PROCESOS ÁGILES

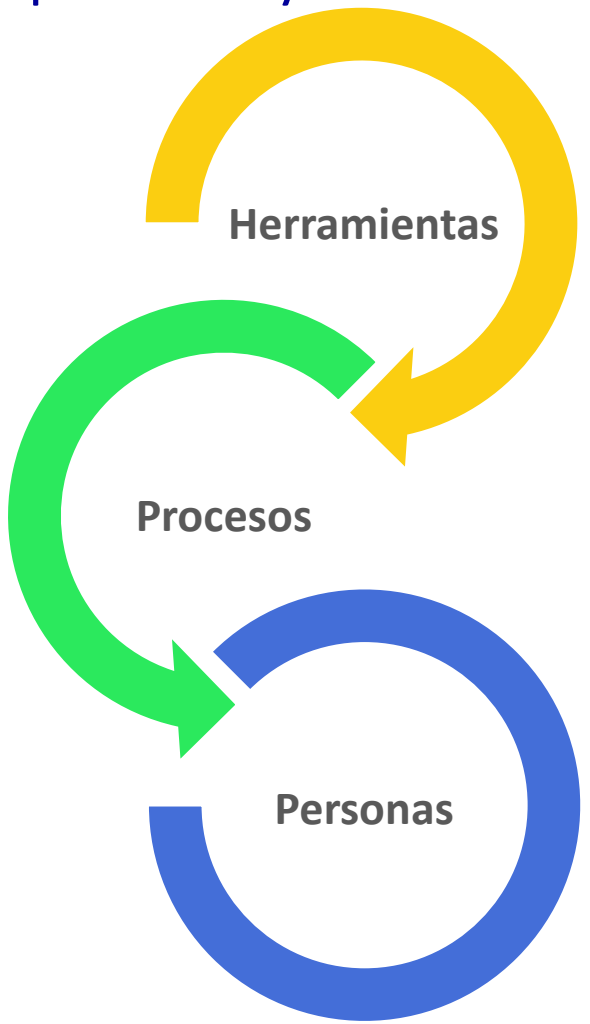
MANIFIESTO POR EL DESARROLLO ÁGIL (2001)

- Hemos aprendido a valorar:
  - personas e interacciones más que procesos y herramientas
  - software funcionando más que documentación extensiva
  - colaboración con el cliente más que negociación contractual
  - respuesta ante el cambio más que seguir un plan.

PERSONAS E INTERACCIONES

… personas e interacciones más que procesos y herramientas

Para producir software se necesita:



Hasta década de los 80

A partir de la década de los 80

A partir del 2001

SOFTWARE FUNCIONANDO

… software funcionando más que documentación extensiva

- Podemos tener una especificación de requisitos o un diseño maravilloso, pero si otro tiene el producto funcionando…
- Lo prefiero, porque
  1. puedo evaluar atributos de calidad externos del sw funcionando. El cliente nos puede dar una retroalimentación.
  2. El sw funcionando (no solo construido) pone el criterio de verificación bastante alto: no dice que sea perfecto, pero tiene que funcionar al punto tal que pueda ser evaluado: criterio de calidad muy alto. Las versiones B no son versiones en condiciones de ponerse en producción, sino de ser evaluadas. Se tarda mucho y lleva mucho esfuerzo tener sw funcionando.
  3. Permite reducir el riesgo de que haya una variación significativa entre la percepción del avance del proyecto y el avance real. P. ej. si tengo 95 % de avance y el cliente me dice que no le sirve…

COLABORACIÓN CON EL CLIENTE

…colaboración con el cliente más que negociación contractual.

El manifiesto ágil se presenta en un momento en que hay fuerte presión por reducir el tiempo de salida al mercado, hay muchos proyectos innovadores (despegue de la web), y hay mucha competencia. Difícil definir con precisión los requisitos al comienzo del proyecto. Es común que cambien y que sean imprecisos.

Cuando el cliente tiene muy claro qué precisa, se puede elaborar un contrato con el cliente. En ambiente de incertidumbre ese enfoque rinde poco. Entonces precisamos que cliente esté colaborando con nosotros y nosotros con él. La gracia de generar sw funcionando es que el cliente lo pueda evaluar.

RESPUESTA ANTE EL CAMBIO

… respuesta ante el cambio más que seguir un plan

En procesos tradicionales el grueso de la planificación se lleva adelante al comienzo. Esto no implica que no tengamos que planificar después (incluso en proceso en cascada), por cambios en los requisitos o en las condiciones (productividad, cantidad de defectos, tecnologías incompatibles).

En el enfoque ágil, en ambiente de incertidumbre: ¿para qué vamos a sobreplanificar? Planifiquemos solo a corto plazo (es como lo meteorológico); al alejarse en el plazo, el nivel de incertidumbre aumenta y planificar a largo plazo es inútil.

MANIFIESTO POR EL DESARROLLO ÁGIL (2001)

- Hemos aprendido a valorar:
  - personas e interacciones más que procesos y herramientas
  - software funcionando más que documentación extensiva
  - colaboración con el cliente más que negociación contractual
  - respuesta ante el cambio más que seguir un plan.
- Aunque valoramos los elementos de la derecha, valoramos más los de la izquierda.

PRINCIPIOS DEL MANIFIESTO ÁGIL

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

PROCESOS GUIADOS POR PLANES Y ÁGILES

- Procesos guiados por planes (predictivos):
  - P. ej. cascada, RUP, MUM
  - El plan se construye fundamentalmente al comienzo.
  - La comunicación se basa en documentos.
- Procesos ágiles (adaptativos):
  - P. ej. XP, SCRUM, FDD (Feature Driven Development), DSDM (Dynamic Systems Development Method)
  - El plan se completa a medida que avanza el proyecto.
  - La comunicación está basada en conversaciones cara a cara.

FACTORES RELEVANTES PARA EL GRADO DE FORMALIDAD/AGILIDAD CONVENIENTE

- Características del proyecto:
  - Riesgos (del proyecto o asociados al uso del producto):
    - Requisitos oscuros o cambiantes (>, > agilidad)
    - Impacto fallas (>, > formalidad)
  - Relación con otros proyectos (>, > formalidad)
- Características de la aplicación
  - Criticidad (>, > formalidad)
- Características del equipo de trabajo
  - Competente (>, > agilidad)
  - Tamaño (>9..., > formalidad)
  - Scrum de scrums, no toda la comunicación es cara a cara
- Características del cliente
  - Cultura, capacidades, posibilidad de involucramiento
- Características de la relación con el cliente
  - Desarrollo interno / contratación externa (+confianza, > agilidad)

PROCESOS GUIADOS POR PLANES

- Cuando usar un proceso guiado por planes:
  - Cuando se trata de una aplicación crítica
  - En un proyecto muy grande con distintos equipos de desarrollo distribuidos.
  - Cuando los que lo van a mantener no son los mismos que los desarrollaron.

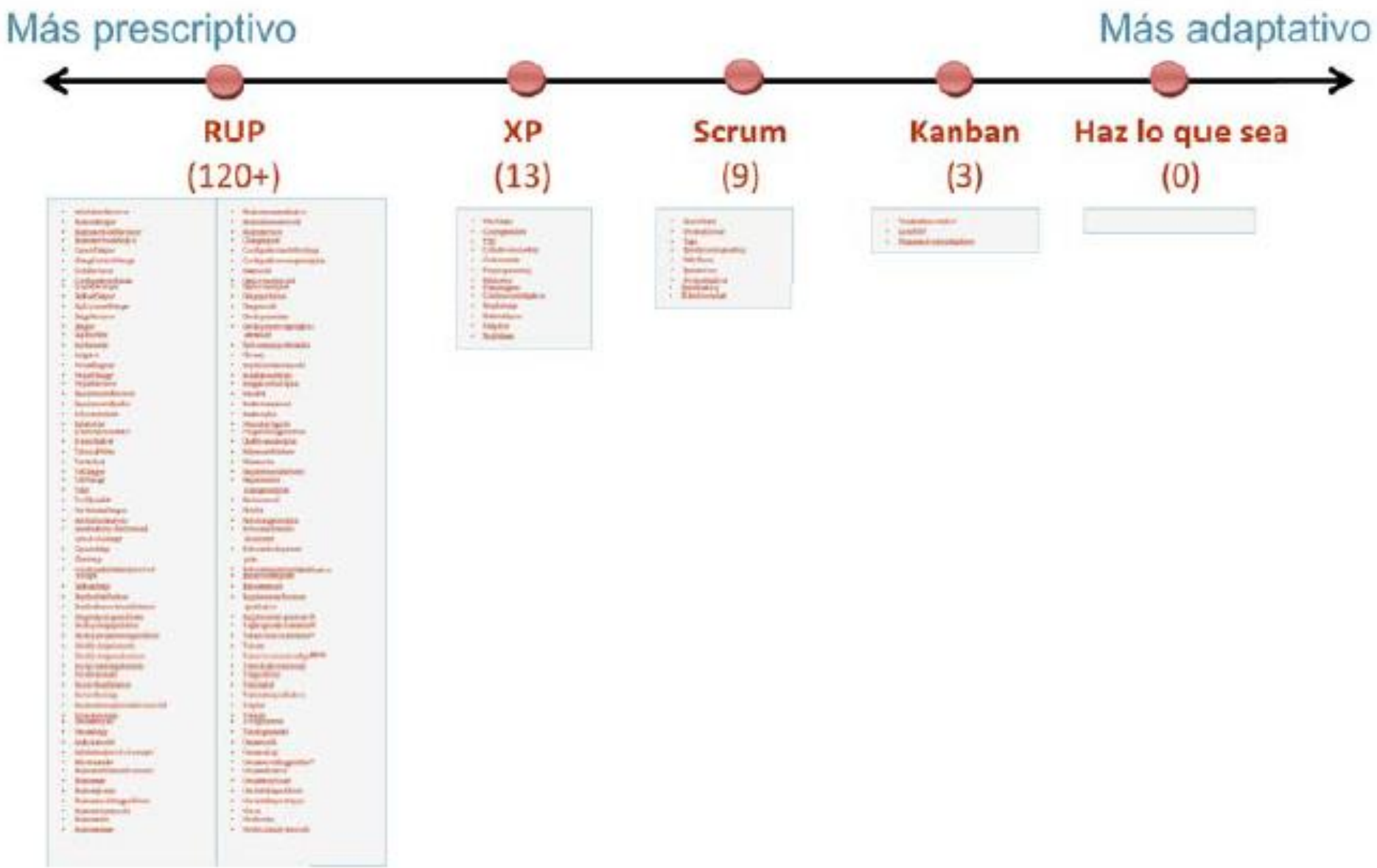
ENFOQUE ÁGIL

- Enfoque apropiado para:
  - Proyectos pequeños (< 10 desarrolladores).
  - No hay condiciones contractuales formales.
  - Los requisitos están cambiando continuamente y se debe entregar funcionalidad a los usuarios de forma frecuente.

- *Agile* son una serie de principios y no una metodología o *framework* en sí.

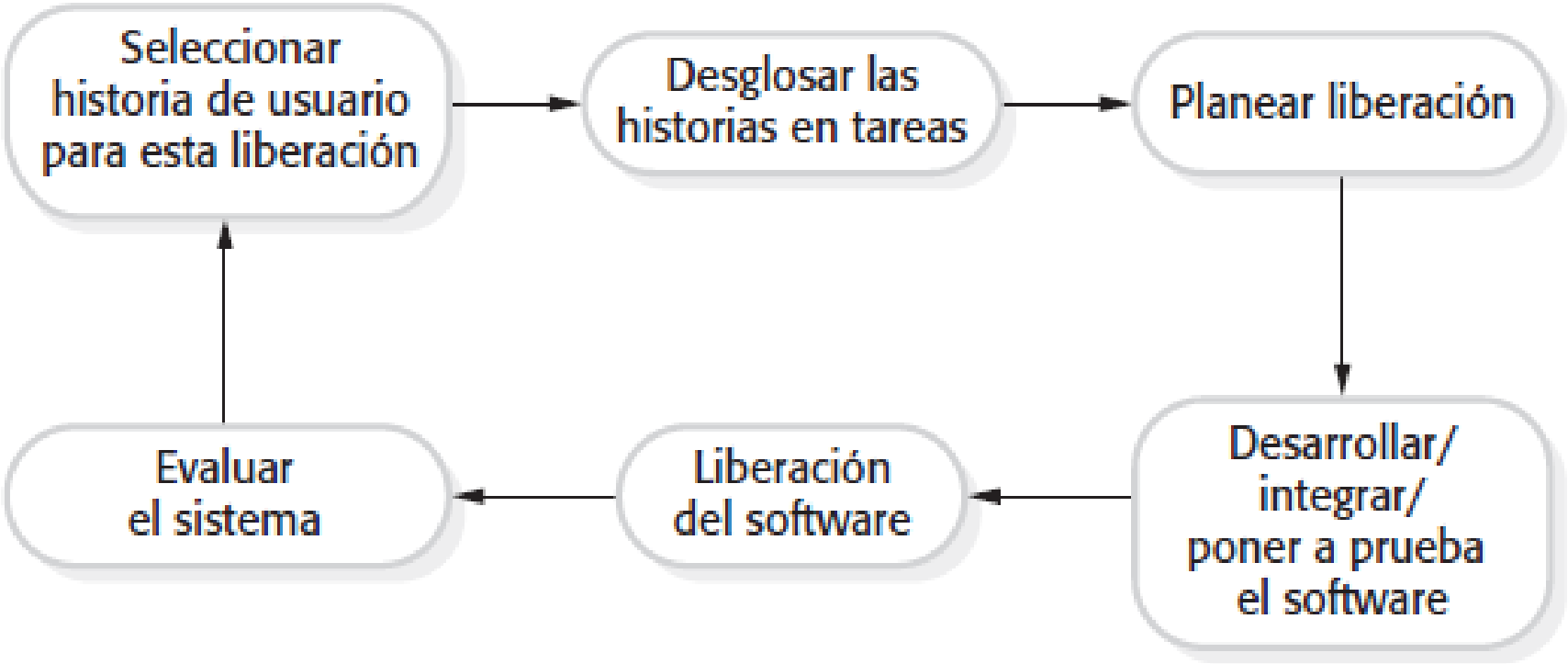
PRINCIPALES PROCESOS ÁGILES

- XP (Extreme Programming)
- Crystal (Cockburn 2002)
- Scrum (1994)
- ASD (Adaptative Software Development)
- Kanban
- FDD
- APF
- APM


Más prescriptivo
Más adaptativo
RUP
(120+)
XP
(13)
Scrum
(9)
Kanban
(3)
Haz lo que sea
(0)

eXtreme Programming (XP)

PROCESO DE XP



* Figura tomada de [Sommerville, 2011].

EXTREME PROGRAMMING (XP)

- Destinado a mejorar la calidad con objetivo de satisfacer las necesidades en constante evolución del cliente.
- Concepto original de Agile
- XP incluye
  - *sprints* cortos,
  - iteraciones frecuentes y
  - colaboración constante con los *stakeholders*.

Extreme Programming (XP)

ROLES

- Cliente
- Equipo de trabajo
  - Desarrollador
  - Verificador
    - Se focaliza en ayudar al cliente a diseñar y escribir pruebas funcionales
    - Responsable de ejecutar las pruebas regularmente y dejar los resultados disponibles para el equipo
  - Coach
    - Encargado de la ejecución técnica y de la evolución del proceso
  - Tracker
    - Registra las estimaciones y las mediciones

XP – EXTREME PROGRAMMING

- El costo del cambio no debe crecer con el tiempo.
- Desarrollo en base a lo actual, sin preocuparse por el mañana.
- 4 valores que deben guiar al equipo:
  - comunicación
  - simplicidad
  - retroalimentación
  - coraje
- 12 prácticas ayudan al equipo a lograr esos valores.

PRÁCTICA DE XP

1. Cliente en el lugar
2. El juego de la planificación
3. Pruebas automatizadas
4. 40 horas semanales
5. Integración continua
6. Pequeñas liberaciones
7. Metáfora
8. Diseño simple
9. *Refactoring*
10. Programación en parejas
11. Estándares de codificación
12. Propiedad colectiva

CLIENTE (O SU REPRESENTANTE) EN EL LUGAR

- Disponible para contestar preguntas y resolver prioridades de pequeña escala
- Escribe los requisitos en historias
- Escribe las pruebas funcionales

EL JUEGO DE LA PLANIFICACIÓN

- Los requisitos se registran como historias en tarjetas.
- Planificación de la liberación
  - Se determina qué requisitos van en una liberación según el tiempo disponible y la prioridad relativa.
  - Participan el cliente y el equipo de desarrollo
  - Cliente: prioridad de cada historia
  - Equipo de desarrollo: Estimación de esfuerzo/tamaño
- Planificación de la iteración
  - Participa solo el equipo de desarrollo
  - Se incluyen historias en función de:
    - Prioridad dada por el cliente y
    - Estimación de esfuerzo/tamaño de quienes vayan a implementarla
  - Se divide la iteración en tareas de pocos días
  - Cada tarea tiene un responsable de que se complete

PRUEBAS AUTOMATIZADAS

- El desarrollo es guiado por el *testing* (*Test Driven Development*)
- Las pruebas son automatizadas (por software)
  - deben codificarse **antes** de escribir el código de la funcionalidad
  - en el momento de diseñarlo
- La implementación de un programa se da por concluida si todas las pruebas automatizadas corren sin defectos

40 HORAS SEMANALES

- La regla es trabajar 40 horas semanales.
- Nunca trabajar extra dos semanas seguidas.
  - Preocupación por las personas y por el equipo (ver desarrollo sostenible del Manifiesto Ágil).
- Ritmo sostenible.
  - De lo contrario se reduce calidad y productividad.

INTEGRACIÓN CONTINUA

- El software se integra varias veces al día.
- Incrementos e iteraciones
  - Se integran para
    - detectar defectos
    - obtener productos que funcionan (tienen comportamiento) y pueden ser evaluados por el cliente
- Todos las pruebas deben correr exitosamente antes de que integrar un nuevo incremento al sistema.

PEQUEÑAS LIBERACIONES

- Liberaciones frecuentes y pequeñas.
- Poner un sistema rápidamente en producción.
- Liberar versiones nuevas en ciclos cortos (desarrollo en fases).

METÁFORA

- Guiar el desarrollo compartiendo una idea simple de cómo funciona el sistema.
- Visión común, nombres y acuerdo sobre forma de abordaje.
- Ejemplo para un software que permita manejar prioridades de tarea: «Debo poder manejar las tareas como las manejo con Post-It en una pizarra».

DISEÑO SIMPLE

- Dado un conjunto de requisitos, existen muchos diseños posibles para satisfacerlos.
- XP establece que hay que elegir el más simple.
- Si se detecta complejidad innecesaria, esta debe ser removida.
- ¿Para qué complicar?
  - Los requisitos pueden cambiar y
  - un diseño simple permite obtener más rápidamente software que funcione, y que, por lo tanto, pueda ser evaluado.
- Solo de necesidades actuales.
- Un caso de la política:
  - «Postergar decisiones tanto como sea responsable hacerlo» (metodología LEAN).

REFACTORING

- Mejora continua del sw con *refactoring*.
- Reestructurar el código para
  - simplificar
  - agregar flexibilidad
  - permitir incorporar nuevos requisitos
- Va de la mano con el diseño simple.

PROGRAMACIÓN EN PAREJAS

- El diseño, las pruebas automatizadas, la codificación, es realizada en parejas
  - Compartiendo un teclado y un *mouse.*
- Las parejas van cambiando
  - Por ejemplo es una en la mañana y en la tarde otra.
- Dentro de una pareja uno hace y el otro revisa.
- Estos roles en la pareja también se rotan.

ESTÁNDARES DE CODIFICACIÓN

- XP enfatiza el código como herramienta de comunicación.
- Todos pueden modificar el código (propiedad colectiva) por lo que debe ser comprensible para todos.
- La documentación en XP está compuesta por:
  - historias
  - pruebas automatizadas
  - pruebas funcionales
  - código

PROPIEDAD COLECTIVA

- Propiedad colectiva del código.
- Todos pueden cambiar cualquier parte del código en cualquier momento.
- Cada integrante del equipo es responsable por el sistema completo.
- Si un par está trabajando y ve la oportunidad de mejorar una porción de código, debe hacerlo si considera que vale el esfuerzo.

SCRUM

Definición de Scrum
Marco de trabajo

- Scrum es un **marco de trabajo** ligero que ayuda a las personas, los equipos y las organizaciones a generar valor a través de **soluciones adaptativas** para problemas complejos.
- Scrum es un marco de trabajo de procesos:
  - no es un proceso, una técnica o método definitivo
  - es un marco de trabajo dentro del cual se pueden emplear varios procesos y técnicas
- El marco de trabajo Scrum es incompleto a propósito.
- En lugar de dar a las personas instrucciones detalladas, las reglas de Scrum guían sus relaciones e interacciones.
- Dentro del marco se pueden emplear distintas técnicas y métodos.
- Diferencias entre Scrum y RUP:
  - En RUP tienes demasiado, y se supone que quitarás aquello que no necesites.
  - En Scrum tienes demasiado poco, y se supone que añadirás el material que falta.

Características de Scrum

- Pone el foco en la gestión ágil del proyecto:
  - emplea un enfoque iterativo incremental para optimizar la predictibilidad y controlar el riesgo.
- No establece ninguna práctica específica de ingeniería.
- Requiere un *Scrum master* que promueva un ambiente donde:
  1. Un *product owner* ordena el trabajo para un problema complejo en un *product backlog*.
  2. El equipo Scrum convierte una porción seleccionada del trabajo en un incremento de valor durante un *sprint*.
  3. El equipo Scrum y sus stakeholders inspeccionan los resultados y ajustan para el próximo *sprint*.
  4. Se repite.

NORMAS

- Prescribe iteraciones de duración fija (*sprints*) de hasta cuatro semanas, que permiten entregar software de forma regular.
- Prescribe equipos interdisciplinarios:
  - Maneja grupos de personas que **colectivamente** tienen todas las habilidades y conocimientos para hacer el trabajo y compartir o adquirir las habilidades que sean necesarias.
- Las responsabilidades son compartidas entre el equipo.

Los pilares de Scrum

Transparencia

- El proceso y el trabajo emergentes deben ser visibles para aquellos que realizan el trabajo, así como para aquellos que lo reciben.
- Las decisiones importantes se basan en el estado percibido de los tres artefactos formales.

Inspección

- Se debe inspeccionar frecuente y diligentemente los artefactos de Scrum y el avance hacia las metas acordadas para detector variaciones potencialmente indeseables o problemas.
- Para ayudar a la inspección, Scrum proporciona una cadencia en la forma de cinco eventos.
- La inspección permite la adaptación. Inspección sin adaptación no tiene sentido. Los eventos de Scrum están diseñados para provocar un cambio.

Adaptación

- Si algún aspecto el proceso se desvía más allá de límites aceptables o si el producto resultante no es aceptable, se debe ajustar el proceso o los materiales producidos.
- El ajuste debe hacerse lo antes posible para minimizar una desviación mayor.

Los cinco valores de Scrum

Compromiso

Para lograr las metas y apoyarse mutuamente.

Focalización

En el trabajo del sprint, para avanzar lo más posible hacia las metas.

Apertura

El equipo Scrum y sus stakeholders son abiertos acerca del trabajo y sus desafíos.

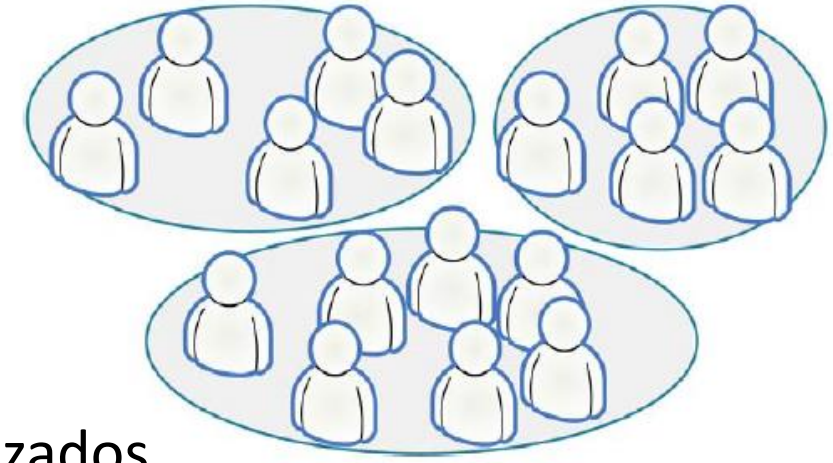
Respeto

Los miembros del equipo se respetan mutuamente como personas capaces, independientes, y son respetados como tales por las personas con las que trabajan.

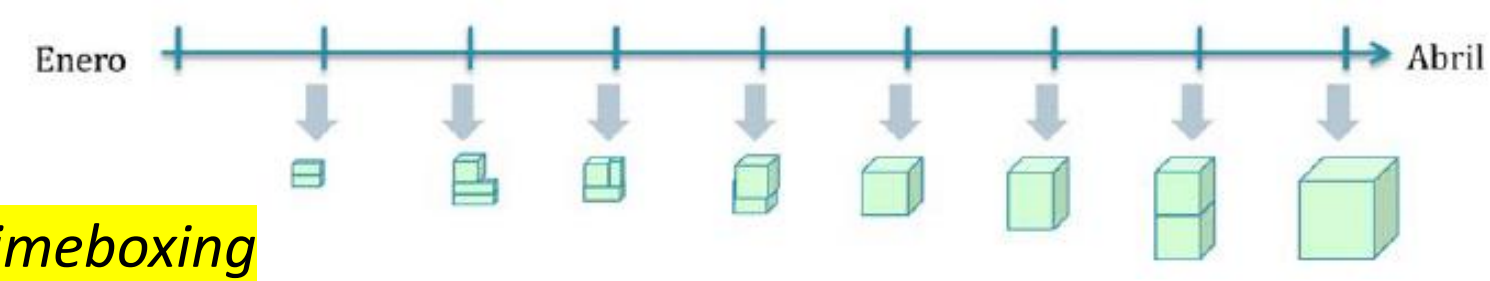
Coraje

Para hacer lo correcto y para trabajar en problemas difíciles.

SCRUM

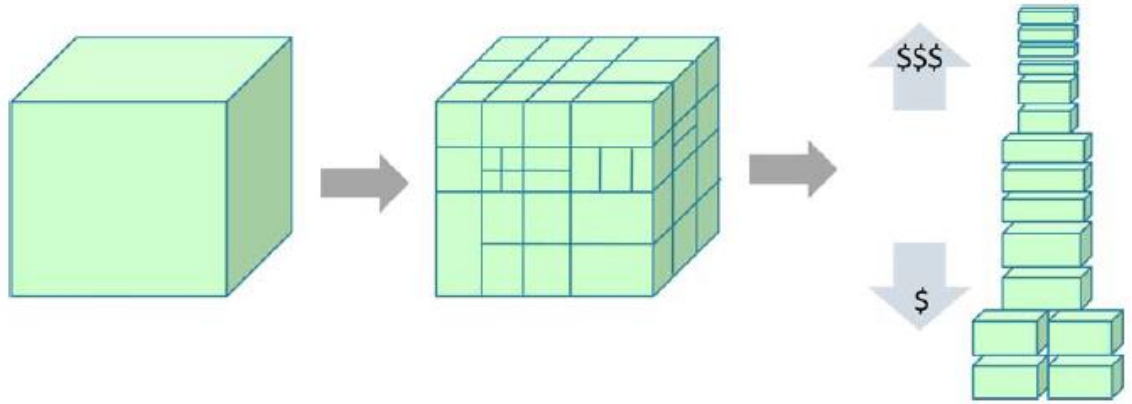


- Equipos pequeños, interdisciplinarios y autoorganizados.
- Divide el tiempo en iteraciones cortas de longitud fija (*sprint*) (de hasta 4 semanas), con código potencialmente entregable y demostrado después de cada iteración.



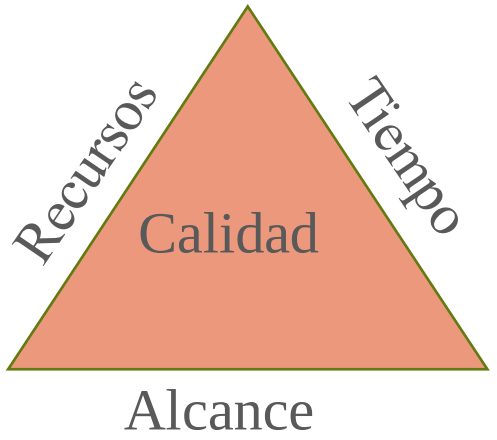
- *Timeboxing*

- Divide el trabajo en una lista de entregables pequeños y concretos. Ordena la lista por orden de prioridad y estima el esfuerzo relativo de cada elemento.



- Optimiza el plan de entregas y actualiza las prioridades en colaboración con el cliente, basada en los conocimientos adquiridos mediante la inspección del entregable después de cada iteración.
- Optimiza el proceso teniendo una retrospectiva después de cada iteración.

TIME-BOXING

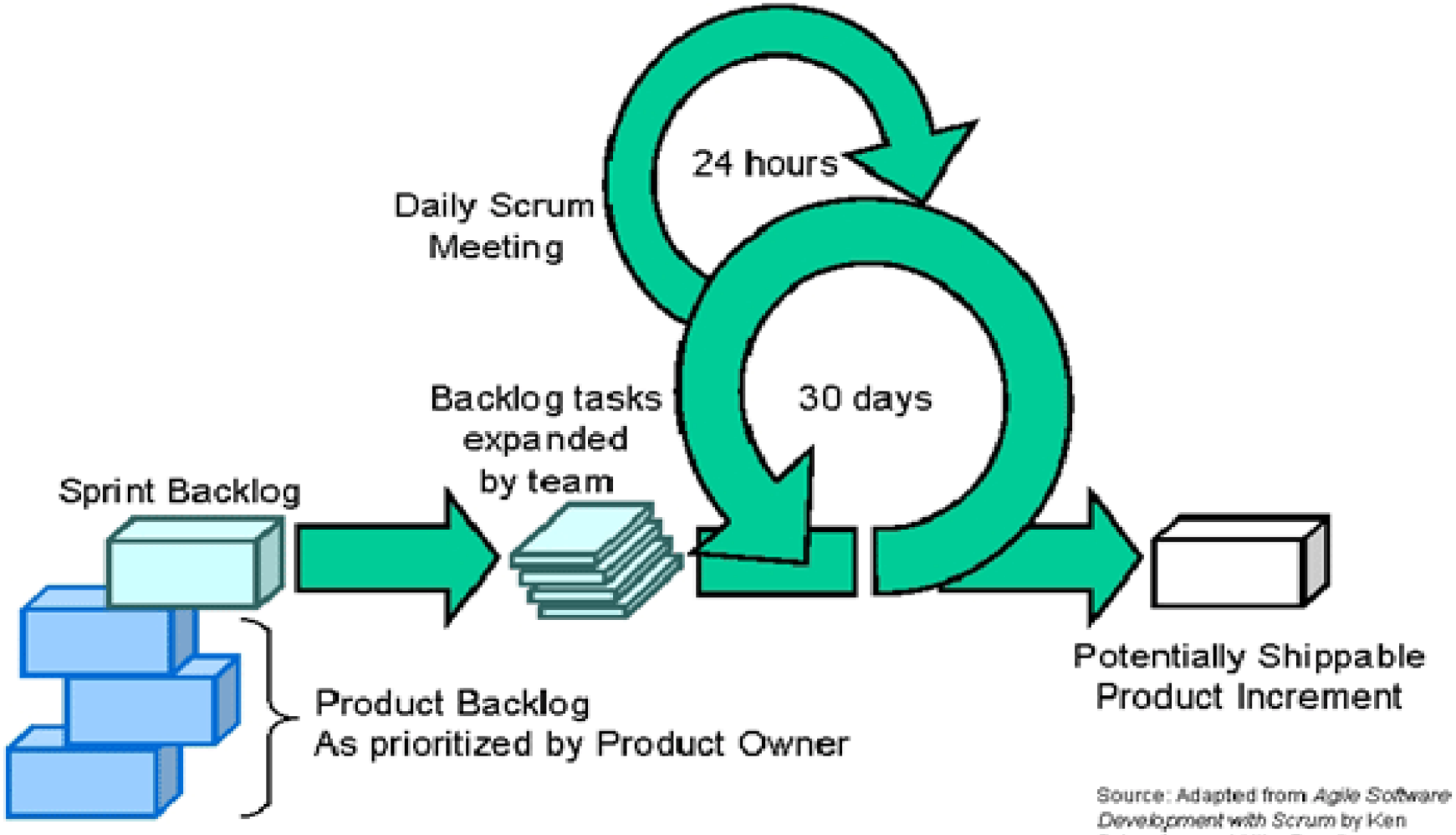


- En gestión de proyectos tenemos la triple restricción:
  - Alcance, recursos, tiempo
- Cambiar una afecta el resto de forma poco predecible. Los proyectos se gestionan a menudo por alcance fijo
  - Si no alcanza el plazo, se extiende o se agregan recursos
- Bueno trabajar en fases, pero …
  - difícil estimar cuánto debe durar cada una para cierto alcance
  - Si se atrasa ¿cuánto permitiremos que se atrase?
  - Para evitar que se estire el proyecto sin tomar medidas...

HAGAMOS AL REVÉS...

- Fijemos el tiempo
  - Dados los recursos
  - Veamos qué alcance logramos.
- Con fases de corta duración
  - La presión del cronograma está siempre presente.
  - El alcance normalmente tiene relevancia dispar (principio de Pareto) y el equipo puede poner foco en lo más relevante y olvidarse de los cambios.
  - Los cambios se pueden incluir en una fase siguiente.

SCRUM



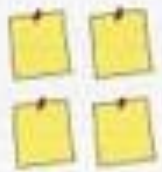
Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle.

Scrum Workflow

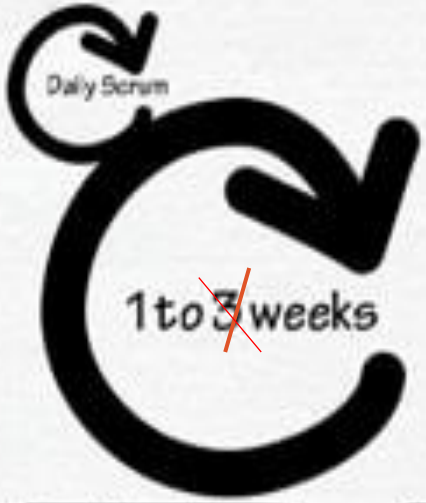
Bringing It All Together

Product Backlog

Sprint Planning



Sprint Backlog



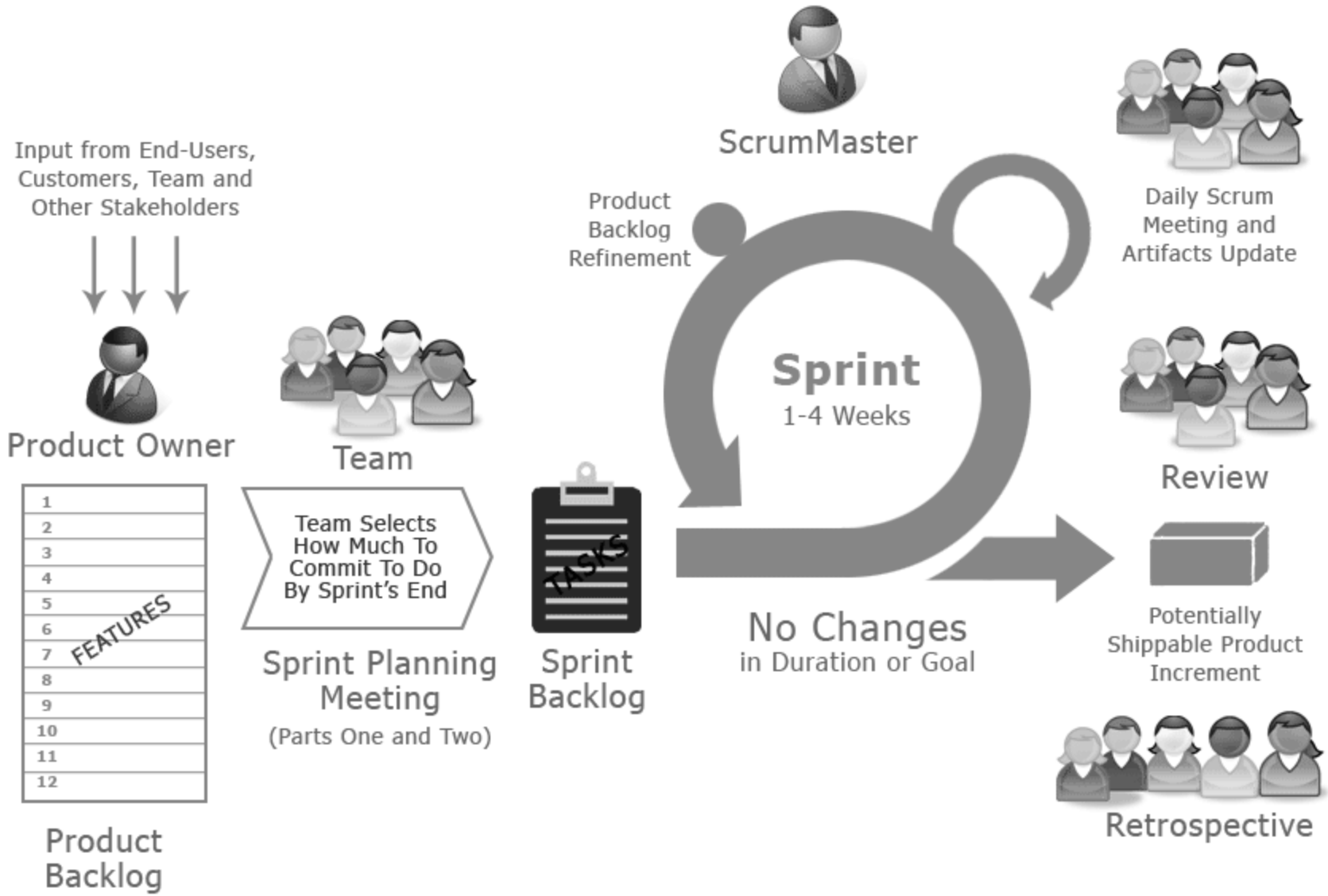
Sprint

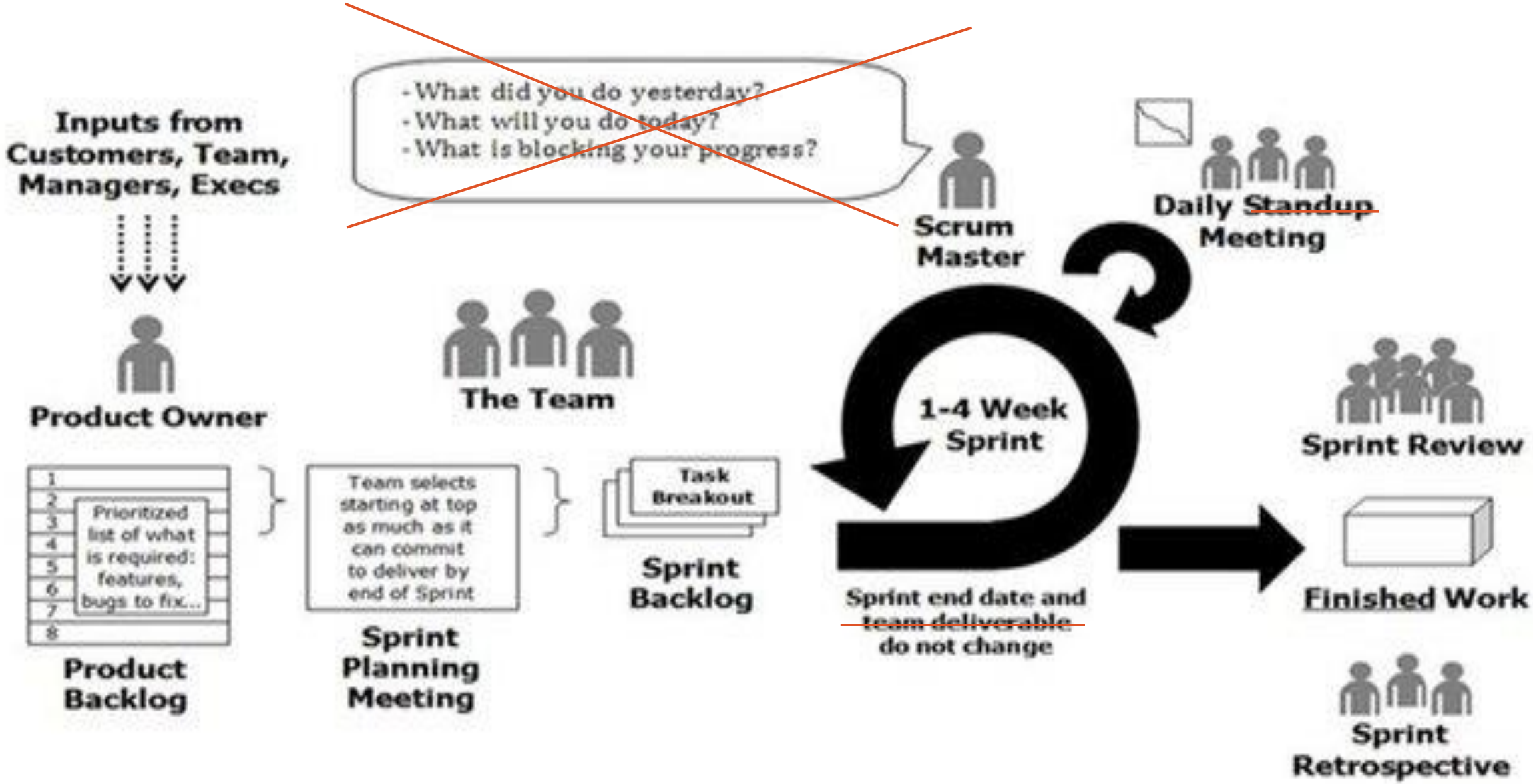
Potentially Shippable Product

Sprint Review

Retrospective


The Agile Scrum Framework at a Glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Burndown/up Charts
Daily Scrum Meeting
Every 24 Hours
1-4 Week Sprint
Sprint Review
Finished Work
Sprint Retrospective
Product Backlog
Sprint Planning Meeting
Sprint Backlog
Sprint end date and team deliverable do not change
AGILEFORALL


Input from End-Users, Customers, Team and Other Stakeholders
Product Owner
Team
1
2
3
4
5
6
7
8
9
10
11
12
FEATURES
Product Backlog
Team Selects How Much To Commit To Do By Sprint's End
Sprint Planning Meeting
(Parts One and Two)
TASKS
Sprint Backlog
ScrumMaster
Product Backlog Refinement
Sprint
1-4 Weeks
No Changes
in Duration or Goal
Daily Scrum Meeting and Artifacts Update
Review
Potentially Shippable Product Increment
Retrospective


Inputs from Customers, Team, Managers, Execs
- What did you do yesterday?
- What will you do today?
- What is blocking your progress?
Scrum Master
Daily Standup Meeting
Product Owner
The Team
1-4 Week Sprint
Sprint Review
1
2
3
4
5
6
7
8
Prioritized list of what is required: features, bugs to fix...
Team selects starting at top as much as it can commit to deliver by end of Sprint
Task Breakout
Sprint Backlog
Finished Work
Sprint end date and team deliverable do not change
Product Backlog
Sprint Planning Meeting
Sprint Retrospective

COMPONENTES

- Los equipos Scrum,
- eventos,
- artefactos y
- reglas asociadas.

EL EQUIPO SCRUM

- Consiste en
  - un *Scrum master.*
  - un dueño del producto (*product owner*),
  - desarrolladores.
- No hay subequipos ni jerarquías. Es una unidad cohesiva de profesionales enfocadas en un solo objetivo por vez: la meta del producto.
- Los equipos Scrum son
  - autoorganizados
    - eligen la mejor forma de llevar a cabo su trabajo y no son dirigidos por personas externas al equipo.
  - multifuncionales
    - Tienen todas las competencias necesarias para crear valor en cada sprint, sin depender de otras personas que no son parte del equipo.

TAMAÑO DEL EQUIPO

- Suficientemente pequeño para permanecer ágil, suficientemente grande para completar trabajo importante en un *sprint*.
- En general, 10 o menos integrantes.
- Equipos pequeños se comunican mejor y son más productivos.
- Si se tornan demasiado grandes, reorganizar en múltiples equipos Scrum enfocados en el mismo producto, que compartan la misma meta del producto, el mismo backlog producto y el mismo *product owner.*

RESPONSABILIDADES DEL EQUIPO

- El equipo Scrum es el encargado de realizar todas las actividades: colaboración con los *stakeholders*, verificación, mantenimiento, operación, experimentación, investigación y desarrollo y cualquier otra cosa que se requiera.
- Trabajar en *sprints* a un paso sostenible mejora la focalización y consistencia del equipo.
- El equipo entero es responsable (*accountable*, debe rendir cuentas) de crear un incremento valioso y útil en cada sprint.
- Hay tres responsabilidades específicas en el equipo Scrum:
  - los desarrolladores
  - el *product owner*
  - el *scrum master*

RESPONSABILIDADES
DESARROLLADORES

- Son los profesionales encargados de crear cualquier aspecto de un incremento usable en cada *sprint*.
- Son responsables de:
  - Crear un **plan** para el sprint: el sprint backlog;
  - Instalar **calidad** al adherirse a una Definition of Done;
  - **Adaptar su plan** cada día hacia la meta del sprint, y
  - hacerse **mutuamente responsables**, como profesionales.

Responsabilidades
PRODUCT OWNER

- Es responsable de
  - maximizar el valor del producto y
  - gestionar efectivamente el *backlog* del producto:
    - desarrollar y comunicar específicamente la meta del producto (*product goal*);
    - crear y comunicar claramente los ítems del producto backlog;
    - ordenar los ítems del producto backlog y
    - asegurar que el producto backlog es transparente, visible y comprendido.
- Puede hacer el trabajo o delegarlo en el equipo de desarrollo. Pero sigue siendo el responsable de dicho trabajo.
- Es una única persona, no un comité. Podría representar las necesidades de muchos stakeholders en el producto backlog, pero aquellos que quieran cambiar el producto backlog deben hacerlo a través de él.
- Toda la organización debe respetar sus decisiones. Estas se reflejan en el contenido y en la priorización de la lista del producto, y en el incremento inspeccionable en la revisión del sprint.

REPONSABILIDADES
SCRUM MASTER

- Es responsable de establecer Scrum como está definido en la Guía de Scrum, ayudando a todos, equipo y organización, a entender la teoría y la práctica de Scrum.
- Es responsable por la efectividad el equipo Scrum, al posibilitar que el equipo mejore sus prácticas, dentro del marco de trabajo Scrum.
- Es un líder al servicio del equipo Scrum y de la organización.

RESPONSABILIDADES
SCRUM MASTER

- **El servicio del *scrum master* al dueño del producto**
  - Ayuda a encontrar técnicas que permitan definir la meta del producto y gestionar el backlog del producto de forma efectiva;
  - ayuda al equipo Scrum a entender la necesidad de contar con ítems del *backlog* del producto claros y concisos,
  - ayuda a establecer una planificación el producto empírica en un ambiente complejo y
  - facilita la colaboración de los *stakeholders*, según se requiera o se necesite.

RESPONSABILIDADES
SCRUM MASTER

- **El servicio del scrum master al equipo Scrum:**
  - Enseña a los miembros del equipo a ser autoorganizados y multifuncionales;
  - ayuda al equipo a enfocarse en crear incrementos de gran valor y cumplir con la Definition of Done;
  - provoca la eliminación de impedimentos para el avance del equipo Scrum;
  - se asegura de que todos los eventos de Scrum tengan lugar y sea positivos, productivos y que se mantengan dentro del plazo estipulado.

RESPONSABILIDADES
SCRUM MASTER

- **El servicio del scrum master a la organización**
  - Lidera, guía y enseña a la organización en la adopción de Scrum;
  - planifica y asesora acerca de las implementaciones de Scrum en la organización;
  - ayuda a los empleados e interesados a entender y llevar a cabo un enfoque empírico del trabajo complejo, y
  - elimina las barreras entre los stakeholders y los equipos Scrum.

Los eventos de Scrum

- Los eventos de Scrum
  - El sprint
  - La planificación del sprint
  - El Scrum diario
  - La revisión del sprint
  - La retrospectiva del sprint
- El sprint contiene a los otros.
- Cada eventos es una oportunidad formal de inspeccionar y adaptar los artefactos de Scrum.
- Se usan para crear regularidad y minimizar la necesidad de otras reuniones no definidas en Scrum.
- Óptimo: que todos los eventos tengan lugar siempre en el mismo horario y lugar, para reducir la complejidad.
- Todos tienen una duración máxima:
  - Una vez que comienza un *sprint*, su duración es fija y no puede acortarse o alargarse.
  - Los demás eventos pueden terminar siempre que se alcance el objetivo del evento, asegurando que se emplee una cantidad apropiada de tiempo sin permitir desperdicio en el proceso.

EL SPRINT

- De largo fijo (hasta un mes) para crear consistencia.
- Un nuevo *sprint* comienza inmediatamente después de terminar el anterior.
- Todo el trabajo necesario para lograr la meta del producto (incluso la planificación del *sprint*, los *scrums* diarios, la revisión del *sprint* y la retrospectiva del *sprint*) tienen lugar dentro del sprint.
- Durante el *sprint*:
  - no se hacen cambios que podrían poner en peligro la meta del *sprint*;
  - no se decrementa la calidad;
  - se refina el *backlog* del producto según se necesite y
  - el alcance puede ser clarificado y renegociado con el *product owner*, a medida que se sabe más.
- Si un sprint es muy largo, la meta del *sprint* puede tornarse inválida, puede aumentar la complejidad y los riesgos. Con *sprints* más cortos se puede generar más ciclos de aprendizaje y limitar el riesgo en costo y esfuerzo a un período más pequeño.
- Cada *sprint* debe ser considerado un pequeño proyecto.
- Aunque existen prácticas útiles para pronosticar el avance (como *burn-downs, burn-ups*, o flujos acumulativos), se desconoce lo que pasará. Solo se puede usar lo que ya ha pasado para tomar decisiones respecto al futuro.
- Se debe cancelar el *sprint* si la meta del *sprint* se vuelve obsoleta. Solo el *product owner* tiene autoridad para cancelar el *sprint*.

Planificación del *sprint*

- Al comienzo de cada *sprint.*
- Se plantea el trabajo a ser realizado para el *sprint.*
- El plan es creado por el trabajo colaborativo del equipo *scrum* entero.
- El *product owner* se asegura que los asistentes estén preparados para discutir los ítems más importantes del *backlog* del producto. El equipo puede invitar a otras personas a participar para dar consejo.
- 8 horas como máximo para un *sprint* de un mes (menos si es más corto).

PLANIFICACIÓN DEL *SPRINT*

- Trata los siguientes temas:
  1. Por qué es este *sprint* valioso
     - El *product owner* propone cómo se podría aumentar el valor y la utilidad del producto en este sprint.
     - El equipo entero define la meta del sprint.
  2. Qué se puede hacer en este *sprint*
     - En conversación con el *product owner*, los desarrolladores seleccionan los ítems del product backlog a incluir en el sprint.
     - El equipo puede **refinar** estos ítems durante el proceso, lo que mejora la comprensión y la seguridad.
     - Cuanto más sepan los desarrolladores sobre su desempeño pasado, su capacidad futura y de su Definition of Done, más confiados estarán en sus predicciones.
  3. Cómo se hará el trabajo seleccionado
     - Para cada ítem seleccionado del *backlog* del producto, los desarrolladores planifican el trabajo necesario para crear un incremento que cumpla con la Definition of Done.
     - Esto se hace a menudo **descomponiendo** los ítems del backlog del producto en ítems más pequeños realizables en un día o menos.

SCRUM DIARIO

- Propósito: revisar el avance hacia la meta del sprint y adaptar el sprint backlog según sea necesario, ajustando el trabajo planificado pendiente.
- En el mismo lugar y hora.
- Duración fija de 15 minutos.
- Para los desarrolladores. (Scrum master y p. o. pueden participar como desarrolladores si es que están trabajando en los ítems del sprint backlog).
- Produce un plan para el día siguiente.
- Mejora la comunicación, se identifican impedimentos, promueve la toma rápida de decisiones y , por lo tanto, elimina la necesidad de otras reuniones.
- No es el único momento en que los desarrolladores pueden ajustar su plan. Se reúnen durante el día para discutir más detalladamente el ajuste o la replanificación del resto del trabajo del sprint.

REVISIÓN DEL *SPRINT*

- Propósito: revisar el resultado del sprint y determinar adaptaciones futuras.
- El equipo Scrum presenta el resultado de su trabajo a los stakeholders principales y se discute el avance hacia la meta del producto.
- El equipo y los stakeholders repasan lo que se ha logrado en el sprint y deciden qué hacer a continuación. También se puede ajustar el product backlog para aprovechar nuevas oportunidades.
- Es una sesión de trabajo, no una mera presentación.
- Es el penúltimo evento del sprint.
- 4 h como máximo (para un sprint de un mes; menos para sprints más cortos).

RETROSPECTIVA DEL *SPRINT*

- Propósito: planificar cómo mejorar la calidad y la efectividad.
- Mejora continua del proceso:
  - Se revisa qué tal nos fue (personas, interacciones, procesos, herramientas y su Definition of Done).
  - Se revisan asunciones erróneas y su origen.
  - ¿Qué funcionó bien?
  - Problemas encontrados y si fueron o no resueltos.
  - ¿Qué se podría mejorar para el próximo *sprint*? (los cambios que mejor ayuden a mejorar la efectividad. Los de mayor impacto se tratan lo antes posible. El resto se agregan al sprint backlog para el próximo sprint.
- Termina el sprint.
- 3 h como máximo (para *sprints* de un mes, o menos para más cortos).

SCRUM
ARTEFACTOS

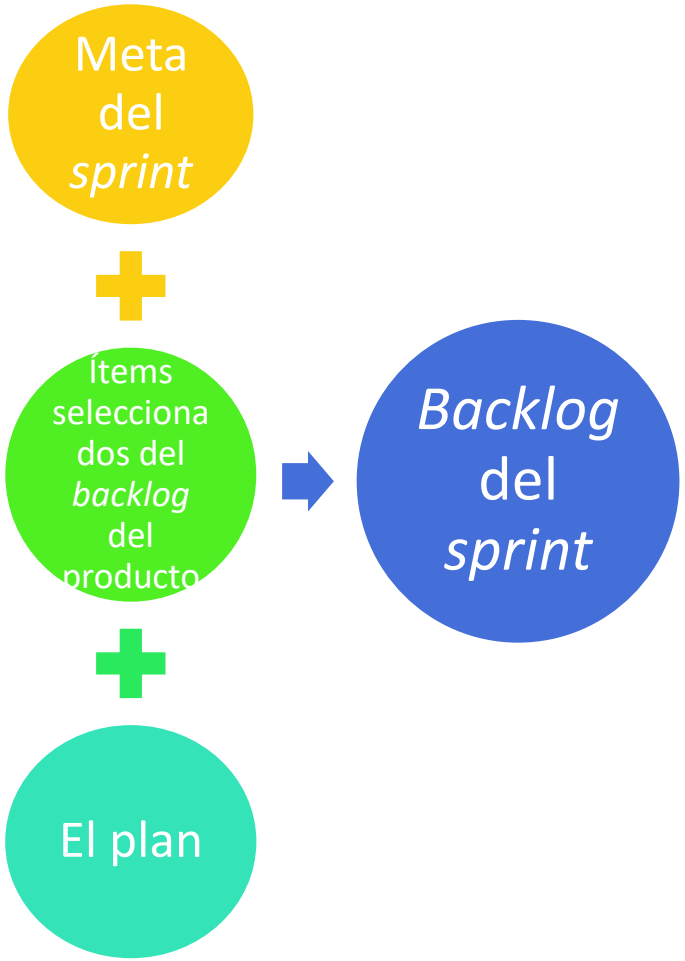
- Los artefactos de Scrum representan trabajo o valor.
- Cada artefacto contiene un compromiso para asegurar que proporciona información que mejora la transparencia y la focalización, contra el cual se puede medir el avance.
- Backlog del producto
  - Compromiso: meta del producto
- Backlog del sprint
  - Compromiso: meta del sprint
- Incremento
  - Compromiso: Definition of Done

Backlog del producto
Compromiso: la meta del producto

- La meta del producto describe un estado futuro del producto.
- Sirve como meta hacia la cual planificar.
- El backlog del producto está compuesto por
  - la meta del producto
  - aquello que realizará la meta del producto.
- La meta del producto es el objetivo a largo plazo del equipo Scrum.
- Deben alcanzar o abandonar un objetivo antes de tomar el siguiente.

EL BACKLOG DEL SPRINT

- El backlog del sprint se compone de:
  - la meta del sprint (el porqué),
  - el conjunto de ítems del backlog del producto seleccionados para el sprint (el qué)
  - un plan para entregar el incremento (el cómo).
- Es un plan por y para los desarrolladores del trabajo que deberán realizar durante el sprint para alcanzar la meta del sprint.
- Se actualiza durante el sprint.
- Debe contener suficiente detalle como para poder medir el avance en el scrum diario.



El backlog del *sprint*
Compromiso: la meta del *sprint*

- Es el único objetivo del sprint.
- Es un compromiso que hacen los desarrolladores.
- Es flexible en cuanto al trabajo exacto necesario para alcanzarla.
- Se crea durante la planificación del sprint y se agrega al backlog del sprint.
- Si el trabajo resulta ser diferente a lo esperado, negocian junto con el product owner el alcance del backlog del sprint sin afectar la meta del sprint.

INCREMENTO

- Cada incremento se agrega a todos los anteriores.
- Se verifica exhaustivamente para asegurar que todos los incrementos funciones bien juntos.
- Para proporcionar valor, el incremento debe ser usable.
- En un sprint se pueden crear múltiples incrementos.
- La suma de los incrementos se presenta en la revisión del sprint.
- Un incremento se puede entregar a los stakeholders antes del fin del sprint. La revisión del sprint nunca debe ser considerada una puerta para liberar valor.
- El trabajo nunca puede ser considerado parte de un incremento a menos que cumpla con la Definition of Done.

Incremento
Compromiso: Definition of Done

- La Definition of Done es una descripción formal del estado del incremento cuando cumple con las medidas de calidad requeridas para el producto.
- Cuando un ítem del backlog del producto cumple con la Definition of Done, se convierte en un incremento.
- Si un ítem del backlog del producto no cumple con la Definition of Done, no puede ser liberado y ni siquiera presentado en la revisión del sprint, sino que vuelve al backlog del producto.

NOTA

- El marco de trabajo de Scrum no puede cambiarse.
- Si bien es posible implementar solo partes de Scrum, el resultado no es Scrum.
- Scrum solo existe cuando está por entero y funciona bien como contenedor de otras técnicas, metodologías y prácticas.

KANBAN

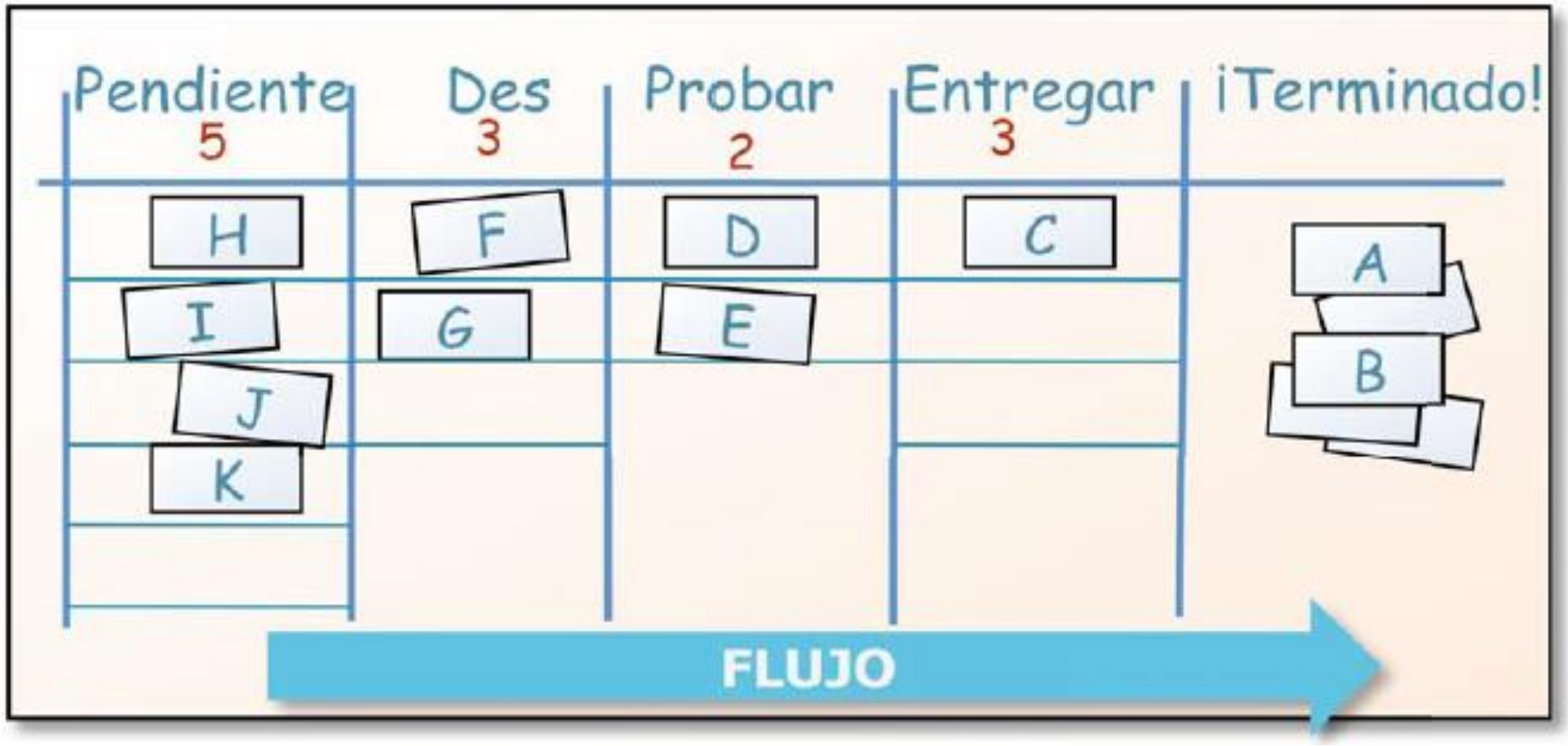
NORMAS

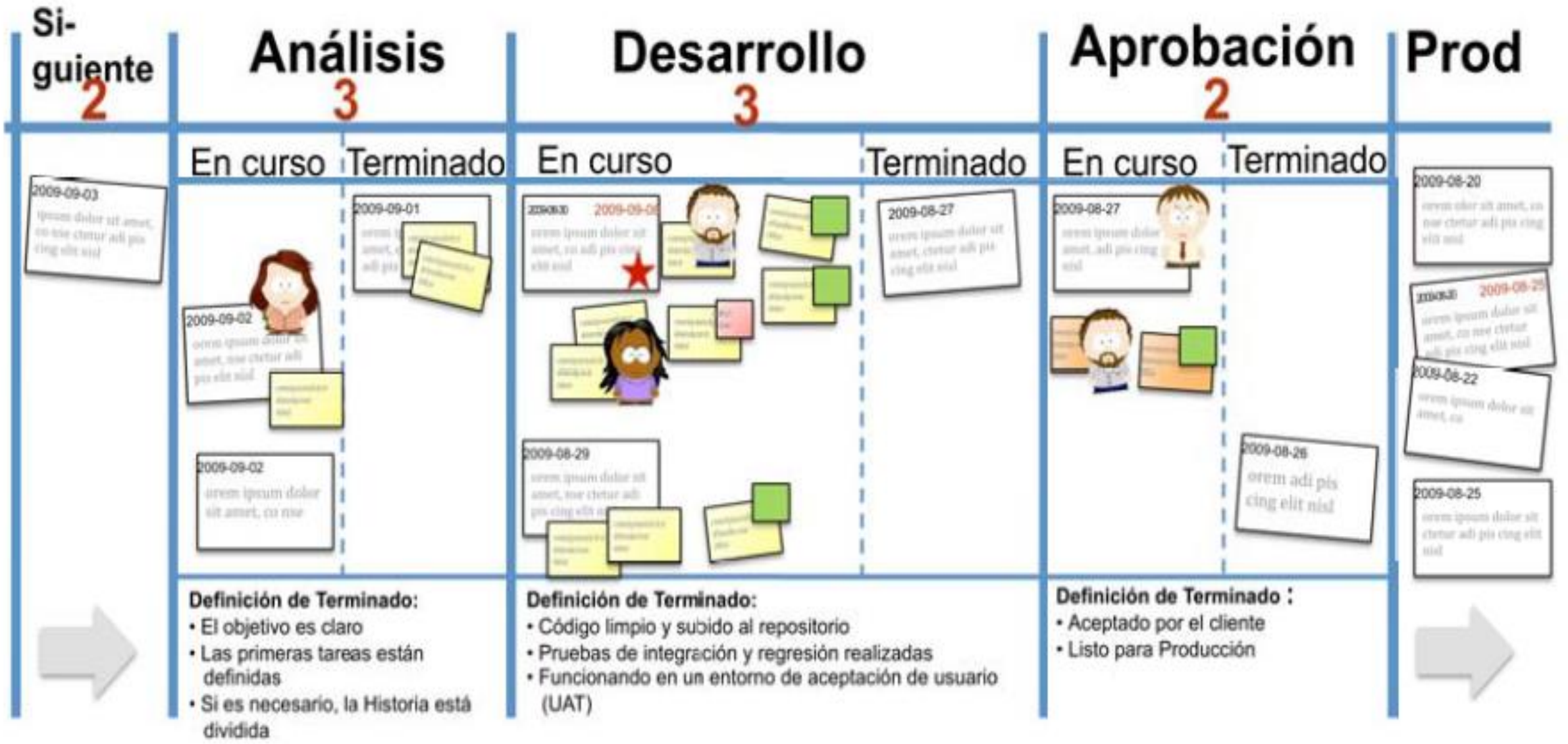
- Kanban deja casi todo abierto.
- Las únicas normas son
  - Visualizar el flujo de trabajo
    - Partir el trabajo en pedazos, anotar cada uno en una tarjeta y ponerla en la pared.
    - Usar columnas con nombre para mostrar en qué etapa del flujo está cada ítem.
    - Usar representaciones visuales de sus tareas con notas adhesivas de colores o utilizando servicios como Hygger, Jira o Trello**.**
  - Limitar el trabajo en curso (*work in progress*)
    - Asignar límites explícitos a la cantidad de elementos en cada estado del flujo del trabajo (limita el tamaño de las colas).
    - En trabajo intelectual tener más de una tarea en curso afecta negativamente la productividad.

KANBAN

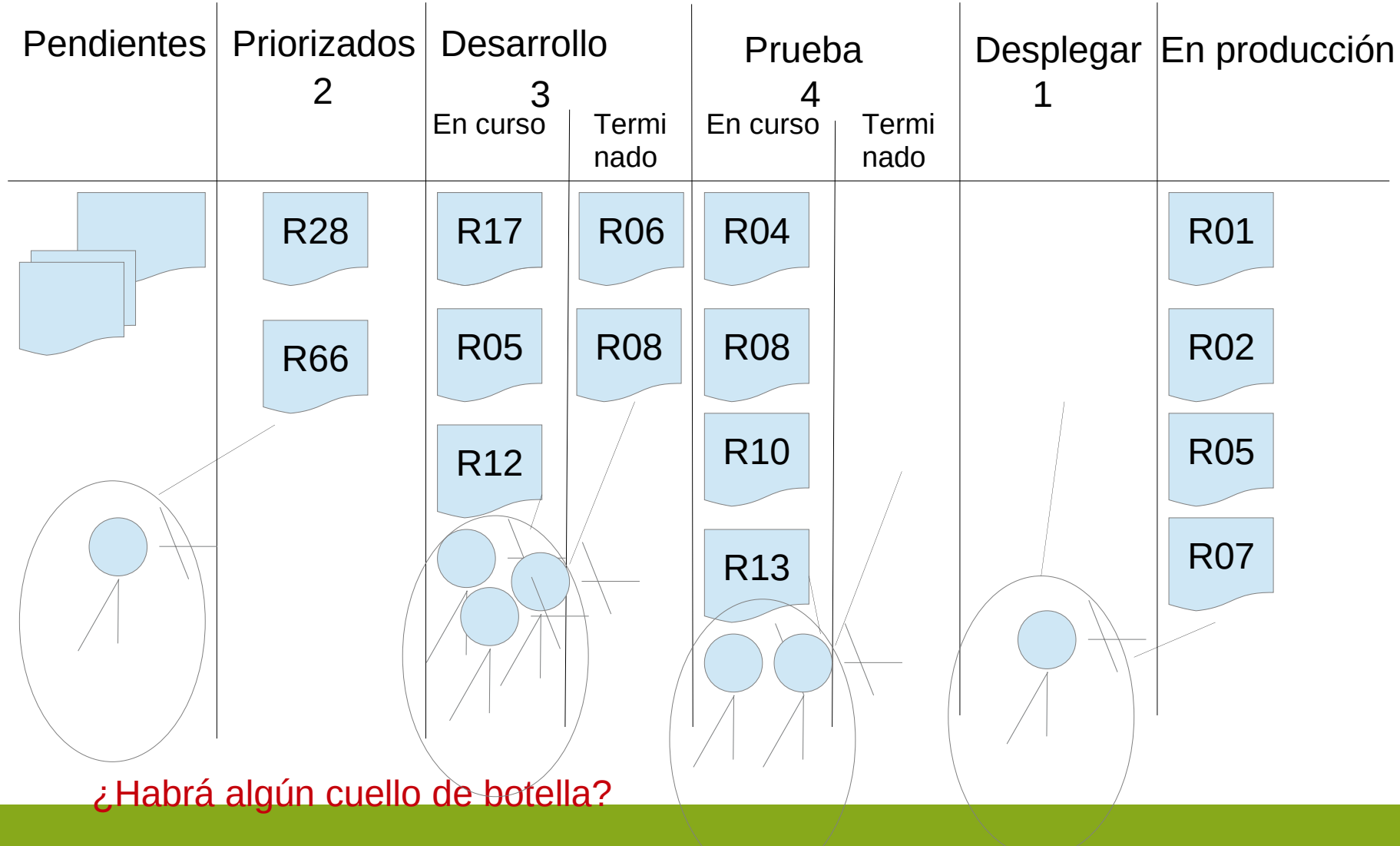
- Kanban (japonés) = tarjeta visual o valla publicitaria.
- Apropiado para gestionar
  - sistemas en producción, con objetivo claro de promover una mejora continua.
  - el trabajo que requiere un rendimiento constante.
- Medir el *lead time* = el tiempo que le lleva a un ítem desde que surge hasta que termina el trabajo (tiempo medio para completar un elemento, a veces llamado *tiempo de ciclo*).
- Ajustar el proceso para volver ese tiempo tan pequeño y predecible como sea posible.
- Las pizarras Kanban son también la solución ideal para que los jefes de producto mantengan el *product backlog*.

CUADRO DE SEGUIMIENTO


Pendiente
5
Des
3
Probar
2
Entregar
3
¡Terminado!
H
I
J
K
F
G
D
E
C
A
B
FLUJO


Si-
guiente
2
Análisis
3
Desarrollo
3
Aprobación
2
Prod
En curso
Terminado
En curso
Terminado
En curso
Terminado
2009-09-03
2009-09-01
2009-09-02
2009-09-02
2009-08-27
2009-08-27
2009-08-26
2009-08-20
2009-08-22
2009-08-25
2009-08-29
Definición de Terminado:
• El objetivo es claro
• Las primeras tareas están definidas
• Si es necesario, la Historia está dividida
Definición de Terminado:
• Código limpio y subido al repositorio
• Pruebas de integración y regresión realizadas
• Funcionando en un entorno de aceptación de usuario (UAT)
Definición de Terminado :
• Aceptado por el cliente
• Listo para Producción

CUADRO DE SEGUIMIENTO. EJEMPLO



¿Habrá algún cuello de botella?

ROLES

- Kanban no prescribe ningún rol.
- Se pueden añadir los roles adicionales que se necesiten.

CONCLUSIONES

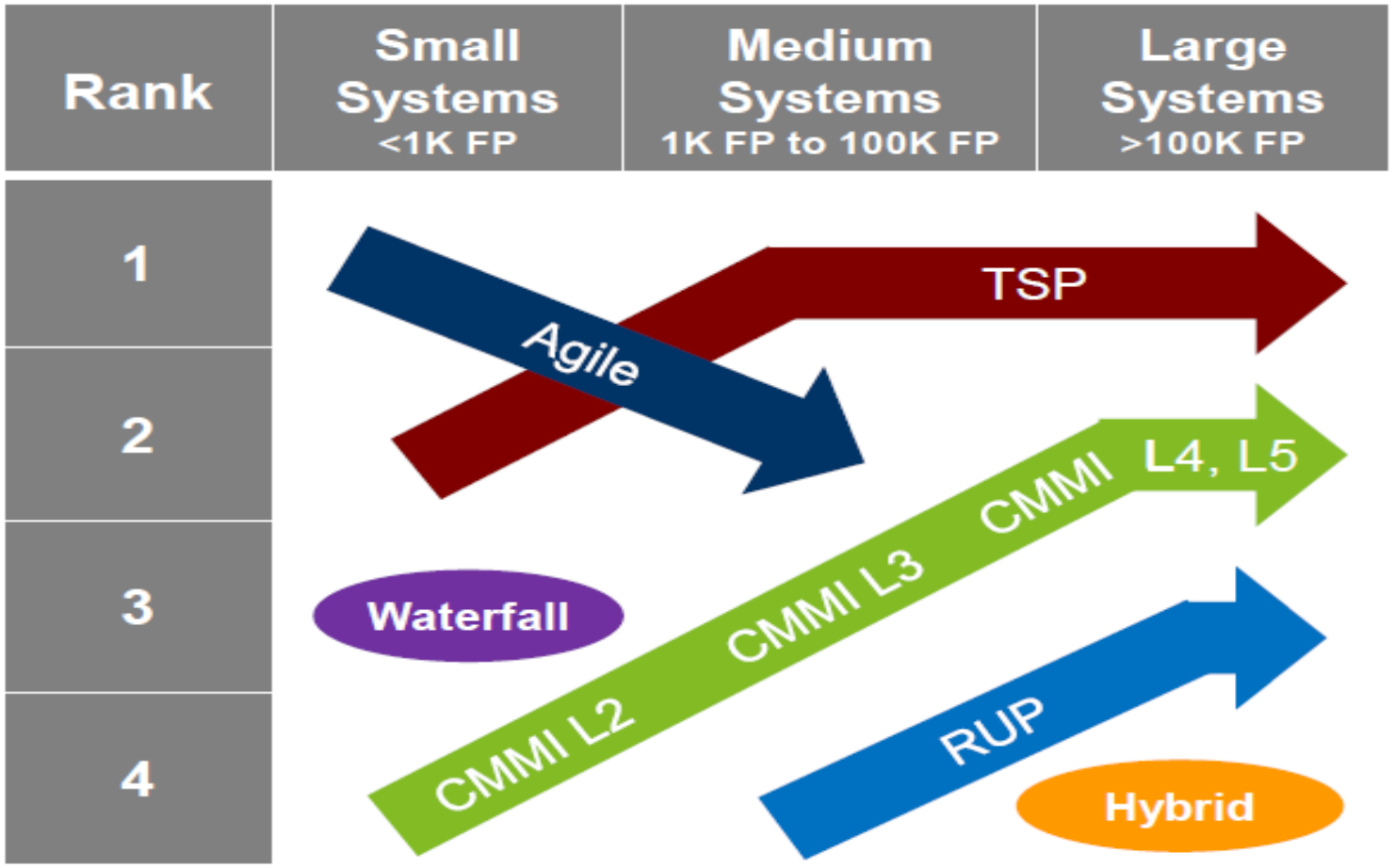
VENTAJAS DE LAS METODOLOGÍAS ÁGILES

- Muy adecuadas :
  - Para equipos pequeños (hasta 7 integrantes)
  - En caso de requisitos oscuros o cambiantes
  - Para innovación en productos
- Manejan muy bien el cambio.
- Permiten obtener rápidamente resultados de valor.
- Eficientes
  - poco desperdicio
  - en especial en comunicación

PROBLEMAS Y LIMITACIONES

- Requiere personal competente.
- Dificultades si la escala es mayor.
- Si bien existen Scrums de Scrums, la gestión se complica...
- El mantenimiento de lo desarrollado con metodologías ágiles puede plantear problemas.
- Depende de la tecnología que se use
  - P. ej. la automatización de pruebas de XP no es viable con cualquier herramienta de desarrollo.
- No siempre es fácil conseguir un «cliente en el lugar» o un «representante del cliente».
- Puede plantear dificultades contractuales.
  - El cliente ¿qué contrata?

DESEMPEÑO RELATIVO DE «AGILE»



Development practices by size of application in function points (FP) [1] [2]

(1FP ≈ 30 to 50 SLOC)

CUESTIONES QUE CAMBIAN

- La modalidad de la gestión
  - Rol del gerente
  - Rol del equipo de trabajo
  - «Empoderamiento» del equipo
- La relación con el cliente
  - Cuestiones contractuales
  - Confianza
- La conformación del equipo de trabajo
  - Mayor importancia de la capacidad de aprendizaje.
  - Los proyectos son menos previsibles, por lo que la asignación de especialistas es más dinámica, se requiere a ellos en función de la evolución del proyecto.

REFERENCIAS

- http://agilemanifesto.org
- Kent Beck (1999). Extreme Programming Explained: Embrace Change;
- Takeuchi, Hirotaka et al. (1986). The New New Product Development Game. Harvard Business Review, www.scrumalliance.org.
- Henrik Kniberg & Mattias Skarin (2010). Kanban and Scrum – making the most of both.
- http://www.nebulon.com/articles/fdd/latestfdd.html.
- A.Cockburn (2002). Agile Software Development.
- Jim Highsmith (2010). Agile Project Management- Creating Innovative Products, 2nd edition.
- Capers Jones (2010). Software Engineering Best Practices: Lessons from Successful Projects in the Top Companies.

PMI

FASES

- Fases secuenciales (iteraciones):
  - Cada iteración tiene como entregable código operativo.
  - Todos los procesos (análisis, diseño, implementación, verificación) se realizan dentro de una sola iteración para producir un incremento de código.
- Fase más grande llamada *release:*
  - Su entregable principal es un conjunto de los incrementos de código que puedan ser puestos en producción o entregados el cliente.

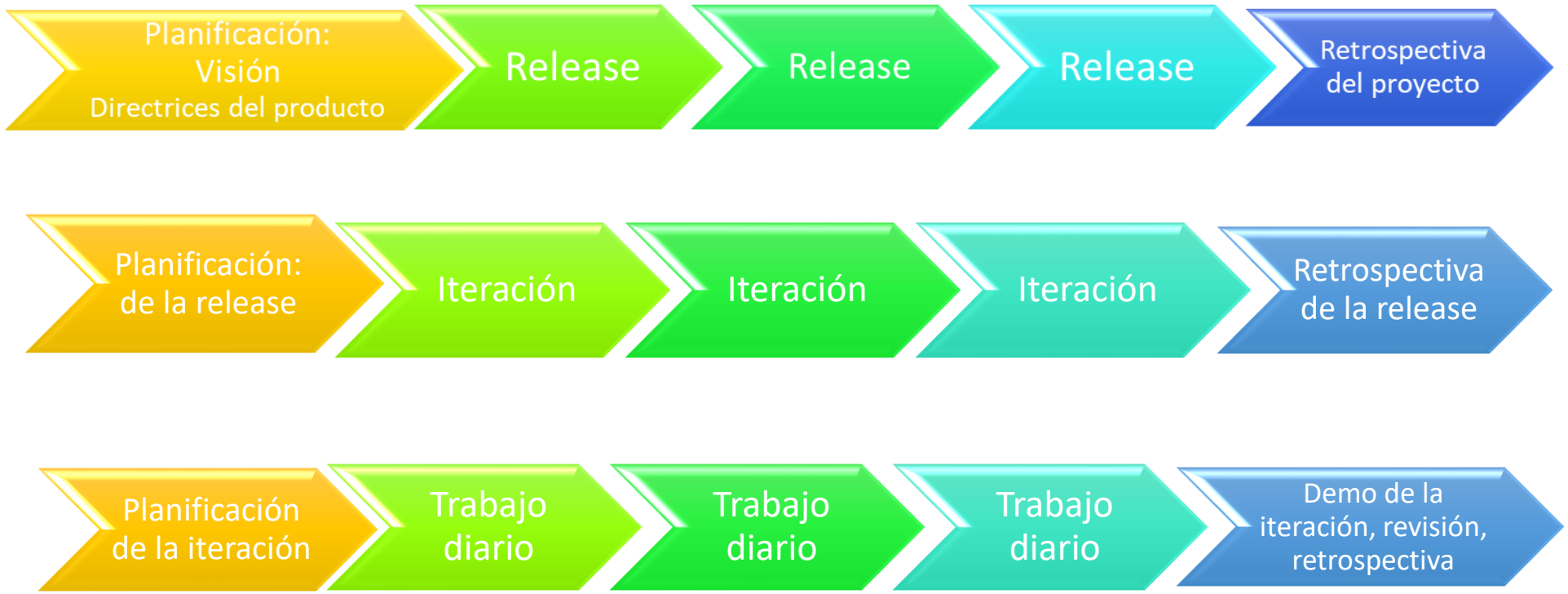
ITERACIONES

- Cantidad de iteraciones
  - determinada por el cliente, con base en lo que este defina como la mínima cantidad de funcionalidad aceptable para una *release*.
- Longitud de las iteraciones
  - Generalmente entre 1 y 4 semanas.
  - Está determinada por el cliente.
  - Factores que afectan la duración de las iteraciones:
    - La volatilidad de la industria
    - La cantidad de riesgo
    - La claridad de la visión

ITERACIONES

- Primera iteración:
  - En proyectos ágiles en la primera iteración hay un proceso de planificación.
  - El entregable es un plan del proyecto a alto nivel.
  - Tmb. puede haber un primer incremento de código entregado.
- Iteración final (de una *release* o de varias):
  - se preparar al producto para su entrega,
  - retrospectiva final del proyecto y
  - otros procesos de cierre.
  - [Pero esto tmb lo hago en cada iteración]
- Cada fase (iteración) debe tener
  - una iniciación formal en la que se defina cuáles son los entregables esperados de esa fase. Las iteraciones ágiles comienzan con una reunión de planificación para definir qué se va a completar en la iteración.
  - y terminan con una revisión formal al final para aprender de los eventos y obtener la aceptación del cliente por los *features* que se entregaron. Durante la revisión, el proyecto puede ser cancelado o aprobado para continuar, o se puede pedir una liberación que se puede implementar inmediatamente o en la siguiente iteración. Para concluir la fase se obtiene el permiso de continuar o se toma la decisión de detener el proyecto.

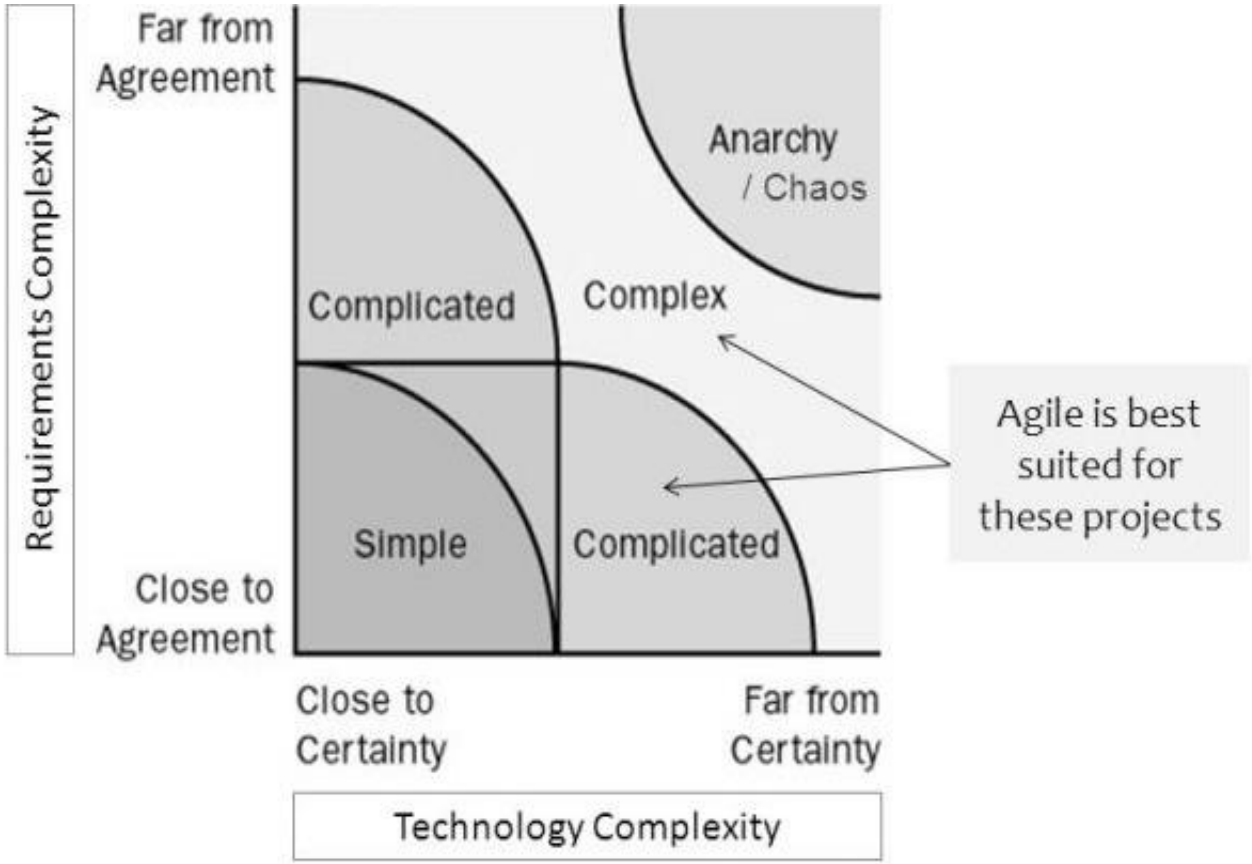
EL FRACTAL ÁGIL



MAPEO DE LOS PROCESOS DE PMI AL FRACTAL ÁGIL

	Grupos de procesos de gestión de proyectos				
Fractal ágil	**Inicio**	**Planificación**	**Ejecución**	**Seguimiento y control**	**Cierre**
Proyecto	Caso de negocio o estudio de factibilidad.	**Kickoff del proyecto y reunión de visionamiento. Planificación del hoja de ruta de la *release.***	Entrega iterativa e incremental de sw operativo.	**Revisiones periódicas de los entregables, de avance y del proceso.**	Retrospectiva del proyecto.
Release	Definición de la hoja de ruta de la *release.*	**Reunión de planificación de la *release.***	Entrega iterativa e incremental de sw operativo.	**Revisiones periódicas de los entregables, de avance y del proceso.**	Retrospectiva de la *reléase.*
Iteración	Reunión de planificación de la iteración.	**Reunión de planificación de la iteración.**	Desarrollo y verificación de los *features.*	**Panel de tareas, gráficas de *burn-down,* reuniones diarias de pie, aceptación de los *features* terminados.**	Demostración, revisión y retrospectiva de la iteración.

LA INCERTIDUMBRE INFLUYE EN LA METODOLOGÍA



Source: "Strategic Management and Organizational Dynamics " by Ralph D Stacey in Agile Software Development with Scrum by Ken Schwaber, Mike Beedle

- Los proyectos más adecuados para ser abordados por la metodología ágil son aquellos que:
  - requieren de investigación
  - tienen altas posibilidad de sufrir cambios
  - presentan requisitos desconocidos o poco claros
  - tienen un propósito difícil de describir