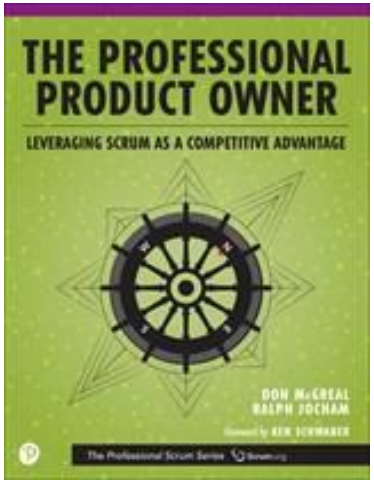
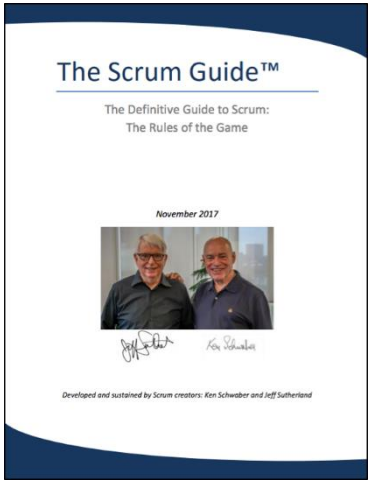
Tema 7

Funciones del Product Owner

Pablo.Bermejo@uclm.es

- Este tema está basado en la siguiente bibliografía, de donde también se obtienen la mayoría de sus figuras:
  - The Professional Product Owner – Don McGreal y Ralph Jochan. Addison-Wesley.2017.
  - Scrum Guide – Scrum.org – 2017.





7.1 Introducción

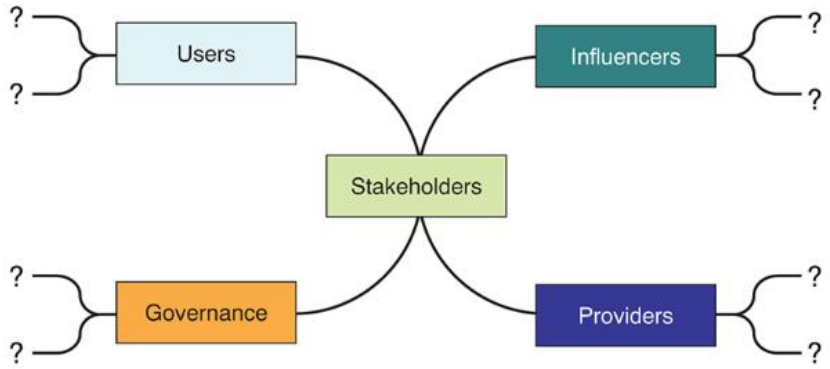
- Los 3 pilares de Scrum son:
  - Transparencia.
  - Inspección.
  - Adaptación.

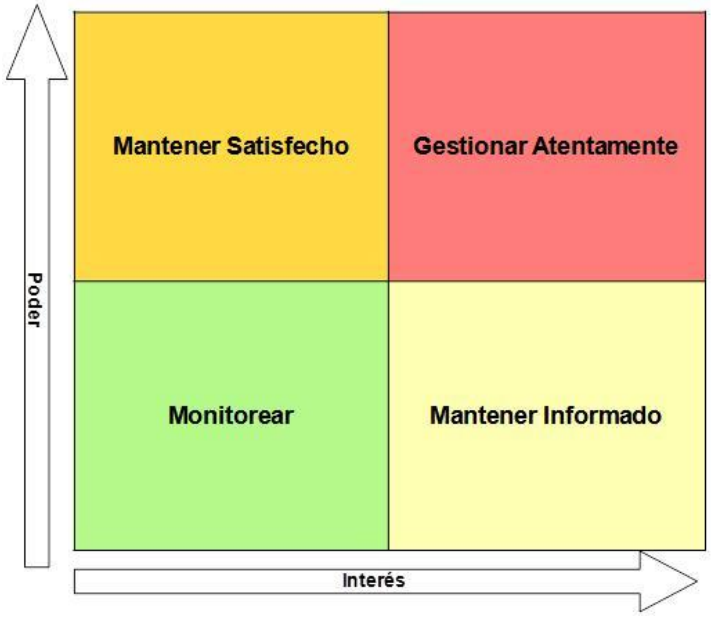
- Las **funciones del Product Owner** son
  - Maximizar el valor del producto entregado al cliente.
  - Gestionar el Product Backlog (crear, refinar y priorizar por valor los PBIs).
  - Asegurarse de que todos entienden los PBIs.
  - Mantener de forma transparente el trabajo realizado y cuánto queda (por lo general, con el PBCh).
  - Realizar predicciones.

7.1 Introducción

7.1 Introducción

- Mediante tormenta de ideas, el PO debe identificar quiénes son los interesados del proyecto para tenerlos en cuenta, y actuar con cada uno según proceda.

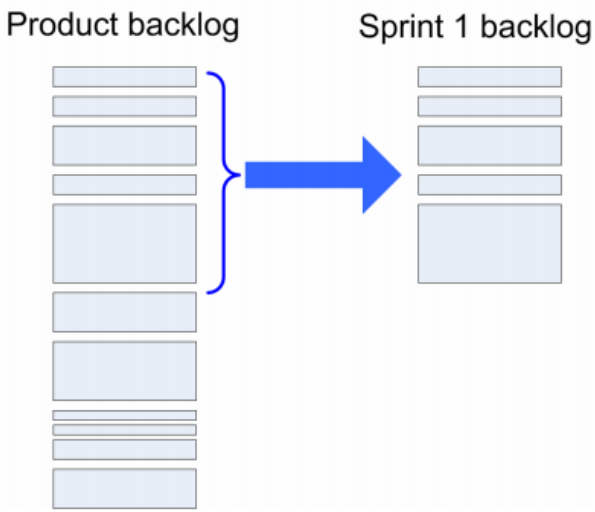




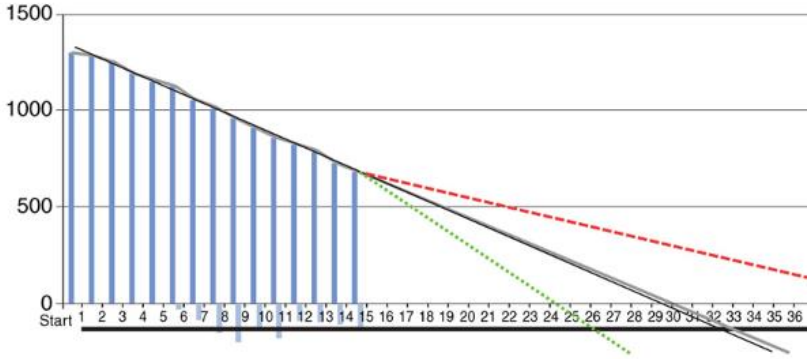
7.1 Introducción

- El Product Backlog debe ser la **única fuente de trabajo** para el Equipo de Desarrollo. Un producto tendrá 1 Product Owner y 1 Product Backlog.

- Los Sprint Backlogs son asunto del Equipo de Desarrollo.



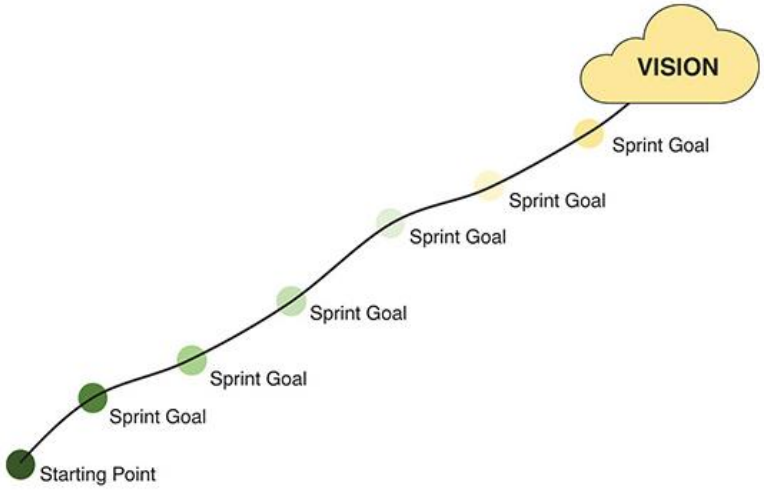
- El PO y los interesados solo les importa el avance y trabajo pendiente a través de los sprints. Por eso**, el PO trabajará con** gráficos de quemado de releases (**Release burn-down**), más comúnmente conocido en Agile como Project Burndown Chart.



7.1 Introducción

- En el Sprint Planning, el PO propone un objetivo, y entre todo el equipo se define el Sprint Goal.

- Si durante un sprint **aparece una nueva característica** funcional, las opciones del PO son:
  - Colocarla en el tope del Pbacklog para que el Equipo de Desarrollo la seleccione en el próximo sprint.
  - Si anula el Sprint Goal actual, entonces cancelar el sprint.



7.1 Introducción

- En el Daily Scrum:
  - El PO puede acudir para mostrar su compromiso.
  - No debe participar.
  - Tras la reunión puede resolver las dudas que haya respecto a los PBI.

- En el Sprint Review:
  1. De nuevo comparte su Visión con el equipo y los interesados.
  2. Explica el Sprint Goal.
  3. Deja que el Equipo de Desarrollo muestre el trabajo “Done”.
  4. Obtiene retroalimentación de los interesados para…
  5. … adaptar el Product Backlog.
  6. Realiza estimaciones.

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.2.1 Tipos de PBI

- Se suele asumir de forma errónea que los **elementos del Product Backlog** son solo Historias de Usuario. Principalmente hay 9 tipos de PBIs:

7.2.1 Tipos de PBI

1. Peticiones de características: peticiones de los interesados.

quiero acceso de administrador, quiero que se pueda ordenar la lista por precio

2. Requisitos no funcionales: desempeño, escalabilidad, términos legales,…

el sistema de evaluación online debe soportar 200 conexiones concurrentes

3. Experimentos: funcionalidad a desplegar solo en un segmento de usuarios para obtener retroalimentación (explícita o implícita).

una función cuya utilidad es dudosa, se despliega a un sector y se monitoriza su uso.

4. Historias de usuario: expresión de un requisito forzando brevedad, forzando así también la comunicación para adquirir más detalle.

Como ... (rol) quiero … (comportamiento) …. para que… (valor)

7.2.1 Tipos de PBI

- **Bugs:** problemas que han aparecido tras la última release o último incremento desplegado.

- **Casos de uso:** lista de acciones entre un actor y un sistema. Es muy poco común actualmente en la planificación del software. Tiene algo más de sentido en la documentación posterior, si se estima que es necesaria (software sanitario, software complejo, trabajo académico, …)

- **Capacidades:** canales o métodos de acceso a la funcionalidad existente.

  Acceso desde móvil, web, nube, proporcionar APIs

- **Otros***: épicas, spikes, foto, documento, cualquier diagrama UML,…*

7.2.1 Tipos de PBI

- Las **historias de usuario** (origen en XP) fuerzan la brevedad y la ambigüedad. Así, se fomenta la comunicación cara a cara.

- De manera informal, pueden verse como el flujo principal que programamos primero de un caso de uso.

- Hay varias propuestas complementarias aceptadas comúnmente, para su formato y para crearlas con calidad:
  - Las 3 Cs (Essential XP. Ron Jeffries. 2011).
  - Formato *Como…, quiero, para*… (User stories applied. Mike Cohn. 2004).
  - INVEST para que las historias sean de calidad (Extreme Programming Explored. Bill Wake. 2002).
  - DEEP (Make the Product Backlog DEEP, Mike Coh. 2009).

7.2.1 Tipos de PBI

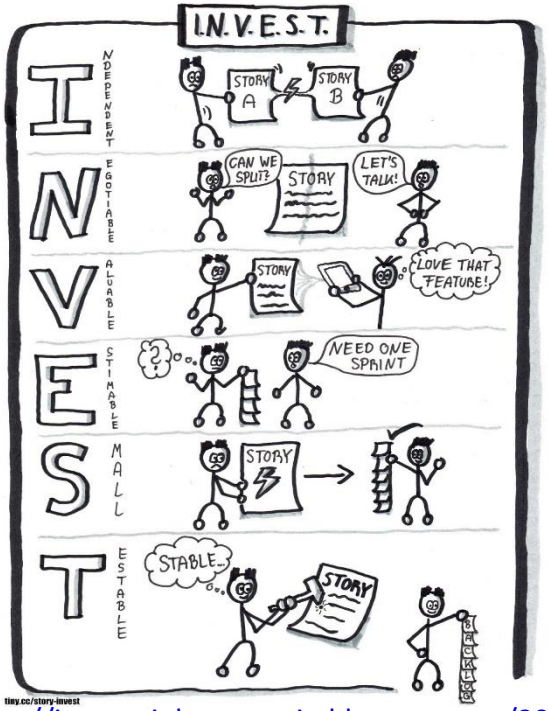
- **Las 3 Cs para una historia de usuario:**
  - **Card**: no escribir requisitos, sino algo que se exprese el valor proporcionado el negocio.
  - **Conversation**: no estresarse pensando detalles al crearla. Habrá que conversar para refinarla.
  - **Confirmation**: los detalles se van capturando como criterios de aceptación, no más de los que cabrían tras una tarjeta de historia de usuario.

- **Plantilla de formato de historia de usuario:**
  - **Como <rol>:** un rol del contexto del negocio. Evitar "usuario" o roles de Scrum.
  - **Quiero <comportamiento>:** acción o flujo orientada al negocio.
  - **Para <valor>:** la razón por la que se pide esa funcionalidad.

As a	blog reader
I want	to comment on a blog entry
so that	I can contribute to the conversation

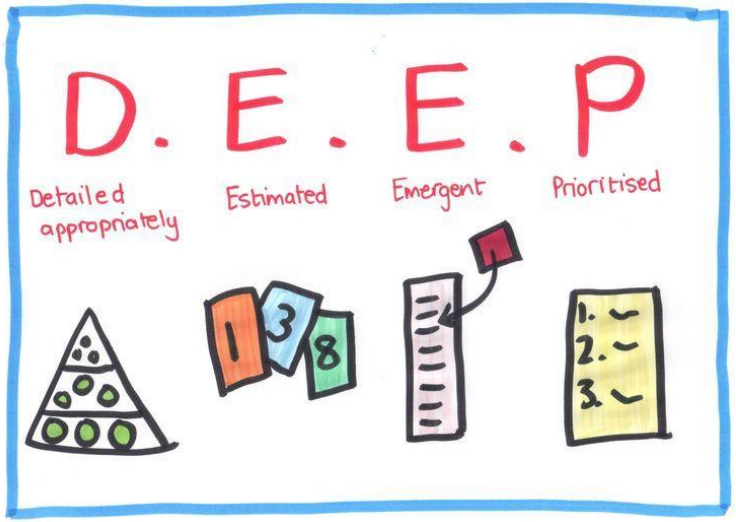
7.2.1 Tipos de PBI

- **INVEST:**
  - Independiente
  - Negociable
  - Valiosa
  - Estimable
  - (Small) Pequeña
  - Testeable.



http://itscertainlyuncertain.blogspot.com/2014/03/invest.html

- **DEEP:**
  - Detallada con los suficientes criterios de aceptación para empezar
  - Emergente: se refina con el tiempo.
  - Estimada: tamaño.
  - Priorizada: por valor, riesgo, costos, dependencias, ….



https://www.slideshare.net/AnaPegan/intro-to-scrum-65115745

7.2.1 Tipos de PBI

- Los **Requisitos No Funcionales** comprenden: usabilidad, escalabilidad, mantenimiento, seguridad, accesibilidad, …

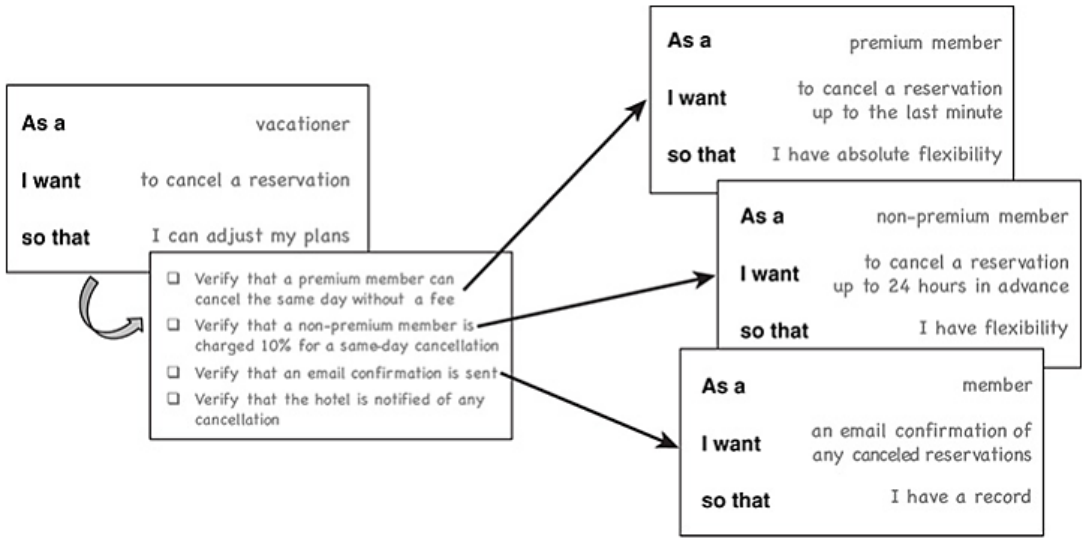
- Pueden **capturarse de 3 maneras**:
  - Como un PBI, pues tienen valor para el negocio.
  - Como criterio de aceptación (‘el Login tarda 2 segundos’, ‘password enmascarado’).
  - Dentro de la DoD (‘todas las páginas deben cargarse en 3s’, ‘contenido en inglés y español’).

7.2.1 Tipos de PBI

- Las épicas son PBIs que no pueden terminarse en 1 sprint, y por lo tanto tienen que dividirse en PBIs más pequeños.

- No conviene ajustar tanto como para que solo pueda seleccionarse 1 historia de usuario para 1 sprint. Esto supone mucho riesgo.

- Cada división de la épica todavía tiene que tener valor:
  - No dividir pensando en componentes tecnológicos (UI, bbdd,…)
  - Basarse en los criterios de aceptación.



7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.2.2 Criterios de aceptación

- Tantos los PBI como los criterios de aceptación son propiedad del PO (aunque se involucra a todo el equipo).

- **Tres formas** en que puedes escribir los **criterios de aceptación**:
  - Probar que … (mentalidad de test).
  - Demostrar que … (mentalidad de Sprint review).
  - Dado <condición> cuando <acción de usuario> entonces <resultado esperado>

- También existen consejos para asegurar calidad en nuestros criterios de aceptación (no lo veremos) en forma de acrónimos: SMART, SAFE.

7.2.2 Criterios de aceptación

Product Backlog Item

Product Name:	AMP			
User Story Name:	Renewal Notification			
User Story Description:	As a sales manager I want to be notified by email of expiring contracts so that customers are not without coverage and we are not missing out on revenue.			
Release:	1	**Sprint:**	1	
Notification email sent to all Account Executives (AE) on the 15th and 30th of every month containing a summary* of all expiring contracts. • Summary email content outlined in analysis doc				Tested YES Accepted
AE summary email contains a clickable URL to AMP, which displays all expiring contracts (see Query User Story) for that AE.				Tested YES Accepted
Once a month, Account Sales Manager (ASM) receives an escalation email* containing all their team's contracts expiring within 60 days. • 60-day Escalation email content outlines in analysis doc				Tested YES Accepted
60-day Escalation email contains a clickable URL to AMP, which displays all Expiring Contracts for the ASM's team.				Tested YES Accepted
Approved for Release:	YES	**Date:**	Jun 26, 2017	

PBI con criterios de aceptación

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.2.3 Ordenar el Product Backlog

- A la hora de priorizar los PBI, conviene tener en cuenta **4 dimensiones**.

- **Valor para el Negocio:** las características más próximas a la Visión deberían estar mas altas en el PB. El valor debería valorarse teniendo en cuenta a los interesados.
  - *"Hacer pago", función muy importante para el cliente.*

- **Riesgo:** de negocio y técnicas. Cuanto más riesgo, más alto en el PB.
  - *Una característica que debe cumplir con una nueva normativa en un plazo dado.*
  - *Una característica que utilizará un API que no hemos probado aún.*

- **Tamaño:** el esfuerzo que requiere el equipo de desarrollo para cumplir el PBI.

(Valor + Riesgo) / Tamaño = Orden

7.2.3 Ordenar el Product Backlog

Si tenemos:

- PBIa → (25 valor + 5 riesgo) / 8 tamaño = 3.75
- PBIb → (15 valor + 10 riesgo) / 5 tamaño = 5

PBIb estará más alto en el Product Backlog.

- **Dependencias:** independientemente de lo anterior, a veces un PBI no puede comenzarse hasta que otro esté Done.
  - *Autenticación antes de poder usar otras funciones.*
  - *Una característica del front que depende de que sea recibida por el end.*

- Usa tu tiempo para ordenar los PBIs para los próximos sprints, no todo el Backlog entero, ya que aún les faltará refinamiento.

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.2.4 DoD

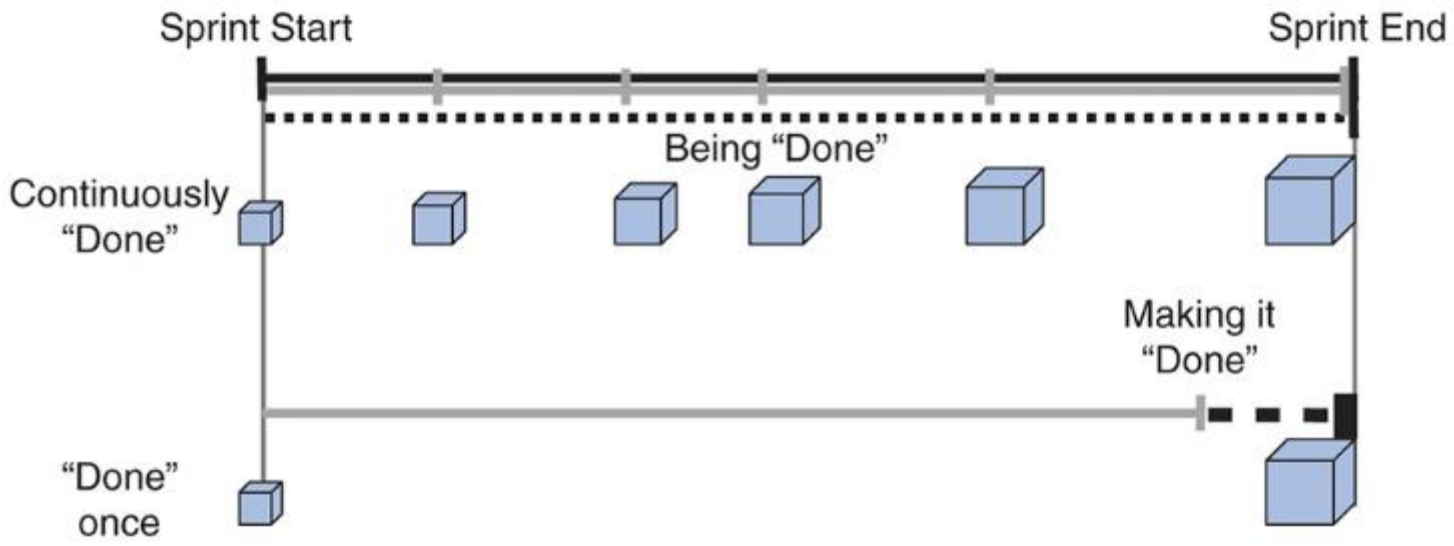
- No hay definición formal de qué es estar “Done”, es algo que debe acordar y entender todo el equipo, y evoluciona con el tiempo.

- La DoD es **para el Incremento** tras el sprint. Los criterios de aceptación son a nivel de PBI.

- Ejemplos de elementos dentro de la DoD:
  - *Tests unitarios, aceptación, regresión o seguridad pasados.*
  - *Estilos de código.*
  - *Sin defectos.*
  - *Subido a rama principal.*
  - *API documentada.*
  - *Aprobado por el PO.*
  - *Guía de usuario actualizada*
  - *Pruebas de desempeño.*

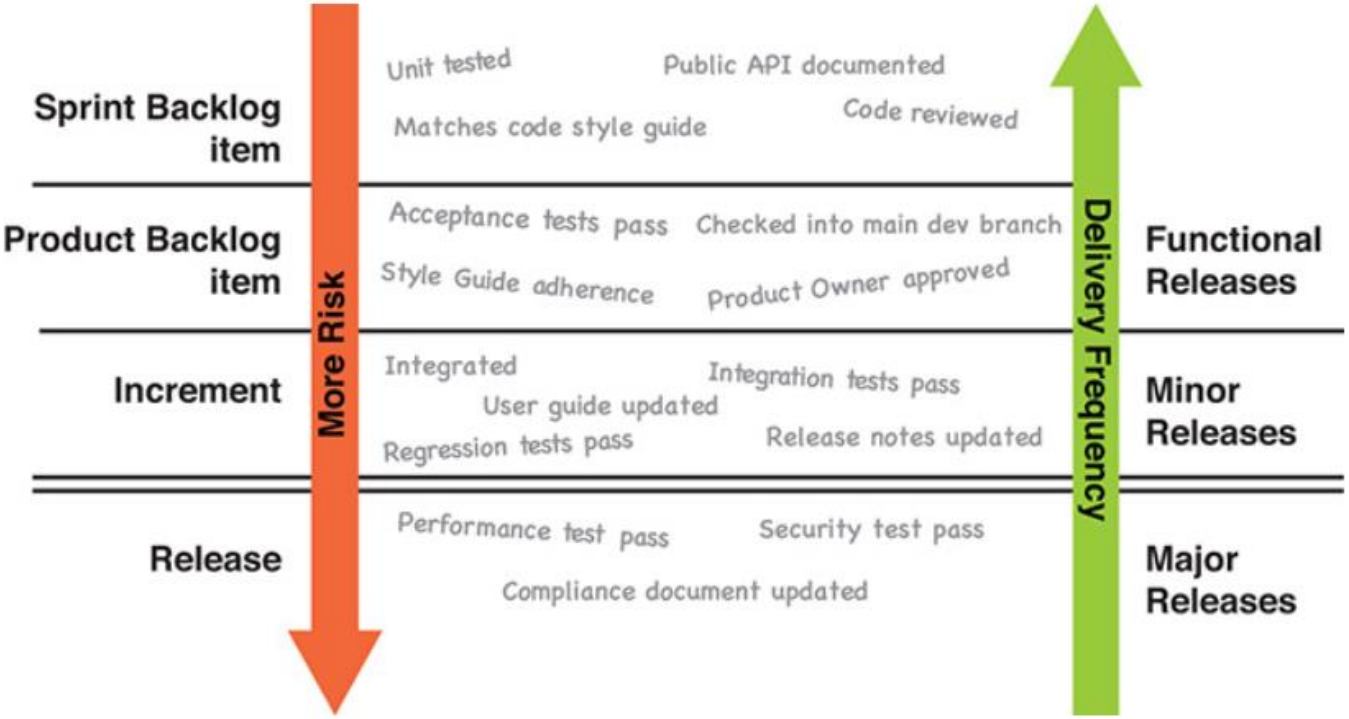
7.2.4 DoD

- ¿En qué momento se debe comenzar a trabajar en cumplir con la DoD?



- Esperar al final para cumplir con tests, subida a repositorio, integración, … puede presentar un riesgo muy alto.
- Hoy en día, mediante CI/CD la DoD puede cumplirse con muchos PBI desde el principio.

7.2.4 DoD



- Muchos elementos de la DoD pueden beneficiarse de la automatización aprendida en la primera vía de DevOps.

7.2.4 DoD

- Jeronimo Palacios:
- It works
- Passes the acceptance criteria
- It is in production
- Peer reviewed
- Covered by tests
- Passes all the tests
- Merged into Master
- No apparent bugs
- Accepted by the PO
- API documented
- Anything not trivial commented
- Argument for binaries documented

- Barry Overeem:
- Created design
- Updated documentation
- Tested
- Approved by Product Owner
- Clear on "how to demo" at Sprint Review

- Francois Desrosiers:
- Code review done
- Unit tests done
- BDD tests done based on the acceptance criteria
- Documentation done (based on the documentation that the client defined
- Features tested/approved by the PO or business analyst
- Features deployed in the staging environment
- Source code of the feature is in the proper branch

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

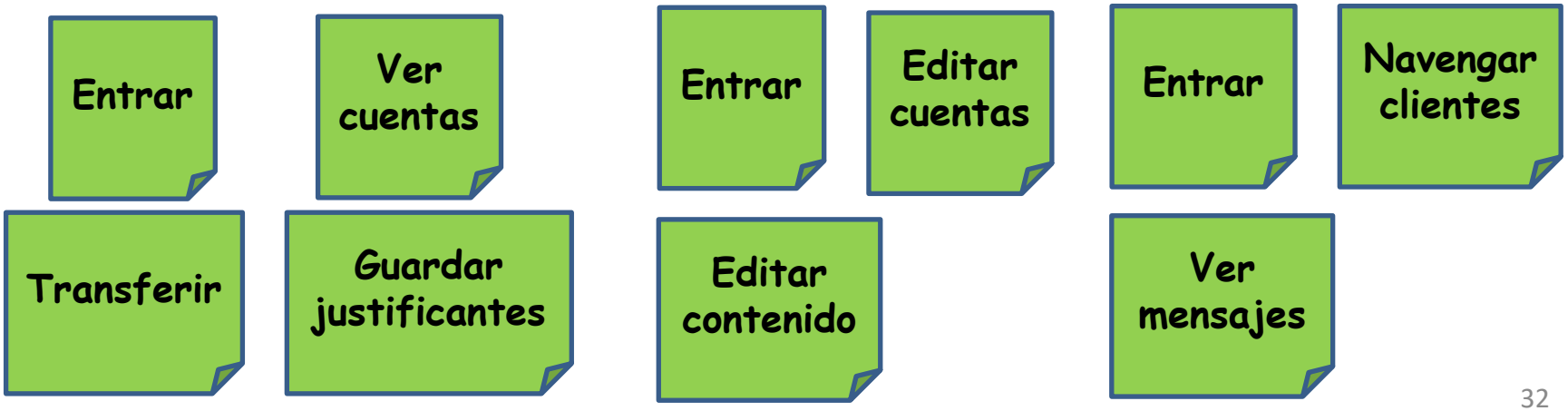
7.3 User Story Mapping

- Técnica propuesta por Jeff Patton.
- Su utilidad principal es **inicializar el Product Backlog** con ayuda del equipo de desarrollo. A la sesión también pueden asistir los interesados.
- Al finalizar, se obtiene un conjunto de PBIs, ordenados y con releases marcadas. Los PBIs de más arriba estarán ***ready*** (ordenado, estimado y detallado).
- Tiene varias fases:
  1. Identificar Personas.
  2. Identificar actividades *(backbone).*
  3. Crear épicas *('walking skeleton')*
  4. Crear historias de usuario. Fusionar las que se repitan entre usuarios o actividades.
  5. Ordenar por valor.
  6. Marcar releases.

7.3 User Story Mapping

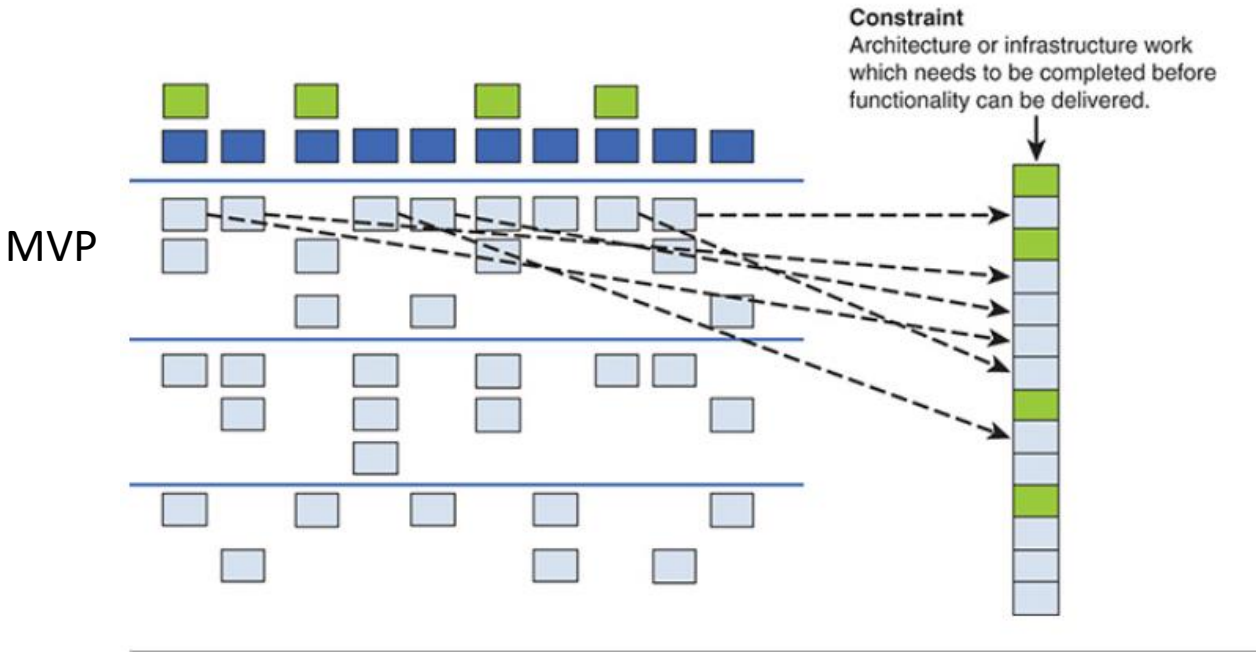
1. Identificar Personas: Pensar qué tipo de usuarios diferentes harán uso del producto. Ordenar de izquierda a derecha según importancia.

2. Identificar actividades (backbone): pensar para cada tipo de persona cómo utilizará el producto, teniendo en cuenta orden temporal si es posible.



7.3 User Story Mapping

- 3,4: Épicas e historia de usuario: pensar épicas y luego sus historias de usuario, en el orden de un día típico de la persona.
- 5,6: Ordenar por las historias de usuario según valor, riesgo, tamaño y dependencias; y separarlas horizontalmente en releases. Después, crear el Product Backlog.



7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.4 Gestión de releases

- Como Product Owner, tendrás **presiones de los interesados** para que les proporciones, acostumbrados a la gestión plan-driven:
  - Cronogramas.
  - Presupuesto.

- El gestor del producto ágil, o Product Owner, pedirá un primer nº **de sprints de prueba** para conocer las primeras limitaciones y hasta dónde puede llegar el equipo de desarrollo.

- Una planificación de un producto software con <tiempo, alcance, costos> fijado y firmado de antemano, es muy probable que no cumpla. O, si cumple, sea a costa de valor. **Cumplir una predicción no es sinónimo de entregar el mayor valor posible.**

- A continuación veremos métodos para planificar releases y estimar tiempo y presupuesto.

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.4.1. Tipos de release

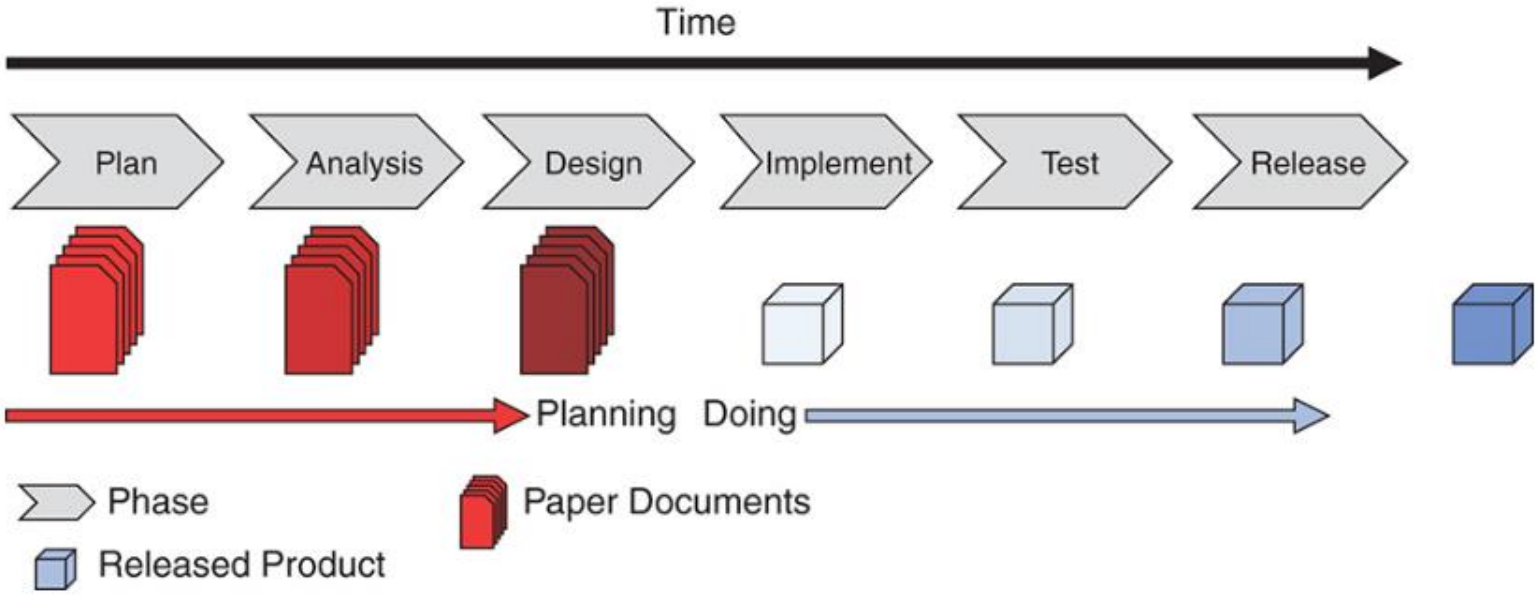
- Con **release**, no nos referimos solo al sentido clásico de entrega grande o versionada. Puede ser eso, o cualquier tipo de incremento desplegado.
- Lo que debemos tener claro es que la release es la única forma de **entregar valor**.
- Podemos distinguir los siguientes tipos de release:

Forma	Proceso	Tipo	Validación
Fases	Waterfall	Mayor (cada 6-12 meses)	Al final.
Fases con desarrollo ágil	Híbrido		Al final.
Scrum entregando los incrementos solo en fechas acordadas	Scrum		Una vez tras cada fecha de release
Scrum con una entrega al final de cada Sprint	Scrum	Menor (cada1-3 meses)	En los límites del Sprint
Scrum con varias entregas durante el Sprint	Scrum	Funcional (CD)	Conforme se completan características.

7.4.1. Tipos de release

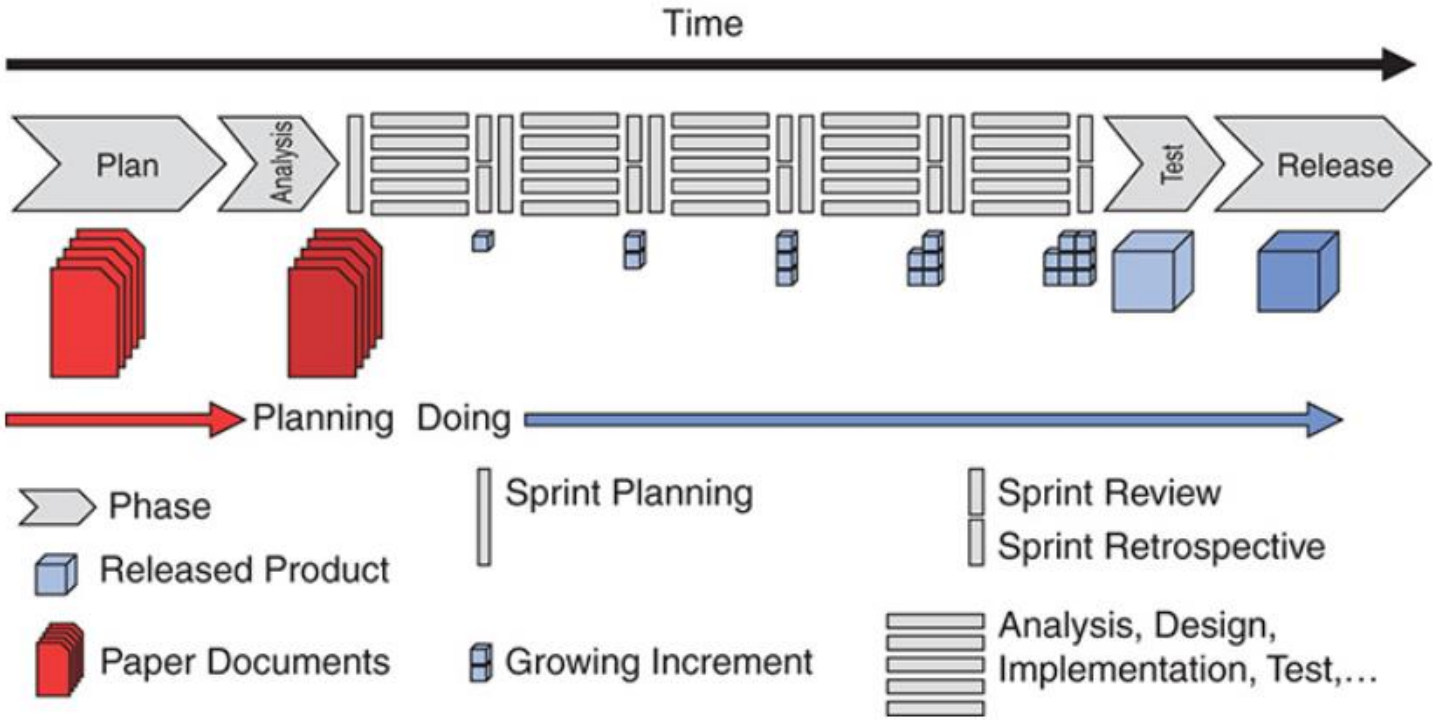
Major Release

- Un **proceso de cascada** siempre lleva a releases grandes. Al seguir un pensamiento lineal, una release es el resultado de un conjunto de fases secuenciales.
- Es lo adecuado cuando el producto es simple, bien entendido y sin riesgos. Si el producto es complejo, hacer la asunción de que el producto será correcto al final es un **gran riesgo**.



7.4.1. Tipos de release

- En procesos **híbridos**, se agrupan fases técnicas en lo que llaman iteraciones o sprints. Normalmente hay un plan aprobado, tras el cual se usa Scrum para implementar requisitos preacordados, y después se hacen algunas pruebas antes de la release. → Water-Scrum-Fall



7.4.1. Tipos de release

- Una major release supone no reducir riesgos durante mucho tiempo y descubrir si hemos desarrollado lo que se esperaba del producto.

- Los principales **motivos** para que una empresa siga una estrategia de releases grandes puede ser:
  - Es necesario realizar una **inversión** para obtener:
    - Tecnología de CI/CD
    - Virtualización.
    - Nuevo HW en producción.
  - Capacidad de **absorción del cliente**: necesita entrenamiento o tiene miedo de cambiar de versión porque los últimos cambios fueron frustrantes (instalación compleja, pérdida de datos,…)

7.4.1. Tipos de release

"I worked on a year-long initiative to create a first-of-its-kind online document printing product. Five months in, we had the ability to upload a file and set some minimal printing options, without the ability to pay or add more complex finishing options (tabs, inserts, bindings, etc.). At the time, management did not want to hear about going to production early as we were not yet "finished." Seven months later, after going to production, we realized that few customers actually used all the extra finishing options we painfully added to the product. Through a Special Instructions text area, we found out what customers really wanted: oddly sized posters and banners. Looking back, the ideal MVP would have been a simple upload with minimal printing options and an instructions text field. We could have then guided product development based on what our actual retail users were asking for – speeding up the feedback loop and generating real value more often."

Don McGreal

7.4.1. Tipos de release

Minor Release

- Pueden ser necesarias por las empresas que realizar releases grandes y no pueden esperar hasta la próxima. Por ejemplo, por la necesidad urgente de parches o solucionar bugs.

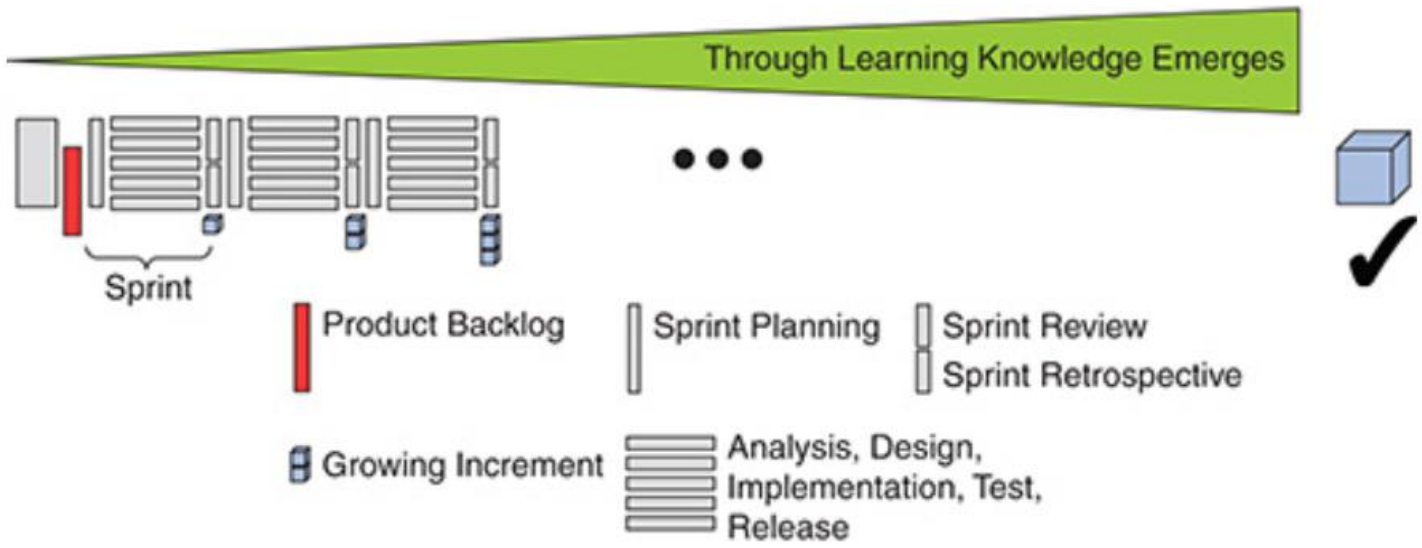
- En equipos Scrum, se despliega una minor release al final de cada sprint o al principio del siguiente. Esto permite una reducción del riesgo y un incremento gradual del coste.

- Las numeraciones de las major y minor releases son de la forma: 3.4, 2.1.5

7.4.1. Tipos de release

Release funcional

- El equipo despliega cada característica **conforme está** lista para ello: CD, la cual lleva a validación y aprendizaje continuo.
- Se añade como último elemento de la DoD "está desplegada".
- Así se elimina el sobrante: cualquier trabajo realizado antes de que el cliente lo reciba se considera *inventario*, que necesita ser mantenido y gestionado sin aportar ningún valor.



7.4.1. Tipos de release

- Cuanto más nos aproximamos a las releases funcionales, más necesitamos técnicas agiles:
  - Mover calidad a la izquierda.
  - Operaciones y seguridad a la izquierda.
  - Tests automáticos.
  - Despliegue automático.
  - Equipos multidisciplinares.
  - Equipos autoorganizativos.
  - Interesados comprometidos.

7.4.1. Tipos de release

- ¿Qué tipo de release provocó el retraso del estreno de esta película?
- ¿Cómo lo podrían haber evitado?

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.4.2 Velocidad y Throughput

- La **Velocidad** del Equipo de Desarrollo es la media del esfuerzo (puntos historia) quemados por sprint.

- Algunos equipos prefieren calcular el **Throughput**, el cual es la media de los PBIs que están Done por sprint (depende menos de la coherencia en estimar tamaños).

- Ambos pueden variarpuede variar:
  - Miembro del equipo enfermo.
  - Conflictos.
  - Problemas de hardware.

- Un equipo cuyo Scrum Master se encargue de que no tenga distracciones y se dedique al mismo producto, tendrá una velocidad y throughputs más **estables** y por lo tanto más **predecibles**.

7.4.2 Velocidad y Throughput

- Imagina que, según tus cálculos, tu Equipo de Desarrollo tiene actualmente una velocidad de 15 SP por sprint y un throughput de 4 PBIs.

- El cliente quiere saber cuánto bajará el PBChart en los siguientes 6 meses con sprints quincenales:

  15 * 12 = 180 SP reducidos

- El cliente está muy interesado en recibir la funcionalidad descrita en los primeros 9 PBIs del Product Backlog. Quiere tener una estimación aproximada de cuándo estarán listos (usas releases funcionales):

  9 / 5 = 2.25 sprints

  2.25 * 14 días/sprint = 31.5 días.

7.4.2 Velocidad y Throughput

- Cuando des tu estimación:
  - No enseñes ninguna fórmula. Evita dar la **sensación** de que hay precisión y **certeza**. En el desarrollo de productos puede ocurrir cualquier cosa.
  - Si el equipo es nuevo o ha sufrido **cambios**, tus datos históricos ya no tienen la misma valía.
  - Si usas velocidad, ojo a la **coherencia** en la estimación del equipo. Una solución suele ser estimar tamaño de tallas (T-shirt size), y asignar luego en tus cálculos a cada talla un número. Pero el Equipo de Desarrollo no tiene por qué conocer dicha asignación.

- Cuidado, estas 2 métricas no deben verse como una representación del valor. Solo **sirven para estimar** cuándo acabaremos.

Ley de Goodhart:

'Cuando una métrica se convierte en el objetivo, deja de ser una buena métrica'

7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

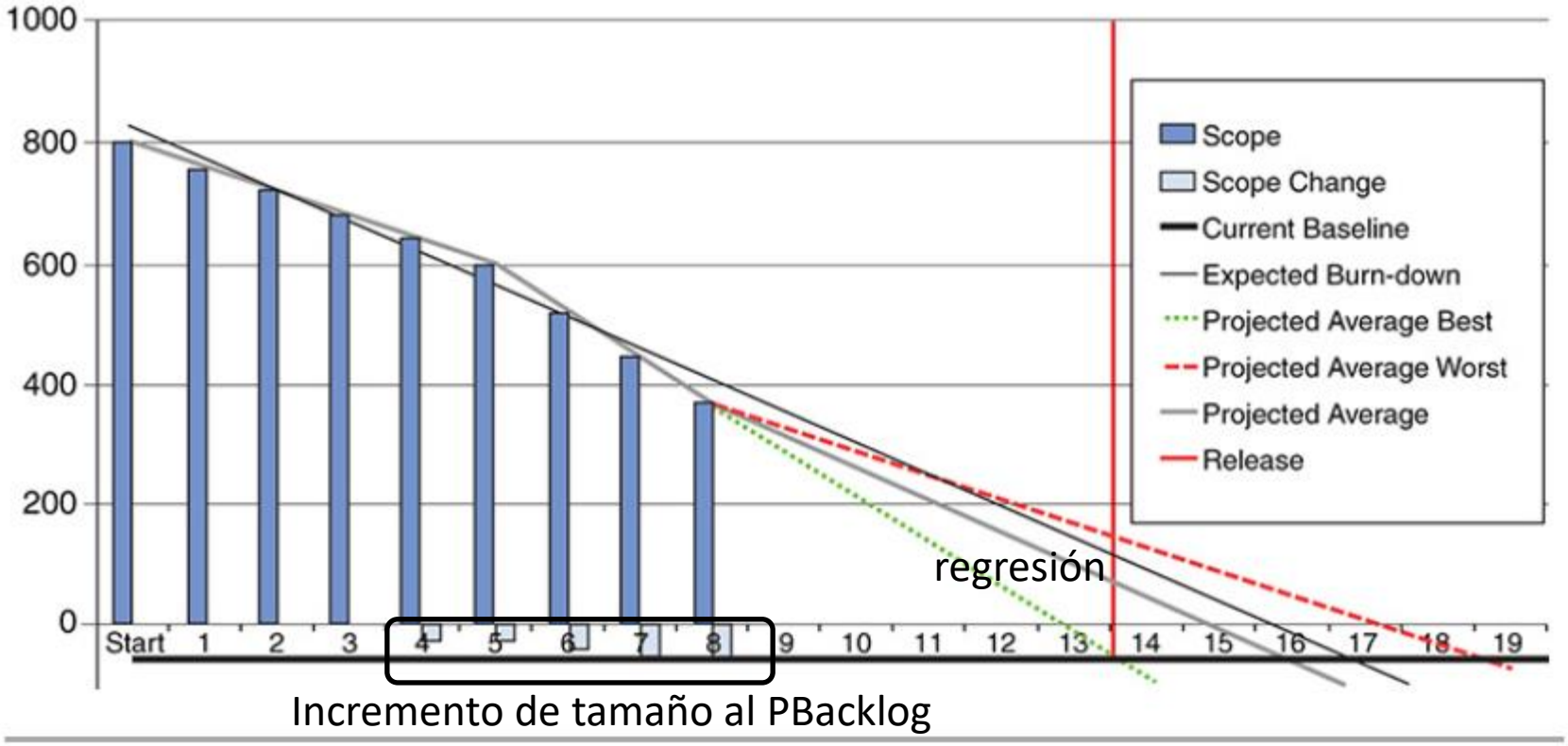
7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

7.4.3 Informes y predicciones

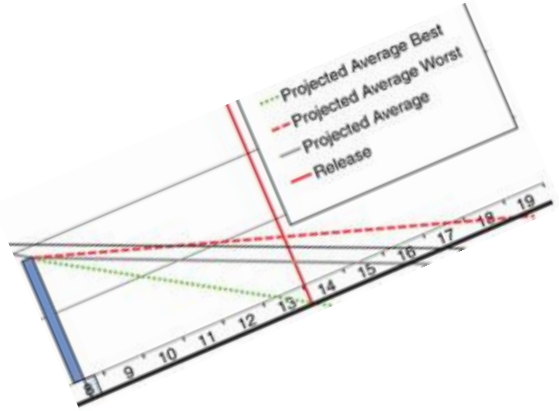
- El informe básico que realiza el PO es el PBChart con predicciones por velocidad.

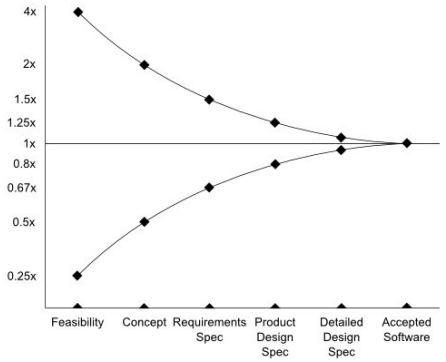


7.4.3 Informes y predicciones

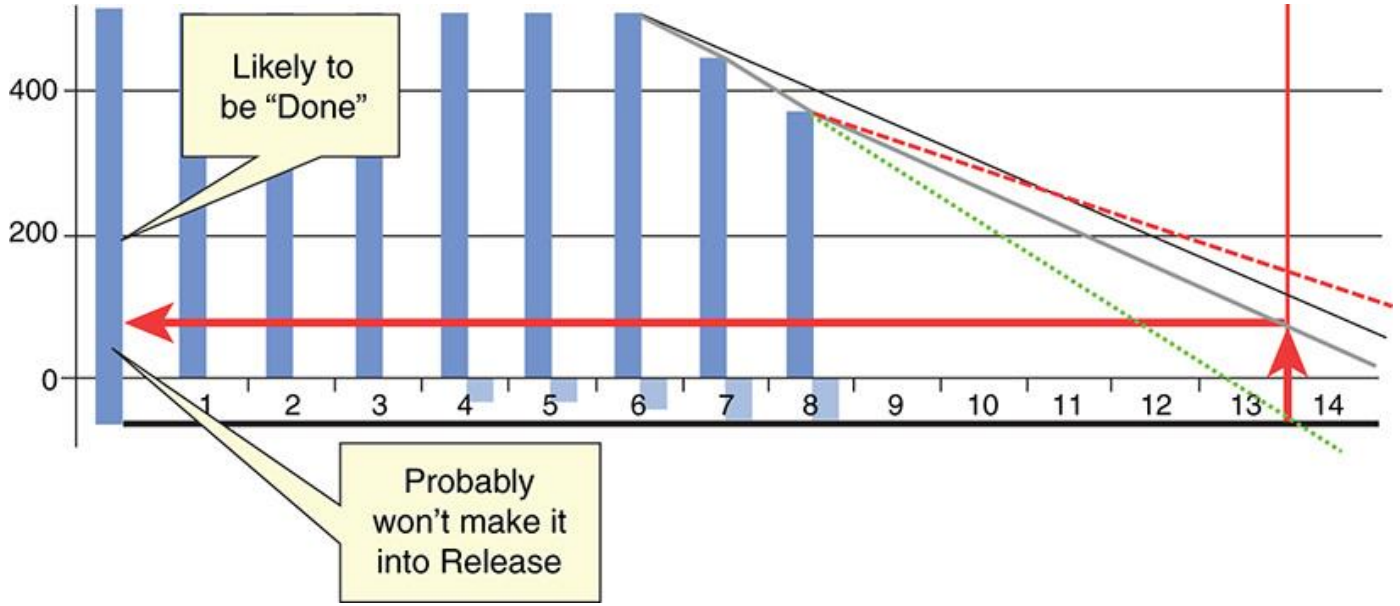
- El incremento del tamaño (sprints 4 a 8) del Product Backlog puede deberse a:
  - Nueva funcionalidad (scope creep) pedida por el cliente.
  - Reestimación de PBIs tras un mejor entendimiento y refinamiento.

- Líneas de regresión por velocidad:
  - Velocidad calculada utilizando todos los sprints
  - Velocidad calculada utilizando los mejores sprints (en la figura los 3 mejores).
  - Velocidad calculada utilizando los peores sprints (en la figura los 3 peores).





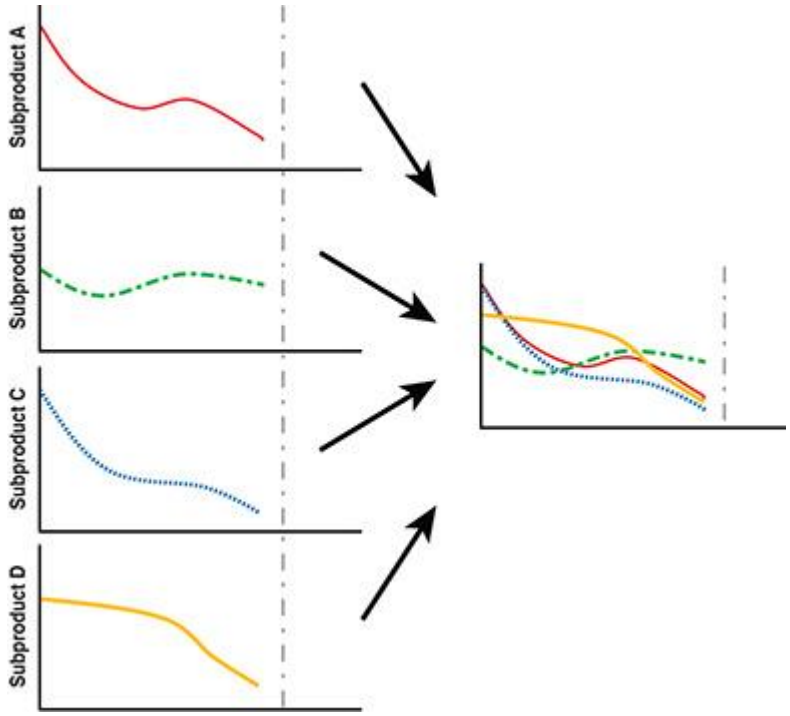
7.4.3 Informes y predicciones



- En el PBCh, la fecha de release es el final del Sprint 13. Vemos que las predicciones normal y pesimista indican que faltará trabajo por realizar. Tenemos 3 opciones:
  - Cambiar la **fecha** de release.
  - Aumentar la **velocidad**: añadir personal el equipo, aumentar el porcentaje de dedicación del equipo a este producto, mejorar la infraestructura y herramientas, escalar con otro equipo.
  - Reducir el alcance pero manteniendo el **MVP**.

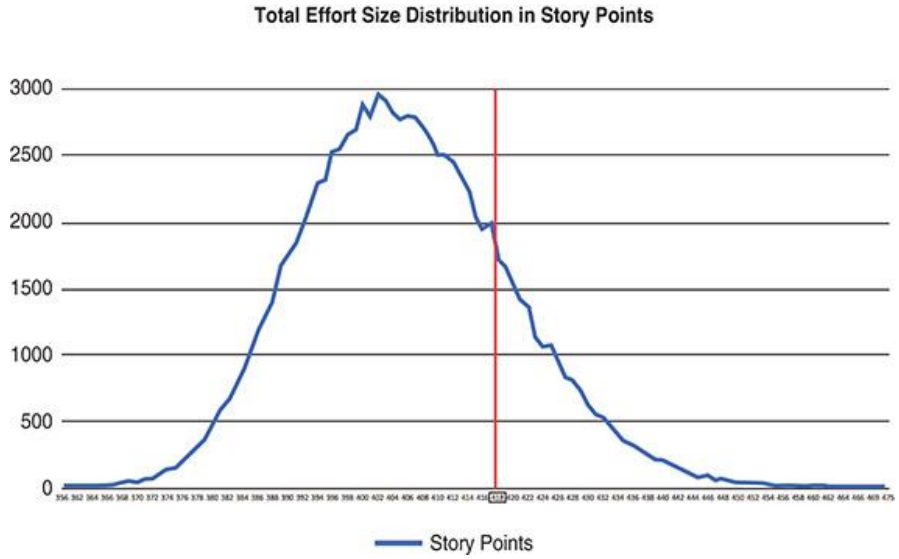
7.4.3 Informes y predicciones

- El CIO o director de programa puede querer comparar cómo van **diferentes proyectos** de desarrollo.
- A la vista de la tendencia de cada uno y la distancia a la fecha de release, pueden decidir traspasar recursos de un proyecto a otro, contratar a alguien más,…



7.4.3 Informes y predicciones

- Si queremos tener un **% de certeza del tamaño de nuestro Product Backlog,** son la simulación **Monte Carlo**:
  - Estimamos tamaño pesimista y optimista a cada PBI.
  - Repetir 10.000 veces:
    - Asignar a cada PBI un nº aleatorio entre el optimista y el pesimista.
    - Sumar el tamaño de todos los PBIs y nos da uno de los probables tamaños totales del PB
  - Hacer gráfico de la distribución de probabilidad a partir de la cantidad de veces que se ha obtenido cada tamaño total.



Hay un 80% de probabilidad
Que el tamaño del PB sea
como mucho 418 SP.

7.4.3 Informes y predicciones

- Estás en el Sprint 12, y quieres hacer una predicción basada en Velocidad para ver cuántos sprints quedan para terminar.

- Tras la última simulación, asumimos que nos queda por quemar un PB=418 SP, y el Equipo de Desarrollo tiene en los últimos 8 sprints Velocidad= 33 SP, habiendo sido la peor 29 y la mejor 37.

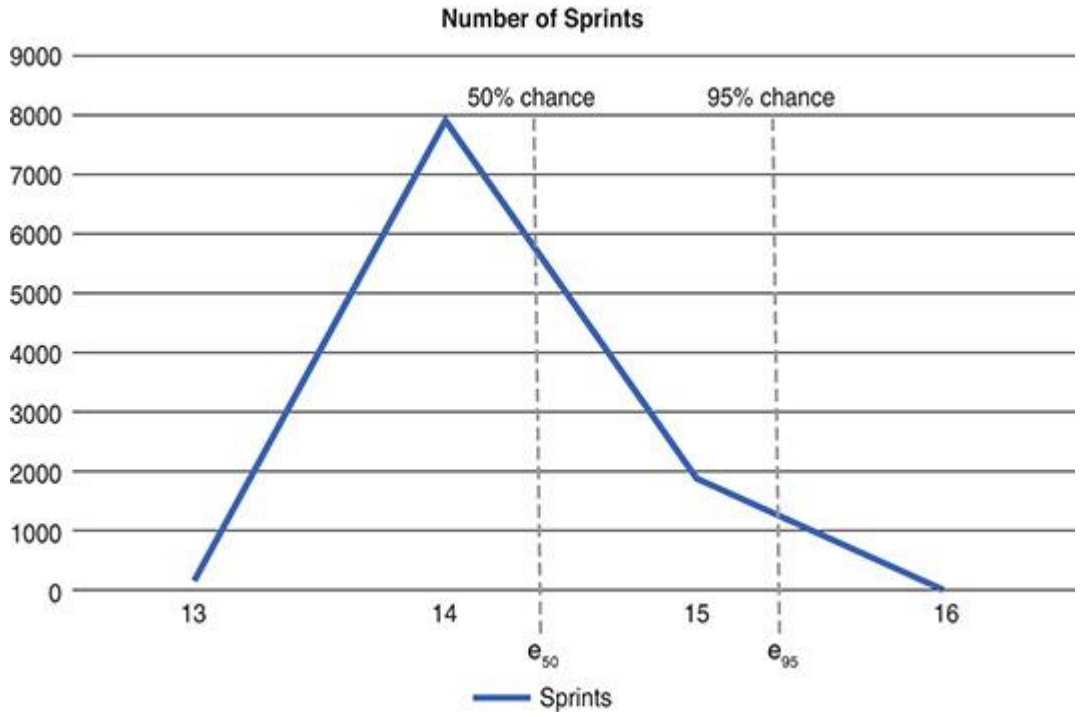
- Sabemos que:

Sprints que faltan = Tamaño que nos queda en el PB / Velocidad

- En vez de usar la velocidad media (33) , vamos a usar Monte Carlo para **simular** 10,000 **velocidades** entre la peor (29) y la mejor (37). Para cada velocidad obtenida, se calcula el nº de sprints necesarios para reducir el PB a 0.

7.4.3 Informes y predicciones

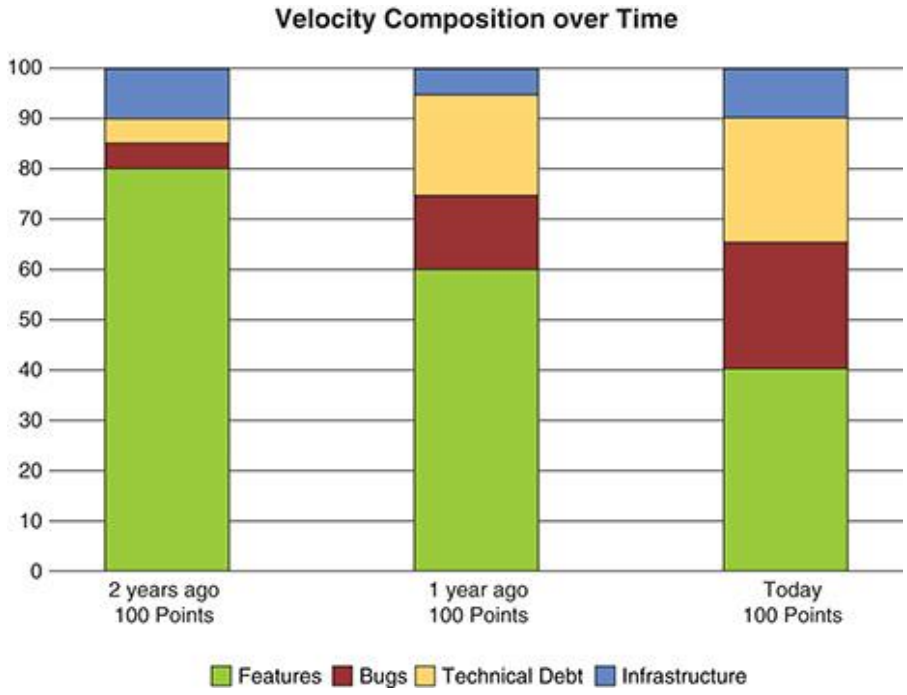
- Tras la simulación, vemos que hay un 90% de probabilidad de necesitar llegar, como mucho, al final del sprint 14.



7.4.3 Informes y predicciones

- Podemos realizar gráficos de qué % hemos dedicado a cada tipo de trabajo (características, bugs, deuda técnica, infraestructura) para un mismo producto.

- Además de como seguimiento, podemos comparar la tendencia (no los números) del equipo.



7.1 Introducción

7.2 Gestión del Product Backlog

- 7.2.1 Tipos de PBI
- 7.2.2 Criterios de Aceptación
- 7.2.3 Ordenar el Product Backlog
- 7.2.4 DoD

7.3 User Story Mapping

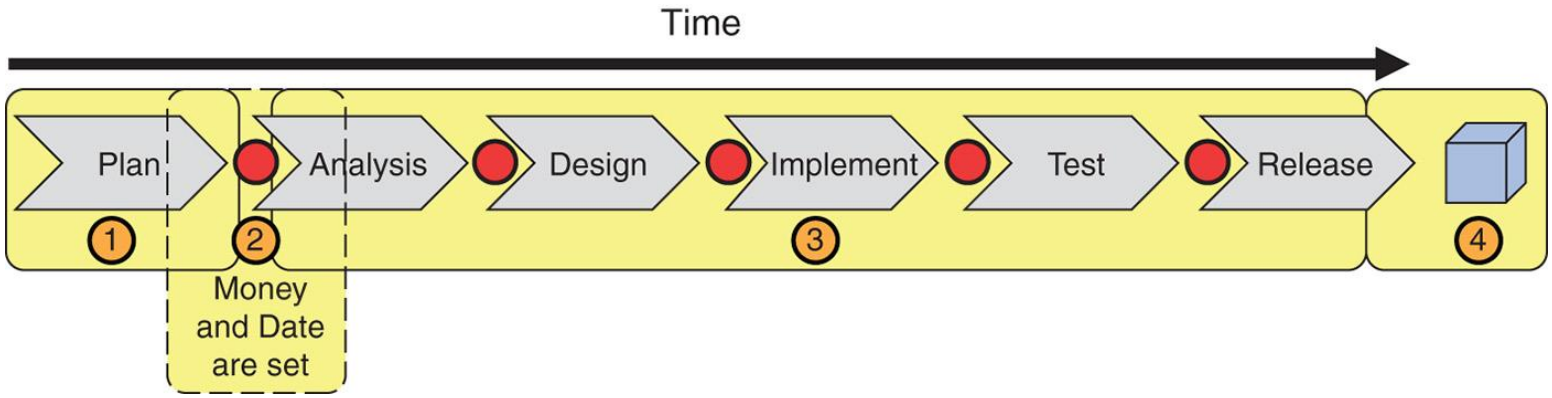
7.4 Gestión de releases

- 7.4.1 Tipos de release
- 7.4.2 Velocidad y Throughput
- 7.4.3 Informes y predicciones

7.5 Presupuesto ágil

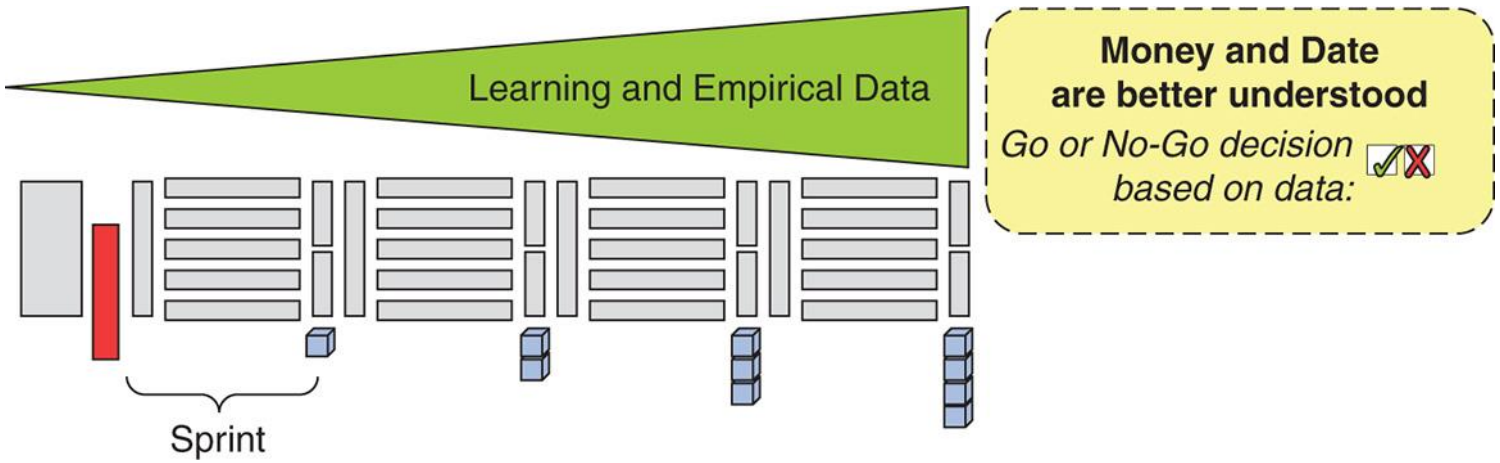
7.4.4 Presupuesto ágil

- Cuando la gestión se realiza de forma **plan-driven**, el plan debe aprobarse antes de comenzar a trabajar y se asigna todo el dinero para el proyecto antes incluso que haya comenzado el desarrollo.



7.4.4 Presupuesto ágil

- Una organización ágil tomará la decisión tras ver el resultado que el equipo de desarrollo obtiene tras unos pocos sprints de prueba, habiendo:
  - Comprobado los primeros riesgos.
  - Ver si es posible cumplir con las características más importantes.
  - Idealmente desplegado, para calcular valor real.
- En vista a los resultados de esa pequeña inversión (Coste Equipo Desarrollo x Sprints), se decide si seguir adelante o no.



7.4.4 Presupuesto ágil

Sobre la guía de presupuesto ágil **FEED-ME,** son importantes los pasos 4 y 6:

1. Financiar productos en lugar de proyectos.
2. Empoderar al Product Owner para tratar el alcance y cronograma.
3. Establecer transparencia, medir constantemente el avance.
4. **Demostrar qué valor se obtiene**, desplegando lo más frecuentemente mposible.
5. Manejar las expectativas de los interesados.
6. **Empirismo para el presupuesto**: éste variará a partir de las evidencias recogidas validando nuestros incrementos.

7.4.4 Presupuesto ágil

- Como Jefe de Proyecto ágil o como Product Owner, puedes encontrarte en 3 casos respecto al presupuesto:

a) Tienes autoridad sobre el presupuesto

1. Construye el Product Backlog.
2. Piensa una velocidad probable de tu equipo.
3. Determina el #Sprints necesarios.
4. Presupuesto = #Sprints x coste por Sprint

b) No tienes autoridad sobre el presupuesto

1. Construye el Product Backlog.
2. Piensa una velocidad probable de tu equipo.
3. Determina cuántos sprints puedes llegar a hacer con el presupuesto actual y piensa qué dejar sin hacer del Product Backlog.

- En ambos casos, conforme despliegues las funcionalidades desarrolladas, es posible que sea más sencillo conseguir más presupuesto.

7.4.4 Presupuesto ágil

c) No tienes autoridad sobre el presupuesto ni sobre el alcance (limitación de ágil)

1. Comunica que el coste será mayor para gestionar los riesgos.
2. Construye el Product Backlog.
3. Piensa una velocidad probable de tu equipo.
4. Determina el #Sprints necesarios.
5. Incrementa el coste por sprint con el coste de 1 o 2 miembros del equipo de desarrollo, como mitigación de riesgos.
6. Presupuesto = #Sprints x coste por Sprint
7. Conforme despliegues funcionalidad, el cliente querrá cambios:
    1. Acéptalos si hay capacidad o si se puede intercambiar por otros PBIs.
    2. Aumenta la predicción de coste en cualquier otro caso, para consideración del comité de cambios.

Conclusiones

- El PO es el nuevo gestor de proyecto ágil, aunque no comparta todas las funciones de un PM clásico.
- A partir del Product Backlog, es posible crear informes de progreso y estimaciones de manera clara.
- No se debe confundir las métricas de progreso con las de valor (las métricas de valor se ven en postgrado, está fuera del alcance de esta asignatura).
- Cuando más nos fijan de antemano el presupuesto y alcance, menos ágil será nuestra gestión.

Conclusiones