Unidad 3

Metodologías ágiles

UAI
Universidad Abierta Interamericana

UAIOnline Ultra

Metodologías ágiles

Unidad 3

- **OBJETIVOS**
  - Entender y aplicar las metodologías ágiles a nivel desarrollo del software.

Planificación y Programación

Unidad 3

- **HABILIDADES Y COMPETENCIAS QUE DESARROLLA LA ASIGNATURA**
  - [Plantea] [el Proceso de Desarrollo de Software] [Para planear, programar, supervisar y controlar los tiempos de un proyecto] [Aplicando métodos ágiles] / [Aplicando el estándar UML]

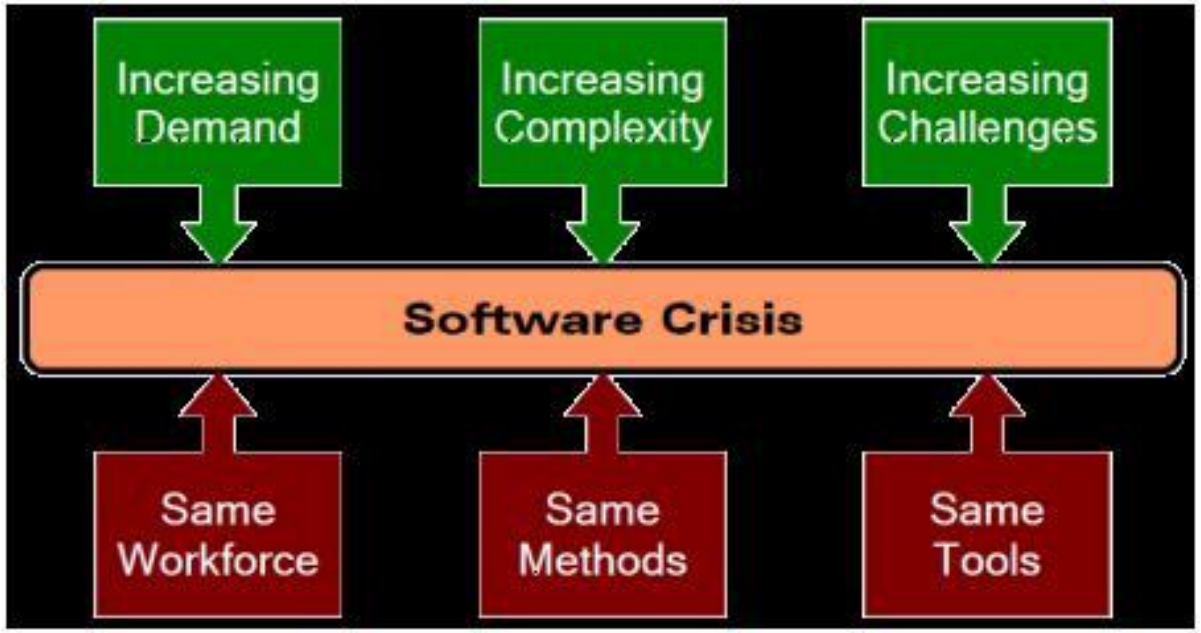
Metodologías ágiles

UAI
Universidad Abierta Interamericana

UAIOnline Ultra

CRISIS DEL SOFTWARE

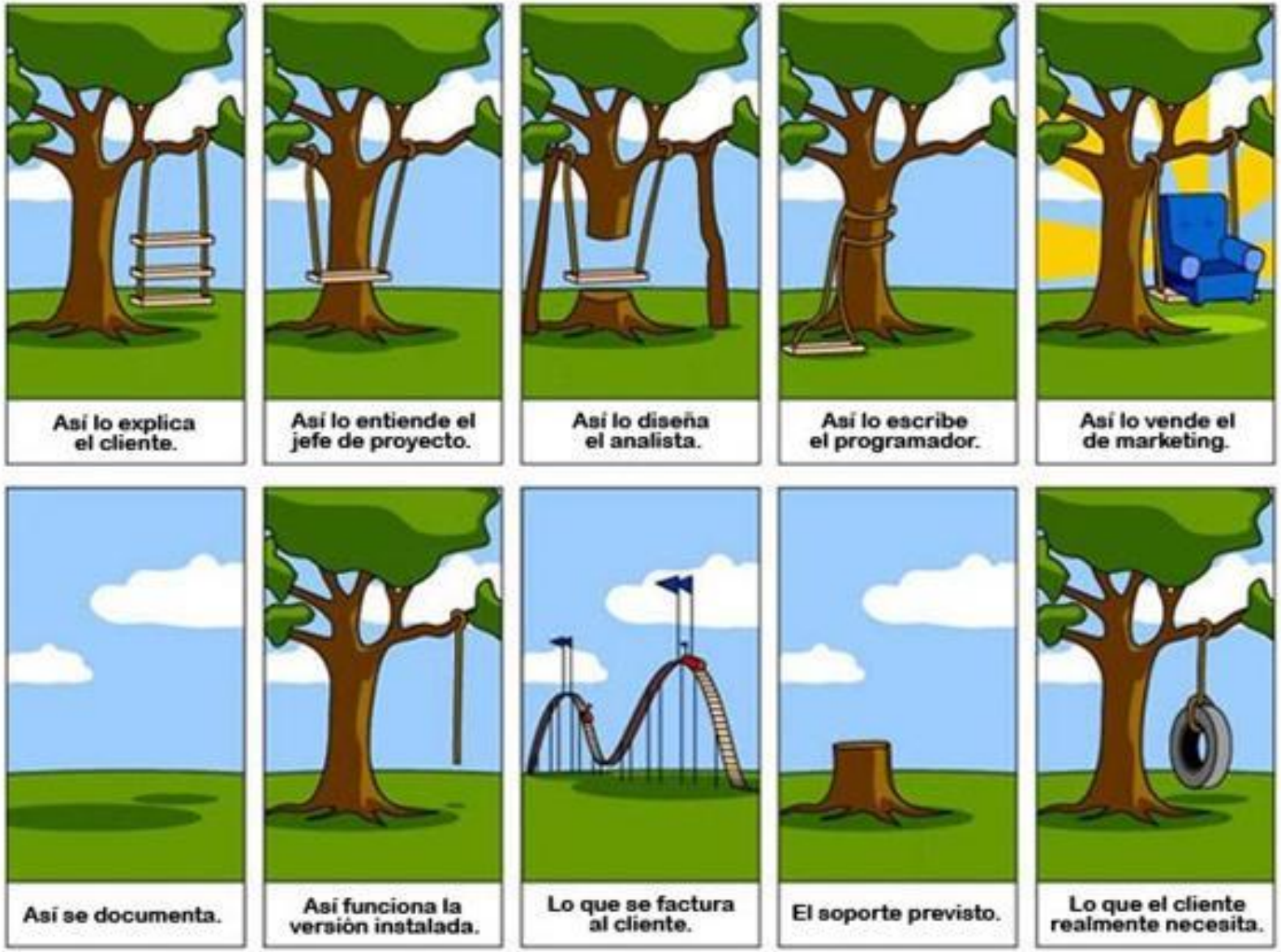
- Término acuñado a principios de los 70
- Dificultades del desarrollo de software frente
  - Al rápido crecimiento de la demanda
  - Al crecimiento de la complejidad
  - Inexistencia de técnicas establecidas para el desarrollo de sistemas que funcionaran bien y puedan ser validados



CRISIS DEL SOFTWARE

- Finales de los 50 la potencia computacional era limitada.
  - Desarrollos "simples" acorde a esta limitación.
  - Proceso "artesanal" de desarrollo.
  - Sin metodologías o caminos a seguir.
- A finales de los 60 la potencia computacional comenzó a aumentar.
  - Nuevos lenguajes de programación de más "alto nivel".
  - Necesidad de programas más complejos
  - Salto importante del hard no acompañado por el soft.
  - Se continuó sin un enfoque acorde al proceso de desarrollo de software.
  - Problemas: dificultad para estimar costos, retrasos en las entregas, errores funcionales, modificaciones con costo muchas veces prohibitivo
- Resultado: software de mala calidad
- Solución: Ingeniería de software

PROBLEMÁTICA DEL DESARROLLO



METODOLOGÍAS ÁGILES - VENTAJAS

Las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes

ÁGILES - VENTAJAS

- Capacidad de respuesta a cambios de requisitos a lo largo del desarrollo
- Entrega continua y en plazos breves de software funcional
- Trabajo conjunto entre el cliente y el equipo de desarrollo
- Importancia de la simplicidad, eliminado el trabajo innecesario
- Atención continua a la excelencia técnica y al buen diseño
- Mejora continua de los procesos y el equipo de desarrollo

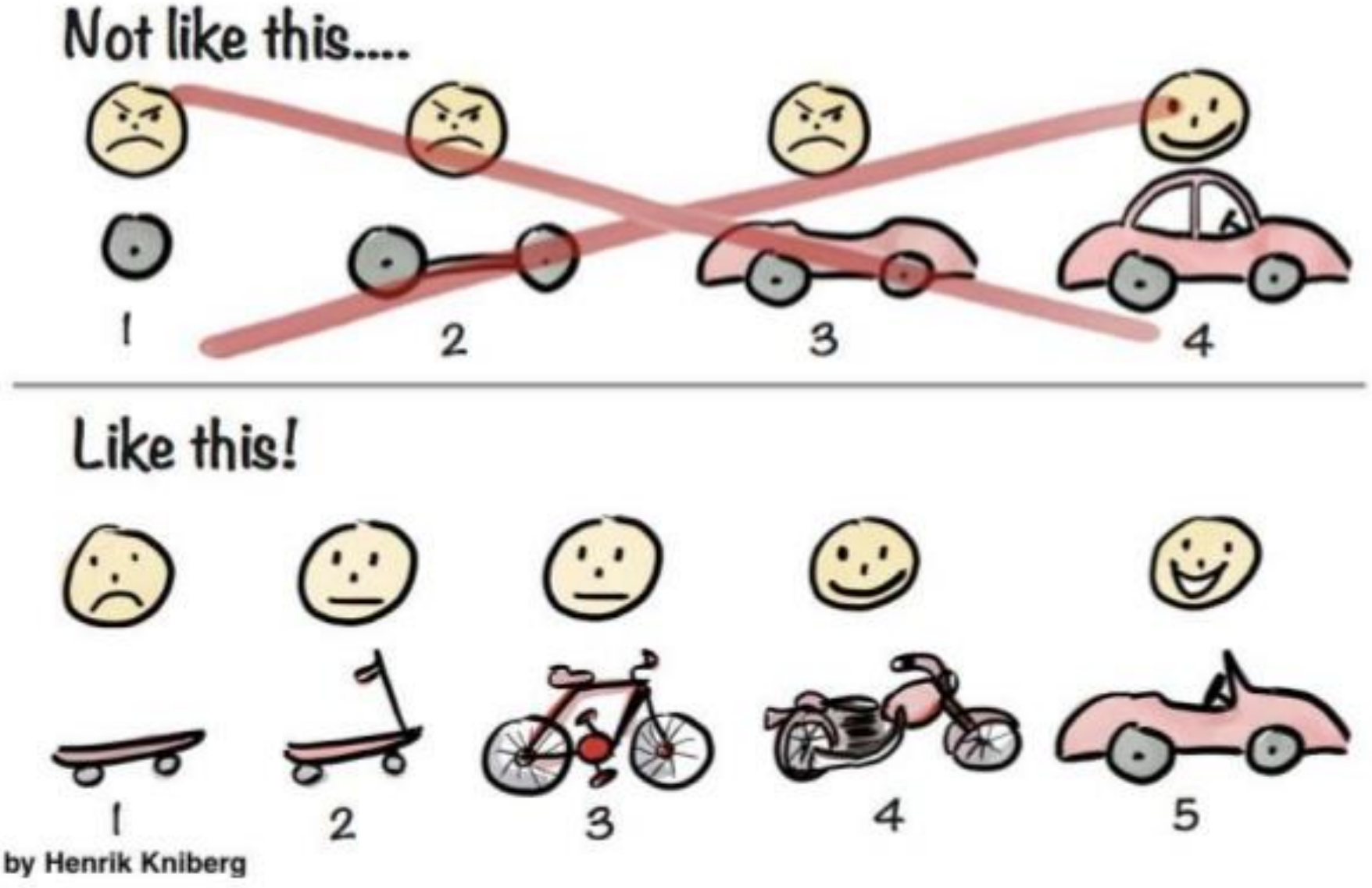
TRADICIONALES VS. ÁGILES

- Entorno muy cambiante
  - Costos, riesgos
- Requisitos
- Burocracia
- Tiempo de Respuesta
- Planificación

- Planificación vs. Respuesta al cambio

TRABAJO ITERATIVO E INCREMENTAL - MVP

- ¿Cómo validar requisitos? ¿Ideas? De manera ágil…
- Minimum Viable Product (MVP)
  - Posee las características mínimas para satisfacer las necesidades del cliente y obtener nuevo feedback para mejorar
    - **Producto**: Ofrece una versión "acotada" de la aplicación, servicio o producto que será ofrecido a los clientes. Debe ser un representante del producto final a ofrecer.
    - **Mínimo**: Sugiere desarrollar el MVP con el menor esfuerzo posible, para ahorrar tiempo y dinerro.
    - **Viable**: El MVP debe ofrecer como objetivo principal una experiencia realista a los clientes, debe parecer que es un producto real que será entregado al cliente final, y no una estrategia de márketing.


Not like this....
1
2
3
4
Like this!
1
2
3
4
5
by Henrik Kniberg

HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT

1 2 3 4

ALSO HOW NOT TO BUILD A MINIMUM VIABLE PRODUCT

1 2 3 4

HOW TO BUILD A MINIMUM VIABLE PRODUCT

1 2 3 4

SCRUM - INTRODUCCIÓN

Método adaptativo de gestión de proyectos que se basa en los principios ágiles:

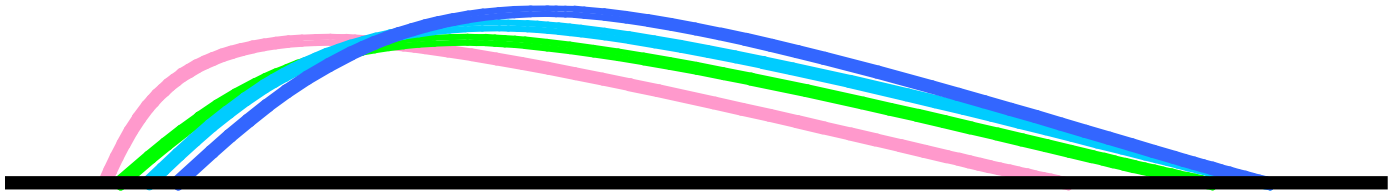
- Colaboración estrecha con el cliente.
- Predisposición y respuesta al cambio
- Prefiere el conocimiento tácito de las personas al explícito de los procesos
- Desarrollo incremental con entregas funcionales frecuentes
- Motivación y responsabilidad de los equipos por la auto-gestión, auto-organización y compromiso.
- Simplicidad. Supresión de artefactos innecesarios en la gestión del proyecto

DESARROLLO SECUENCIAL VS. SUPERPUESTO

Requisitos | Diseño | Código | Test

En lugar de hacer todo de una cosa a la vez ...

...los equipos Scrum hacen un poco de todo todo el tiempo



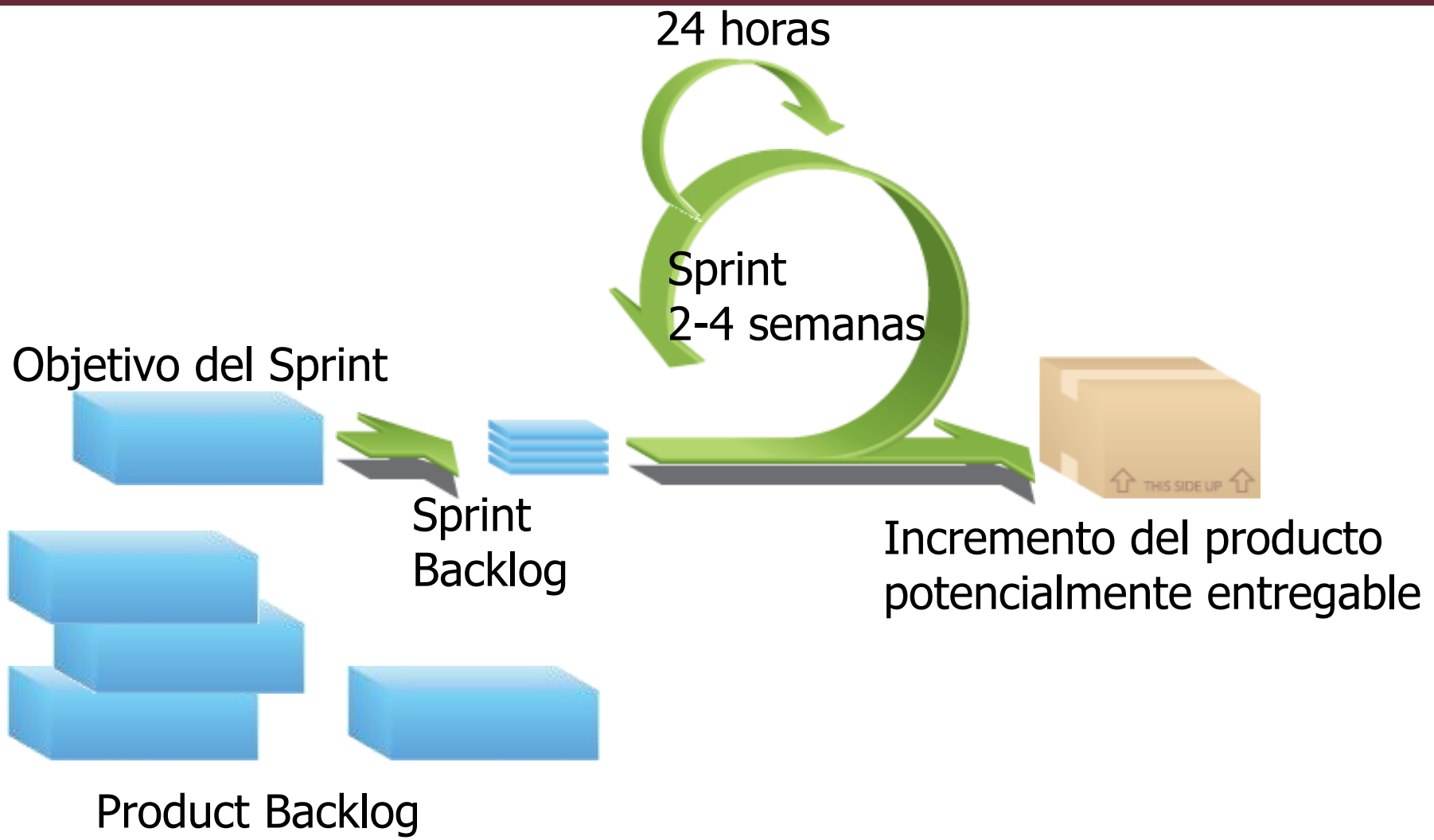
Scrum en 100 palabras

- Scrum es un proceso ágil que nos permite centrarnos en ofrecer el más alto valor de negocio en el menor tiempo.
- Nos permite rápidamente y en repetidas ocasiones inspeccionar software real de trabajo (cada dos semanas o un mes).
- El negocio fija las prioridades. Los equipos se auto-organizan a fin de determinar la mejor manera de entregar las funcionalidades de más alta prioridad.
- Cada dos semanas o un mes, cualquiera puede ver el software real funcionando y decidir si liberarlo o seguir mejorandolo en otro sprint.

SCRUM - INTRODUCCIÓN

- Conjunto de prácticas y roles
- Posee documentos y artefactos específicos
- Equipos auto-organizados
- Gestión de la expectativa del cliente (cambios)
- Mejoras
  - Productividad
  - Calidad
  - Motivación

ESCENCIA

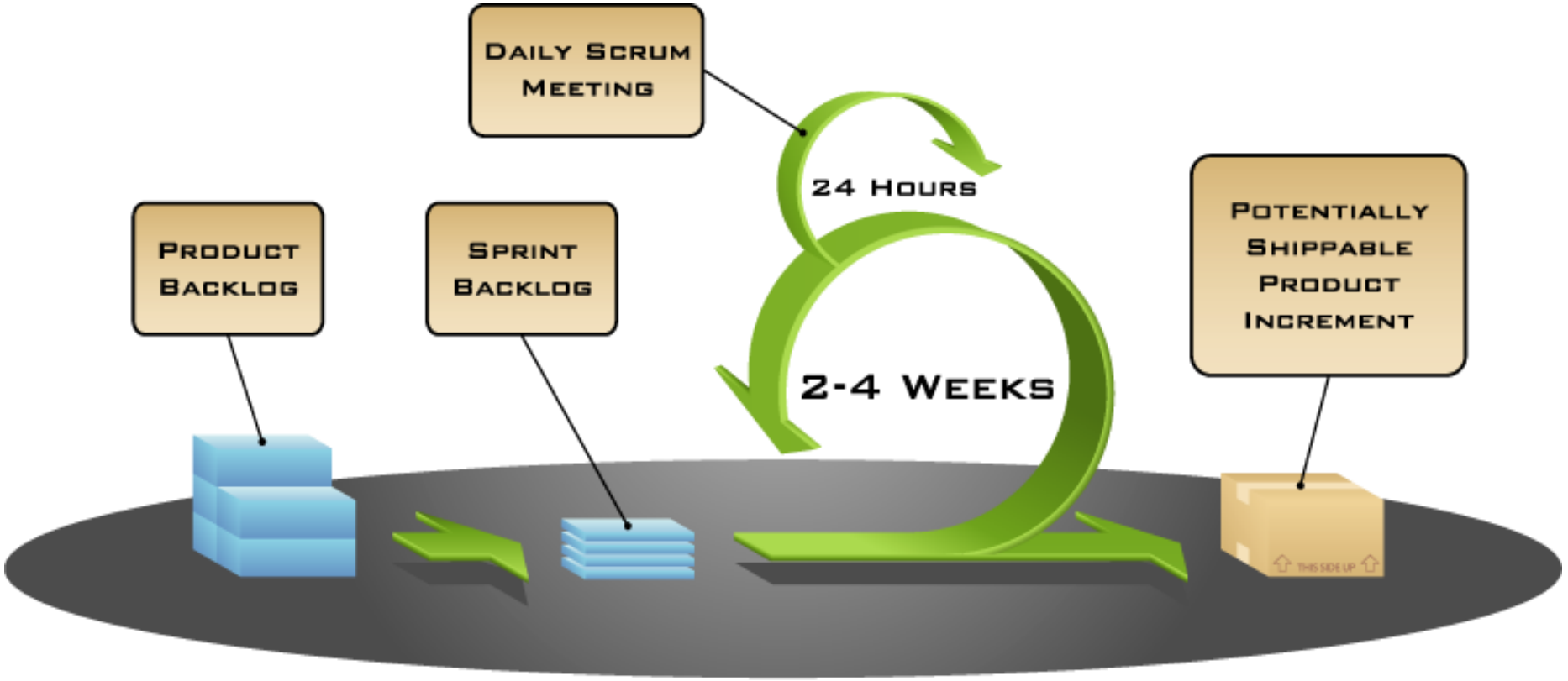

24 horas
Sprint
2-4 semanas
Objetivo del Sprint
Sprint
Backlog
Incremento del producto
potencialmente entregable
Product Backlog

EL MANIFESTO ÁGIL – UNA DECLARACIÓN DE VALORES

Individuos e interacciones	sobre	Procesos y herramientas
Software que funciona	sobre	Documentación exhaustiva
Colaboración con el cliente	sobre	Negociación de contratos
Responder ante el cambio	sobre	Seguimiento de un plan

Fuente: www.agilemanifesto.org

ARTEFACTOS


Daily Scrum Meeting
24 Hours
Product Backlog
Sprint Backlog
2-4 Weeks
Potentially Shippable Product Increment
Copyright © 2005, Mountain Goat Software

SPRINTS

- En Scrum los proyectos avanzan en una serie de "Sprints".
  - Análogo a las iteraciones.
- La duración típica es 2–4 semanas o a lo sumo un mes calendario.
- La duración constante conduce a un mejor ritmo.
- El producto es diseñado, codificado y testeado durante el Sprint.

SCRUM FRAMEWORK

Roles

- Product owner
- ScrumMaster
- Team

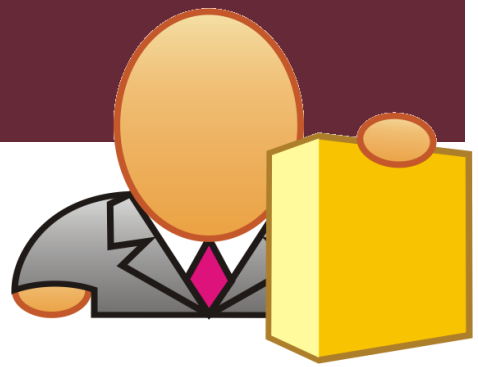
Reuniones

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefactos

- Product backlog
- Sprint backlog
- Burndown charts

PRODUCT OWNER



- Define las funcionalidades del producto
- Decide sobre las fechas y contenidos de los releases.
- Es responsable por la rentabilidad del producto (ROI).
- Prioriza funcionalidades de acuerdo al valor del mercado/negocio.
- Ajusta funcionalidades y prioridades en cada iteración si es necesario.
- Acepta o rechaza los resultados del trabajo del equipo.

EL SCRUMMASTER

- Representa a la gestión del proyecto
- Responsable de promover los valores y prácticas de Scrum
- Remueve impedimentos
- Se asegura de que el equipo es completamente funcional y productivo
- Permite la estrecha cooperación en todos los roles y funciones
- Escudo del equipo de interferencias externas

EL TEAM

- Típicamente de 5 a 9 personas
- Multi-funcional:
  - Programadores, testers, analistas, diseñadores, etc.
- Los miembros deben ser full-time
  - Puede haber excepciones (Ej.: Infraestructura, SCM, etc.)
- Los equipos son auto-organizativos
  - Idealmente, no existen títulos pero a veces se utilizan de acuerdo a la organización
- Solo puede haber cambio de miembros entre los sprints

SCRUM FRAMEWORK

Roles

- Product owner
- ScrumMaster
- Team

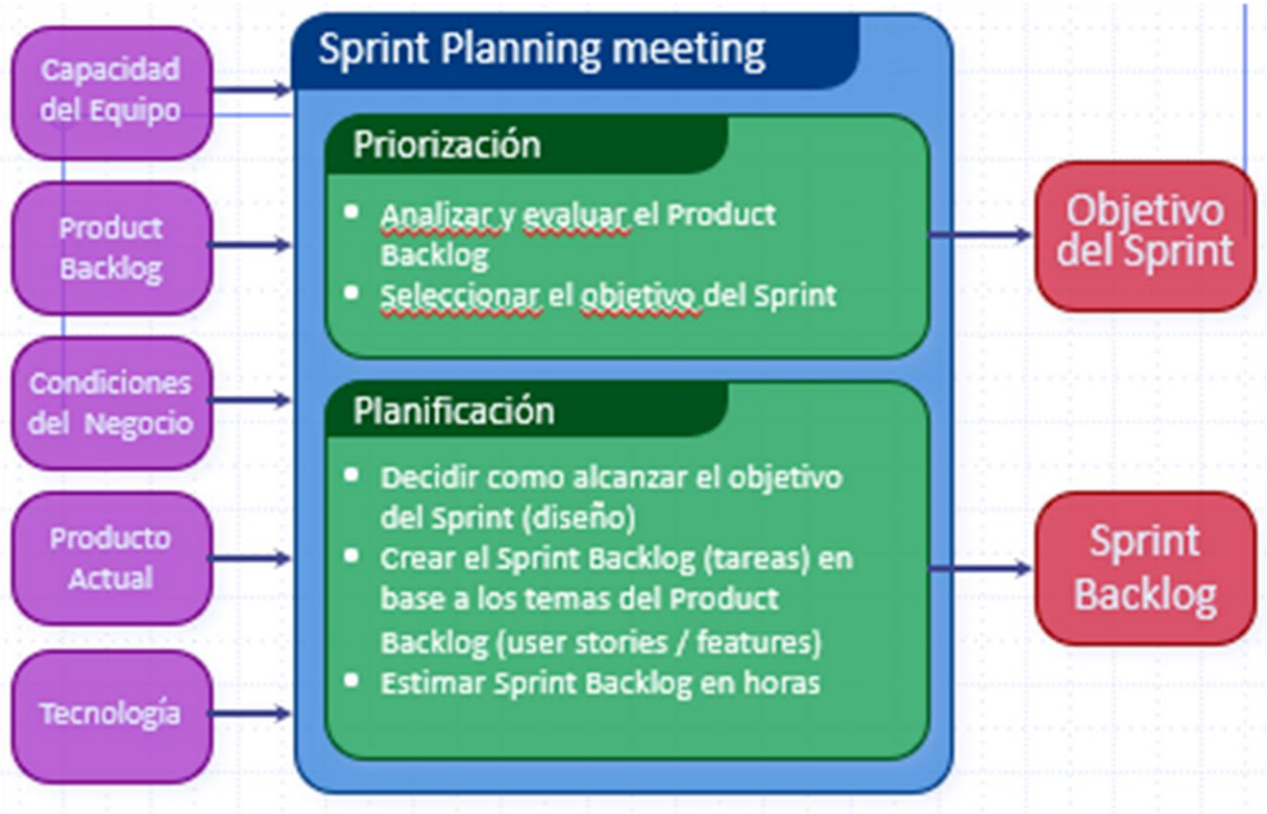
Reuniones

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefactos

- Product backlog
- Sprint backlog
- Burndown charts

DAILY SCRUM


Capacidad del Equipo
Product Backlog
Condiciones del Negocio
Producto Actual
Tecnología
Sprint Planning meeting
Priorización
Analizar y evaluar el Product Backlog
Seleccionar el objetivo del Sprint
Planificación
Decidir como alcanzar el objetivo del Sprint (diseño)
Crear el Sprint Backlog (tareas) en base a los temas del Product Backlog (user stories / features)
Estimar Sprint Backlog en horas
Objetivo del Sprint
Sprint Backlog

DAILY SCRUM

- Parámetros
  - Diaria
  - Dura 15 minutos
  - Parados
- No para la solución de problemas
  - Todo el mundo está invitado
  - Sólo los miembros del equipo, ScrumMaster y Product Owner, pueden hablar
  - Ayuda a evitar otras reuniones innecesarias

TODOS RESPONDEN 3 PREGUNTAS

1. ¿Qué hiciste ayer?
2. ¿Qué vas a hacer hoy?
3. ¿Hay obstáculos en tu camino?

- No es dar un status report al Scrum Master
- Se trata de compromisos delante de pares

SPRINT REVIEW

- El equipo presenta lo realizado durante el sprint
- Normalmente adopta la forma de una demo de las nuevas características o la arquitectura subyacente
- Informal
  - Regla de 2 hs preparación
  - No usar diapositivas
- Todo el equipo participa
- Se invita a todo el mundo

SPRINT RETROSPECTIVE

- Periódicamente, se echa un vistazo a lo que funciona y lo que no
- Normalmente 15 a 30 minutos
- Se realiza luego de cada sprint
- Todo el equipo participa
  - Posiblemente también clientes

SCRUM FRAMEWORK

Roles

- Product owner
- ScrumMaster
- Team

Reuniones

- Sprint planning
- Sprint review
- Sprint retrospective
- Daily scrum meeting

Artefactos

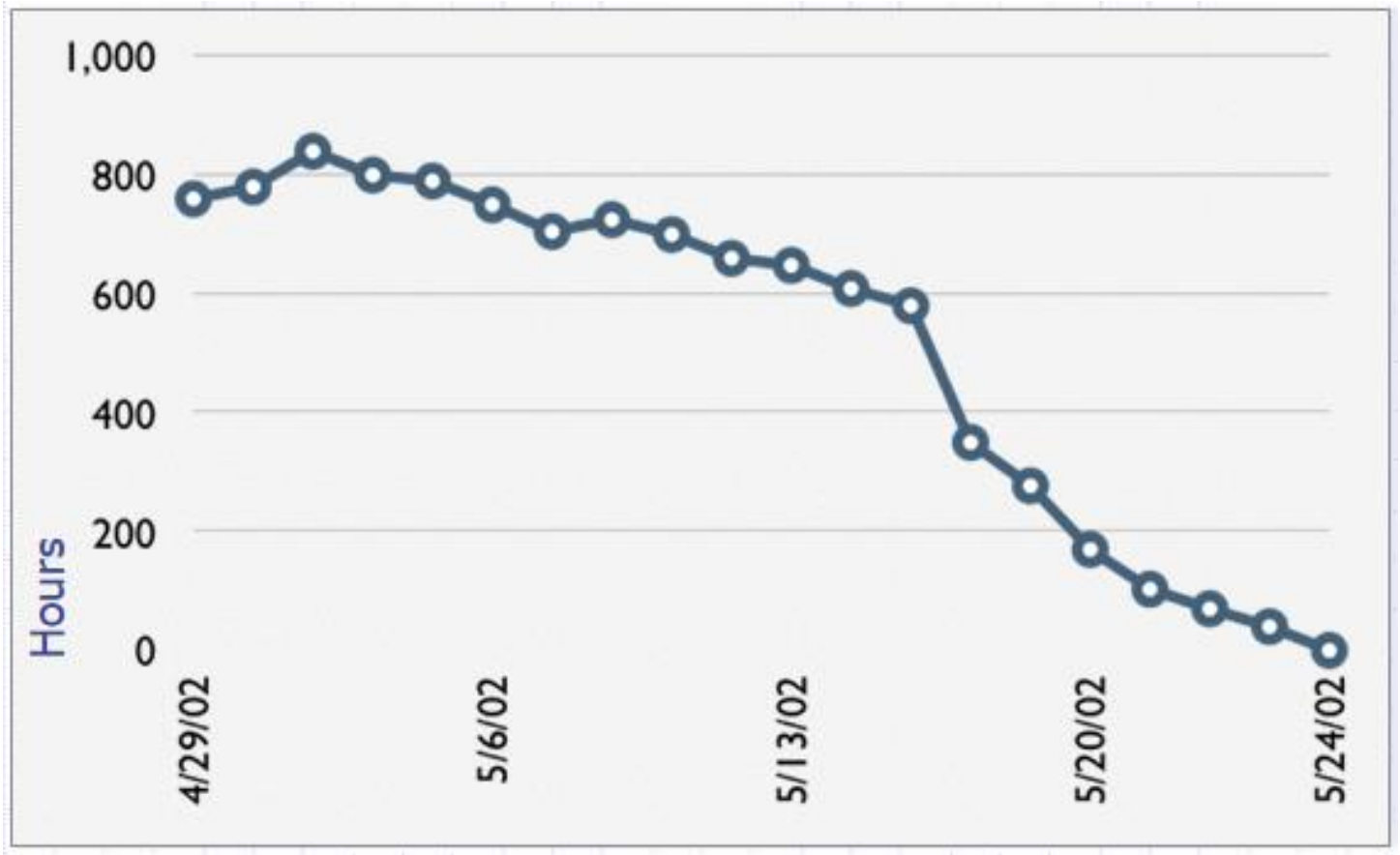
- Product backlog
- Sprint backlog
- Burndown charts

PRODUCT BACKLOG

Este es el product backlog

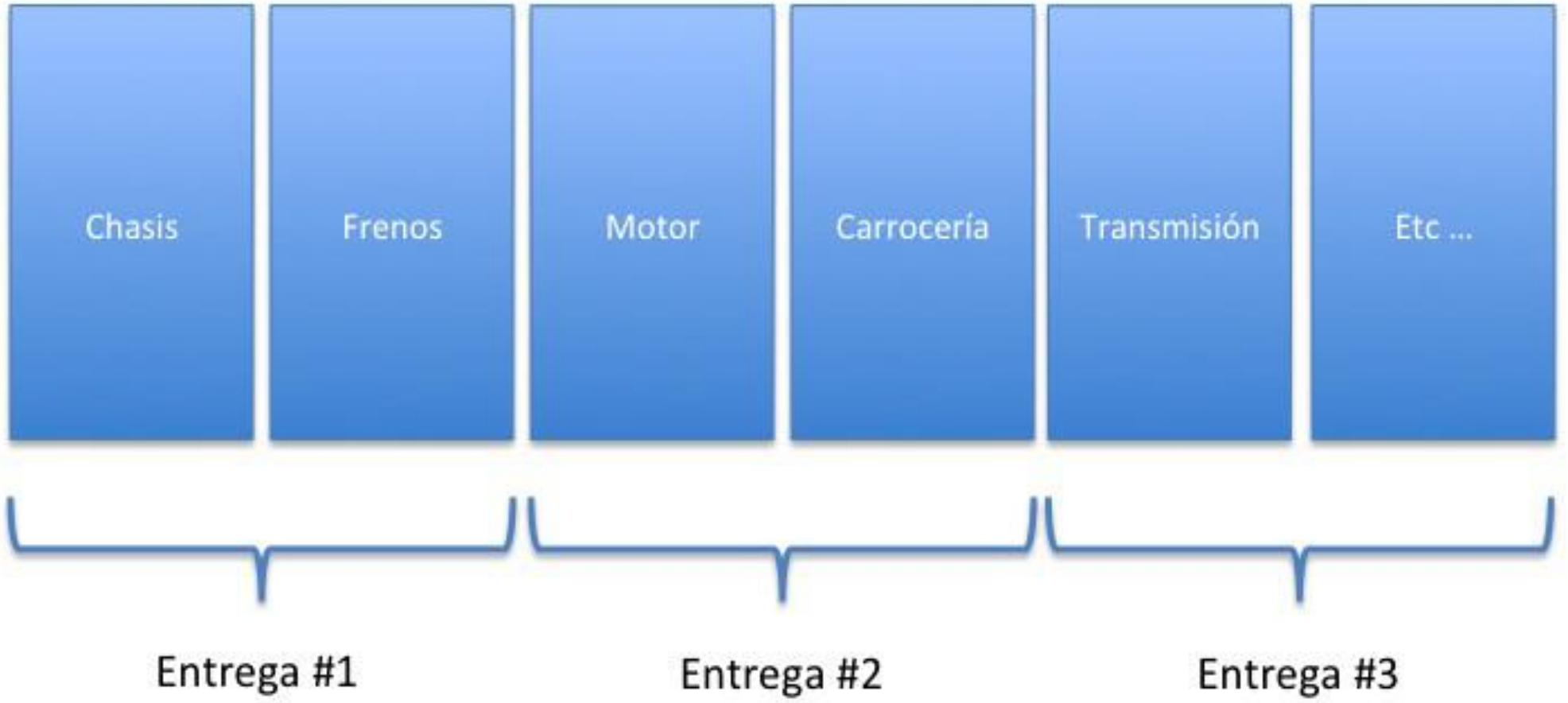
- Los requisitos
- Una lista de todos los trabajos deseados en el proyecto
- Idealmente cada tema tiene valor para el usuarios o el cliente
- Priorizada por el Product Owner
- Repriorizada al comienzo de cada Sprint

UN SPRINT BURNDOWN CHART



EJEMPLO CONSTRUCCIÓN DE VEHÍCULO

EJEMPLO CONSTRUCCIÓN DE VEHÍCULO



EJEMPLO CONSTRUCCIÓN DE VEHÍCULO

ESTIMACIÓN Y PLANIFICACIÓN / I

Saber estimar y planificar es fundamental a la hora de encarar proyectos donde el producto necesita de un grado importante de creatividad y/o innovación.

*Una de las características de la gestión de proyectos ágiles es el ser una actividad **adaptativa** en vez de predictiva.*

ESTIMACIÓN Y PLANIFICACIÓN /2

En Scrum el control empírico del proyecto se aplica de manera regular mediante las siguientes prácticas:

-El ***cliente*** aporta feedback del proyecto inspeccionando los resultados en la demostración que se realiza al final de cada iteración. Con los objetivos se realizan las adaptaciones necesarias en la lista de *requisitos priorizada*.

-El ***equipo*** mejora de manera continua su metodología de trabajo y el entorno del proyecto en la *retrospectiva* que se realiza al final de cada iteración. Las tareas resultantes se incluyen en las *planificaciones de la siguientes iteraciones*.

PLANIFICACIÓN ÁGIL VS PLANIFICACIÓN TRADICIONAL

- Planificación tradicional es generalmente predecible.
- **Metodologías ágiles atacan este problema.**

El desarrollo de software es una actividad de creación y transmutación de conocimiento.

Como tal, no puede ser predicha ni estimada en forma precisa.

AGILES ES ADAPTATIVO

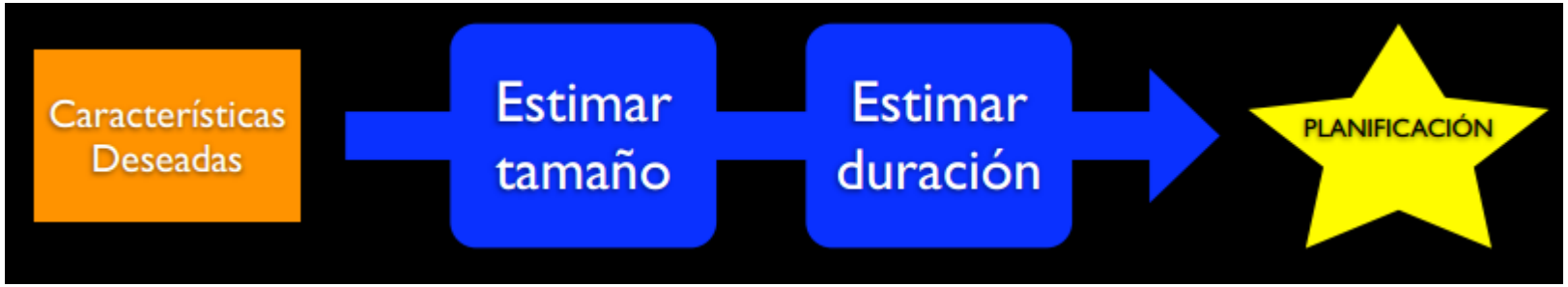
Pocas organizaciones están dispuestas a embarcarse en un proyecto sin tener siquiera una idea aproximada de cuánto va a costar o cuándo va a estar terminado el producto.

PLANIFICACIÓN ÁGIL VS PLANIFICACIÓN TRADICIONAL

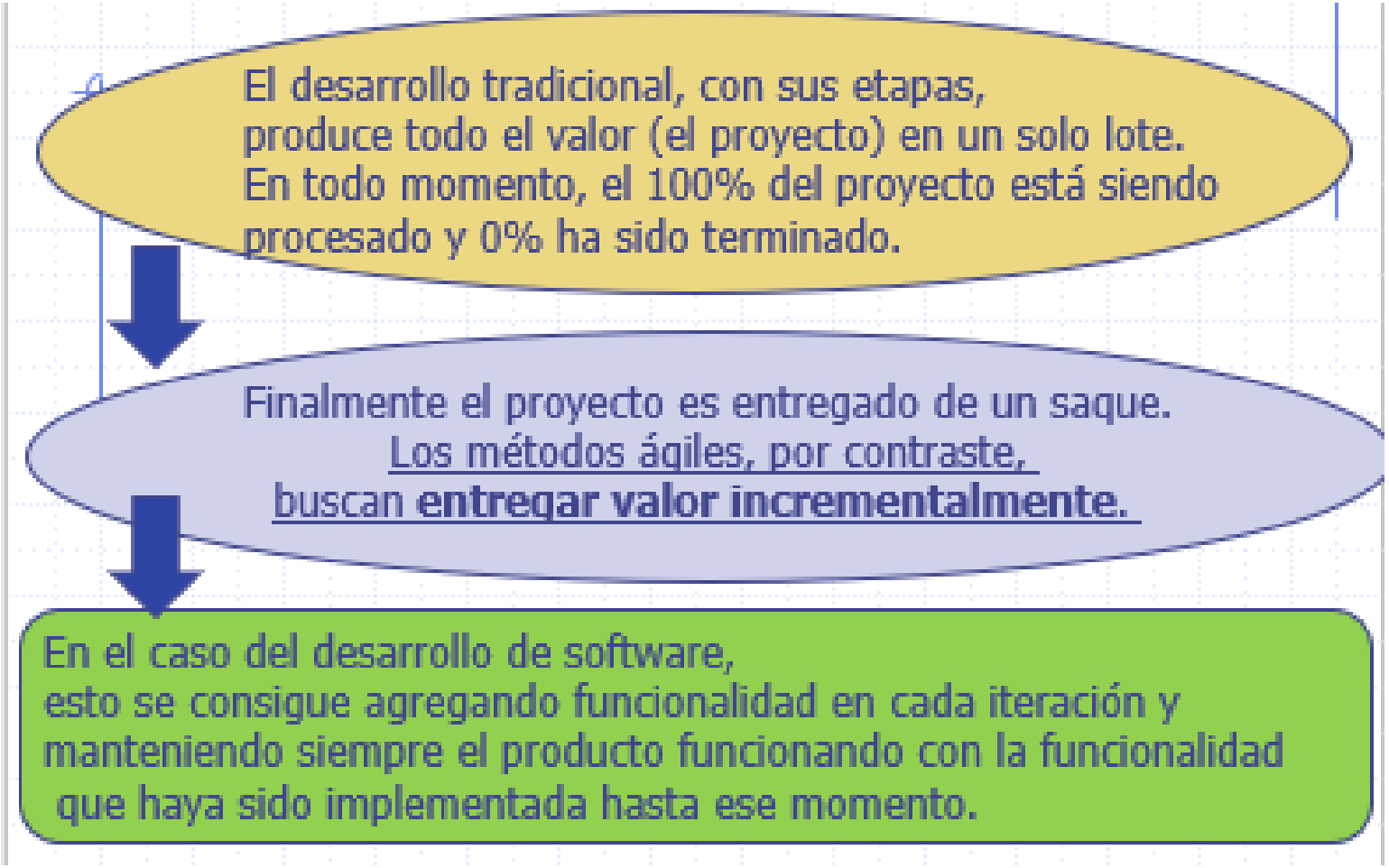
La planificación ágil parte de la idea de planificar en función de objetivos de negocio en lugar de tareas (a diferencia de la planificación tradicional), priorizando los que aportan más valor, y esperando a dar detalle a objetivos y tareas conforme se va acercando el momento de construcción de estos objetivos, cuando la indeterminación se va reduciendo, de manera que se amortiza el esfuerzo de planificar de manera detallada.

PLANIFICACIÓN ÁGIL

Estimación de Tamaño
≠
Estimación de Duración



PLANIFICACIÓN ÁGIL



OBJETIVOS COMO HISTORIAS DE USUARIO

La estimación y planificación ágil es la **creación del product backlog**, o sea la definición del proyecto a realizar.

Se puede dividir en objetivos expresados como **historias de usuario** (user stories), cada una aportando valor de negocios incremental e individual.

Se escriben con el siguiente formato:
"***Como xxx (rol)***
quiero hacer yyy (RF)
con el objetivo de zzz",

OBJETIVOS COMO HISTORIAS DE USUARIO

Ejemplo:
"Como cliente del banco, quiero pedir un préstamo para poder comprar una casa" .

LA ESTIMACIÓN LA LLEVA A CABO EL EQUIPO

La estimación no la realiza el Cliente / Product Owner, por que no es él quien se va a "ensuciar las manos" y luchar por cumplir con fechas.

Todo el equipo realiza la estimación para:

- Reforzar el compromiso de todo el equipo respecto a las fechas que dan al cliente.

- Reforzar el compromiso de cada miembro del equipo respecto al resto.

- Hacer que todo el mundo se sienta oído.

LA ESTIMACIÓN LA LLEVA A CABO EL EQUIPO
UNIDADES PARA ESTIMAR

La estimación se puede realizar utilizando alguna de las siguientes unidades:

Días ideales: los días necesarios para que el equipo pueda completar un objetivo, sin considerar interrupciones.

Puntos de historia de usuario: la complejidad que tiene cada historia de usuario. Un equipo en un proyecto determinado es capaz de completar un número semi-regular de puntos de historia cada iteración ("velocity").

LA **PIEDRA ANGULAR** es el punto de referencia a partir del cual se realizan las estimaciones

METODOS DE ESTIMACIÓN

- **La estimación ágil se basa en el conocimiento, la experiencia**
- **Estimación con metodo delphi**

Su objetivo es la consecución de un consenso basado en la discusión entre expertos. Es un proceso repetitivo. Su funcionamiento se basa en la elaboración de un cuestionario que ha de ser contestado por los expertos. Una vez recibida la información, se vuelve a realizar otro cuestionario basado en el anterior para ser contestado de nuevo.

- **JUICIO EXPERTO**

VELOCIDAD DE DESARROLLO (VELOCITY)

- Cantidad de puntos sumando todas las historias de usuario realizadas
- Si el equipo completó tres historias en la iteración, cada una de cinco puntos historia, entonces su velocidad en esta iteración fue quince.

- el factor más importante para obtener la velocidad es basarse en históricos, es decir, en la velocidad de los sprint previos. En el mundo Scrum a esto se le llama "Yesterday's weather" (estimo el tiempo que hará hoy en base al que hizo ayer

Las estimaciones (y la velocidad) se ven afectadas cuando se introducen nuevas tecnologías o conceptos de negocio en medio del proyecto, aunque después se vuelven a estabilizar.

ESTIMACIÓN CON PLANNING POKER

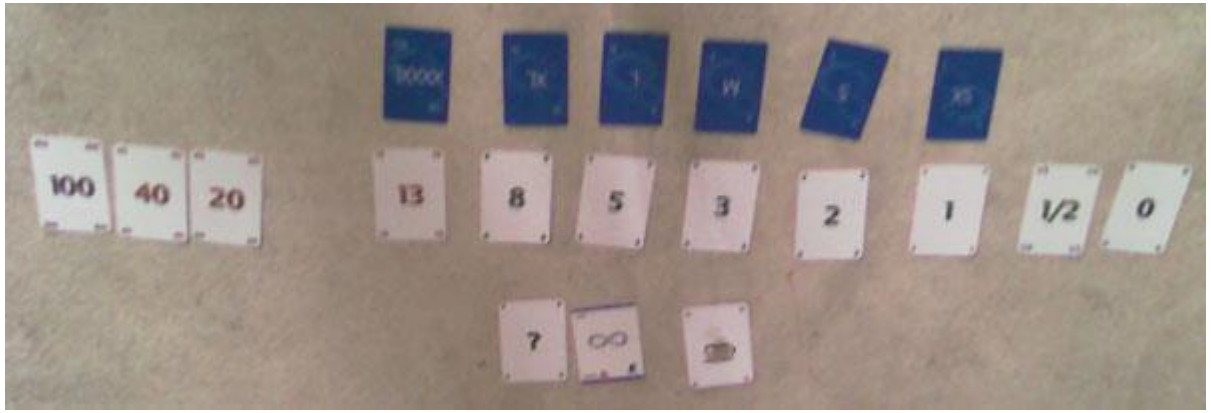
El **product backlog** es, para ser exactos, una lista *priorizada y estimada* de historias. Por ahora sólo tenemos historias. Falta estimarlas y priorizarlas.

El proceso de estimación se puede hacer utilizando una técnica llamada **planning poker** (póker de planificación). El objetivo del *planning poker* es obtener una medida de **tamaño relativo** de todas las historias respecto a sí mismas.

ESTIMACIÓN CON PLANNING POKER

La técnica de planning poker permite hacer una **estimación inicial del proyecto rápida y fiable**, dado que todos los miembros del equipo comparten sus diferentes informaciones y expresan su opinión sin sentirse condicionados por el resto.

A continuación se puede ver diferentes barajas de cartas de planning poker.



ESTIMACIÓN Y PLANIFICACIÓN DE ITERACIÓN BASADA EN COMPROMISO /I

En la ***reunión de planificación de iteración (Sprint Planning)*** el equipo va seleccionando objetivos del Product ***Backlog*** (historias de usuario) y descomponiéndolos en tareas.

Escoge tantos objetivos como cree que es capaz de completar en una iteración (**commitment driven sprint planning**).

La velocidad permite verificar (**checkpoint**) que el total de puntos de historia de la nueva iteración es congruente con las anteriores iteraciones, si están escogiendo demasiados objetivos o demasiado pocos.

ESTIMACIÓN Y PLANIFICACIÓN DE ITERACIÓN BASADA EN COMPROMISO /2

Cada miembro del equipo que escoge una tarea hace su estimación en horas delante de todos, de manera que:

- Se compromete respecto a sus compañeros.

- Evita estimaciones demasiado optimistas (que comprometerían las fechas del proyecto) o demasiado pesimistas (que atentarían contra la competitividad de la empresa). De esta manera se disminuyen y ajustan los buffers.

LA ESTIMACIÓN ÁGIL AYUDA A CREAR "CONCIENCIA DE EQUIPO"

Como consecuencia de las técnicas descritas, la estimación ágil crea conciencia de equipo.

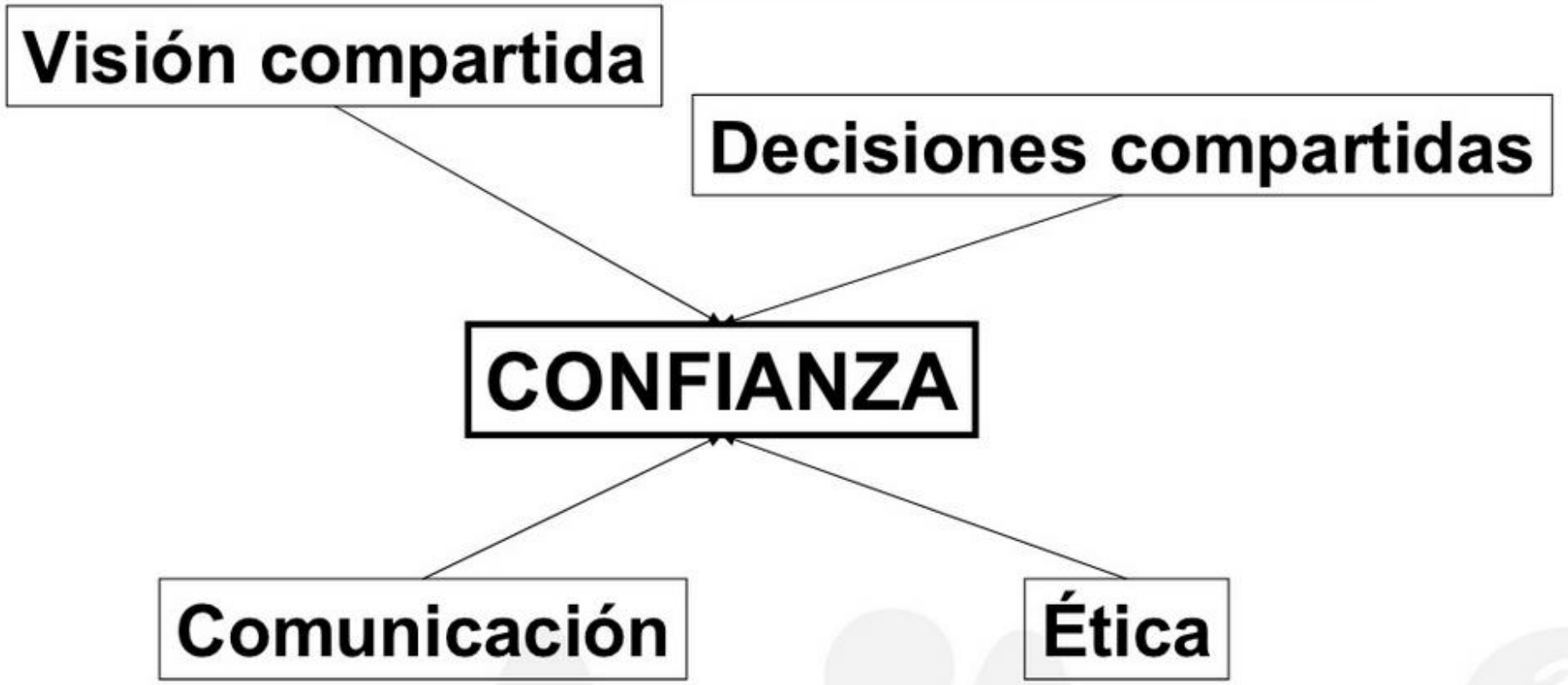
Todos participan, tienen voz y voto.

Cuando se produce una equivocación en una estimación, se ha equivocado todo el equipo. Por ello, no es para individualistas. No sirve que alguien diga "Yo he hecho mi parte de la historia de usuario".

PLANIFICACIÓN

- Ejemplo
  - Nos contratan desde un zoológico con el fin de bañar sus animales (product backlog)
  - Con que Animales cuentan? (User stories)
  - Estimar y pesar cada tarea.
  - Planificar un sprint
  - Priorizar

TÉCNICAS DE RETROSPECTIVA

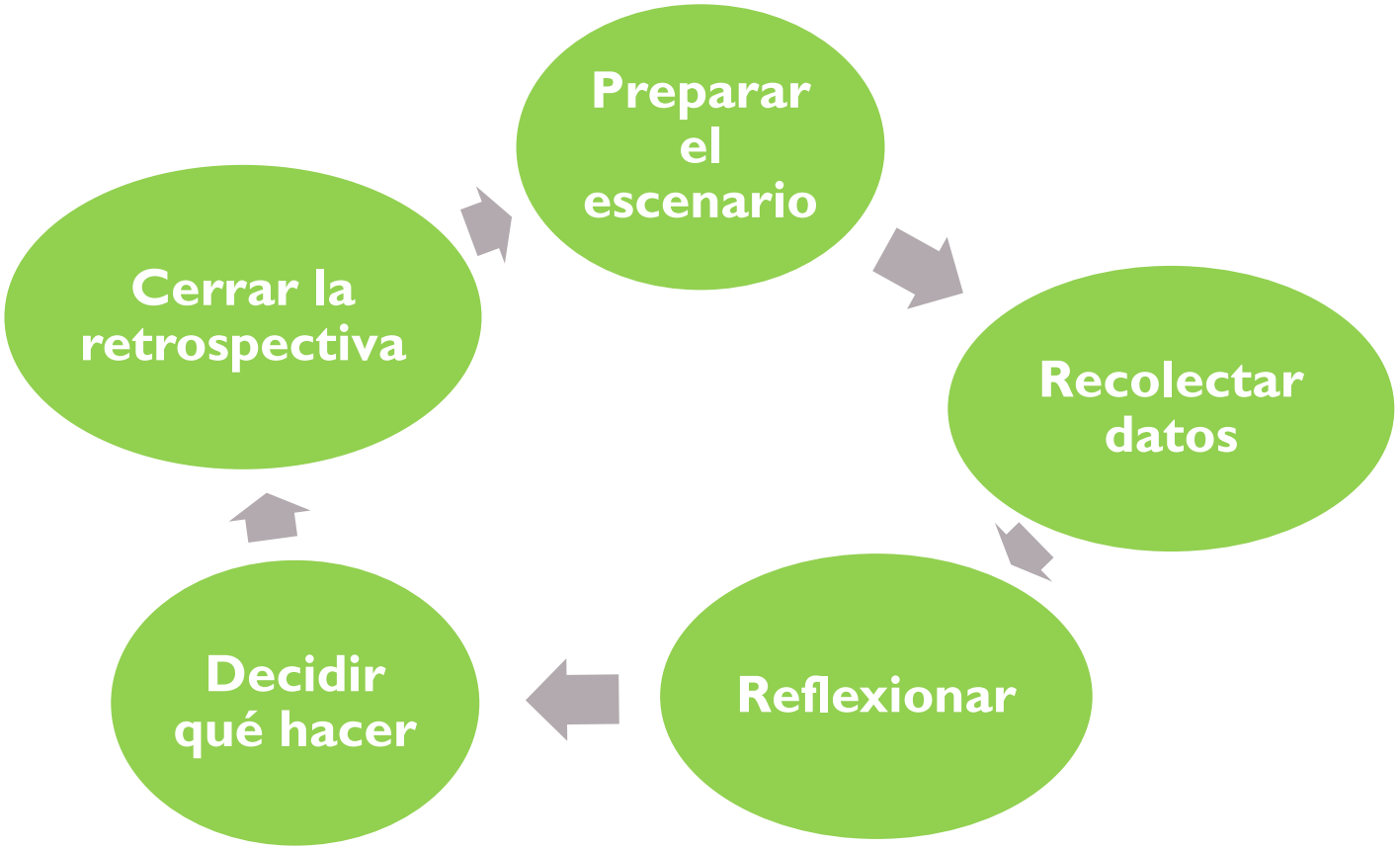


John Boyd (estratega militar):

TÉCNICAS DE RETROSPECTIVA

TÉCNICAS DE RETROSPECTIVA

Las 5 etapas de una retrospectiva



XP

¿ Qué es XP?

- Es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck
- Es el más destacado de los procesos ágiles de desarrollo de software
- Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad

XP - CARACTERÍSTICAS

- **Desarrollo iterativo e incremental,**
- **Pruebas unitarias continuas**,
- **Programación en parejas**:
- Frecuente **integración del equipo de programación con el cliente**
- **Corrección de todos los errores**
- **Refactorización del código**
- **Simplicidad en el código**

XP - VALORES

- **Simplicidad**
- **Comunicación**
- **Realimentación**
- **Coraje o valentía**
- **Respeto**

XP - ROLES

- **Programador (Piloto)**
- **Navegante (Copiloto)**
- **Cliente** Escribe las historias de usuario y las pruebas funcionales para validar su implementación
- **Tester** Ayuda al cliente a escribir las pruebas funcionales
- **Tracker** Es el encargado de seguimiento. Proporciona realimentación al equipo.
- **Coach** Responsable del proceso global. Guía a los miembros del equipo para seguir el proceso correctamente
- **Consultor** Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto.
- **Gestor (Big Boss):** Es el dueño de la tienda y el vínculo entre clientes y programadores. Su labor esencial es la coordinación.

XP - VENTAJAS /DESVENTAJAS

Ventajas

- **Mejora la calidad del código**
- **Mejora la concentración del equipo de desarrollo**
- **Mejora la distribución del conocimiento entre el equipo**

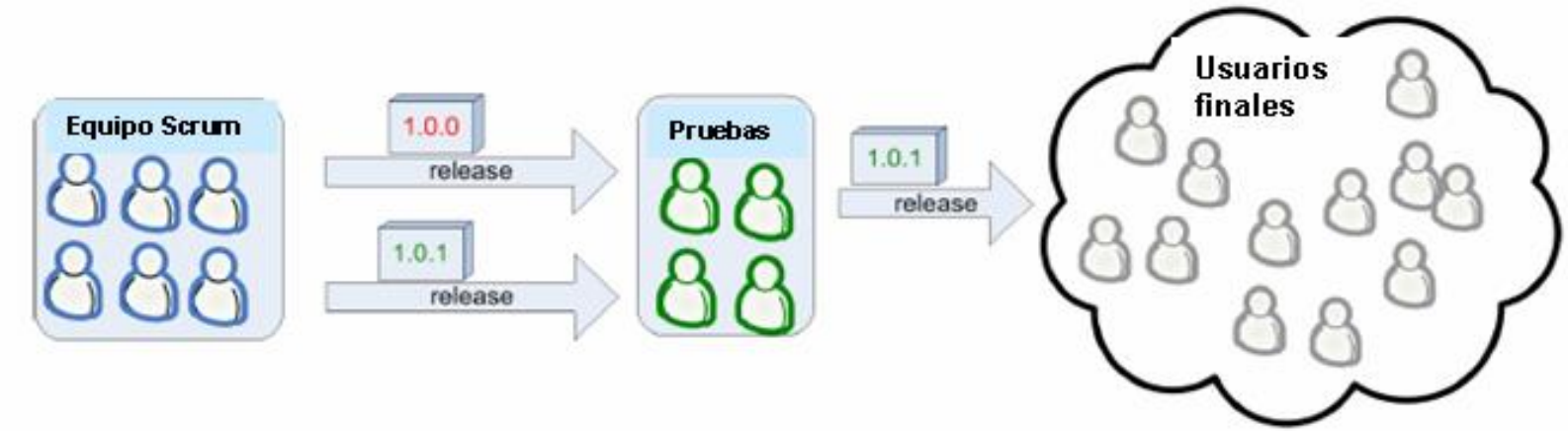
Desventajas

- **No debería hacerse durante todo el día (Cansancio)**
- **Algunas personas no están comodas programando de a pares**
- **Es necesario cambiar de pareja con frecuencia**

COMBINANDO SCRUM Y XP

Scrum se centra en las prácticas de organización y gestión mientras que XP se centra en las prácticas de desarrollo.

tratan de áreas diferentes y se complementan entre ellas



TDD
DESARROLLO GUIADO POR TEST

Desarrollo guiado por pruebas significa que escribes un test automático y, a continuación, escribes el código suficiente para pasar dicho test y después refactorizas el código, principalmente para mejorar la legibilidad y eliminar duplicaciones. Aclarar y repetir.

TDD - ¿CÓMO FUNCIONA?

Normalmente cuando se realizan metodologías ágiles de gestión, por ejemplo Scrum y donde se trabaja en equipos para desarrollar, por ejemplo XP

MEJORA NOTABLEMENTE EL RENDIMIENTO

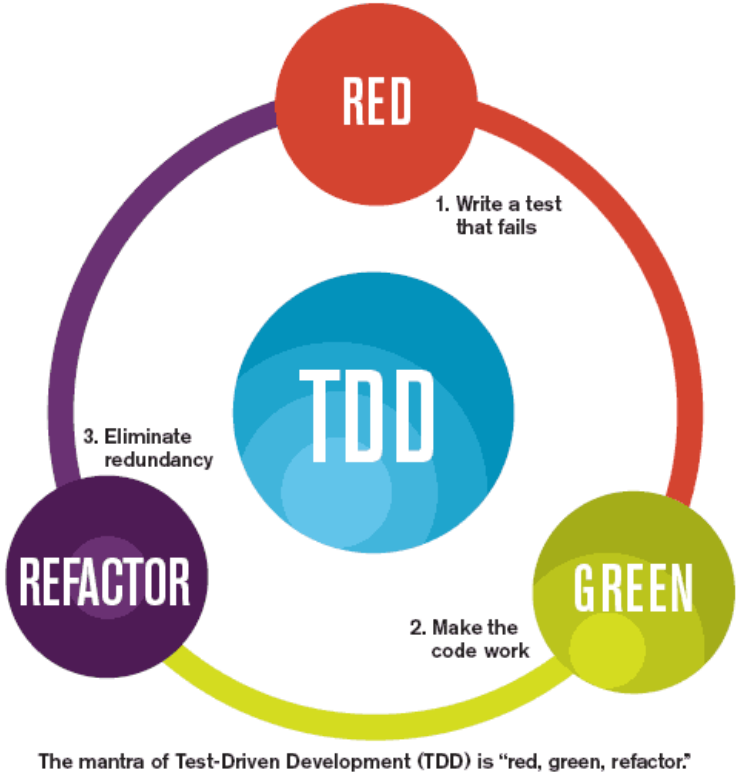
Dependiendo de la naturaleza del proyecto

"No empieces a desarrollar nada nuevo hasta que esté en producción lo que has desarrollado"

CICLO DE VIDA R-G-R

¿Cómo funciona ?

- Escritura de código (Baby Steps)
- Obtener ROJO
- Escritura de código
- Obtener Verde



The mantra of Test-Driven Development (TDD) is "red, green, refactor."

Este es el ciclo de vida del desarrollo guiado por test, Rojo-Verde-Refactor y se debería repetir esto una y otra vez hasta que tengamos las características funcionales y completas.

CARACTERÍSTICAS

- La obtención de un buen resultado depende de la calidad de las pruebas.
- No solo se basa en pruebas, depende también del refactor del código (mejor código, limpio, mantenible).
- El refactor solo se aplica si es necesario.
- El test es escrito por el desarrollador. (Visión desde su punto de vista)
- Es DURO para comprender
- Tiene un efecto muy positivo en el sistema
- El foco principal de TDD es testar las funcionalidad de bajo nivel

VENTAJAS

¿Por qué es mejor hacer las pruebas antes qué el código?

- Se condicionan las pruebas a lo implementado: pueden obviarse casos de prueba.
- Se realiza un profundo análisis funcional y se refinan los requisitos.
- Se implementa solamente código necesario:
  - Baja redundancia
  - Sin código muerto (“por si acaso”)
  - Código limpio

BUENAS PRÁCTICAS

- Tener el código separado de los tests, en carpetas diferentes.
- Los tests deben fallar la primera vez que son escritos.
- Los nombre de los tests deben ir acorde con la intención, deben ser nombres expresivos.
- Refactorizar para eliminar código duplicado después de pasar los tests.
- Repetir las pruebas después de cada refactorización.
- Solo se debe escribir nuevo código, cuando algún test falla. Cada test debe comprar un nuevo comportamiento, o diferente.
- Escribe primero el *assert*.
- Todos los tests deben ser pasados antes de escribir otro test.
- Solo se refactoriza cuando todos los tests pasan.
- Escribe el mínimo y simple código para pasar las pruebas.
- No usar las dependencias entre tests. Los tests deben pasar en cualquier orden.
- Los tests deben ser rápidos.
- Crear solo un test por iteracion

BDD Y ATDD

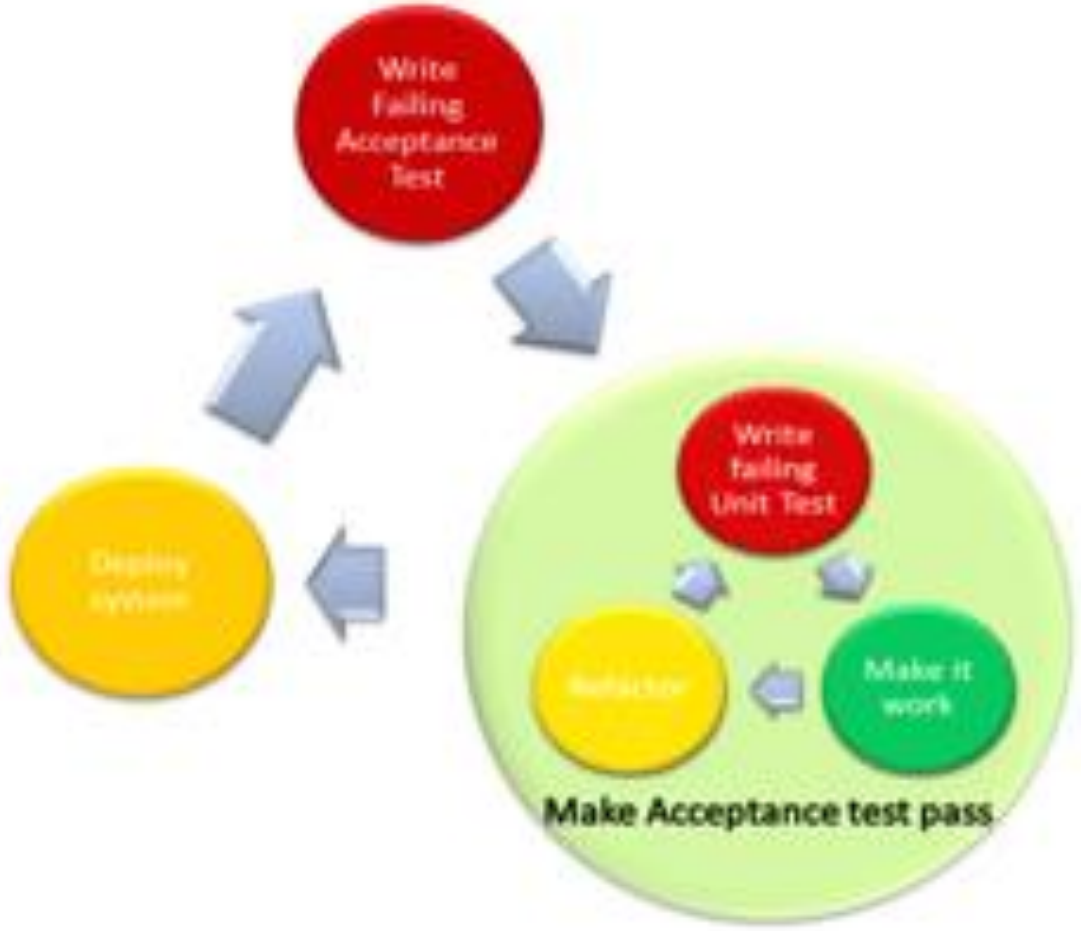
- **BDD (Behavior Driven Development)**
  - **Similar a TDD**
  - **Metodología de desarrollo**
  - **Usuario / Stakeholder escriben los test en alto nivel**
  - **El desarrollador automatiza el test**
  - **Casos de test más alineados con el negocio**
  - **Podria reemplazar al requisito funcional**
  - **Visión desde ambos puntos de vista: cliente y desarrollador.**

As a [X] (persona o Rol)
I want [Y] (alguna característica)
so that [Z] (resultado esperado para esa característica)

BDD Y ATDD

- **ATDD (Acceptanec Test Driven Develpment)**
  - **Los criterios de aceptación son definidos antes del proceso de desarrollo y van a guiar al proceso subsecuente.**
  - **Ejercicio colaborativo que involucra al product owner, analistas de negocio, testers y desaarrolladores, y ayuda a asegurar que todos los miembros del proyecto entendieron que se necesita para que el proyecto este correcto**
  - **los equipos crean uno o mas test de nivel de aceptación para una característica antes de comenzar a trabaja**
  - **Los test unitarios aseguran que la aplicación se construyo correctamente**
  - **Los test de aceptación aseguran que se ha desarrollado la funcionalidad esperada**

ATDD


Write Failing Acceptance Test
Write failing Unit Test
Make it work
Make Acceptance test pass

Fin de la clase

UAI
Universidad Abierta Interamericana