

## Burn-down chart

El objetivo de este gráfico facilitar el seguimiento del **trabajo** (esfuerzo) **que queda por realizar** en un periodo de tiempo concreto (iteración). En los equipos de desarrollo ágiles, el gráfico Burn-down se suele **aplicar por sprint (o release)** y ayuda al equipo a la **visualización del esfuerzo realizado** y el que queda por realizar hasta la finalización del sprint (o release).



### UTILIZACIÓN

- En el eje **X se representa el tiempo**, bien en semanas, en días, etc, dependiendo de la duración de la iteración.
- En el eje **Y se representa el esfuerzo** a realizar, siendo lo más extendido el uso de puntos de esfuerzo (también podrían utilizarse horas, días-hombre, etc.).
- Si el gráfico representa una iteración, entonces la Y debería ser los **puntos totales de esfuerzo** para esta iteración, y la X la **duración de la iteración**.
- Se han tomado como referencia para el ejemplo una iteración de 2 semanas (10 días) y una estimación de 45 puntos de esfuerzo a realizar por el equipo.
- Podemos trazar la **línea de esfuerzo ideal** uniendo dos puntos:
  1. Inicio de iteración, total de esfuerzo estimado ( $x = D0$ ,  $y = 45$  en el ejemplo).
  2. Final de iteración, esfuerzo a cero ( $x = D10$ ,  $y = 0$ ).
- El gráfico se construye añadiendo **un punto al final de cada día**, con las coordenadas  $x = \text{día}$  e  $y = \text{puntos de esfuerzo estimados para finalizar}$ . Tras esta acción, se debe unir este punto con el del día anterior trazando una recta.
- Si estamos **por debajo de la línea de esfuerzo ideal**, el avance es **mayor del previsto** (queda menos por hacer), y si estamos **por encima**, el avance es **menor**.

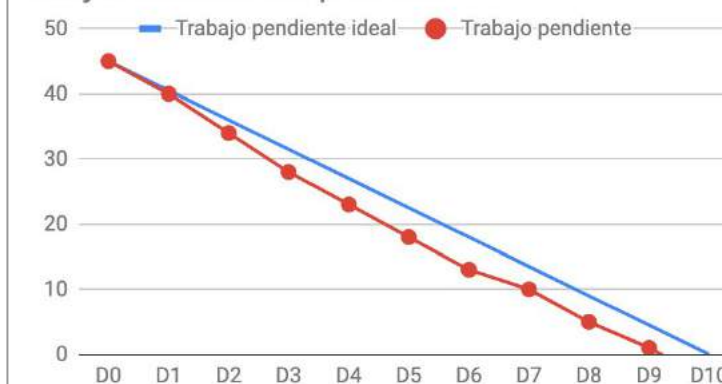
Menor avance del previsto



Avance según lo previsto



Mayor avance del previsto



## Burn-up chart

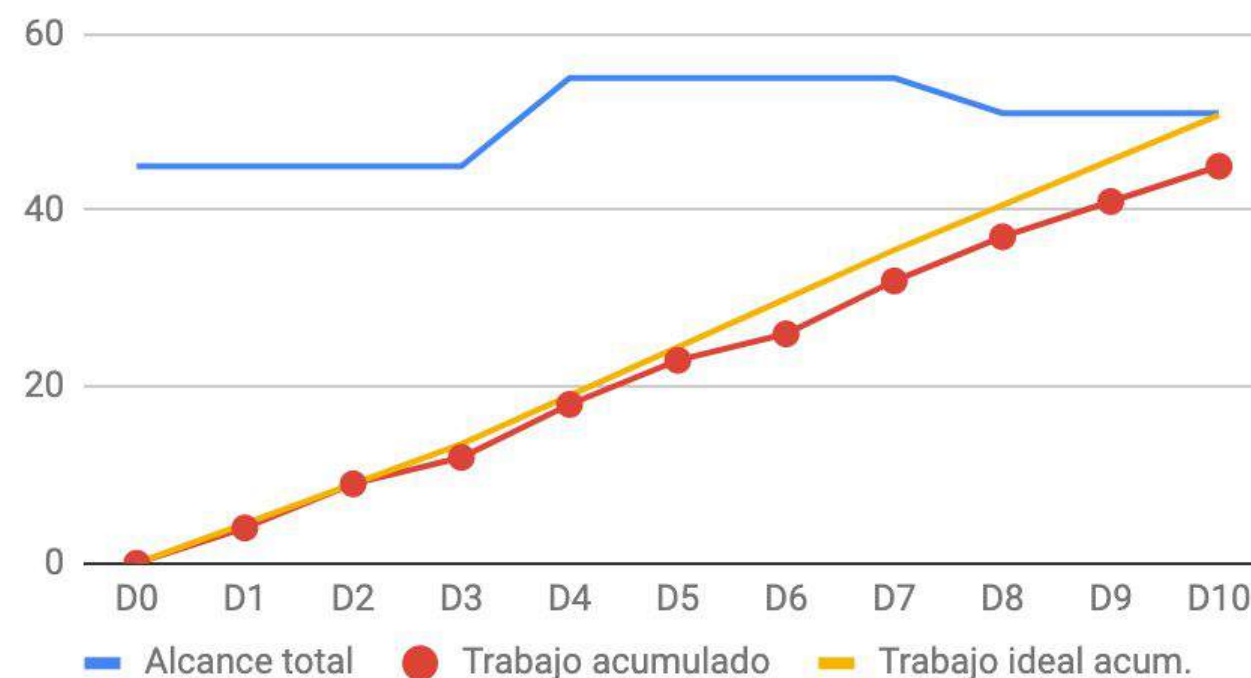
El objetivo de este gráfico es facilitar el seguimiento del **trabajo** (esfuerzo) **realizado** en un periodo de tiempo concreto (iteración). Para ello, muestra por separado el progreso del trabajo realizado y el alcance (trabajo total a realizar). Con ello, permite ver claramente los **cambios de alcance** en la iteración. Al igual que el Burn-down chart, se suele utilizar a nivel de iteración o release.

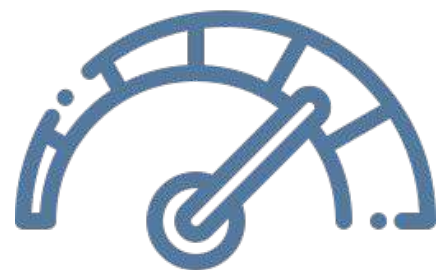


### UTILIZACIÓN

- En el eje **X se representa el tiempo**, bien en semanas, en días, etc, dependiendo de la duración de la iteración.
- En el eje **Y se representa el esfuerzo** a realizar, siendo lo más extendido el uso de puntos de historia (también podrían utilizarse horas, días-hombre, etc.).
- Si el gráfico representa una iteración, se dibuja una línea con **puntos totales de esfuerzo** para esta iteración. Los incrementos y reducciones de alcance se reflejan en la misma.
- Se han tomado como referencia para el ejemplo una iteración de 2 semanas (10 días) y una estimación de 45 puntos de esfuerzo a realizar por el equipo. El alcance el D4 sube a 55 puntos y el D8 baja a 51.
- Podemos trazar la **línea de esfuerzo ideal** uniendo dos puntos:
  1. Inicio de iteración, esfuerzo a cero ( $x = D0, y = 0$  en el ejemplo).
  2. Final de iteración, total de esfuerzo estimado ( $x = D10, y = 51$ ).
- El gráfico se construye añadiendo **un punto al final de cada día**, con las coordenadas  $x = \text{día}$  e  $y = \text{puntos de esfuerzo estimados para finalizar}$ . Tras esta acción, se debe unir este punto con el del día anterior trazando una recta.
- Si estamos **por encima de la línea de esfuerzo ideal**, el avance es **mayor del previsto**, y si estamos **por debajo**, el avance es **menor**.

Burn-up chart con cambios de alcance





## Velocity chart

El objetivo de este gráfico es poder visualizar la **evolución de la velocidad del equipo** a lo largo de múltiples iteraciones. Para ello, muestra (para cada iteración) los puntos de esfuerzo estimados del equipo, así como los realmente alcanzados al finalizar cada iteración.



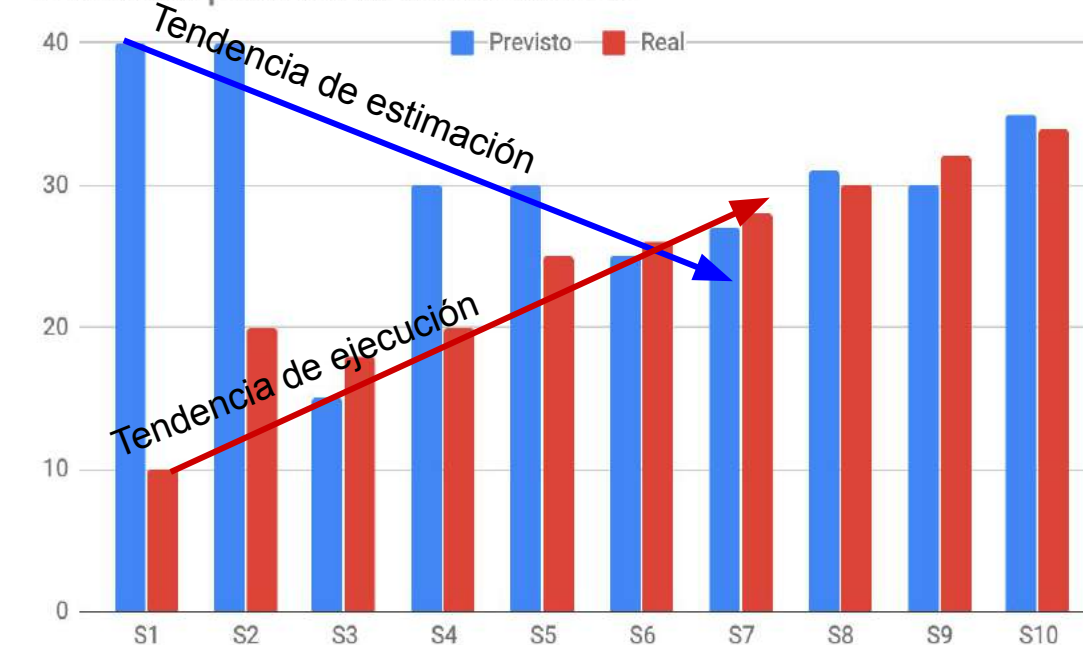
### UTILIZACIÓN



**¡Atención!** La velocidad es una métrica que no se puede utilizar para comparar el rendimiento de equipos diferentes. Se debe limitar a la evolución del propio equipo.

- En el eje **X se representan las iteraciones**.
- En el eje **Y se representa el esfuerzo**, siendo lo más extendido el uso de puntos de esfuerzo (también podrían utilizarse horas, días-hombre, etc.).
- Se ha tomado para el ejemplo un equipo que lleva realizados 10 sprints o iteraciones. El esfuerzo lo reflejan en puntos de esfuerzo.
- Para crear el diagrama son necesarios dos datos **por cada iteración**:
  1. **Estimación de esfuerzo total** a realizar por el equipo, obtenido al planificar el trabajo al comienzo de la iteración (en la Sprint Planning para un equipo Scrum).
  2. **Esfuerzo total del trabajo realizado** durante la iteración, obtenido al final de la misma (en la Sprint Review para un equipo Scrum).
- El gráfico puede reflejar grandes **divergencias entre estimado y realizado**, siendo más acentuadas en las primeras iteraciones. Esto sucede especialmente en equipos nuevos con inmadurez tecnológica o metodológica.
- Según avanzan las iteraciones, las divergencias deberían reducirse.
- Un **equipo maduro en mejora continua** se quedará unas veces por encima y otras por debajo de la estimación, pero siempre cerca de la misma. Además **mostrará una tendencia creciente** hasta llegar a su **umbral de eficiencia**. Así sabremos que ni ha caído en la zona de confort ni está saturado de trabajo.

Velocidad prevista vs velocidad real



## Throughput chart

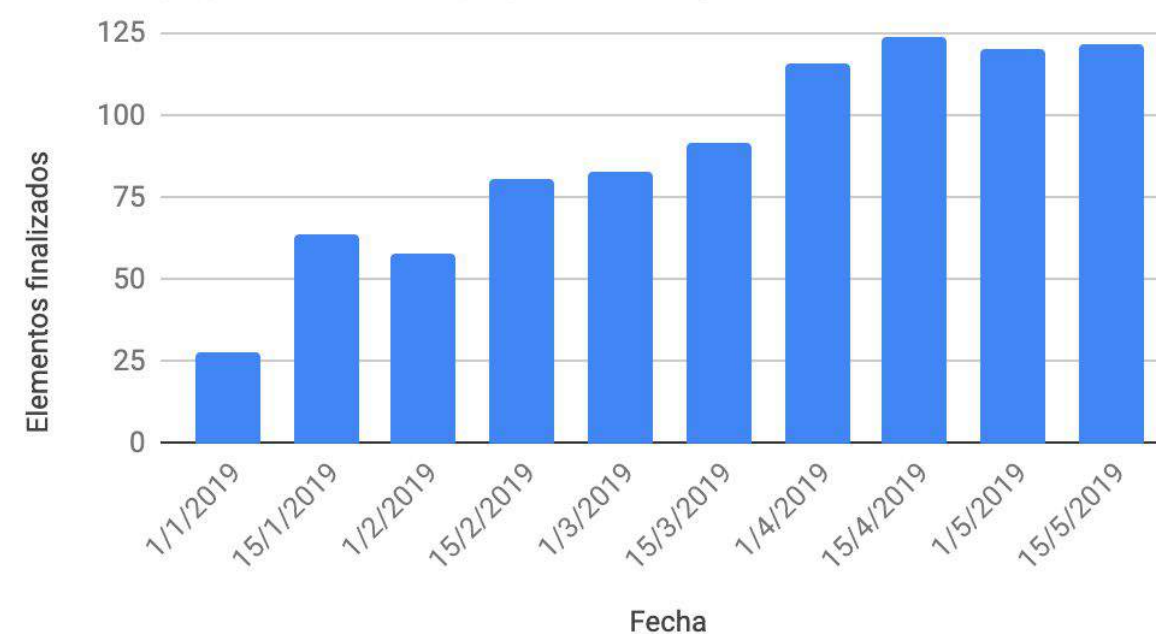
El **throughput** indica la cantidad de **elementos de trabajo realizados** (no puntos de esfuerzo) en un periodo de tiempo concreto. Este gráfico muestra su **evolución** con la cadencia que nos interese (cada día, semana, mes, ...). Esto permite a un equipo comprometido con la mejora continua visualizar sus avances en términos de productividad y predecir las fechas de finalización de sus elementos de trabajo.



### UTILIZACIÓN

- En el eje **X se representa el tiempo**, bien en semanas, en días, etc., dependiendo de la cadencia que nos interese.
- En el eje **Y se representan los elementos de trabajo realizados**.
- Se han tomado como referencia para el ejemplo un equipo que contabiliza el trabajo entregado cada 15 días.
- Para crear este gráfico, se deben hacer dos cosas:
  1. **Determinar la cadencia** que queremos usar.
  2. **Contabilizar** los elementos de trabajo **finalizados** en dicho periodo temporal.
- Se puede hacer una **previsión de fechas de finalización** de un determinado trabajo. Para ello se debe:
  1. Dividir el trabajo en n elementos de trabajo homogéneos.
  2. Determinar el throughput medio, una vez el equipo sea estable.
  3. Calcular el tiempo que llevará completar el trabajo aplicando la **Ley de Little** (tiempo = n° elementos de trabajo / throughput).
  4. Sumar el tiempo obtenido a la fecha de inicio del trabajo.
- Para que el throughput sea una métrica consistente y precisa, es necesario que el **tamaño medio** de los elementos de trabajo sea **homogéneo**.
- Aunque el throughput debería mostrar la mejora del equipo, tiene un **umbral de eficiencia**. En el ejemplo gráfico, el equipo se estabiliza con un throughput de 120 elementos de trabajo realizados cada 15 días.

Throughput de un equipo en mejora continua





## Cumulative Flow Diagram (CFD)

El diagrama de flujo acumulado o **CFD** muestra la **evolución en el tiempo del flujo de trabajo** de un equipo. Para ello, refleja la **cantidad de trabajo en curso (WIP) en cada estado**, acumulando o apilando estas cantidades gráficamente, en periodos regulares.



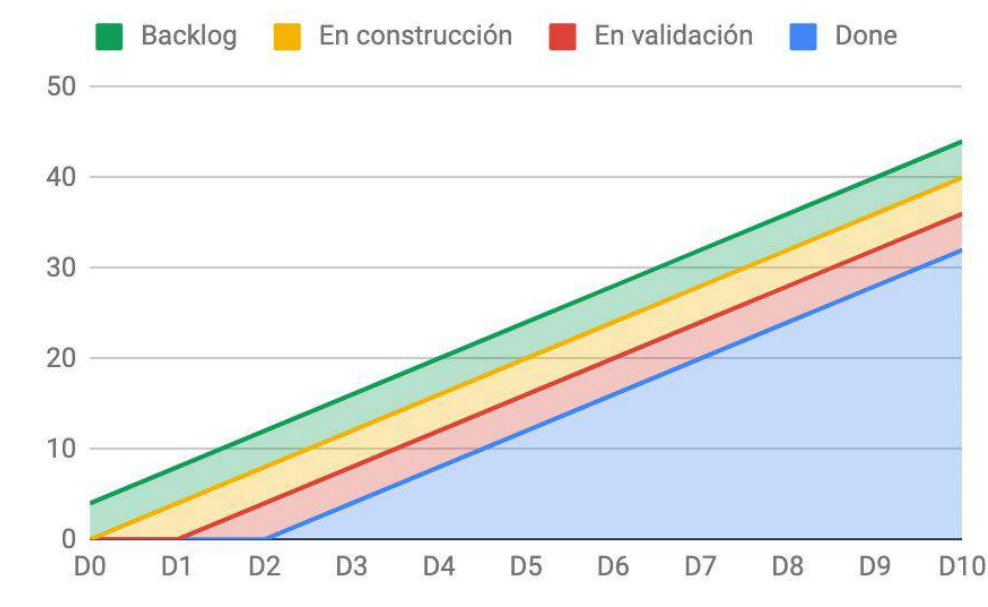
### UTILIZACIÓN

- En el eje **X se representa el tiempo**, bien en semanas, en días, etc., siendo lo habitual hacerlo en días.
- En el eje **Y se representa el número de elementos de trabajo en cada estado** del flujo de trabajo del equipo. Estos se apilan en orden inverso respecto al flujo. Alternativamente, se podrían reflejar puntos de esfuerzo.
- Para construir este gráfico es necesario disponer de la cantidad de elementos de trabajo que hay en cada estado cada día:
  1. Se comienza por el último estado del flujo (Done) y se marca el número de elementos.
  2. Se toma el anterior estado del flujo (En validación) y se marca el número de elementos apilándolo.
  3. Se continua realizando el paso 2 (En construcción) hasta llegar al estado inicial del flujo (Backlog).
 Finalmente, se unen los puntos de cada estado y se colorean las distintas áreas resultantes.
- Cuando las bandas del CFD varían suavemente y su altura es estable, es síntoma de un buen equilibrio entre demanda y capacidad.
- Aunque este gráfico es ampliamente utilizado con **Kanban**, también es una herramienta potente para otros frameworks ágiles.

CFD con 3 estados



CFD con 4 estados



## Métricas en un CFD

Un CDF muestra una **gran cantidad de información** de manera muy compacta: el **burn-up chart**, **lead time**, **cycle time**, el trabajo en curso (**WIP**) e incluso **cuellos de botella**.

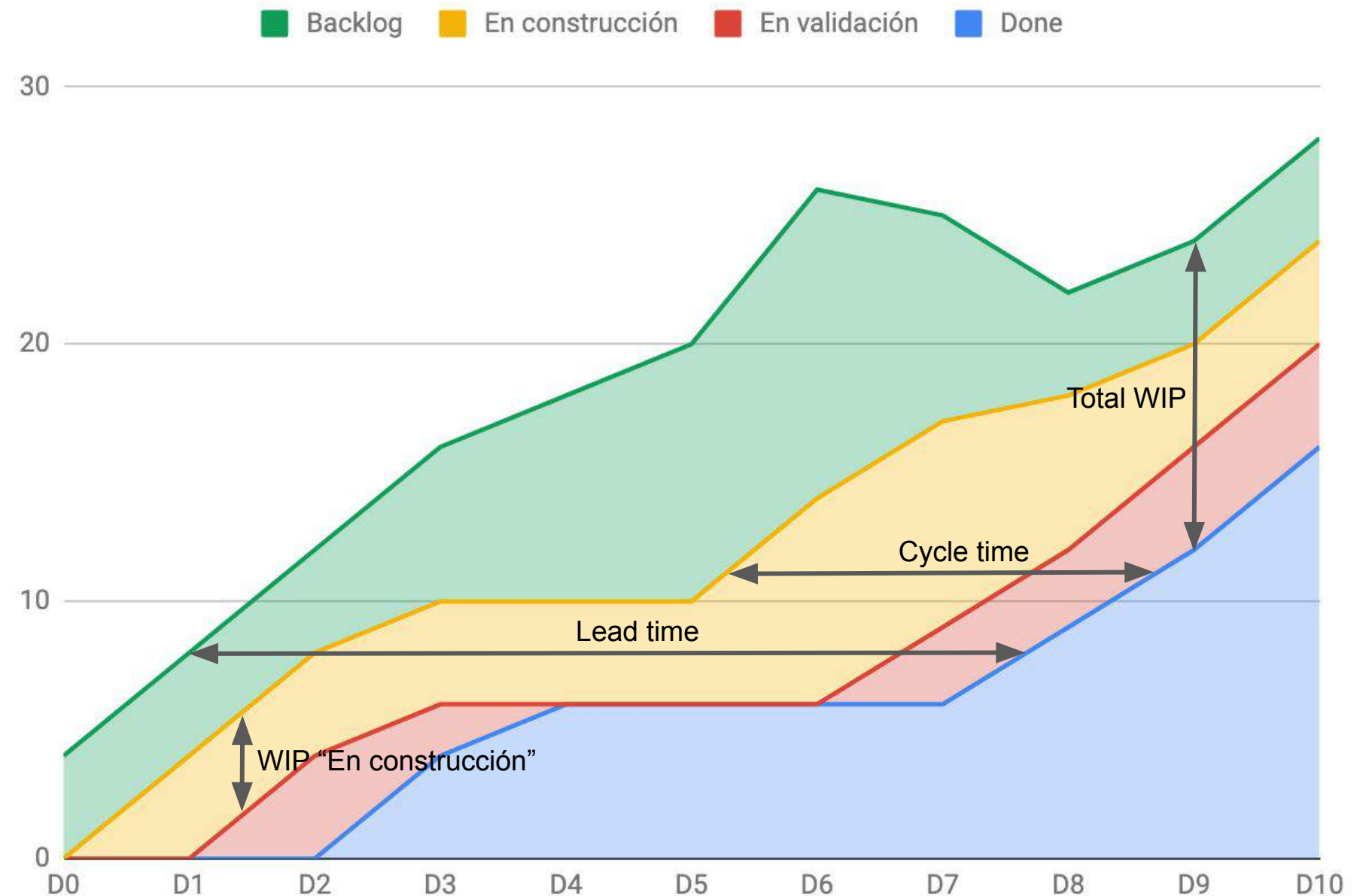
### Definición de las métricas:

- **Cycle time:** es el tiempo que pasa desde que se inicia un trabajo hasta que se entrega. Para el ejemplo: desde que entra en el estado "En construcción" hasta que llega a "Done".
- **Lead time:** es el tiempo que pasa desde que se solicita un trabajo (entra en el Backlog) hasta que se entrega (Done).
- **WIP (Work in Progress):** es la cantidad de elementos en progreso para uno o varios estados del flujo de trabajo.
- **Burn-up:** la evolución del último estado del flujo (Done) muestra cómo se va completando el trabajo, al igual que en un burn-up chart.

## ¿Por qué el Total WIP incluye el Backlog?

Para entenderlo hay que ver el sistema **desde fuera**, como una caja negra. Entran elementos (Backlog), se procesan y se entregan (Done). Todo lo que no se ha entregado está en proceso (Total WIP).

## Métricas en un CFD



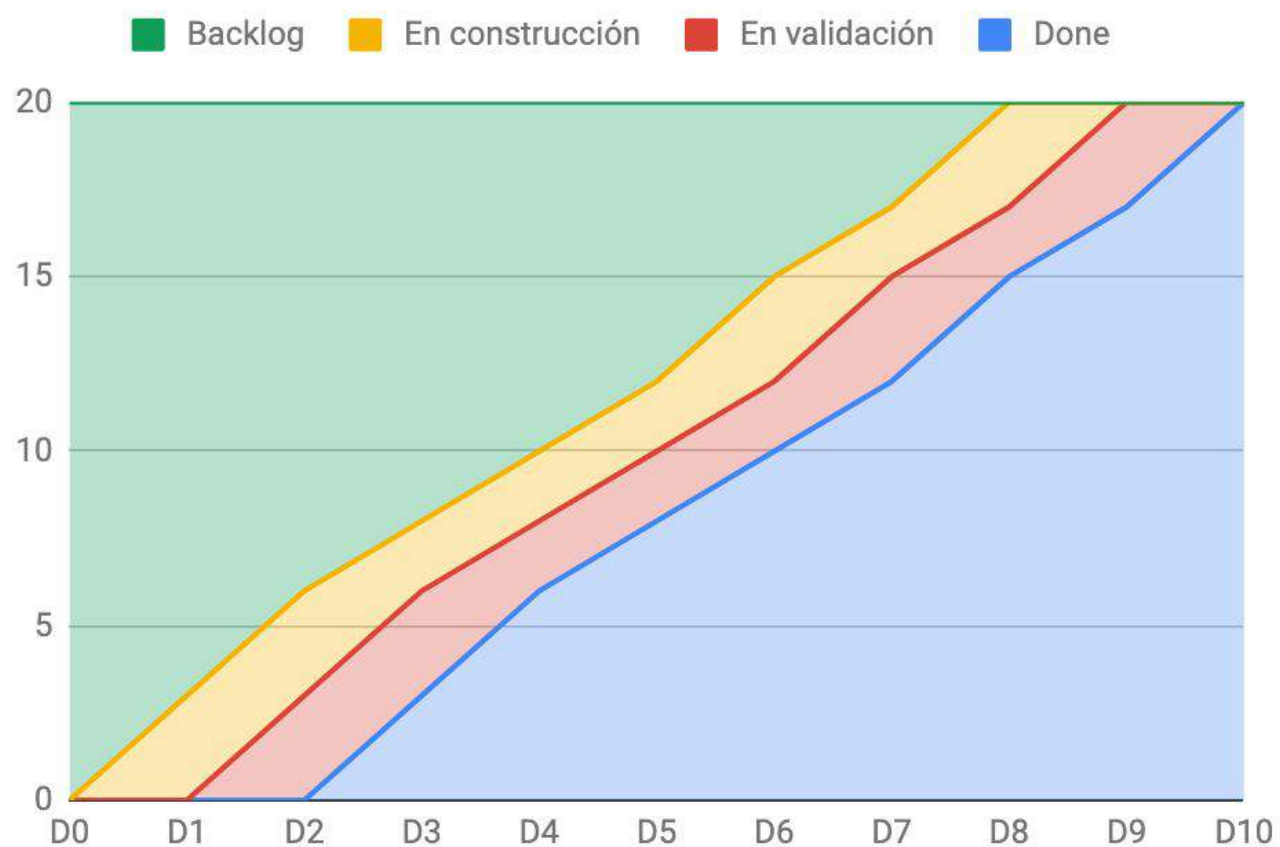
## Entendiendo los "cortes" en un CFD

- Los **cortes verticales** muestran la **cantidad de elementos de trabajo** de cada estado para un momento dado.
- Los **cortes horizontales** muestran los **tiempos medios** que los elementos de trabajo pasan en cada estado.

## EJEMPLO - Sprint Scrum

- Un CFD puede reflejar perfectamente la evolución de los elementos de trabajo en Scrum.
- En el ejemplo se refleja un Sprint, lo cual tiene algunas particularidades:
  - El alcance, se limita al Sprint Backlog, confeccionado en la sesión de Sprint Planning.
  - El eje X (tiempo) se limita a la duración del Sprint (10 días).
  - Al partir de un alcance dado (20), el Backlog WIP comienza en su máximo y se reduce conforme pasa el tiempo.

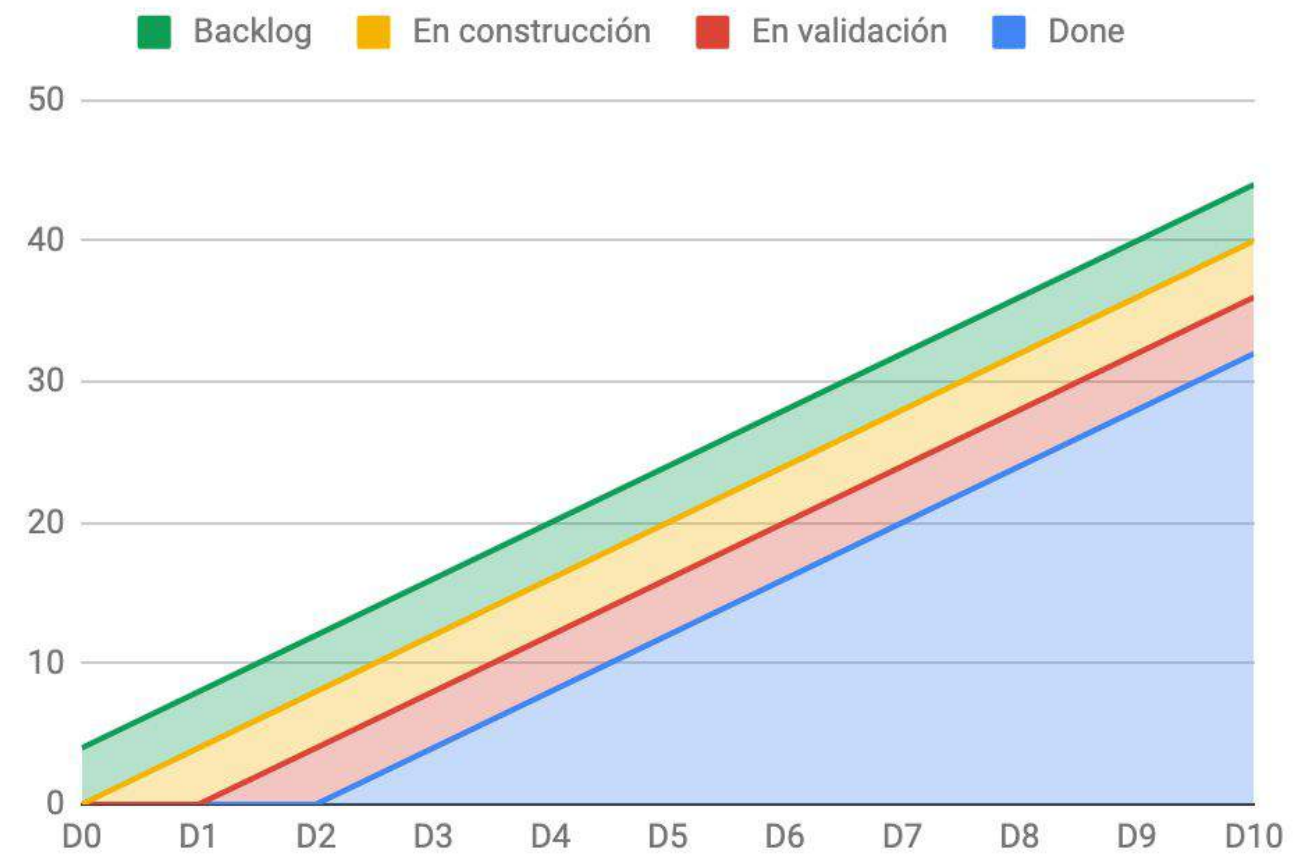
CFD de un Sprint Scrum (alcance congelado)



## EJEMPLO - Flujo continuo equilibrado

- Un CFD habitualmente muestra un flujo continuo.
- La demanda se refleja con el primer estado del flujo de trabajo (en el ejemplo, el Backlog).
- La capacidad global se refleja con el último estado (Done).
- Cuando las pendientes de las distintas rectas son iguales (son paralelas), significa que demanda y capacidad están alineadas.
- En el ejemplo se ve una situación ideal.

Demanda alineada con la capacidad

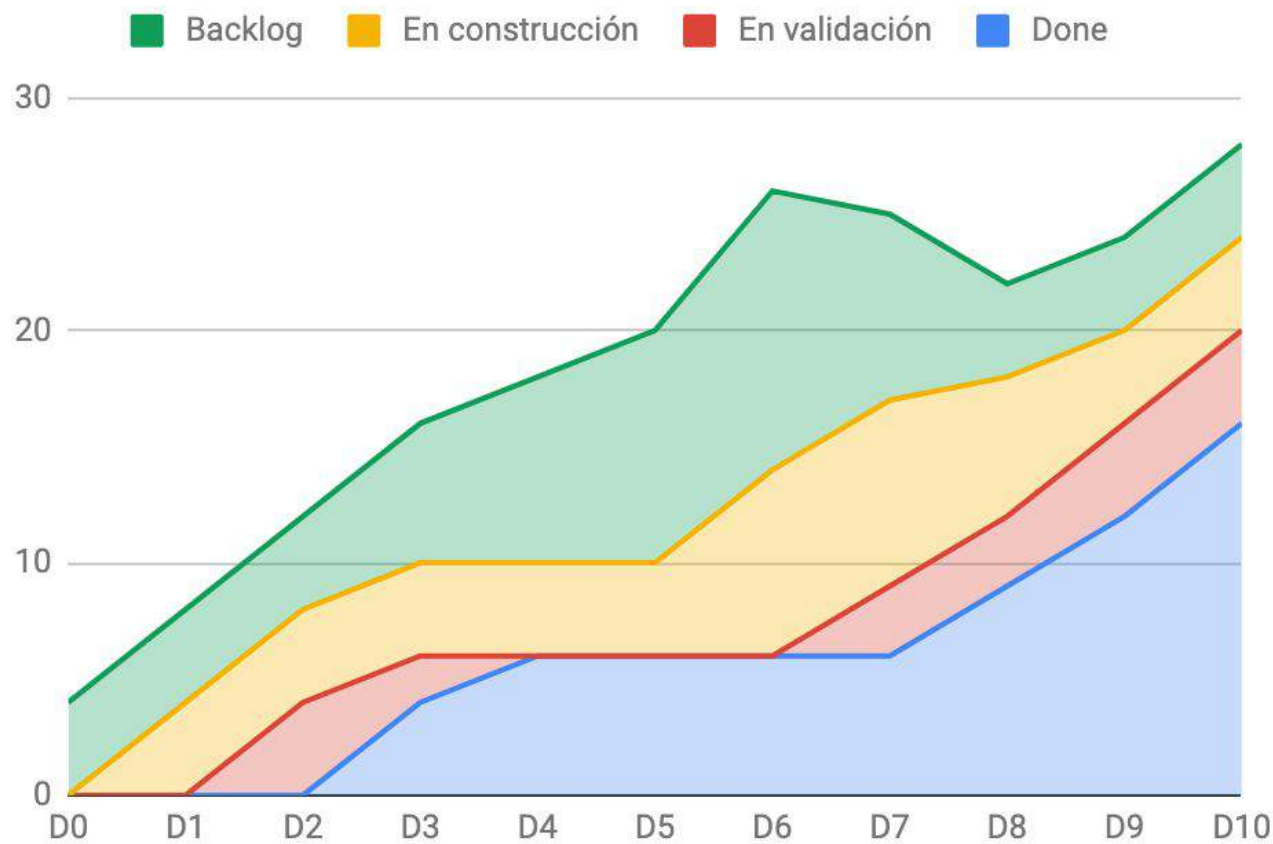




## EJEMPLO - Cuello de botella

- Hasta el día 2 todo avanza equilibradamente.
- A partir de ahí algo se atasca en el estado "En construcción". El tamaño del backlog comienza a crecer y en el estado "En validación" se van quedando sin trabajo.
- La situación se reconduce poniendo el foco en el estado donde hay problemas (se aumenta su capacidad), hasta que vuelven a equilibrarse capacidad y demanda.

CFD cuello de botella



## EJEMPLO - Equipo saturado

- Este CFD muestra como la demanda (representada en el backlog) crece a ritmo superior a la capacidad disponible.
- Se puede apreciar que el backlog va creciendo de tamaño, a pesar de que el equipo de trabajo tiene varios puntos de sobreesfuerzo para tratar de controlar la demanda.
- Si esta situación se mantiene, el equipo se saturará y probablemente baje la calidad de las entregas.

CFD mayor demanda que capacidad

