

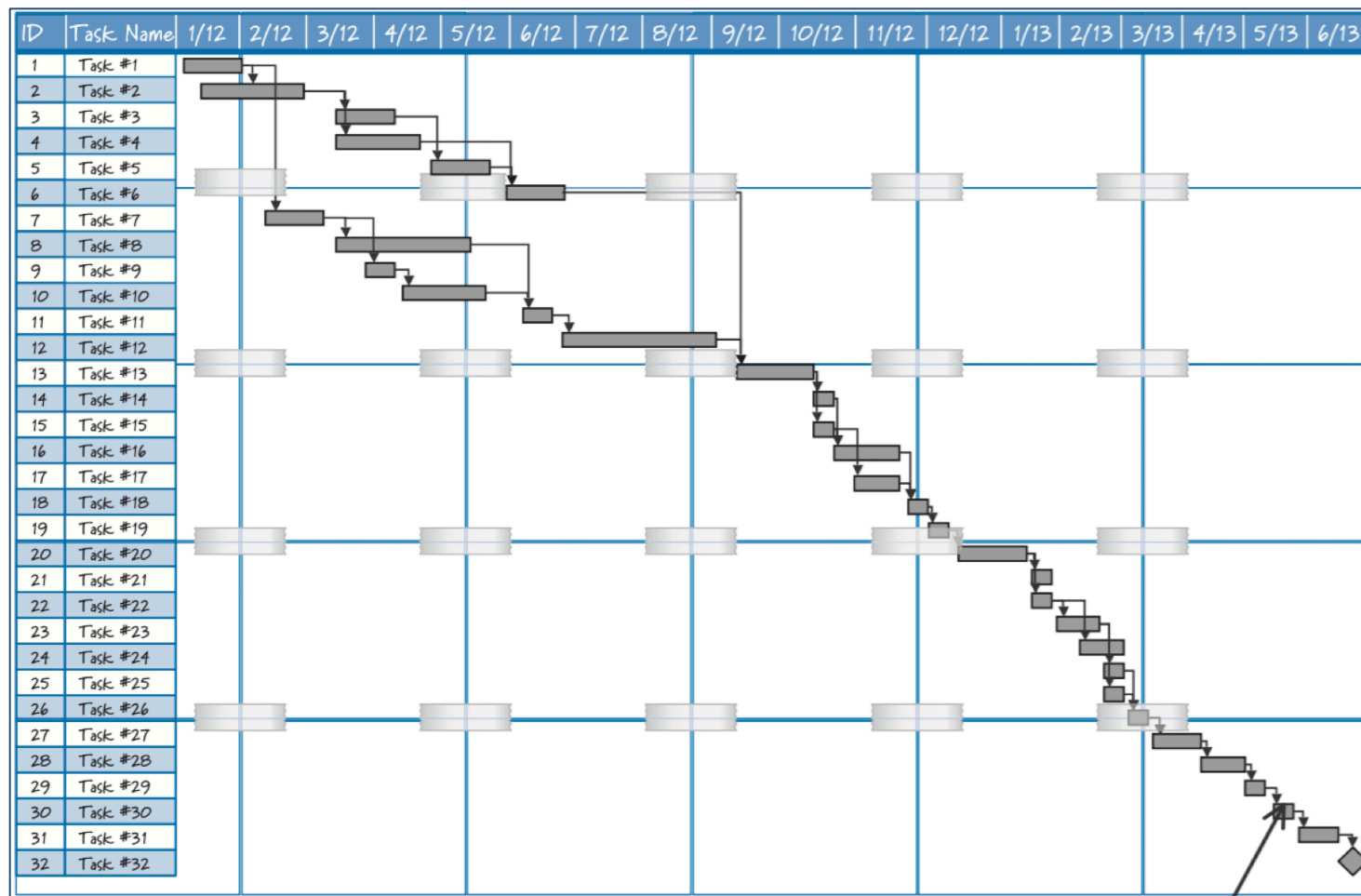
# Planificación:

Un mapa para lograr nuestro propósito

Yadran Eterovic S. ( [yadran@uc.cl](mailto:yadran@uc.cl) )

## La Carta Gantt

- Se requiere mucho trabajo para construirla
- Se requiere bastante trabajo para modificarla constantemente
- Rara vez es posible mantenerse dentro del plan inicial
- Puede insistirse en mantener un plan que está fuera de la realidad





En desarrollos ágiles la planificación es tan vital como en desarrollos tipo cascada

... pero los compromisos no se toman desde un inicio

... sino que se dejan para el último momento (en que aún es) responsable (hacerlo)

... y se va planificando iterativa e incrementalmente, cada vez en mayor detalle a medida que el horizonte de tiempo se acorta:

... desde la visión y el *roadmap* del producto

... hasta las iteraciones y las tareas individuales

En la práctica, se han propuesto, y se usan, muchos (modelos de) procesos ágiles de desarrollo

En lo que sigue y cuando ayude a fijar mejor las ideas, usaremos el vocabulario y las prácticas de *Scrum*

La planificación, ágil o no, comienza con la definición de la *visión del producto*

—incluyendo posibles funcionalidades de alto nivel, riesgos potenciales y dependencias—

... a partir de la cual el *Product Owner* construye el *backlog de alto nivel del producto* y un *roadmap aproximado para el producto*

# La visión del producto

7

“Any progress starts with a mission, a vision.” : [Arnold Schwarzenegger](#).

Define a los clientes

... sus problemas u oportunidades

... y la experiencia y beneficios que se prevén para los clientes

Tiene que ser atractivo tanto intelectual como emocionalmente

Debe proporcionar claridad sobre el estado futuro del producto

... y al mismo tiempo dar suficiente libertad para adaptarse según la retroalimentación

Nada importa más que entregar la visión

Vender más hardware empaquetando mejor software

Atraer clientes más grandes escalando más eficazmente

Vender más servicios *cloud* ofreciendo tecnología de aprendizaje de máquinas

“8 Great Product Vision Examples,” por Janna Bastow  
<https://www.prodpad.com/blog/product-vision-examples/>

# Roadmap del producto

La entrega del producto va a ocurrir a lo largo del tiempo, p.ej, un año

... en un cierto número de *liberaciones* ( o *releases* ), p.ej., 4

El *roadmap* especifica, y permite visualizar, para cada liberación,

- el objetivo de la misma
- las capacidades (o *features*) requeridas
- ... y los criterios de éxito



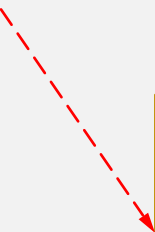
# Roadmap del producto

La entrega del producto va a ocurrir a lo largo del tiempo, p.ej, un año

... en un cierto número de *liberaciones* ( o *releases* ), p.ej., 4

El *roadmap* especifica, y permite visualizar, para cada liberación,

- el objetivo de la misma
- las capacidades (o *features*) requeridas
- ... y los criterios de éxito



Se planifica a partir del MVP —típicamente, la primera liberación— de manera que cada nueva liberación agregue incrementalmente capacidades hacia el producto completo de la Visión

# Roadmap del producto

La entrega del producto va a ocurrir a lo largo del tiempo, p.ej, un año

... en un cierto número de *liberaciones* ( o *releases* ), p.ej., 4

El *roadmap* especifica, y permite visualizar, para cada liberación,

- el objetivo de la misma
- las capacidades (o *features*) requeridas
- ... y los criterios de éxito

El *Product Backlog* de alto nivel especifica capacidades como

- “Ordenar online”, “Sugerir combo”, “Aplicar descuentos”, “Arma tu propio <producto>”

... o también

- “Mejorar la capacidad de ...”, “Integrarse con la plataforma de ...”, “Residencia de datos”

Jira Your work Projects Filters Dashboards People Apps Create

Search

Appreciation app Discovery project

Views: All ideas, Impact assessment, Roadmap, Impact vs Effort, Timeline roadmap, Create a view, Archive, Project settings, Give feedback

Roadmap

Column Roadmap Group by Filter Sort Fields 4

Find an idea in this view

Now 3

- Improve discoverability of configuration options  
Priority score 100  
Goal Increase CSAT  
Delivery 25% Done
- Sharing to non-technical users  
Priority score 90  
Goal Increase CSAT  
Delivery 13% Done
- Improve performance of sharing feature  
Priority score 78  
Goal Increase CSAT  
Delivery 10% Done

Next 3

- New integration with video sharing platforms  
Priority score 60  
Goal Differentiate
- Dashboards for non-technical users  
Priority score 55  
Goal Increase CSAT
- Better search  
Priority score 45  
Goal Increase CSAT

Later 3

- Improve evaluation and conversions  
Priority score 34  
Goal Improve conversion
- Video capabilities  
Priority score 27  
Goal Differentiate
- Enterprise data residency  
Priority score 8  
Goal Improve conversion
- Platform improvements  
Priority score 7  
Goal Improve conversion

No value

- First-time ex  
Priority score 6
- Chat-bot su  
Priority score 6  
Goal

+ Add idea

“Product Roadmap Guide: What is it & How to Create One,” por Bree Davies

<https://www.atlassian.com/agile/product-management/product-roadmaps#:~:text=What%20is%20a%20product%20roadmap,how%20they%20will%20be%20achieved.>

En lugar de hacer una liberación del producto al mercado (o una entrega del producto al cliente) sólo cuando el producto lo haga todo

... —común en los desarrollos tipo cascada, pero que aumenta los costos y, normalmente, los riesgos de que no satisfaga las necesidades de los usuarios—

... en desarrollos ágiles se privilegia entregar algo temprano a los clientes, para que ellos lo puedan usar y así poder dar *feedback*

El producto viable mínimo (MVP):

... *una forma de validar las ideas relativas al producto*

En lugar de hacer una liberación del producto al mercado (o una entrega del producto al cliente) sólo cuando el producto lo haga todo

... —común en los desarrollos tipo cascada, pero que aumenta los costos y, normalmente, los riesgos de que no satisfaga las necesidades de los usuarios—

... en desarrollos ágiles se privilegia entregar algo temprano a los clientes, para que ellos lo puedan usar y así poder dar *feedback*

El producto viable mínimo (MVP):

... *una forma de validar las ideas relativas al producto*

Para identificar un MVP, hay que:

- primero, elegir a los usuarios iniciales
- luego, definir cuál es el trabajo crítico que tienen que hacer y cuál es la mínima funcionalidad con la cual se puede lograr
- finalmente, especificar una manera de validar si el MVP fue exitoso

## What are Examples of the Minimum Viable Product?

If you're wondering what this would look like in practice, let's review how a couple of familiar brands launched successful MVPs.

### Airbnb

With no money to build a business, the founders of Airbnb used their own apartment to validate their idea to create a market offering short-term, peer-to-peer rental housing online. They created a minimalist website, published photos and other details about their property, and found several paying guests almost immediately.

### Foursquare

The location-based social network Foursquare started as just a one-feature MVP, offering only check-ins and gamification rewards. The Foursquare development team began adding recommendations, city guides, and other features until they had validated the idea with an eager and growing user base.

[“Minimum Viable Product \(MVP\),” by ProductPlan](https://www.productplan.com/glossary/minimum-viable-product/)

<https://www.productplan.com/glossary/minimum-viable-product/>

# Backlog del producto

La lista priorizada de requisitos

... tanto funcionales

... como de mejoras

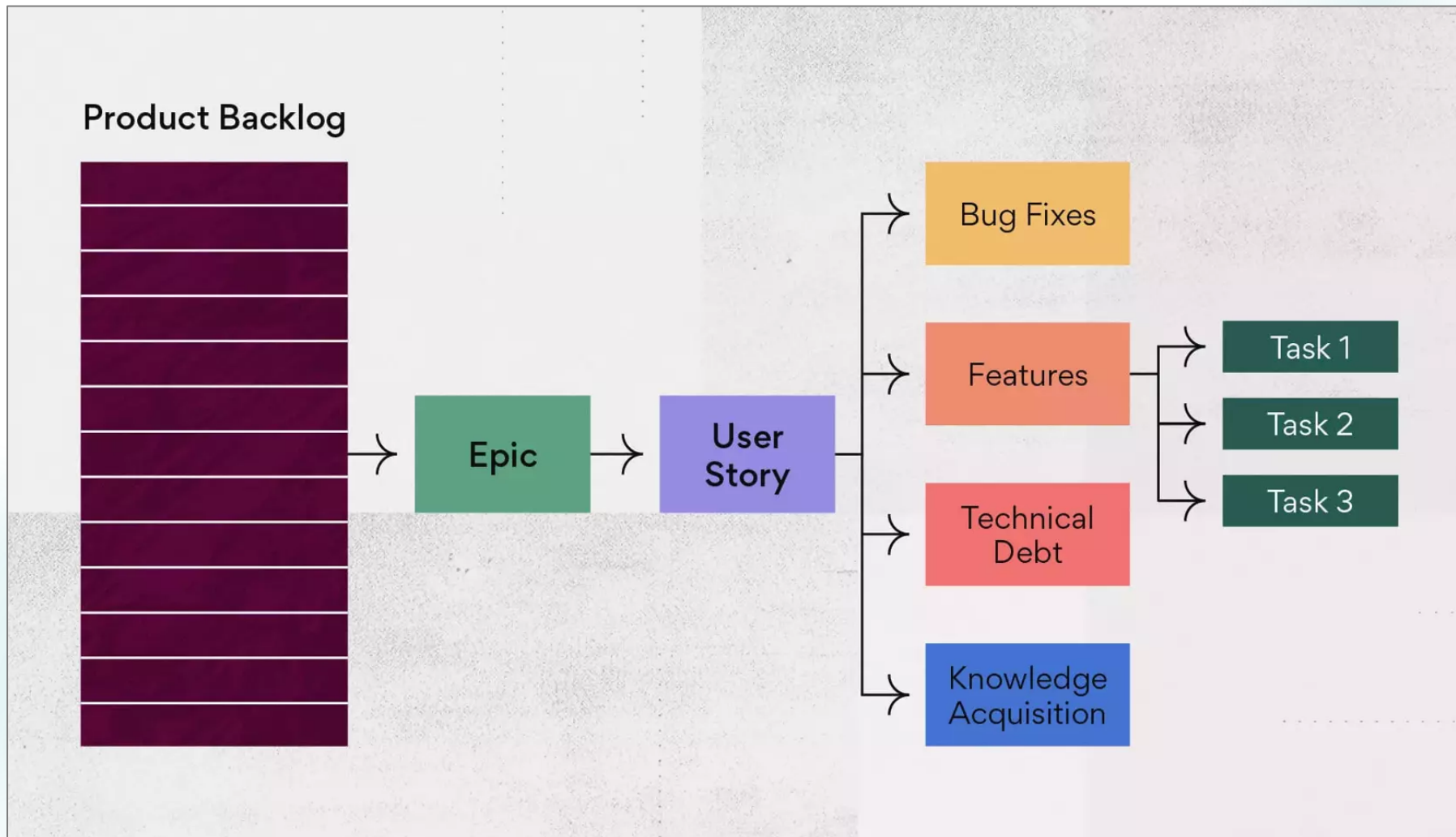
... y de correcciones

... del producto

Es la única fuente de tareas para el equipo

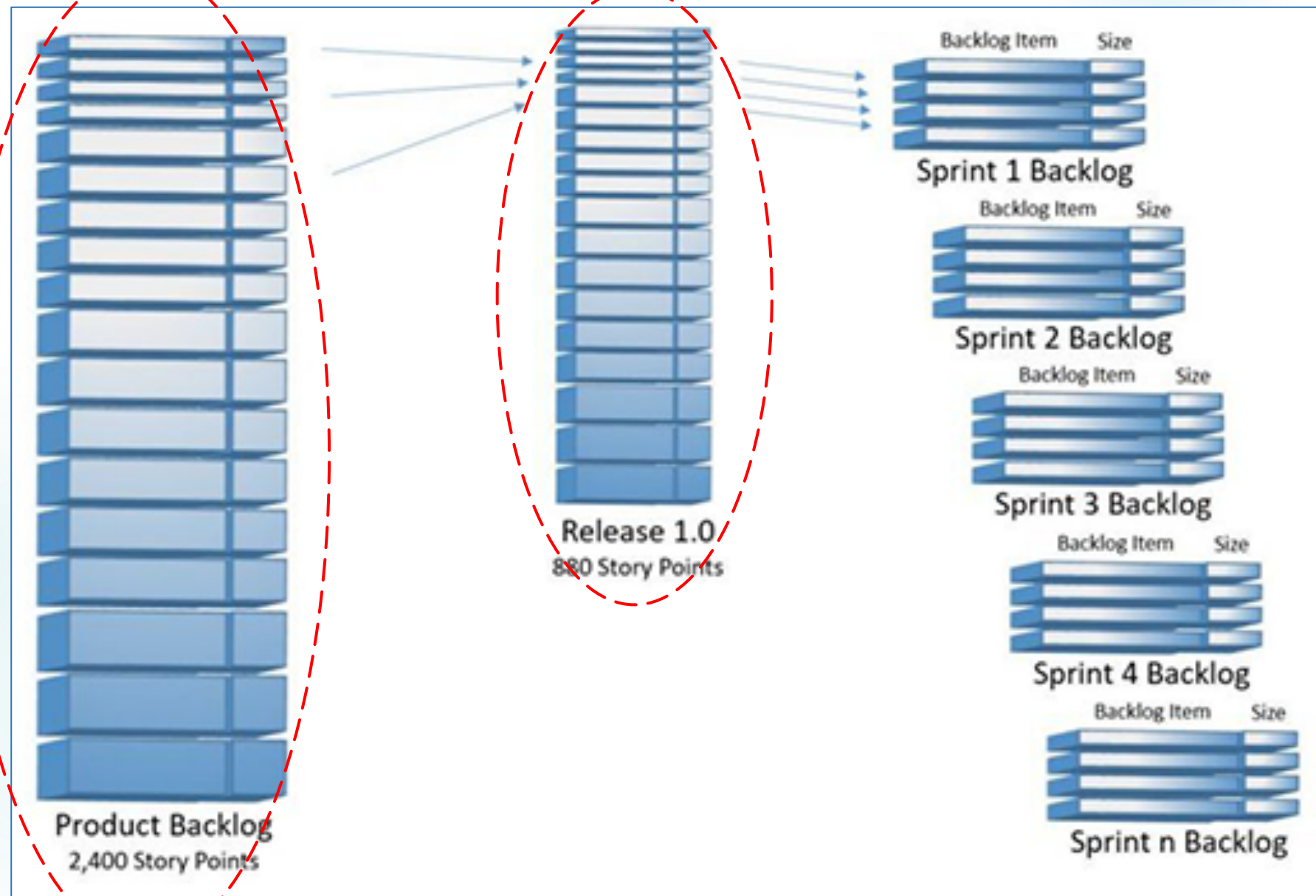
La entrega del producto va a ocurrir a lo largo del tiempo

... en un cierto número de *liberaciones* ( o *releases* )



“What is a product backlog? (And how to create one),”  
por Alicia Raeburn  
<https://asana.com/resources/product-backlog>





Una de las responsabilidades clave del *Product Owner*:

... asegurarse de que el backlog del producto esté al día para poder hacer la planificación de las iteraciones:

- identificar cuáles ítemes del backlog son los mejores candidatos para la próxima iteración

Esta priorización toma en cuenta varios criterios:

- el valor para el negocio —aumento de ventas, reducción de costos, mejora de la experiencia del usuario, etc.
- la validación de hipótesis
- reducir la deuda técnica y los riesgos

Para evitar que esta priorización se convierta sólo en la opinión de la persona que gana más (o que grita más fuerte), conviene adoptar un modelo más objetivo

p.ej., hacer primero el trabajo más corto o el que entrega el mayor valor

... o hacer primero el trabajo que tenga el mayor cociente entre valor y tiempo de desarrollo

Normalmente, la priorización inicial está enfocada en lograr el MVP

En desarrollos ágiles, es fundamental decidir la duración de las iteraciones

Si bien ésta puede depender de las funcionalidades específicas (ítemes del backlog) que se pretende implementar

... muchos procesos privilegian iteraciones de una misma duración a lo largo de todo el proyecto

P.ej., iteraciones de una semana, en el caso de productos nuevos en que hay pocas dependencias

... o de cuatro semanas, en el caso de productos ya establecidos en que las integraciones son complejas

La duración más común para una iteración es dos semanas:

- esta consistencia en la duración de las iteraciones permite al equipo establecer un ritmo de trabajo

... y pronosticar cuántos *story points* (ver más adelante) pueden completar en una iteración, llamado la velocidad del equipo

Al inicio de una iteración

... —durante la planificación de la iteración—

... el equipo, en conjunto con el Product Owner, selecciona un subconjunto de los ítemes del backlog del producto, tal que todos están de acuerdo en que puede completarse dentro de la iteración

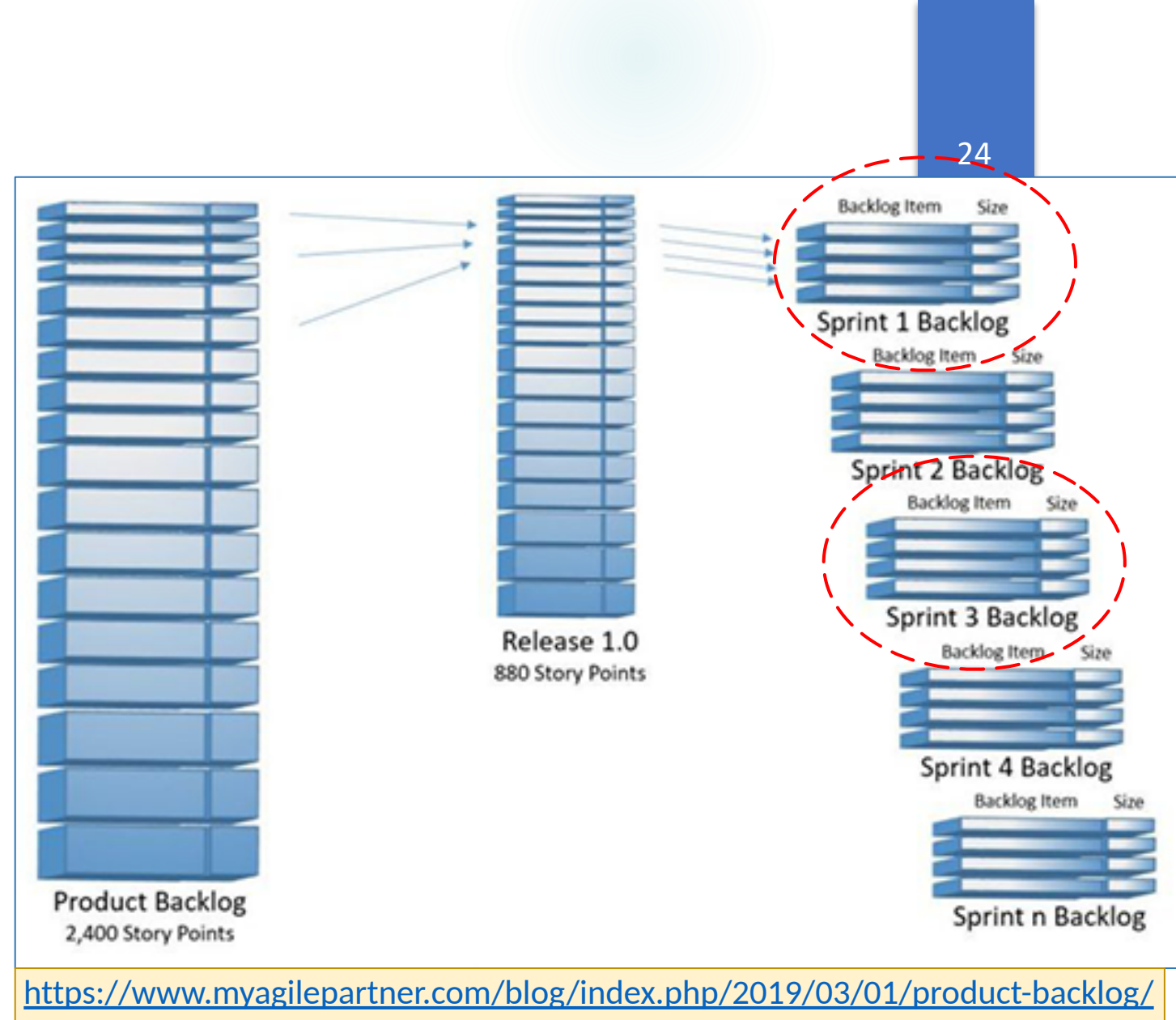
Este subconjunto de ítemes pasa a ser el backlog de la iteración

Este subconjunto de ítemes pasa a ser

... el backlog de la iteración

Para cada ítem del backlog del producto:

- el Product Owner lo presenta
- el equipo hace preguntas para entender bien qué es lo que habría que hacer y le asigna una cantidad de *story points* según complejidad, riesgo y esfuerzo necesario, y
- si el equipo está acuerdo en que se puede completar en una iteración, entonces lo agrega al backlog de la iteración





Esto se repite hasta que la suma de los *story points* en este backlog supere apenas la *velocidad* del equipo

Los *story points* son una forma de cuantificar la magnitud de un relato (o *historia*) de usuario

Los valores que se usan normalmente son 1, 2, 3, 5, 8, 13, 20, 40 y 100

... y un procedimiento habitual para hacer la asignación es *planning poker*:

- cada integrante del equipo de desarrollo hace su propia estimación de la magnitud, y la explica al resto del equipo;
- luego de conocer las estimaciones y escuchar las explicaciones de los otros integrantes, vuelve a hacer su propia estimación;
- este procedimiento se repite hasta que las estimaciones converjan

Todos los que van a hacer la estimación —principalmente, los desarrolladores— se sientan alrededor de una mesa

Al centro de la mesa se coloca un relato de usuario

Cada persona recibe una baraja de 14 naipes:

... hace individualmente su estimación para el relato de usuario

... y pone el naipe correspondiente cara abajo sobre la mesa

Finalmente, todos dan vuelta sus naipes al mismo tiempo



Ya está hecho

Cada naipe representa una  
estimación relativa del esfuerzo

*“No tengo suficiente informa-  
ción para hacer la estimación”*

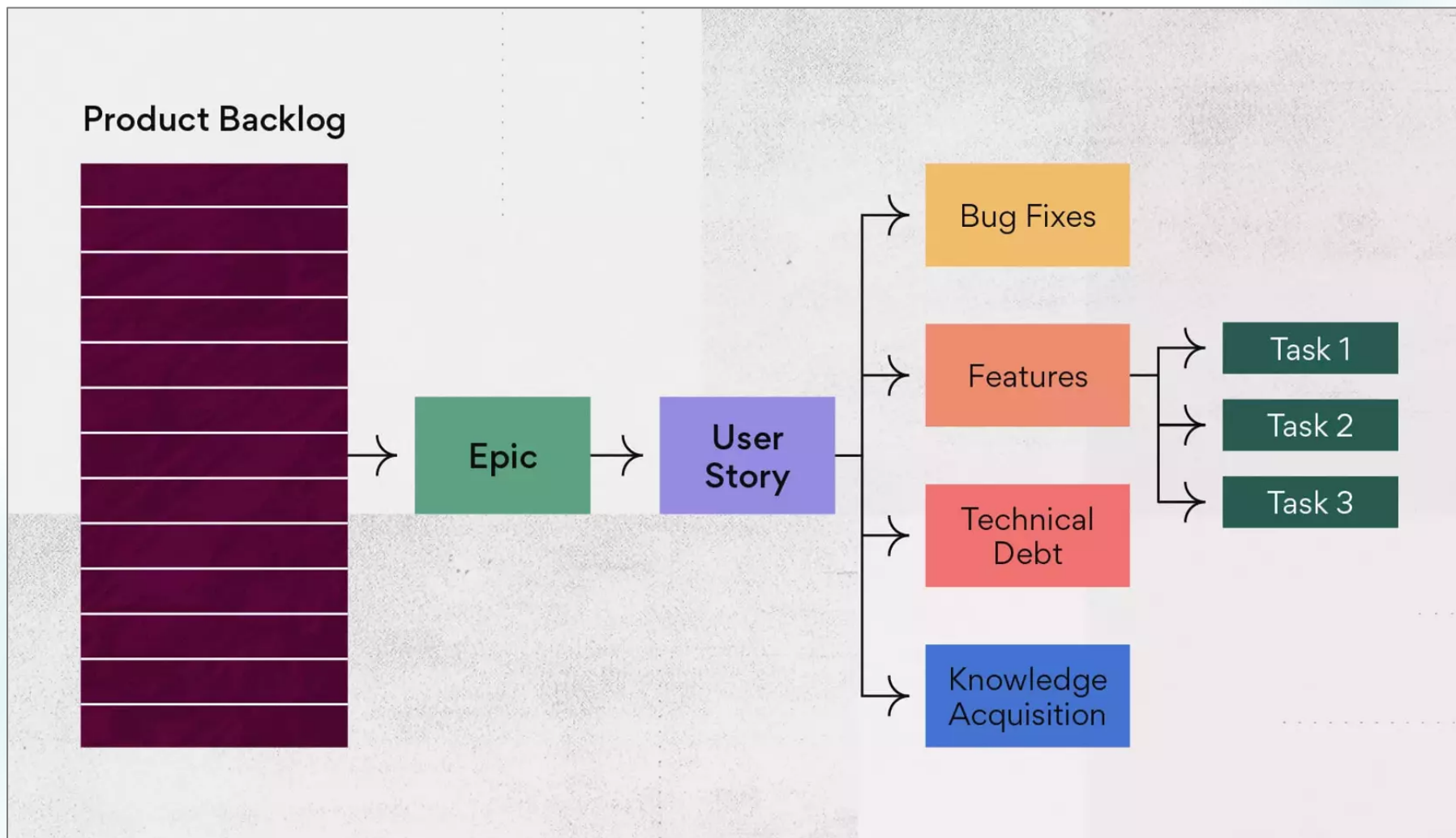
*“Necesito un  
descanso”*

Finalmente, es necesario planificar detalladamente el trabajo que hay que hacer para poder entregar al final de la iteración los relatos de usuario comprometidos

... —sí, sólo se planifica detalladamente de a una iteración a la vez

Como esta planificación es de corto plazo, debe tomar en cuenta:

- el número de horas diarias que el equipo va a dedicar al proyecto
- el número de días que cada integrante del equipo va a estar disponible durante la iteración



“What is a product backlog? (And how to create one),”  
por Alicia Raeburn  
<https://asana.com/resources/product-backlog>

Para cada relato que se va a abordar en la iteración

... se debe considerar las tareas que es necesario llevar a cabo

... incluyendo las más obvias como diseño, programación y testing

... pero también tareas específicas a cada relato que deben ser identificadas durante la planificación de la iteración

La ingeniería de software—sus procesos, métodos, técnicas, prácticas, etc.—es de aplicabilidad general

Dondequiera que haya una necesidad de desarrollar una aplicación de software como solución a un problema

... se pueden usar estos procesos, métodos, técnicas, prácticas, etc. para hacer un mejor trabajo (si se usan bien)

La ciencia de datos puede ser vista como un área de aplicación de la ingeniería de software

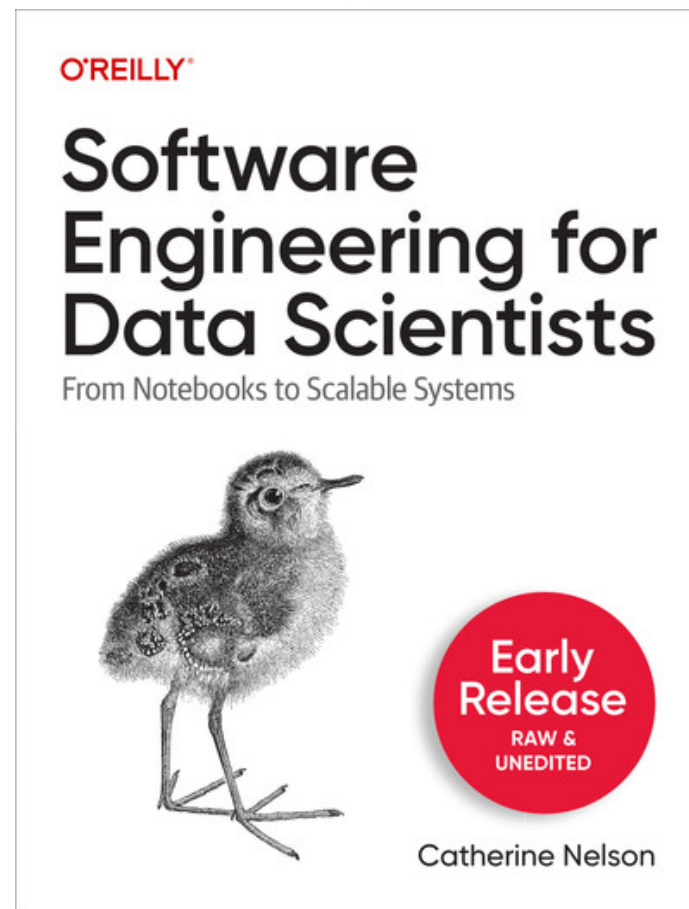
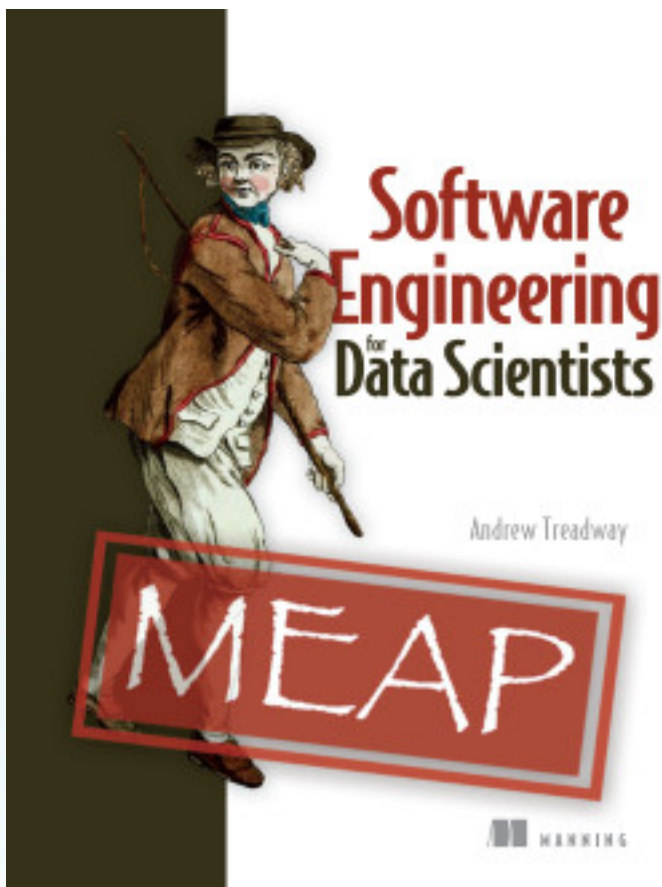
Considerando por una parte los volúmenes, dispersión y variedades de datos involucrados

... y por otra la rapidez con la que se quiere tener resultados frente a consultas complejas, muy variadas

... es sólo natural y lógico pensar en desarrollar productos de software que permitan hacer esto haciendo uso de todo el poder computacional que nos ofrece la tecnología moderna



Aún en desarrollo:  
prometidos para fines de 2023 o principios de 2024



Crear aplicaciones que usan los datos  
(ingeniería de software, en general)

Organizar y manejar datos  
(bases de datos)

Procesar datos (algoritmos)

Importancia de la ingeniería de software  
para la ciencia de datos (de acuerdo con  
S. Raju y S. Gupta, de Great Learning\*):

Visualizar datos

Testing y evaluación de datos

\*<https://medium.com/@mygreatlearning/importance-of-software-engineering-for-data-science-a4ca17d6c99>

Escribir código de alta calidad

Usar la línea de comandos

E. Berge\* propone que la/os data scientists deben aprender/desarrollar estas cuatro habilidades/competencias clave de desarrollo de software al inicio de sus carreras:

Usar control de versiones

Escribir tests

\*<https://towardsdatascience.com/why-software-development-skills-are-essential-for-data-science-f53364670bd7#5f93>

The Cross Industry Standard Process for Data Mining (CRISP-DM) is a process model that serves as the base for a [data science process](#). It has six sequential phases\*:

Business understanding – What does the business need?

Data understanding – What data do we have / need? Is it clean?

Data preparation – How do we organize the data for modeling?

Modeling – What modeling techniques should we apply?

Evaluation – Which model best meets the business objectives?

Deployment – How do stakeholders access the results?

Published in 1999 to standardize data mining processes across industries, it has since become the [most common methodology](#) for data mining, analytics, and data science projects.

Data science teams that combine a loose implementation of CRISP-DM with overarching team-based [agile](#) project management approaches will likely see the best results.

\*<https://www.datascience-pm.com/crisp-dm-2/>

Pueden aplicarse siguiendo el estilo “cascada”, definiendo planes detallados para cada fase al inicio del proyecto

Requisitos funcionales:  
- casos de uso  
- relatos de usuario  
Product Owner

### Six sequential phases (CRISP-DM):

1. Business understanding – What does the business need?
2. Data understanding – What data do we have / need? Is it clean?
3. Data preparation – How do we organize the data for modeling?
4. Modeling – What modeling techniques should we apply?
5. Evaluation – Which model best meets the business objectives?
6. Deployment – How do stakeholders access the results?

Patrones de diseño:  
- Estrategia  
- Fachada  
- Adaptador  
- Fábrica  
- Decorador

Revisión del Sprint  
Retrospectiva del Sprint

Pueden aplicarse siguiendo el estilo “ágil, iterativo, incremental”, moviéndose de ida y vuelta entre las fases, de acuerdo con las necesidades (muchas veces cambiantes) del proyecto, y repitiendo varias veces la secuencia abordando distintas necesidades en cada repetición

Arquitectura de software:  
- estratificada  
- cliente-servidor  
- publicador-suscriptor

# Ejemplo

38

<https://www.datascience-pm.com/crisp-dm-2/>

Un proyecto *churn* con tres entregables:

- un modelo *churn* voluntario
- un modelo *churn* de no pago y desconexión
- un modelo de propensión a aceptar una oferta para retención

En *cascada*, ejecutas completamente una de las 6 fases de CRISP-DM para los tres modelos antes de pasar a la siguiente fase para cualquiera de los modelos

En *ágil*, ejecutas completamente las 6 fases de CRISP-DM para uno de los modelos (una iteración) antes de iniciar la secuencia de 6 fases para otro modelo (otra iteración)

... sin embargo, ...

# Desafíos en data science y soluciones ofrecidas por la ingeniería de software

¿Cómo manejamos cambios al mismo código hechos por diferentes desarrolladores?

¿Cómo hacemos testing a nuevas funcionalidades?

¿Cómo facilitamos que el código escrito por nosotros— código confuso escrito en varios lenguajes de programación y distribuido entre múltiples archivos —pueda seguir siendo desarrollado por otros?



Código mejor estructurado, al aplicar principios y patrones de diseño

Colaboración entre varios desarrolladores produciendo código, mediante control de versiones

Código que puede “escalar”, usando algoritmos y estructuras de datos más eficientes y técnicas de caching

Poner modelos en producción usando arquitecturas de software y manejo de excepciones

Testear eficazmente el software usando test driven development

# El patrón *Adaptador*

Aumenta la compatibilidad entre interfaces

... p.ej., formatos de datos

... por lo tanto, es usado a menudo para leer datos almacenados en diferentes formatos

... y convertirlos a un objeto de datos estándar

... p.ej., Pandas tiene unos 20 adaptadores para leer la mayoría de los tipos de archivos a su Dataframe

Format Type	Data Description	Reader	Writer
text	CSV	read_csv	to_csv
text	Fixed-Width Text File	read_fwf	
text	JSON	read_json	to_json
text	HTML	read_html	to_html
text	LaTeX		Styler.to_latex
text	XML	read_xml	to_xml
text	Local clipboard	read_clipboard	to_clipboard
binary	MS Excel	read_excel	to_excel
binary	OpenDocument	read_excel	
binary	HDF5 Format	read_hdf	to_hdf
binary	Feather Format	read_feather	to_feather
binary	Parquet Format	read_parquet	to_parquet

<https://eugeneyan.com/writing/design-patterns/>



# El patrón *Fábrica*

Desacopla los objetos (el uso de éstos)

... de cómo son creados

(crear objetos puede ser complejo; una fábrica de objetos simplifica el trabajo de los programadores y previene errores)

Se puede definir, p.ej., mediante una interfaz o clase abstracta

... luego, producimos una subclase y la equipamos con nuestra propia implementación

```
from torch.utils.data import Dataset
```

```
class SequencesDataset(Dataset):
```

```
    def __init__(self, sequences: Sequences, neg_sample_size=5):
```

```
        self.sequences = sequences
```

```
        self.neg_sample_size = neg_sample_size
```

```
    def __len__(self):
```

```
        return self.sequences.n_sequences
```

```
    def __getitem__(self, idx):
```

```
        pairs = self.sequences.get_pairs(idx)
```

```
        neg_samples = []
```

```
        for center, context in pairs:
```

```
            neg_samples.append(self.sequences.get_negative_samples(context))
```

```
        return pairs, neg_samples
```

<https://eugeneyan.com/writing/design-patterns/>

# El patrón *Decorador*

Queremos hacer algo antes y/o después de la ejecución de una función

... pero no queremos modificar la función propiamente dicha

(especialmente si tenemos que modificar muchísimas funciones similarmente)

Básicamente, capturamos un cierto estado antes de la ejecución de la función, y luego capturamos un cierto estado después de su ejecución

```
def log_time(func):
    """Logs the time it took for func to execute"""
    def wrapper(*args, **kwargs):
        start = time()
        val = func(*args, **kwargs)
        end = time()
        duration = end - start
        print(f'{func.__name__} took {duration} seconds to run')
        return val
    return wrapper

@log_time
def get_data(db, query):
    """Gets data from a SQL-based database"""
    data = db.get(query)
    return data

if __name__ == '__main__':
    # Decorated function will print 'get_data took X seconds to run'
    db = SQLDB()
    query = 'SELECT * FROM foo'
    data = get_data(db, query)
```