy grande al principio

Artefactos de Scrum:

Historias de usuario I

Es un requerimiento que agrega valor al cliente (no es técnico)
No es una descripción detallada
Es una característica que el cliente puede utilizar

Como <rol> quiero <acción> [para qué]
Como comprador quiero seleccionar el producto [para luego comprarlo]

La nota para implementar en el sprint debe ser hablada con el PO
El PO debe responder todas la preguntas y estas deben quedar por escrito y documentadas
Además se deben definir los criterios de aceptación (que son cambiantes) en forma clara

Artefactos de Scrum:

Historias de usuario II

No hay que confundir un criterio de aceptación con el concepto de finalizado

El concepto de finalizado es una lista que aplica a todas las historias de usuario y cada historia de usuario debe pasar por este control:

Por ejemplo: desarrollado, testeado y aceptado por el usuario

Los criterios de aceptación son particulares de una historia

Artefactos de Scrum:

Estimación por puntos

La estimación se basa en

Esfuerzo, Riesgo, Complejidad e Incertidumbre

Se utiliza usualmente escalas como Fibonacci o escalas que no van de unidad en unidad

Se comparan las historias y se estima en forma relativa (comparando).

A veces se utiliza una historia como referencia (Una pequeña) y se le asigna un puntaje bajo (P. ej 3).

Se somete a consulta de los miembros del equipo realizando una votación, sin mostrar los resultados hasta que todos votaron.

Si hay una diferencia muy grande, se vuelve a votar.

Como máximo tres votaciones

¡Tener en cuenta que es una estimación!

Esto se hace con todas las historias de usuario

Utilizar horas no es buena idea

Artefactos de Scrum:

Estimación por puntos

Los puntos son una medida universal y no tienen sesgo (por ejemplo experiencia)
Es algo estable
Es creible
Habla del tamaño de la historia y no de tiempos
Es una medida constante
¡Es mas exacta!

Perder tiempo en las estimaciones no agrega exactitud

Artefactos de Scrum:

Sprint Backlog

El Sprint backlog es la extracción del Product Backlog de los ítems que vamos a poder realizar en un sprint determinado

Lo determina el equipo

Se pueden desglosar las historias de usuario en tareas = como lo vamos a lograr

El Sprint Backlog esta fijo el desglose no y el equipo lo puede cambiar

Artefactos de Scrum:

Incremento del producto

Al finalizar un sprint el equipo es responsable de presentar un **incremento** de producto **potencialmente** entregable (No necesariamente se le entregará al usuario, esto lo decide el PO, pero el producto es utilizable)
El Product Owner decide si esta terminado o no y si se entrega o no al usuario

Scrum en 12 Minútos

https://www.youtube.com/watch?v=d3Ro7p-JAoA

¿Qué es un scrum master?

https://www.youtube.com/watch?v=CYCgVWX5N2Q

Reuniones de Scrum:

Reunión de planeación

Participantes:

Equipo de desarrollo, PO, Scrum Master, Otros

Cuando

Al inicio de cada Sprint

Duración

8 horas por cada 4 semanas de sprint.

Entradas:

Product Backlog, último incremento de producto, capacidad proyectada del equipo de desarrollo durante el sprint (cantidad de personas) y anterior desempeño del equipo de desarrollo (Puntos).

Salidas:

Qué se va a obtener y cómo (El sprint backlog)

Reuniones de Scrum:

Reunión de planeación

Seleccionar los puntos a completar del Product Backlog

El PO puede tener metas

Desglose del PB

El PO define prioridades y el PO establece una meta basada en la velocidad histórica en puntos. Se puede discutir

El equipo coge el desglose de tareas que se pueden terminar en un día o menos para revisar en las reuniones diarias

El scrum master ayuda y marca tiempos de reunión

Hay que tener en cuenta las capacidades y tiempos de personas que no estarán disponibles

Velocidad de trabajo

mantener la duración del sprint consistente con lo datos históricos

Reuniones de Scrum:

Reunión de Scrum Diario o Daily Scrum

Participantes:

Equipo de desarrollo, Scrum Master (no es obligatorio), si puede aportar

Cuando

Todos los días a la misma hora

Duración

15 min.

Entradas:

Que hice ayer, que haré hoy y que impedimentos veo para lograr los objetivos

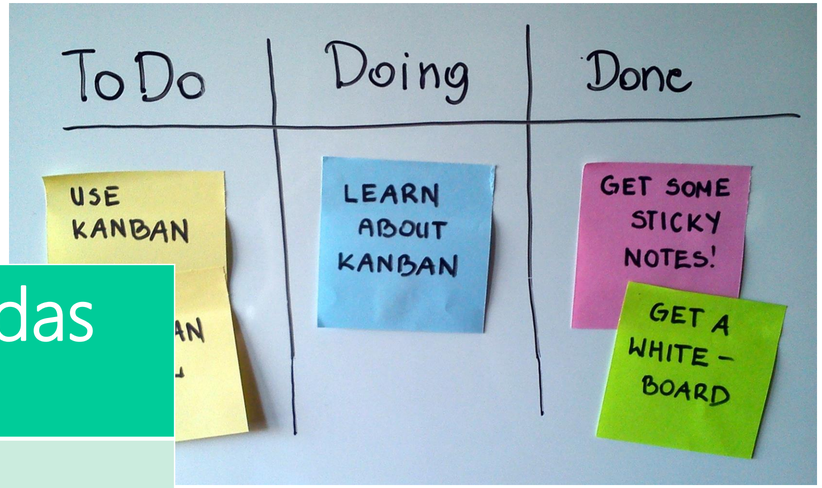
Salidas:

Entender que es importante para conseguir la meta

Sprint Backlog actualizado

Impedimentos (el Scrum Master actuará para resolverlos)

Reuniones de Scrum: Tablero Kanban



Tareas	Para Hacer	Haciendo	Terminadas
Tarea1	xx		
Tarea 2		YY	
Tarea 3			zz

Se pueden agregar otras columnas como

- En espera
- Holgura

Reuniones de Scrum:

Tablero Kanban

Es importante trabajar en función de las prioridades y los objetivos del Sprint
Tener en cuenta que solo se genera valor cuando algo esta terminado

No trabajar Multitarea, ya que cada tarea implica tiempo de programación previa. Debiéramos comenzar y terminar lo que hacemos. Multitarea tiene impacto dañino en el rendimiento.

Ver Video

http://forbes.es/business/10387/por-que-la-gente-inteligente-no-cae-en-la-multitarea/

Reuniones de Scrum:

Reunión de Revisión del sprint

Participantes:

Equipo de desarrollo, Scrum Master, producto Owner y Stakeholders (Optativo)

Cuando:

Al finalizar cada Sprint

Duración

4 horas por cada 4 semanas de sprint.

Entradas:

El incremento y el producto backlog

Salidas:

Producto incrementado para potencial envío

Se observa que esta terminado

La velocidad en que se realizó (puntos)

Retroalimentación. Esto va al producto Backlog y se planifica para un futuro Sprint

Reuniones de Scrum:

Reunión de Revisión del sprint

- Se muestra lo trabajado a los stakeholders
- Se muestra **SOLO** lo que esta totalmente terminado
- Se puede mostrar algo sobre lo que se quiere consultar
- El PO confirma o no lo terminado
- Retroalimentación de los stakeholders
- Eso va al PB

Reuniones de Scrum:

Reunión de Retrospectiva del sprint (enfocado a proceso)

Participantes:

Equipo de desarrollo, Scrum Master, producto Owner

Cuando:

Despues de la revisión del sprint

Duración

3 horas por cada 4 semanas de sprint.

Entradas:

Información de los equipos acerca del último Sprint

Salidas:

Que fue bien

Que podemos mejorar

Plan de mejora

https://proyectosagiles.org/retrospectiva-sprint-retrospective/

Reuniones de Scrum:

Reunión de Retrospectiva del sprint (enfocado a proceso)

Método

Saludo a participantes, revisión de agenda y plantear una meta

Pregunta breve sobre que se espera de la reunión

Escuchar respuestas

Recolectar datos

Dividir al grupo en pequeños grupos (2, 3 personas)

Pedir a miembros que recuerden hechos significativos del proyecto (buenos o no)

En una línea de tiempo poner estos hechos

Generar aprendizaje

Ver problemas o cosas positivas y encontrar las causas

Puntuar la importancia

Determinar que hacer utilizando SMART (Specific (específico), Measurable (medible), Achievable (realizable), Realistic (realista) y Time-Bound (limitado en tiempo) https://ivanmb.com/definir-objetivos-smart/)

Reuniones de Scrum:

Reunión de refinamiento del producto Backlog

Participantes:

Equipo de desarrollo, Product Owner

Cuando:

Es un proceso continuo y cuando se necesite

Duración

10% de tiempo del sprint

Entradas:

Visión de producto, producto backlog y velocidad

Salidas:

Product backlog refinado

Reuniones de Scrum:

Reunión plan de lanzamiento (no esta en la guía de scrum)

Objetivo: definir cuando se va a realizar la entrega del producto

Participantes:

Equipo de desarrollo, Product Owner, Scrum master, stakeholders

Cuando:

Cuando se necesite

Duración

ND

Entradas:

Visión de producto, producto backlog, Meta de lanzamiento y velocidad

Salidas:

Plan de lanzamiento

Formas: basado en el alcance (Cuando tendré el alcance) o basado en el tiempo (fecha fija, que tendré en ese tiempo)

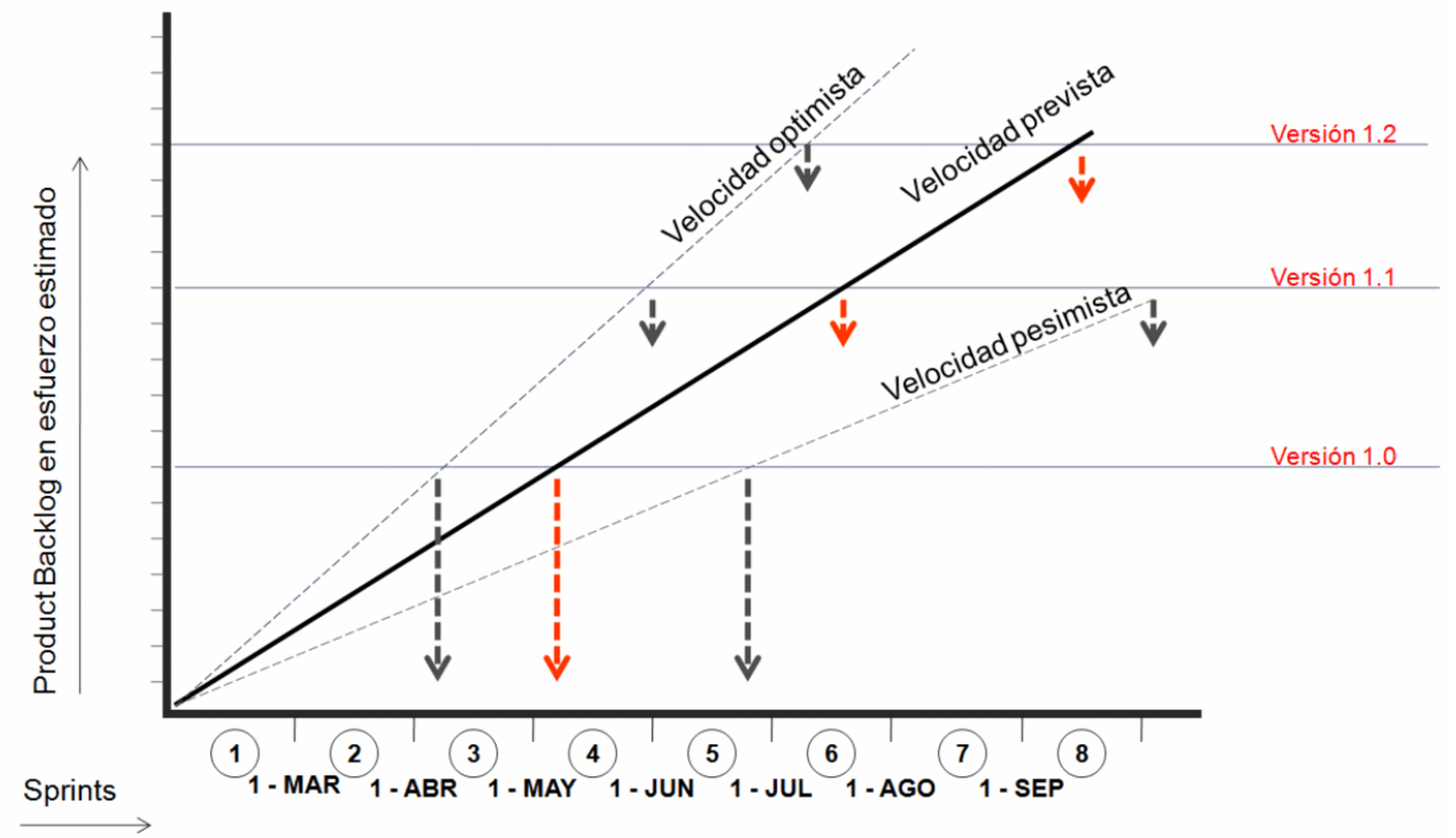
Reuniones de Scrum:

Reunión plan de lanzamiento (no está en la guía de scrum)
Basado en alcance:

1) Refinar
2) Progreso gráfico
3) Pronosticar

Basado en el tiempo

1) Refinar
2) Definir la velocidad histórica
3) Calcular velocidad media
4) Pronosticar hasta donde llegamos en el backlog


Product Backlog en esfuerzo estimado
Velocidad optimista
Velocidad prevista
Velocidad pesimista
Versión 1.2
Versión 1.1
Versión 1.0
1
2
3
4
5
6
7
8
1 - MAR
1 - ABR
1 - MAY
1 - JUN
1 - JUL
1 - AGO
1 - SEP
Sprints

Implantación de scrum:

UN 70% de las aplicaciones de scrum son de industria
Es muy importante que la empresa adopte el manifiesto scrum
Los equipos debieran estar lo mas cercanos posible

Video recomendado:

https://www.youtube.com/watch?v=p9MYRrQEOGI

FORMADORES { IT }

www.formadoresfreelance.es

Consultores y formadores freelance