Gestión de Proyectos Software

Scrum – Sprints

Contenidos

- Sprints
- Definición de hecho
- Actividades de un sprint
  - Planificación
  - Ejecución
  - Revisión
  - Retrospectiva

Sprints

Sprints

- Scrum organiza el trabajo en iteraciones de hasta un mes de duración llamadas sprints
  - Duración limitada (*timeboxed*), corta y consistente
  - Con un objetivo que no debería cambiar una vez se empieza
  - Deben acabar alcanzando, para cada PBI, el estado que el equipo acuerde como "hecho"

Duración limitada (*timeboxed*) y corta

- Fechas de inicio y fin predefinidas e inmutables
- Permite limitar la cantidad de trabajo en marcha (*work in process, WIP)* (principio ágil)
  - Se trabaja solo en lo que se planea que se puede comenzar y terminar durante el sprint
  - Esto nos obliga a priorizar, eligiendo una cantidad pequeña de trabajo que sea lo más importante
- Demuestra progreso validando resultados (principio ágil)
  - Tenemos una fecha conocida y cercana donde habrá partes del trabajo valiosas ya completadas
  - Esto es mucho más fiable como medida de progreso que, p.ej., la conformidad con un plan
- Evita el perfeccionismo innecesario
  - Lo perfecto es enemigo de lo suficientemente bueno

Duración limitada (*timeboxed*) y corta

- Motiva el cierre
  - Las cosas se terminan cuando tienen una fecha de finalización firme establecida
    - Y más si esta es cercana
  - Entregar cosas frecuentemente es gratificante y por tanto mantiene más alto el interés del equipo
- Mejora las predicciones (equilibrar predicción y adaptación, principio ágil)
  - Cada vez es más fácil predecir lo que podemos hacer en un sprint, porque hemos hecho muchos otros de la misma duración antes
  - Es más fácil predecir lo que podemos hacer en una semana que en un año

Duración limitada (*timeboxed*) y corta

- Tenemos *feedback* pronto (adaptación, principio ágil)
  - Al final de cada corto sprint, es decir, frecuentemente, tenemos *feedback* con los clientes, patrocinadores etc.
    - Esto genera oportunidades para la inspección y la adaptación en base a este feedback
    - Podemos tomar decisiones basadas en hechos
- Mejora el retorno de la inversión
  - Al generar mejoras de producto antes, y más frecuentemente, también podemos empezar a generar antes los beneficios debidos a esas mejoras
- Error acotado
  - ¿Cómo de mal lo puedes hacer en un sprint de dos semanas? En el peor de los casos, si lo hicieras todo mal habrás perdido tan solo dos semanas

Duración consistente

- Como regla, un equipo debería elegir una duración consistente (siempre igual) para sus sprints en un proyecto
- Podemos cambiar con una buena razón:
  - Tender hacia sprints más cortos conforme cogemos experiencia
  - Las vacaciones llegan y es más práctico tener un sprint de tres semanas antes de agosto que los normales de dos
  - Etc.

Duración consistente

- No poder acabar el trabajo previsto en el sprint actual **no** es una buena razón para cambiar su duración
  - Es un síntoma de algún problema y una oportunidad para mejorar
- En la práctica, una semana suelen ser cinco días
  - Si hay algún día de fiesta de por medio en un sprint de una semana, se podrá hacer menos trabajo en ese sprint, pero su duración no cambia
- Tener sprints de la misma duración nos proporciona **cadencia**
  - Un ritmo regular y predecible
  - Esto facilita habituarse a una forma de trabajar
    - Los lunes solemos hacer tal…, para los miércoles generalmente ya hemos hecho cual...
  - También permite mantener más constante la intensidad del trabajo
    - En un proyecto tradicional, el trabajo es más intenso más cerca del final
  - Facilita la agenda
    - Es más fácil organizar reuniones cuando sabemos las fechas de las próximas N con anticipación
  - Facilita coordinar a varios equipos en el mismo proyecto
  - Si tenemos una fecha de lanzamiento/entrega predefinida, sabemos fácilmente cuántos sprints tendremos hasta esa fecha

Objetivo de un sprint

- El objetivo describe el propósito y valor del sprint
- Idealmente es algo claro y único. Ejemplos:
  - Implementar la generación inicial de informes
  - Cargar y revisar los datos para el mapa de Europa
  - Demostrar la capacidad de enviar un mensaje a través de una pila integrada de software, firmware y hardware
- Pero también puede tener varias partes
  - Que la impresión básica funcione y que se pueda buscar por fecha

Compromiso mutuo

- Durante la planificación de un sprint, el equipo de desarrollo ayuda a decidir el objetivo del sprint
  - Este objetivo se usa para elegir las entradas de la pila del producto que se pueden completar en el sprint
    - A su vez estas entradas permiten refinar el objetivo del sprint
- El objetivo del sprint es un **compromiso** entre el equipo de desarrollo y el dueño del producto
  - El equipo cumplirá el objetivo y el dueño no lo cambiará durante el sprint
- El objetivo del sprint no debe cambiar, pero puede ser clarificado
  - Una clarificación proporciona detalles adicionales para ayudar al equipo

Consecuencias del cambio

- Scrum acepta el cambio, pero cuando es equilibrado y económicamente sensato
- El coste del cambio aumenta conforme invertimos más en un trabajo
  - Con un sprint empezado, ya hemos hecho una inversión en las tareas de la pila del sprint
    - Como mínimo planificarlas
    - A mitad del sprint ya habremos diseñado/implementado y probado cosas
  - Cualquier cambio a mitad de sprint exigirá replanificar el resto del sprint
  - El equipo se desmotivará si el dueño del producto puede incumplir sus compromisos en cualquier momento
  - Es difícil concentrarse en una tarea si en cualquier momento te pueden decir que esa tarea ya no es necesaria

Ser pragmático

- No cambiar el objetivo de un sprint es una regla
- Pero puede ocurrir algo que haga que el objetivo se convierta en irrelevante. Lo lógico será cambiarlo
  - Un competidor acaba de lanzar un producto. Nuestro objetivo para el sprint actual pasa a ser económicamente inviable
  - Un sistema en producción falla y hace falta gente de nuestro equipo para solucionarlo inmediatamente

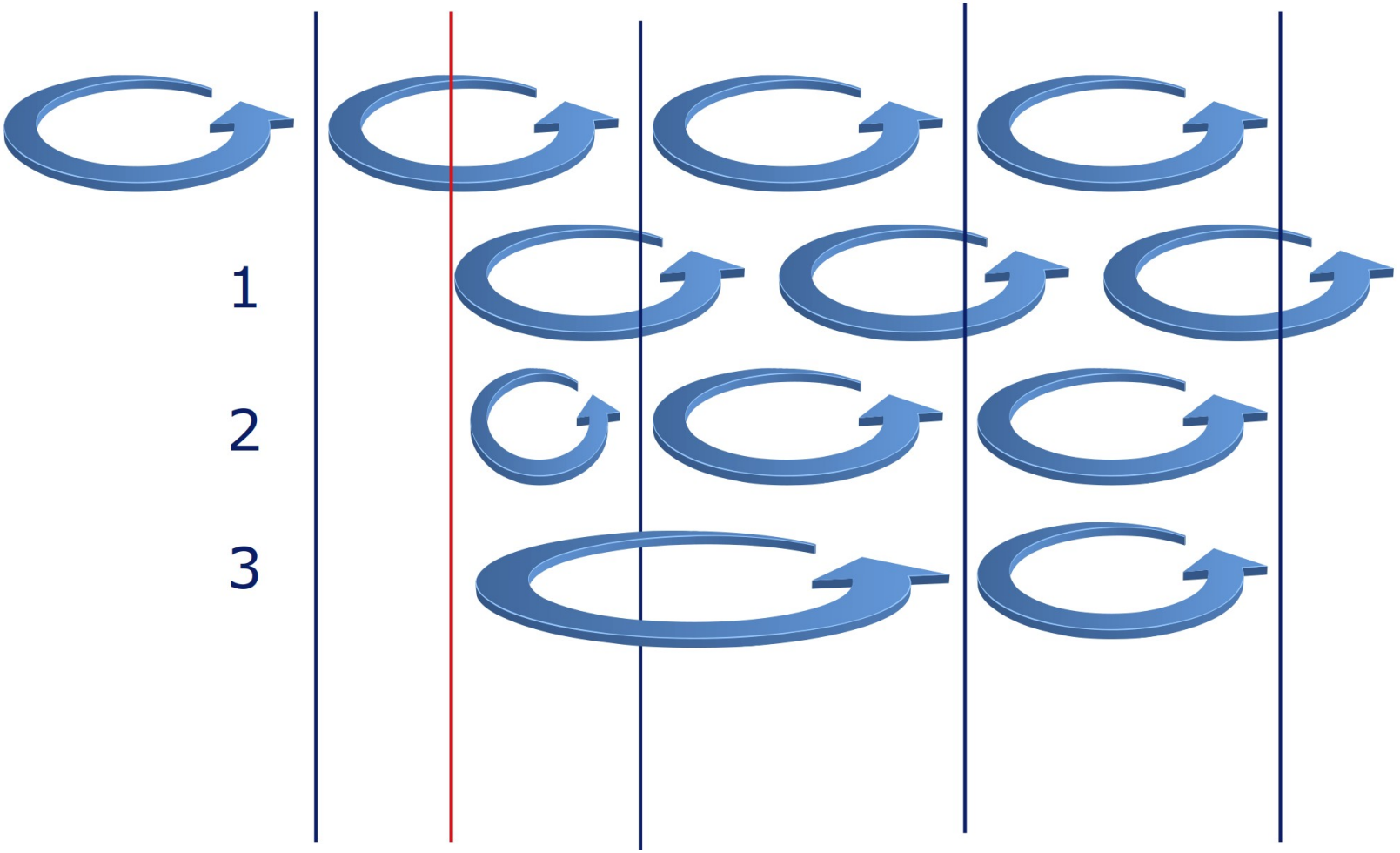
Terminación anormal de un sprint

- Si el objetivo del sprint se convierte en inválido, el equipo Scrum puede decidir terminar anormalmente el sprint
  - No hay incremento de producto potencialmente entregable y por tanto no se puede hacer la revisión con el cliente
  - Así que se pasa a la retrospectiva y a planificar el siguiente sprint
- Muchas veces será económicamente más sensato terminar el sprint normalmente
  - Sobre todo si queda poco para el final
  - Puede ser preferible retirar algo de la pila del sprint en lugar de terminar el sprint anormalmente (hay que ver qué es más costoso)
- Terminar anormalmente anula las ventajas que tiene la duración consistente de los sprints
  - Es un último recurso

Terminación anormal

- Si se decide terminar anormalmente, hay que determinar la longitud del siguiente sprint. 3 opciones
  - 1. Mantener la duración original, alterando las fechas de inicio y fin
  - 2. Acortar el siguiente sprint para terminar en la fecha prevista del terminado anormalmente
  - 3. Alargar el siguiente sprint para incluir lo que falte del terminado anormalmente y el siguiente

Terminación anormal



Definición de hecho

Definición de hecho

- El resultado de un sprint debería ser un incremento del producto potencialmente entregable al cliente
  - Aunque no se vaya a entregar en cada sprint
- Potencialmente entregable es un estado de confianza en lo que se ha hecho
  - No hay nada sustancial que no se haya tenido en cuenta
  - El producto alcanza un nivel de calidad establecido
- Cada equipo Scrum tiene que acordar qué consideran que está hecho

Definición de hecho

- Algo está hecho (potencialmente entregable) cuando una lista de tareas se ha realizado exitosamente. Por ejemplo:
  - Diseño revisado, código completo y verificado (estándar de codificación, comentado, revisado), documentación de usuario actualizada, tests pasados (unitarios, de integración, de regresión, de plataforma, de idioma), cero errores detectados, tests de aceptación por el dueño de producto pasados y está funcionando en los servidores de producción
- La lista anterior depende del tipo de producto, tecnologías, la organización que lo crea y posibles impedimentos actuales
- En todo caso siempre se tiene que entregar algo de valor para el cliente
- Si algún elemento no pasa la lista entera, no está hecho
  - Por ejemplo, si queda un bug en alguna entrada de la pila del sprint, esta se devuelve a la pila del producto y se abordará en otro sprint

Definición de hecho

- Hecho puede incluir alguna tarea muy costosa, que no cabe en un sprint
  - Si es por falta de automatización, habrá que resolverlo
  - Si no, habrá que aceptar que algunas cosas se comiencen en un sprint y se acaben en el siguiente
    - Por ejemplo podemos requerir que se pase un test de carga/estrés de 2 semanas cuando trabajamos con sprints de 1
- La definición de hecho puede cambiar con el tiempo
- Muchos equipos empiezan con una definición de hecho que no permitiría entregar al cliente
  - Y una vez tienen una base de producto desarrollada, refinan la definición de hecho
- A veces hay que alterar temporalmente algún aspecto de la definición de hecho
  - Por ejemplo, si hecho implica "probado en un dispositivo real" que aún no ha llegado, se puede cambiar temporalmente a "probado en el emulador"

Hecho frente a aceptado

- El dueño del producto establece unas condiciones de satisfacción (criterios de aceptación) para cada entrada de la pila del producto
  - Que luego se validan con tests de aceptación (posiblemente automáticos)
  - Pasar estos tests puede planificarse como una tarea dentro del sprint en el que esa entrada de pila se implementa
    - Tanto si son automáticos como si son manuales, pasarlos va a requerir algo de tiempo de alguien, aunque solo sea para comprobar los resultados
- Los criterios de aceptación son adicionales a la definición de hecho
  - Una entrada de la pila del producto tiene que pasar sus criterios de aceptación **y** la definición de hecho para estar completa/aceptada
    - Frente a estar "simplemente" hecha
  - Muchos equipos incluyen como una entrada más de la definición de hecho "criterios de aceptación pasados"
    - En ese caso toda entrada de la pila hecha también estará completa/aceptada

Definición de hecho en GitLab

Definition of done

If you contribute to GitLab, please know that changes involve more than just code. We use the following definition of done. To reach the definition of done, the merge request must create no regressions and meet all these criteria:

- Verified as working in production on GitLab.com.
- Verified as working for self-managed instances.
- Verified as supporting Geo through the self-service framework. For more information, see Geo is a requirement in the definition of done.

If a regression occurs, we prefer you revert the change. Your contribution is *incomplete* until you have made sure it meets all of these requirements.

Functionality

1. Working and clean code that is commented where needed.
2. The change is evaluated to limit the impact of far-reaching work.
3. Performance guidelines have been followed.
4. Secure coding guidelines have been followed.
5. Application and rate limit guidelines have been followed.
6. Documented in the `/doc` directory.
7. If your MR touches code that executes shell commands, reads or opens files, or handles paths to files on disk, make sure it adheres to the shell command guidelines
8. Code changes should include observability instrumentation.
9. If your code needs to handle file storage, see the 📎 uploads documentation.
10. If your merge request adds one or more migrations, make sure to execute all migrations on a fresh database before the MR is reviewed. If the review leads to large changes in the MR, execute the migrations again after the review is complete.
11. If your merge request adds new validations to existing models, to make sure the data processing is backwards compatible:
    - Ask in the `#database` Slack channel for assistance to execute the database query that checks the existing rows to ensure existing rows aren't impacted by the change.
    - Add the necessary validation with a feature flag to be gradually rolled out following the rollout steps.

    If this merge request is urgent, the code owners should make the final call on whether reviewing existing rows should be included as an immediate follow-up task to the merge request.

    NOTE: There isn't a way to know anything about our customers' data on their self-managed instances, so keep that in mind for any data implications with your merge request.
12. Consider self-managed functionality and upgrade paths. The change should consider both:
    - If additional work needs to be done for self-managed availability, and
    - If the change requires a required stop when upgrading GitLab versions.

    Upgrade stops are sometimes requested when a GitLab code change is dependent upon a background migration being already complete. Ideally, changes causing required upgrade stops should be held for the next major release, or at least a 3 milestones notice in advance if unavoidable.

https://gitlab.com/gitlab-org/gitlab-foss/-/blob/master/doc/development/contributing/merge_request_workflow.md#definition-of-done

Testing

1. Unit, integration, and system tests that all pass on the CI server.
2. Peer member testing is optional but recommended when the risk of a change is high. This includes when the changes are far-reaching or are for components critical for security.
3. Regressions and bugs are covered with tests that reduce the risk of the issue happening again.
4. For tests that use Capybara, read how to write reliable, asynchronous integration tests.
5. Black-box tests/end-to-end tests added if required. Please contact the quality team with any questions.
6. The change is tested in a review app where possible and if appropriate.
7. Code affected by a feature flag is covered by automated tests with the feature flag enabled and disabled, or both states are tested as part of peer member testing or as part of the rollout plan.
8. If your merge request adds one or more migrations, write tests for more complex migrations.

UI changes

1. Use available components from the GitLab Design System, Pajamas.
2. The MR must include *Before* and *After* screenshots if UI changes are made.
3. If the MR changes CSS classes, please include the list of affected pages, which can be found by running `grep css-class ./app -R`.

Description of changes

1. Clear title and description explaining the relevancy of the contribution.
2. Description includes any steps or setup required to ensure reviewers can view the changes you've made (for example, include any information about feature flags).
3. Changelog entry added, if necessary.
4. If your merge request introduces changes that require additional steps when installing GitLab from source, add them to `doc/install/installation.md` in the same merge request.
5. If your merge request introduces changes that require additional steps when upgrading GitLab from source, add them to `doc/update/upgrading_from_source.md` in the same merge request. If these instructions are specific to a version, add them to the "Version specific upgrading instructions" section.

Approval

1. The MR acceptance checklist has been checked as confirmed in the MR.
2. Create an issue in the infrastructure issue tracker to inform the Infrastructure department when your contribution is changing default settings or introduces a new setting, if relevant.
3. An agreed-upon rollout plan.
4. Reviewed by relevant reviewers, and all concerns are addressed for Availability, Regressions, and Security. Documentation reviews should take place as soon as possible, but they should not block a merge request.
5. Your merge request has at least 1 approval, but depending on your changes you might need additional approvals. Refer to the Approval guidelines.
    - You don't have to select any specific approvers, but you can if you really want specific people to approve your merge request.
6. Merged by a project maintainer.

Production use

The following items are checked after the merge request has been merged:

1. Confirmed to be working in staging before implementing the change in production, where possible.
2. Confirmed to be working in the production with no new Sentry errors after the contribution is deployed.
3. Confirmed that the rollout plan has been completed.
4. If there is a performance risk in the change, you have analyzed the performance of the system before and after the change.
5. *If the merge request uses feature flags, per-project or per-group enablement, and a staged rollout:*
    - Confirmed to be working on GitLab projects.
    - Confirmed to be working at each stage for all projects added.
6. Added to the release post, if relevant.
7. Added to the website, if relevant.

Contributions do not require approval from the Product team.

https://gitlab.com/gitlab-org/gitlab-foss/-/blob/master/doc/development/contributing/merge_request_workflow.md#definition-of-done

Dependencies

If you add a dependency in GitLab (such as an operating system package) please consider updating the following, and note the applicability of each in your merge request:

1. Note the addition in the release blog post (create one if it doesn't exist yet).
2. The upgrade guide.
3. The GitLab Installation Guide.
4. The GitLab Development Kit.
5. The CI environment preparation.
6. The Omnibus package creator.
7. The Cloud Native GitLab Dockerfiles

Incremental improvements

We allow engineering time to fix small problems (with or without an issue) that are incremental improvements, such as:

1. Unprioritized bug fixes (for example, Banner alerting of project move is showing up everywhere)
2. Documentation improvements
3. RuboCop or Code Quality improvements

Tag a merge request with Stuff that should Just Work to track work in this area.

https://gitlab.com/gitlab-org/gitlab-foss/-/blob/master/doc/development/contributing/merge_request_workflow.md#definition-of-done

Actividades de un sprint

Planificación

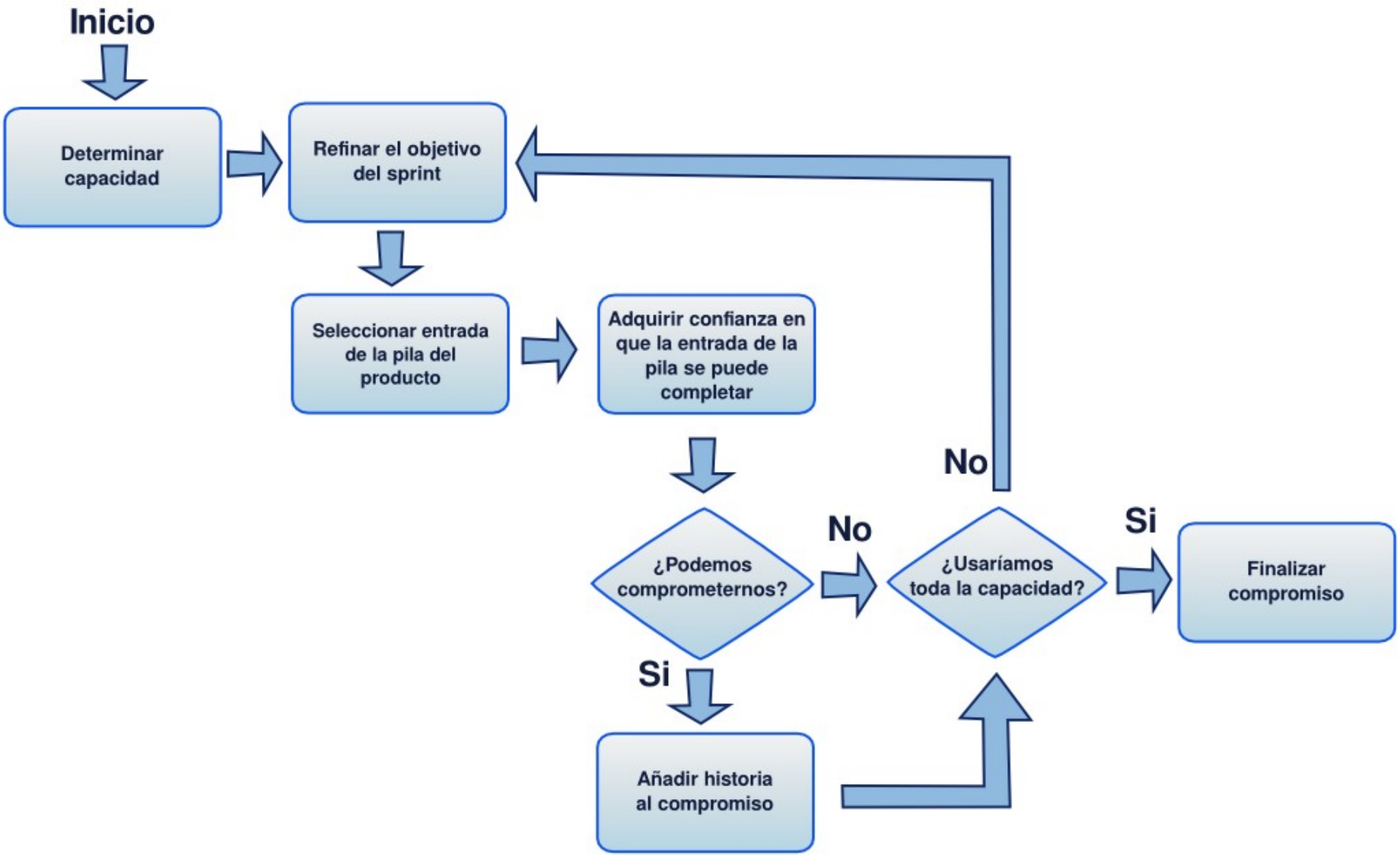
Planificación de un sprint

- Se decide un objetivo para el sprint, y se eligen las entradas de la pila que encajan con este objetivo
- Se planifica el sprint para que el equipo adquiera confianza en que podrá completarlo
  - Justo al principio de cada sprint
  - Se pueden dedicar hasta 2 horas por cada semana del sprint
    - Este tiempo puede disminuir conforme los equipos adquieren experiencia en el producto en el que trabajan
  - "Adquirir confianza" significa que una vez concluida la planificación, el equipo confía en que podrán terminar todo lo que han planificado dentro del sprint
- Las entradas de la pila elegidas y el plan del sprint forman la pila del sprint

Entradas del proceso de planificación de un sprint

- Una pila del producto adecuadamente mantenida (*groomed*)
  - Las entradas de la parte superior están listas
    - Criterios de aceptación claros, tamaño adecuado, estimadas, priorizadas
- Un objetivo claro para el sprint (por parte del dueño del producto)
  - Que se debe refinar durante la planificación
- La velocidad del equipo
- Otras consideraciones
  - Disponibilidad de los miembros del equipo, sus conocimientos y habilidades, y restricciones de negocios o tecnológicas que puedan afectar

Planificación de sprint



Determinar la capacidad

- Para la planificación tenemos que calcular **la capacidad que tiene el equipo para trabajar en entradas de la pila** durante el sprint
- El tiempo disponible (p.ej., "jornadas laborables en el sprint") no se usa solo para trabajar en entradas de la pila
  - Otras actividades del sprint
    - Planificación, revisión, retrospectiva, *grooming* de la pila del producto
  - Actividades relacionadas con la mejora del proceso Scrum, posiblemente decididas en alguna retrospectiva anterior
- Y también habrá tiempo laboral no disponible para el sprint
    - Vacaciones, médico...
    - Compromisos profesionales fuera del sprint
      - Trabajo en otros proyectos, tareas administrativas, otras reuniones...

Capacidad en puntos de historia

- Si expresamos la velocidad del equipo en puntos de historia, determinar la capacidad disponible para un sprint es predecir la velocidad para ese sprint
- Se coge la velocidad media del equipo, y si el sprint no va a ser un sprint "promedio", entonces se ajusta (normalmente a la baja)
  - Por ejemplo si va a haber mucha gente de vacaciones u otros proyectos robando mucho tiempo habrá que bajarla

Capacidad en horas-persona

- Si queremos afinar más que con los puntos de historia, podemos determinar la capacidad en horas-persona
- Cada miembro del equipo de desarrollo estima cuántas horas tendrá disponibles para trabajar en las entradas de la pila del sprint
  - Como casi toda estimación, es buena idea darla como un rango [mínimo – máximo] de nº de horas disponibles previstas
  - Recordad que en esas horas no entran las que hagan falta para otras actividades Scrum del proyecto, ni para otros proyectos, vacaciones etc.

Seleccionar entradas de la pila del producto

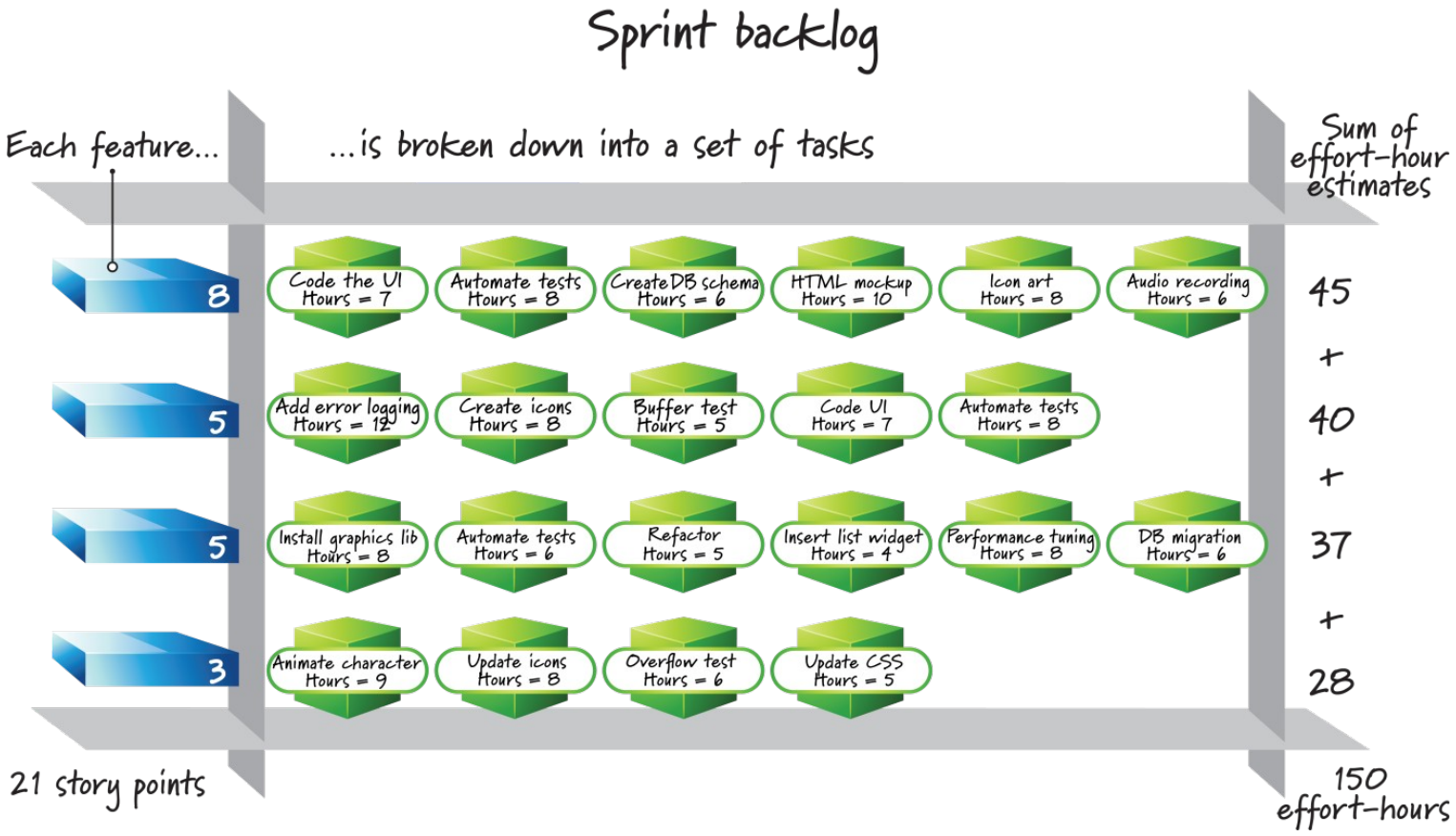
- Si hemos definido un objetivo claro para el sprint, elegimos las entradas que encajen con él
- Si no, elegiremos entradas de las de la cima de la pila en orden de más a menos prioridad
  - Si, p.ej., tras elegir las dos primeras, y aún con capacidad disponible, vemos que no queda capacidad para la tercera, probaremos con la cuarta que puede ser más corta
  - O trataremos de partir la tercera en dos o más
- Se empieza solo lo que se puede terminar en el sprint

Adquirir confianza

- Podemos usar la velocidad del equipo para determinar si el compromiso es realista
  - Si haces entre 25 y 30 puntos de historia por sprint, comprometerte a hacer 25 es razonable a priori
  - Un equipo con experiencia que lleva tiempo trabajando en un producto puede encontrar que esto es suficiente
- Pero es aún más fiable dividir las entradas elegidas en tareas, porque hasta que no haces esta división, hay cosas que no sabes
  - Interdependencias, demasiadas tareas que solo un miembro del equipo sabe hacer etc.
- ¿Cuántas tareas? ¿Cuáles?
  - Las que hagan falta para cumplir la definición de hecho y ser aceptadas por el dueño de producto
  - Estimadas en horas-persona y restadas de la capacidad del equipo para comprobar si son un compromiso razonable

La pila del sprint

- El resultado de la planificación del sprint es la pila del sprint



La pila del sprint

- En la pila del sprint no está toda la información para saber si hemos adquirido un compromiso razonable
  - P.ej., ¿qué pasa si el especialista en interfaz de usuario va a estar de vacaciones más de medio sprint?
- En Scrum generalmente no se asignan miembros del equipo a tareas durante la planificación del sprint
  - Pero al menos hay que tener esto en cuenta lo suficiente como para determinar si el compromiso para el sprint es o no es razonable

Finalizar el compromiso

- El objetivo del sprint y las entradas de la pila del producto que se han seleccionado representan el compromiso del equipo
  - El objetivo del sprint se ha podido refinar durante la planificación si, p.ej., se ha visto que era demasiado ambicioso
  - A veces en lugar de compromiso se dice predicción, para minimizar una actitud defensiva a la hora de hacer las estimaciones

Pila del sprint

- Teniendo en cuenta que durante un sprint no se debe cambiar el trabajo que nos hemos comprometido a hacer
- ¿Permitiremos añadir nuevas tareas al sprint en medio del mismo?

Ejecución

Ejecución de un sprint

- Empieza tras la planificación del sprint y termina antes de la revisión
- Ocupa la mayor parte del tiempo del sprint
  - También habrá que dejar tiempo para hacer grooming de la pila y para las reuniones de planificación, revisión y retrospectiva
- Los miembros del equipo se auto-organizan para determinar la mejor forma de alcanzar el objetivo establecido en la planificación
  - El ScrumMaster facilita el trabajo al equipo
    - No se encarga de repartir trabajo ni de decir cómo se hace
  - El dueño del producto debe estar disponible para clarificar cosas, revisar trabajo intermedio, proporcionar *feedback* y verificar que los criterios de aceptación se cumplen

Planificación de la ejecución de un sprint

- El equipo ha creado una pila del sprint durante su planificación
- No es un plan de trabajo detallado en forma, p.ej., de diagrama de Gantt
  - El coste de crear algo así de detallado para cada sprint sería difícil de justificar
- Sí que es importante hacer la planificación suficiente para exponer dependencias fuertes entre tareas y así poder ordenarlas en el tiempo
  - El resto se puede abordar de forma oportunista, durante la ejecución del sprint

Organizar el trabajo

- Primero se elige por dónde empezar
  - Por las PBI más prioritarias de la pila del sprint es lo más obvio
  - Dependencias técnicas o restricciones por el personal disponible pueden afectar a esto
- También hay que evitar pensar en cada PBI como un "miniproyecto" con metodología en cascada
  - No: diseño la PBI, luego la implemento y al final del sprint la pruebo
  - Sí: diseño un poco de la PBI, lo implemento, lo pruebo, diseño un poco más…
    - Esto facilita aún más el repartir y paralelizar el trabajo
- El equipo decide qué tareas hay que hacer en cada momento y quién hará cada una

Organizar el trabajo

- Paralelizar el trabajo ayuda a poder terminar más cosas
  - Pero trabajar en demasiadas cosas al mismo tiempo requiere demasiados cambios de contexto, lo que es ineficiente
- En general, los miembros del equipo con capacidad disponible deberían trabajar en cosas que están empezadas en lugar de en cosas nuevas
  - P.ej., ponerse con los tests de una PBI que está medio desarrollada en lugar de empezar a desarrollar una nueva
  - Esto requiere que todo el mundo sepa hacer distintas cosas
- No es que haya que evitar siempre el estar trabajando en varias PBI al mismo tiempo
  - Hay que evaluar el trabajo pendiente, las habilidades de cada cual etc.
- Igual que con las PBI, también hay que evitar abordar cada sprint como si fuera un "miniproyecto" con una aproximación en cascada
  - Por ejemplo, no queremos analizar todas las PBI antes de empezar a diseñarlas, ni diseñar/implementar todo antes de empezar a hacer pruebas
  - El riesgo de no completar ninguna entrada de la pila es grande si lo hacemos así

Scrum diario

- Es parte de la inspección y adaptación (principio ágil)
  - Cada día se ve lo que está haciendo todo el mundo
    - No solo lo que te afecta más directamente, que esto lo verías casi seguro aunque no se hiciera el Scrum diario
  - Salen a la luz posibles problemas sin esperar más de 24 horas
  - Facilita la planificación oportunista
- 15 minutos, siempre a la misma hora
- Es una actividad crítica para que Scrum funcione bien
  - La auto-organización del equipo es imposible si el equipo no habla con frecuencia, y esta reunión es la forma de asegurar que se habla al menos una vez al día
- Con una metodología tradicional esta reunión se vería como un desperdicio
  - Un tiempo en el que la gente no está "produciendo"
  - Innecesaria, porque en este tipo de proyectos se asume que solo la directora del proyecto necesita saber lo que hace la gente, tener visión global del proyecto y coordinar/repartir las tareas

Comunicación

- Generalmente se usa un tablero de tareas (*task board*) y diagramas de trabajo pendiente (*burndown*) y/o de trabajo completado (*burnup*)
  - El tablero de tareas se crea inicialmente a partir de la pila del sprint
- El tablero va reflejando los cambios durante el transcurso del sprint
  - Se suele recoger el estado en el que se encuentran las tareas necesarias para completar cada entrada de la pila
    - Por ejemplo pueden aparecer en tres estados posibles: por hacer, en progreso y completada

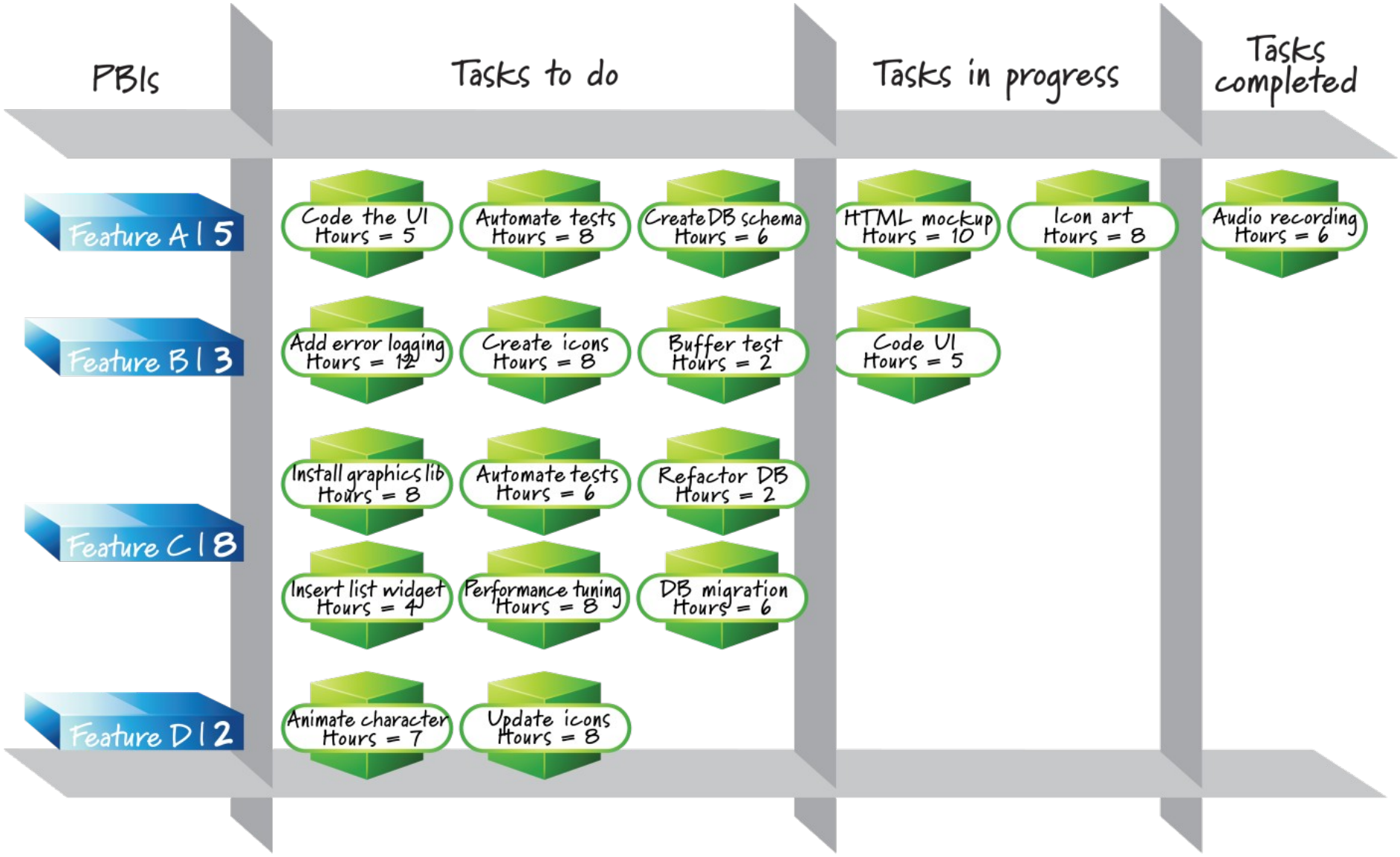


Diagrama de *burndown* del sprint

- El eje vertical son horas-persona
- El eje horizontal son los días dentro del sprint
- El diagrama representa las horas de esfuerzo que nos quedan para completar todas las tareas de la pila del sprint
  - Horas **estimadas**. Este diagrama no representa horas reales invertidas
    - Si una tarea se estimó en 4 horas y nos ha costado 8, en este diagrama pondremos 4
    - Por tanto no se puede usar para el control de esfuerzos del equipo Scrum

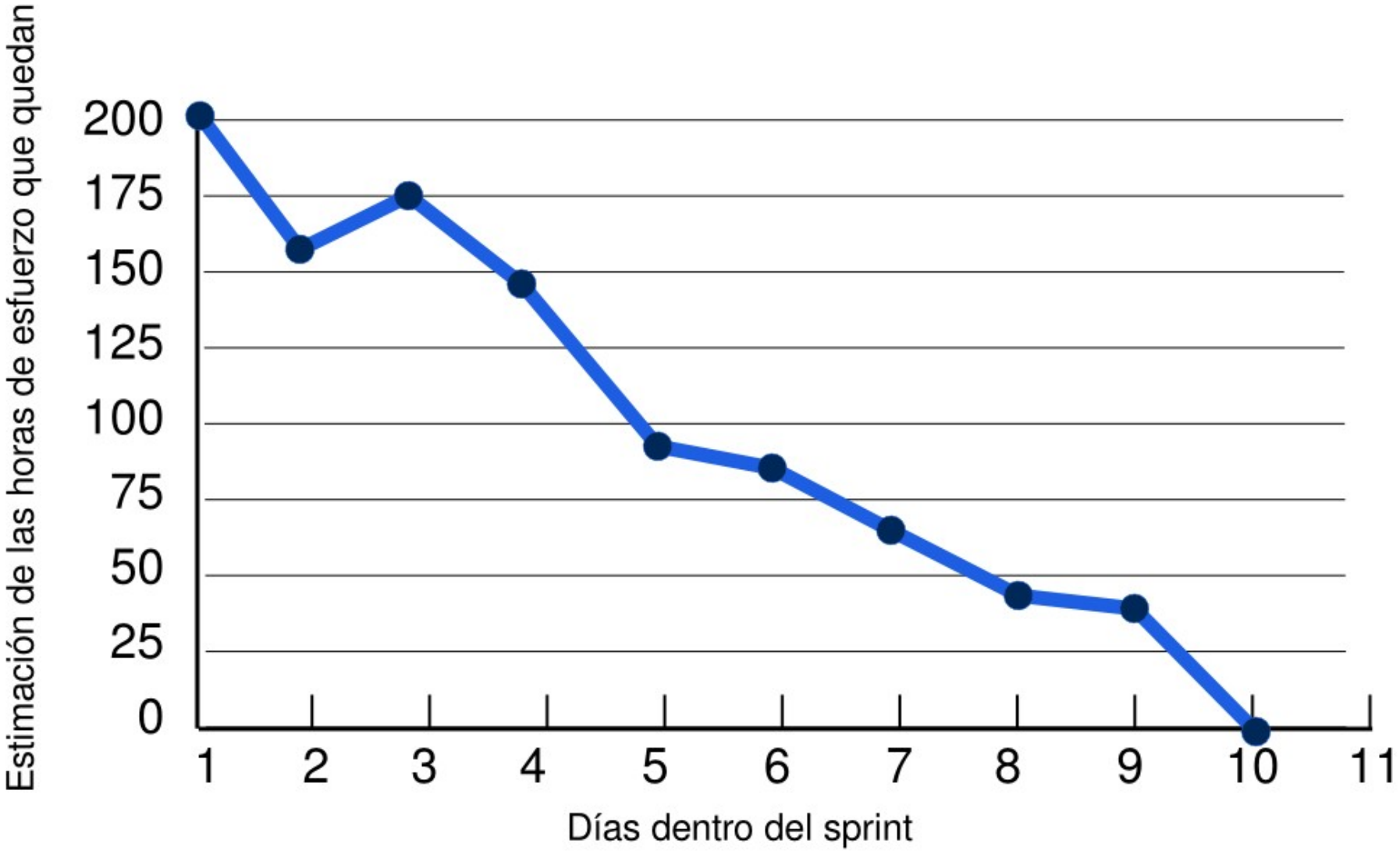

Estimación de las horas de esfuerzo que quedan
200
175
150
125
100
75
50
25
0
1
2
3
4
5
6
7
8
9
10
11
Días dentro del sprint

Diagrama de *burndown*

- ¿Qué significa que del día 2 al día 3 hayan aumentado las horas de esfuerzo de las tareas pendientes de hacer?

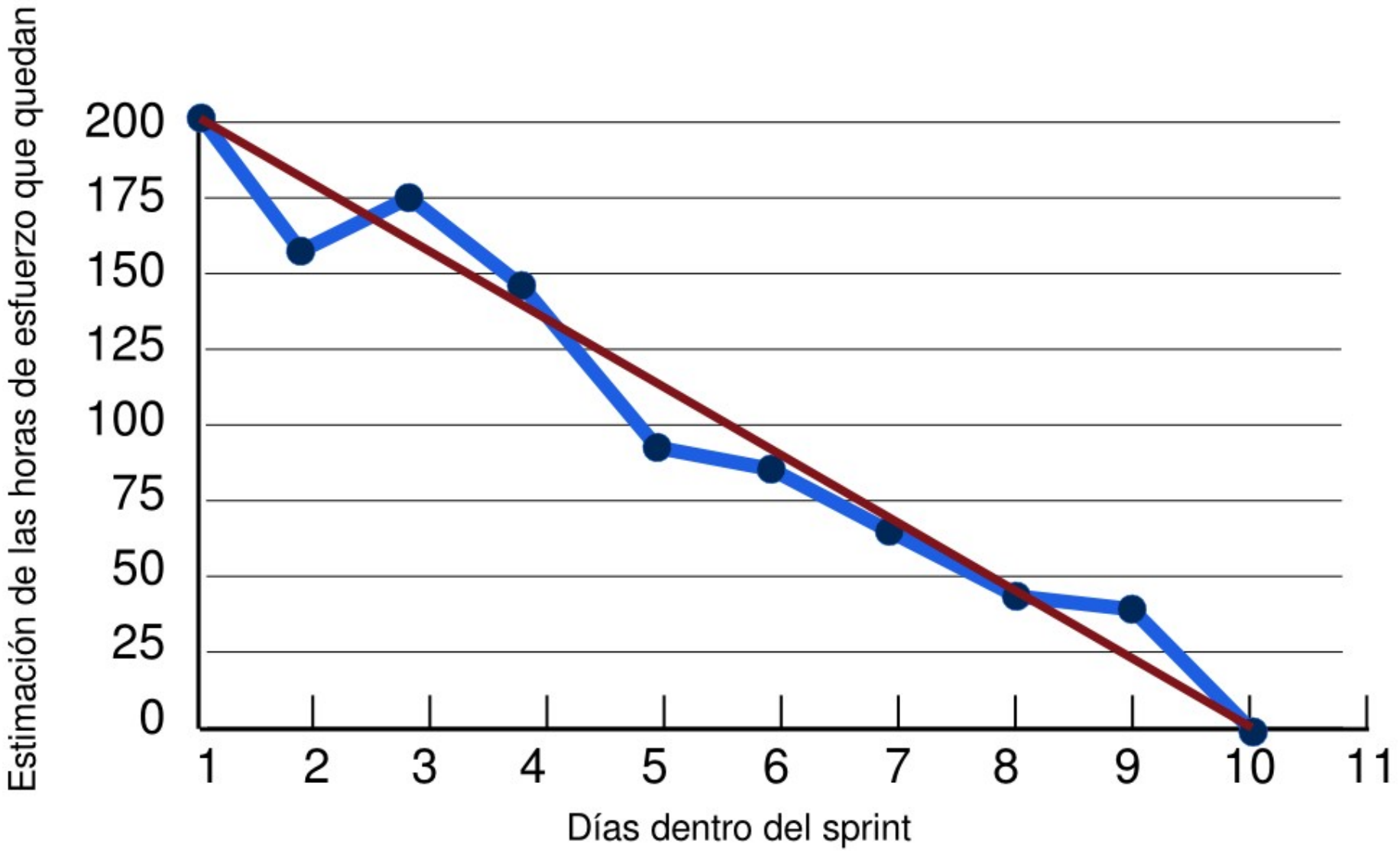

Estimación de las horas de esfuerzo que quedan
200
175
150
125
100
75
50
25
0
1
2
3
4
5
6
7
8
9
10
11
Días dentro del sprint

Diagrama de *burnup* del sprint

- El eje vertical son horas-persona o puntos de historia
  - Los puntos de historia serán en general más adecuados porque son una medida de progreso real
    - Porque sí o sí representan PBI terminadas
- El eje horizontal son los días dentro del sprint
- El diagrama representa el progreso llevado a cabo en el sprint

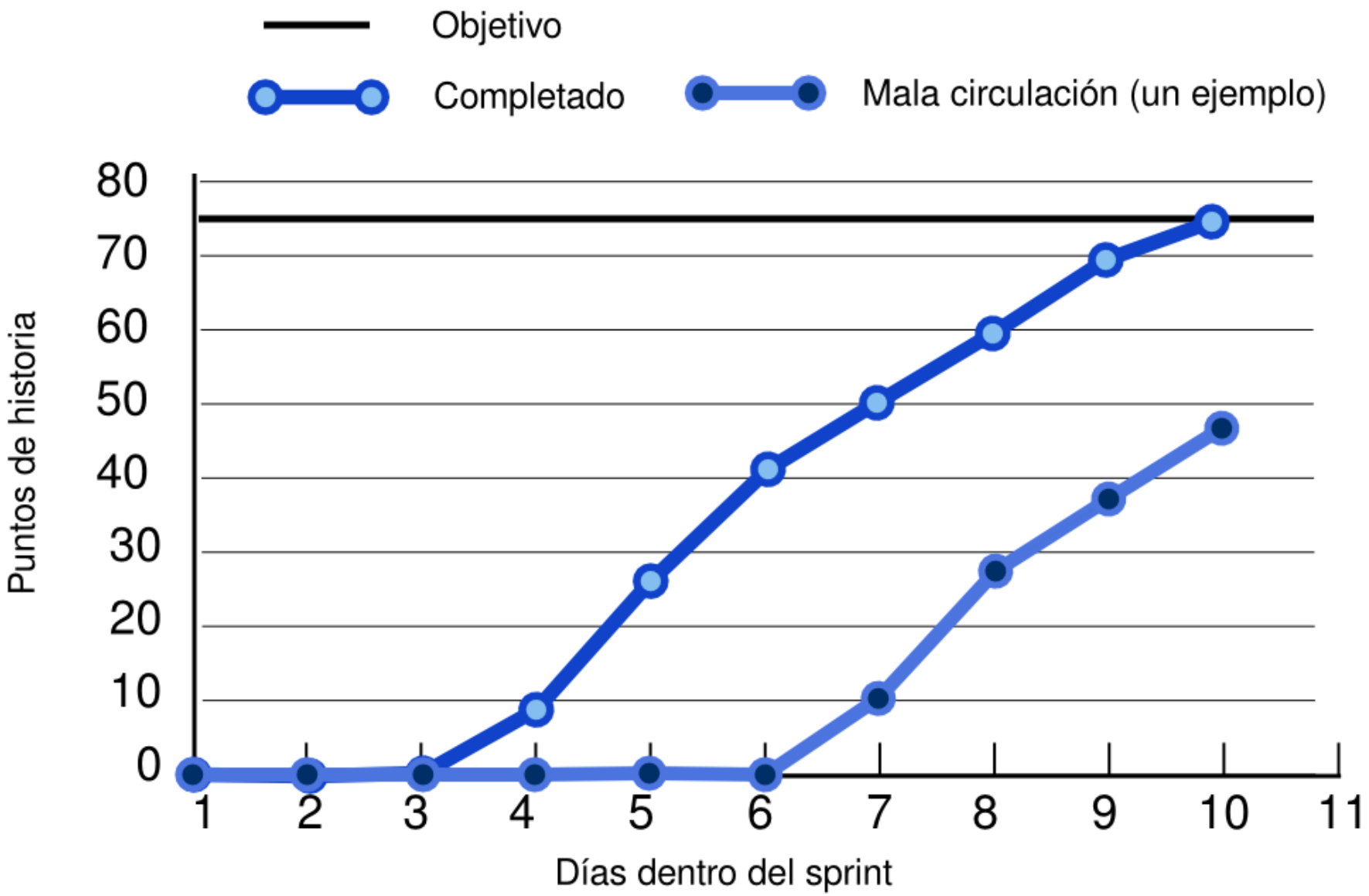

Objetivo
Completado
Mala circulación (un ejemplo)
Puntos de historia
80
70
60
50
40
30
20
10
0
1
2
3
4
5
6
7
8
9
10
11
Días dentro del sprint

Diagrama de *burnup*

- ¿Cuántas entradas de la pila del producto se habían planificado para el sprint anterior según el diagrama de *burnup* que muestra buena circulación (*flow*)?

Inciso. ¿Por qué tableros y diagramas bien visibles?

- Tenerlos actualizados y bien visibles los convierte en "radiadores" de información
  - Metáfora del radiador (da calor solo con pasar cerca) frente a la nevera (tienes que ir a buscar algo y abrirla para ver lo que hay dentro)
  - Cualquiera que pase delante los ve, y esto puede traducirse en oportunidades
    - Detección temprana de problemas, surgen ideas nuevas etc.
- Por otra parte, un meta-análisis realizado por la Universidad de Sheffield concluyó que
  - Monitorizar tus progresos hacia un objetivo, y más si es frecuentemente, ayuda a alcanzarlos
  - Esto tiene un efecto mayor si esa monitorización se graba físicamente y es pública
  - https://www.apa.org/news/press/releases/2015/10/progress-goals.aspx

Revisión

Revisión de un sprint

- Tras la ejecución del sprint, inspeccionamos sus resultados
  - El estado actual del producto
- Además del equipo Scrum completo, debe incluir gente (interesada) que no está disponible durante el sprint
  - Internos
    - Ejecutivos de la empresa, usuarios internos (si es un dpto. de la empresa el que va a hacer uso de la aplicación), personal de ventas (que tendrá que vender la aplicación), personal de soporte a usuarios (que tendrá que ayudar con sus dudas)...
  - Externos
    - Clientes, usuarios, socios…

Preparación de la revisión

- Determinar los asistentes, roles de cada uno y acordar horarios
- Asegurarse de que está claro el trabajo **completado** durante el sprint
  - Lo único que se puede presentar
  - El dueño del producto ya ha tenido que dar el visto bueno a ese trabajo durante la ejecución
- Preparar la demostración
  - El software real
  - Generalmente solo aquello que se ha hecho para alcanzar el objetivo del sprint

Ejecución de la revisión

- Alguien (típicamente el dueño del producto) cuenta el objetivo del sprint, las entradas de la pila asociadas y un resumen de lo que se ha completado
- Se hace una demo de las características relevantes implementadas
  - A veces es buena idea que sean clientes/usuarios "toquen" el producto con sus manos
- Se discute y se opina sobre lo que se ha visto (ideas, problemas, mejoras...)
  - No se solucionan problemas graves, pero se sacan a la luz
- Muchas veces se aprovecha para hacer algo de *grooming* de la pila del producto

Ejecución de la revisión – Preguntas y respuestas

- ¿A los clientes/usuarios les gusta lo que ven? ¿Querrían cambios?
- ¿Es todavía relevante lo que estamos haciendo, dados posibles cambios en nuestro entorno/mercado/contexto legal...?
- ¿Se nos ha pasado alguna característica importante?
- ¿Estamos trabajando demasiado en algo que no es tan importante?

¿Qué hacer si...

- Durante la revisión, un ejecutivo de alto nivel de la empresa del cliente indica que una de las entradas de la pila que ha visto demostradas no le parece que esté realmente terminada

¿Qué hacer si...

- Hay varios equipos Scrum trabajando en el mismo proyecto

Retrospectiva

Retrospectiva de un sprint

- Oportunidad para que el equipo Scrum examine su forma de trabajar, identifique formas de mejorar y haga planes para llevar a cabo esas mejoras
  - ¿Qué ha funcionado y debemos continuar haciendo?, ¿qué no ha funcionado y hay que dejar de hacer?, ¿qué debemos empezar a hacer o mejorar?
  - Mejora continua (Kaizen - 改善)
- Se examinan procesos, prácticas, comunicación, entorno, herramientas etc.
- Participa todo el equipo Scrum
  - Se puede invitar a alguien más, pero no será común

Preparación de la retrospectiva

- Decidir el foco
  - Por defecto todos los aspectos del proceso Scrum
  - Se puede elegir algo más concreto que sea importante para el equipo y que vaya a requerir esfuerzo
    - P.ej., "mejorar nuestro proceso de testing"
  - Con el foco determinado, se puede decidir si se invita a alguien externo al equipo Scrum
- Decidir los ejercicios a realizar
- Recopilar datos objetivos
  - P.ej., entradas de la pila que no se terminaron, los diagramas de *burndown*/*burnup* del sprint, bugs detectados etc.
  - Depende del foco
- Establecer duración
  - Como pauta, entre 30 y 45 minutos por cada semana de sprint

Ejecución de la retrospectiva

- Dejar claro que todo el mundo puede expresar sus opiniones, también las negativas, sin temor
  - Pero la retrospectiva no es el sitio para sacar a la luz problemas con personas concretas
    - Es para hablar del proceso y de la organización del trabajo
- Usar los datos objetivos como forma de que todo el mundo comparta una visión común de lo ocurrido durante el sprint
  - Y luego compartir las posibles visiones subjetiva de cada uno para terminar de asegurarnos de que no hay visiones dispares que no se han compartido
- Realizar los ejercicios decididos en la preparación

Actividad: línea temporal de eventos

- Representación visual de eventos del sprint
  - "Se colapsó el servidor", "hubo que parar para atender un problema en producción", "Ángel faltó dos días por enfermedad"
- Se pinta una línea horizontal dividida en los días del sprint
  - Cada miembro del equipo va añadiendo lo que considera significativo (según el foco de la retrospectiva) que ha ocurrido durante el sprint
  - Puede ser útil anotar también si algunos días alguien se sintió especialmente estresado o pesimista, o quizás todo lo contrario
    - Podría ayudar a identificar malas y buenas prácticas

Ejercicio: identificar cosas que se pueden mejorar

- Examinar la línea temporal de eventos para responder a
  - ¿Qué funcionó bien?
  - ¿Qué funcionó mal?
  - ¿Dónde hay oportunidades para hacer las cosas de otra forma?
- Los participantes recopilan las ideas que se les ocurran
  - También se pueden coger ideas que surgieron durante retrospectivas anteriores y aún no se han abordado
- Se agrupan las ideas en grupos por similitud
  - También se pueden agrupar en "cosas a seguir haciendo", "cosas a dejar de hacer" y "cosas para probar"
- Se discuten las ideas y se identifican áreas de mejora

Dinamizar la reunión

- Para que salgan a la luz las cosas se pueden preguntar distintas cosas a todos los miembros
  - ¿Qué crees que hay que empezar a hacer, dejar de hacer y seguir haciendo?
  - O usar las cuatro L: Liked, Learned, Lacked y Longed For
    - ¿Que te gustó, aprendiste, careciste, deseaste?
- También se puede examinar la definición de hecho, quizás para ampliarla

Determinar acciones

- Decidir qué hacer para solucionar los problemas identificados
  - Y comprobar si acciones decididas con anterioridad funcionan
- Lo primero es decidir qué problemas son más urgentes
  - Se puede hacer como una votación ponderada de los miembros del equipo
    - P.ej., cada persona tiene 5 votos y los puede repartir como quiera entre los problemas identificados
- Luego se determina cuánto tiempo se dedicará a solucionarlos
  - El dueño del producto, presente en la reunión, tiene mucho que decir aquí
- Finalmente se deciden las acciones que se llevarán a cabo

Determinar acciones

- Muchas acciones tomarán la forma de tareas específicas que algún miembro del equipo Scrum realizará en el próximo sprint
  - P.ej., si "el proceso de pasar los tests manuales en móviles reales" cuesta mucho tiempo, una acción puede ser "implementar un sistema que automatice el despliegue y que abra automáticamente las apps en los móviles de pruebas"
    - Buscando que los testers no pierdan el tiempo con esa parte
  - El equipo decide quién lo hará, y calcula cuánto tiempo llevará
    - Esto se tendrá en cuenta en la planificación del próximo sprint
- Hay acciones que no restarán capacidad al equipo
  - "Hay que ser puntual en el Scrum diario"
- Hay problemas que dependen de otros
  - La acción la hará un miembro del equipo, pero el resultado dependerá de otros
    - "Conseguir que los de administración nos permitan comprar un nuevo servidor de pruebas"
- Habrá problemas que no entendemos lo suficiente como para pensar una acción para solucionarlos
  - Podemos establecer como acción una exploración para comprender mejor el problema

Determinar acciones

- Hay cosas que pueden parecer acciones de mejora, pero que no son muy "accionables"
  - Trabajar en lotes más pequeños
  - Hacer requisitos más fáciles de entender
  - Escribir más tests unitarios
  - Mejorar las estimaciones
  - ...
- Las anteriores son vagas y más buenos deseos que acciones concretas de mejora
- Se deben expresar cosas más concretas, cuantitativas, medibles... que contribuyan a mejorar esos problemas
  - Cada persona tiene que hacer commit del código al Git al menos dos veces al día
  - Cada PBI debe llevar criterios de aceptación definidos estilo BDD/A-TDD
  - Hay que crear al menos un test automático para cada bug que se detecte antes de proceder a arreglarlo
  - Usaremos siempre póquer de planificación al estimar PBI
  - ...

Realizar las acciones

- Las acciones decididas no deben quedarse en la retrospectiva, hay que abordarlas
- No es buena idea tener un plan de mejora separado del trabajo habitual del sprint
  - Lo normal es que este plan de mejora acabe como algo secundario en comparación con el resto del trabajo que siempre será más prioritario
- Para que las acciones se hagan, lo mejor es considerarlas como tareas durante la planificación del siguiente sprint
  - Así restan capacidad al equipo
    - O sea, que estamos reservando tiempo para realizarlas de manera explícita
- Y dedicar algo de tiempo en cada retrospectiva a analizar si lo que se decidió hacer en la anterior se ha hecho, y si ha funcionado o no

Posibles problemas

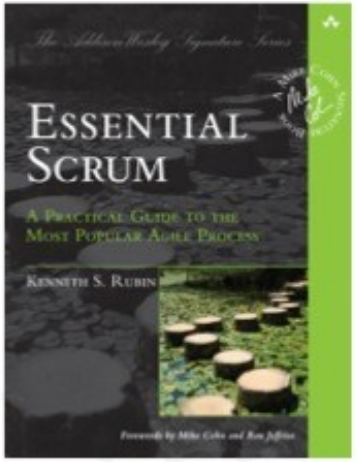
- La retrospectiva es importante para la mejora continua y hay que hacerlo entender a todo el mundo
  - Incluso los equipos experimentados tienen margen para mejorar
- El objetivo es determinar **acciones** de mejora
  - No es una sesión de terapia de grupo para quejarse en voz alta
  - Tampoco tiene que ser una reunión depresiva
    - Se habla de problemas, pero es para determinar cómo se van a resolver
  - Los problemas no se resuelven durante la retrospectiva
    - Se detectan y se determinan acciones de mejora que se llevan a cabo **después**
- Si hay problemas graves que no salen a la luz en las retrospectivas, esos no se van a mejorar sustancialmente
  - El ScrumMaster tiene que detectar esto y ayudar a resolverlo

Bibliografía

- Kenneth S. Rubin. *Essential Scrum. A practical guide to the most popular agile process*
  - Capítulo 4 (*Sprints*), Capítulo 19 (*Sprint Planning*), 20 (*Sprint Execution*), 21 (*Sprint review*), 22 (*Sprint retrospective*)

Visual AGILExicon®

- Slides in this presentation contain items from the Visual AGILExicon®, which is a trademark of Innolution, LLC and Kenneth S. Rubin.
- The Visual AGILExicon is used and described in the book: "***Essential Scrum: A Practical Guide to the Most Popular Agile Process***"
- You can learn more about the Visual AGILExicon and permitted uses at: http://innolution.com/resources/val-home-page



Connect with Innolution:

Facebook.com/InnolutionLLC

Twitter.com/krubinagile

Visual AGILExicon®