Desarrollo de Software I

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación

Mauricio Gaona
mauricio.gaona@correounivalle.edu.co
Profesor

2022-II

Desarrollo de Software I

01 **RESUMEN**
Aspectos generales vistos en la clase anterior.

02 **METODOLOGÍA ÁGIL XP**
Prácticas ágiles en la metodología XP y criterios de aceptación.

03 **COMO INICIAR UN PROYECTO DE SOFTWARE**
Sprint Cero.

04 **EXPOSICIONES**
Definir equipos para el Proyecto y exposiciones

Resumen

Desarrollo de software ágil

Resumen

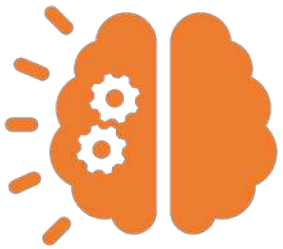
Ágil es un término que se usa para describir los enfoques iterativos del desarrollo de software que enfatizan en la entrega incremental, la colaboración en equipo, la planificación continua y el aprendizaje continuo, en lugar de intentar entregarlo todo de una vez cerca del final.

Desarrollo de software ágil

Resumen

Ágil es una forma de pensar y de hacer las cosas y un comportamiento durante un proceso de desarrollo

Ágil es una mentalidad

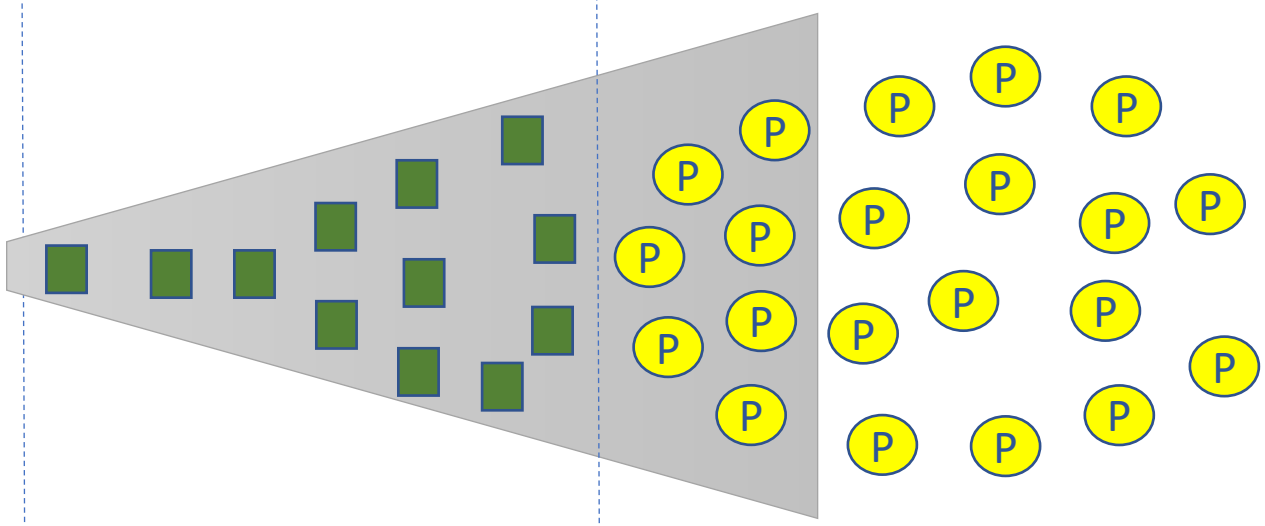


Descrita por 4 valores

Definido por 12 principios

Manifestada a través de un

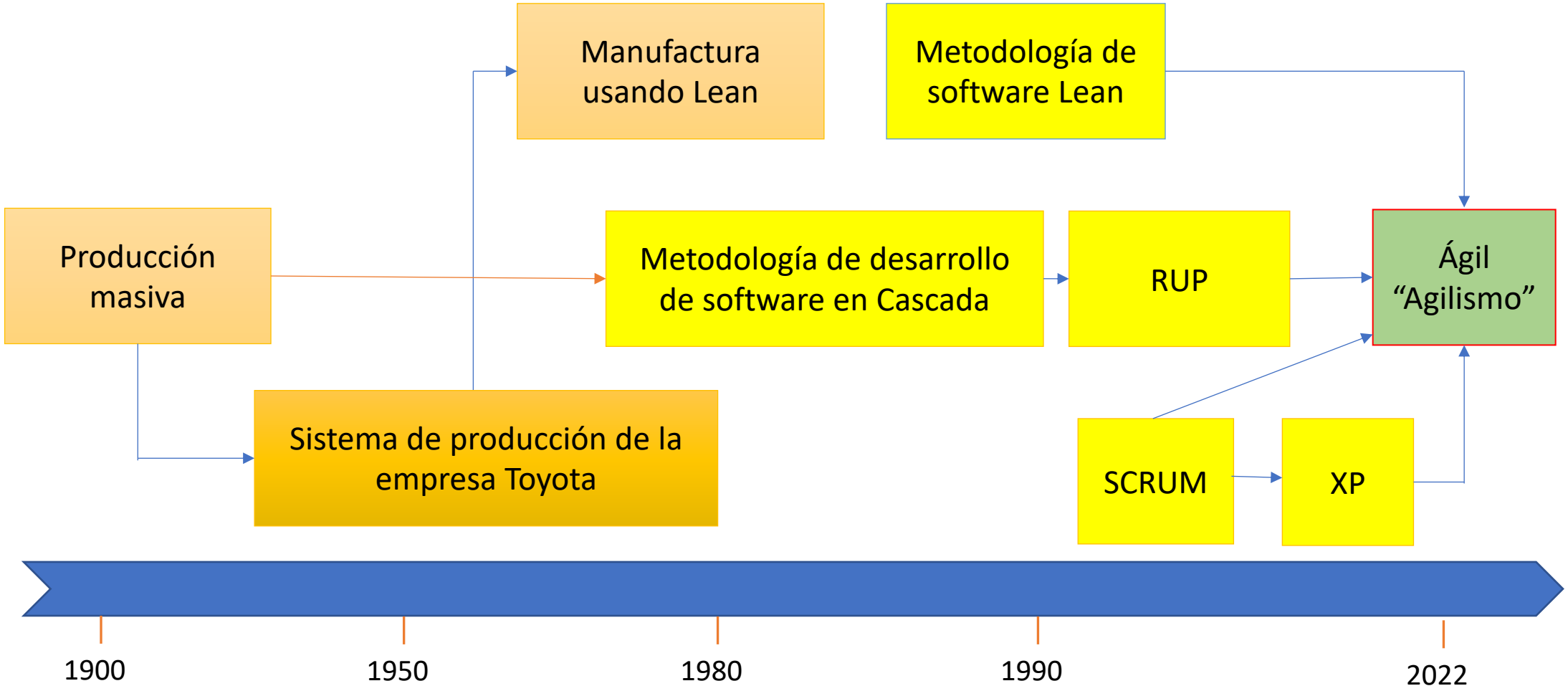
Número ilimitado de prácticas



Agilismo
Es una forma de pensar

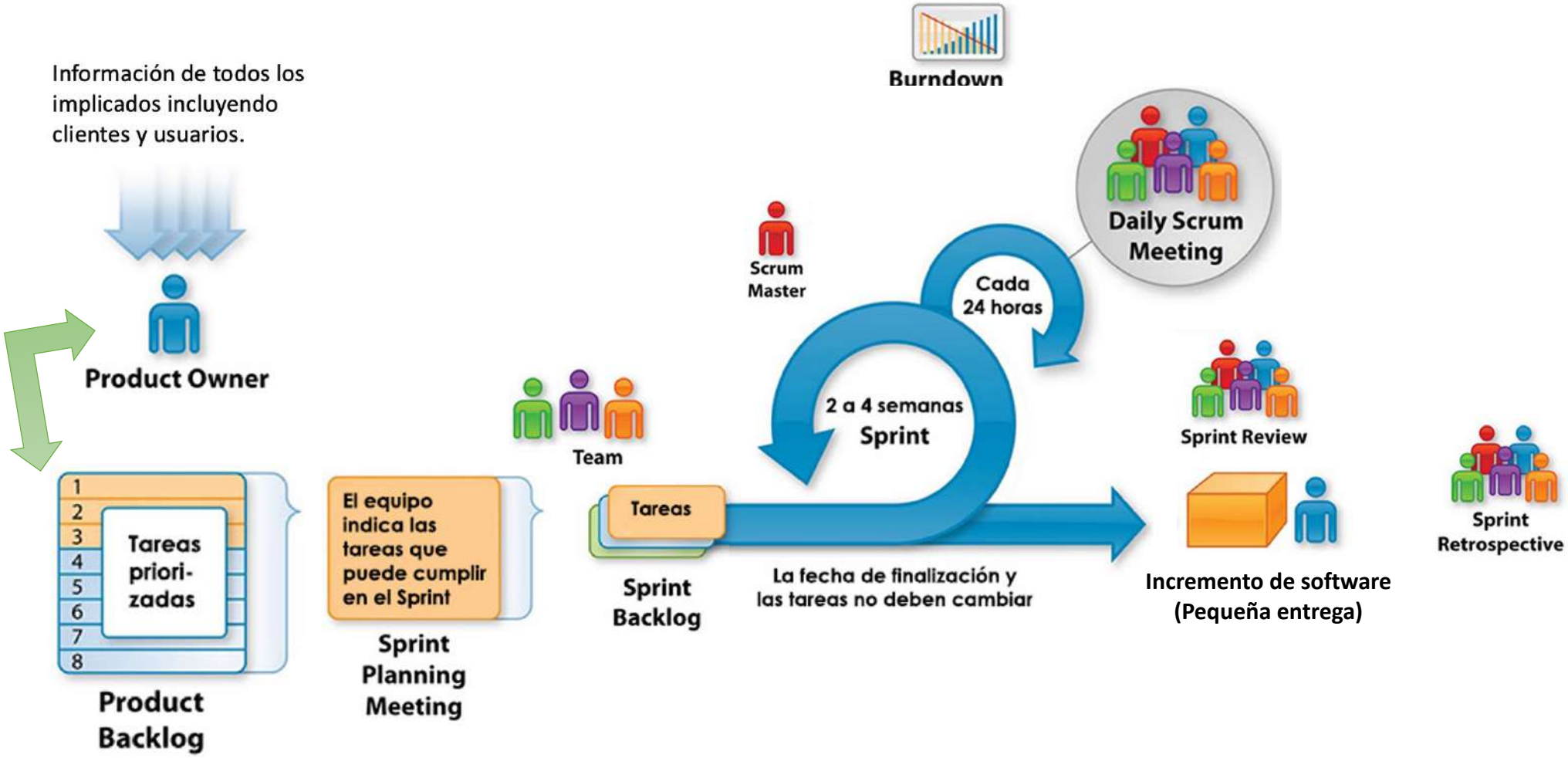
P= práctica ágil

Historia de las metodologías ágiles



Conceptos

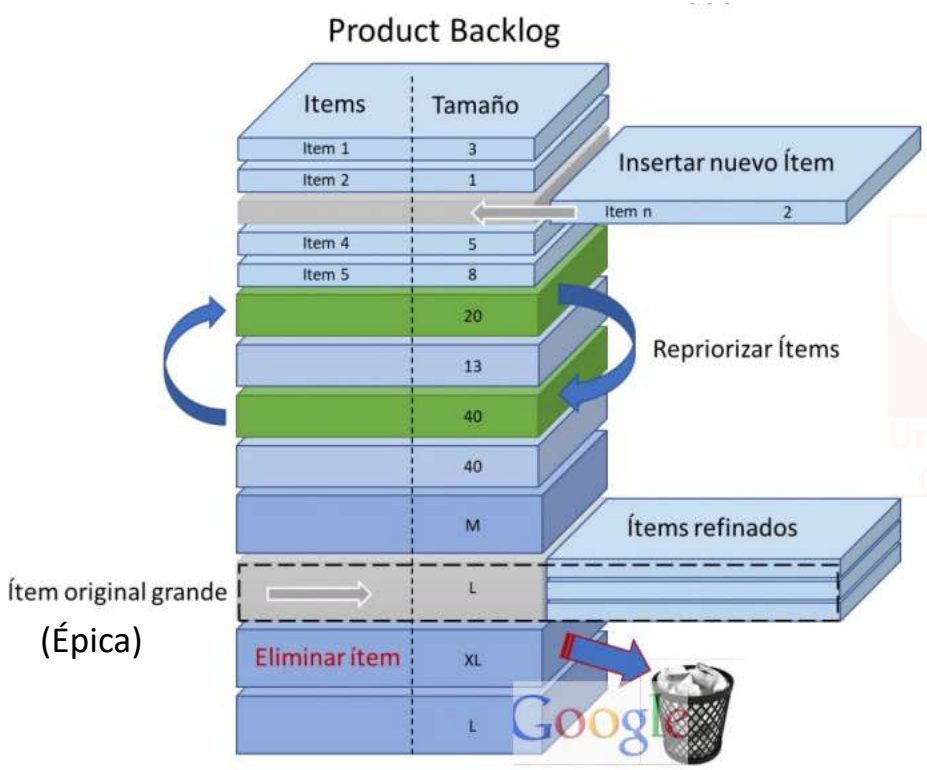
Metodología ágil Scrum



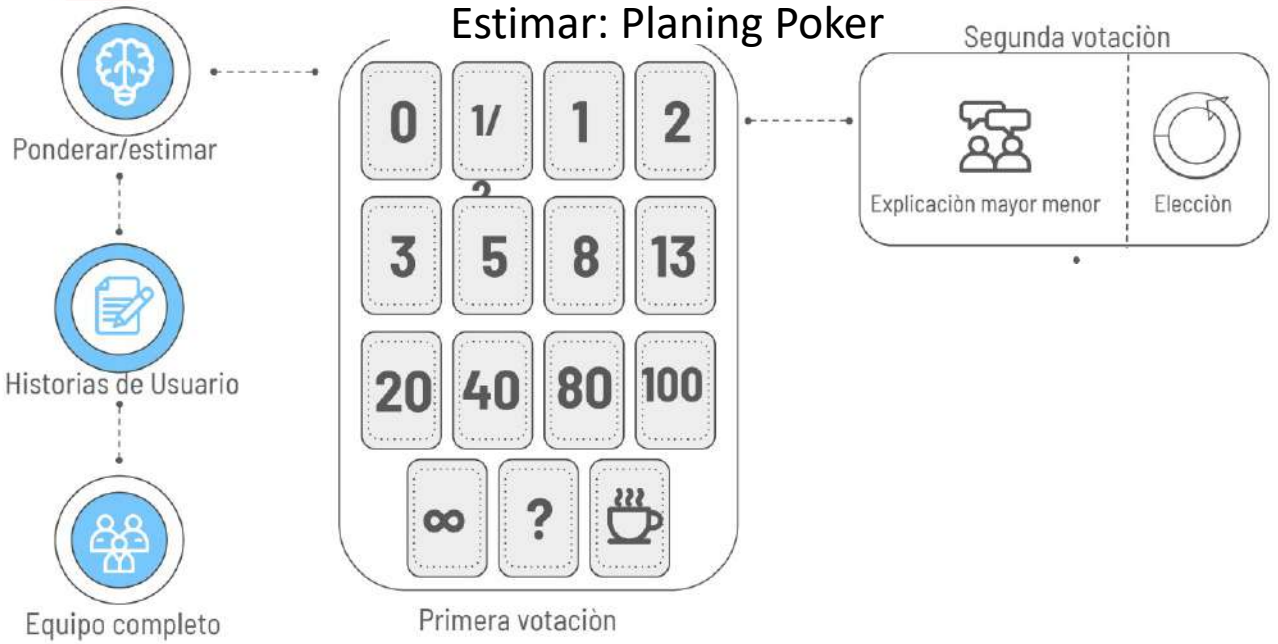
Product Backlog

Características

- El Backlog es un documento “dinámico”.
- Los ítems del Backlog deben agregar siempre valor.
- Los ítems se deben priorizar.
- Todos los ítems deben estimarse.



PBI: Product Backlog Items



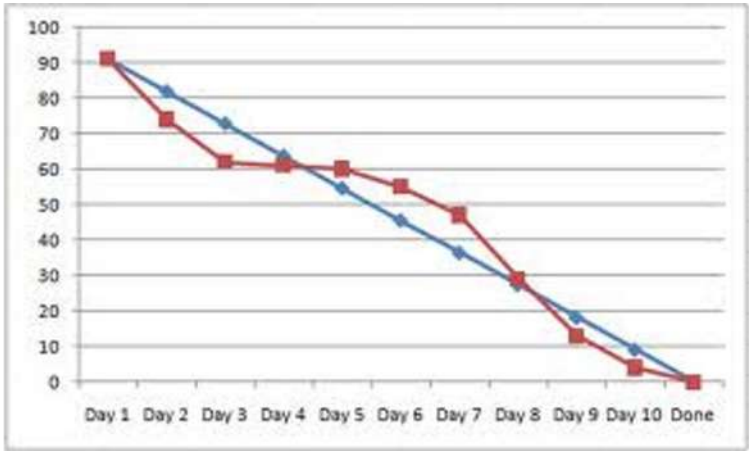
REUNIONES EN SCRUM

Reuniones o ceremonias que se realizan en un Sprint

PRÁCTICAS ÁGILES EN SCRUM

BurnDown chart

Gráficos de trabajo pendiente (**Burndown charts**)
Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los objetivos/requisitos. Permite extrapolar si el Equipo podrá completar el trabajo en el tiempo estimado.

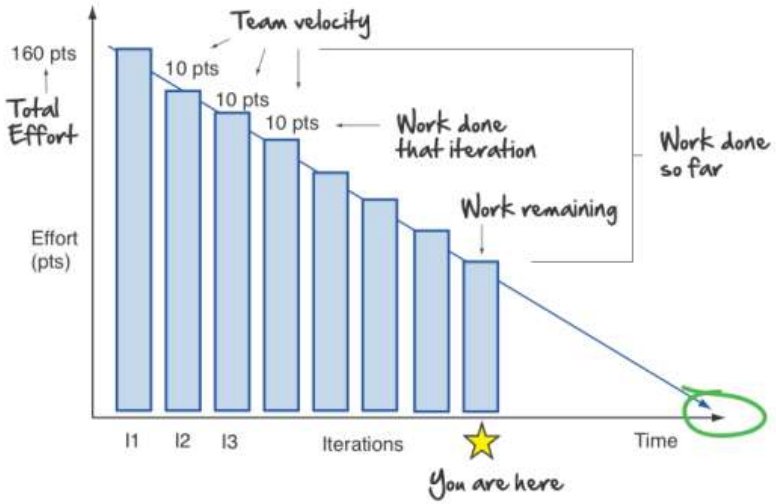


Recuerda cuantas HU quedan pendientes

Se pueden utilizan los siguientes gráficos de esfuerzo pendiente:

- Días pendientes para completar las HU del producto o proyecto (burndown chart), realizado a partir de la lista de HU priorizada (Product Backlog).

- Puntos pendientes para completar las tareas de la iteración (sprint burndown chart), realizado a partir de la lista de tareas de la iteración.



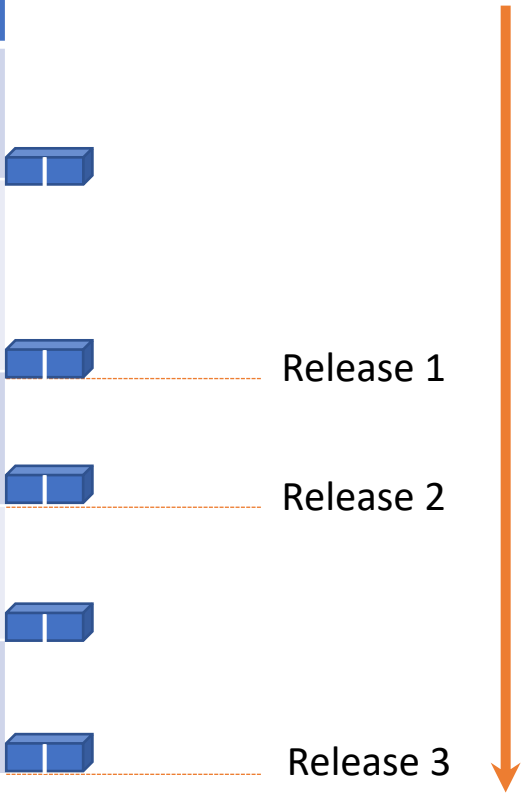
PRÁCTICAS ÁGILES EN SCRUM

Release Plan

Velocidad

Sprint	HU	Puntos de HU		Desarrollador
1	HU1 HU2	3 4	7	Pedro Juan
2	HU3 HU4 HU6	2 4 2	8	Maria Pedro/Ricardo Juan
3	HU5 HU7	4 3	7	Maria Juan
4	HU9 HU8	3 3	6	Pedro Juan
5	HU10 HU11	4 3	7	Maria Pedro

Road Map



Velocidad: Cantidad de puntos que un equipo es capaz de hacer en un sprint

Preguntas ?

Metodología Ágil XP

Actividad de evaluación

Metodología Ágil XP

«Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar.» **Kent Beck**.

¿Qué es XP?

•La Programación Extrema es una metodología ágil de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado.

Metodología Ágil XP

- XP surgió como respuesta y posible solución a los problemas derivados del cambio en los requerimientos
- XP es orientada principalmente a la codificación
- Programación basada en los “deseos” del cliente.
- Los participantes en un proyecto lo conforman los jefes de proyecto, desarrolladores, el cliente y otros (stakeholders)
- Se rige por valores y principios definidos en el manifiesto ágil

Metodología Ágil XP

Actividades

Codificación: La parte mas importante de XP.

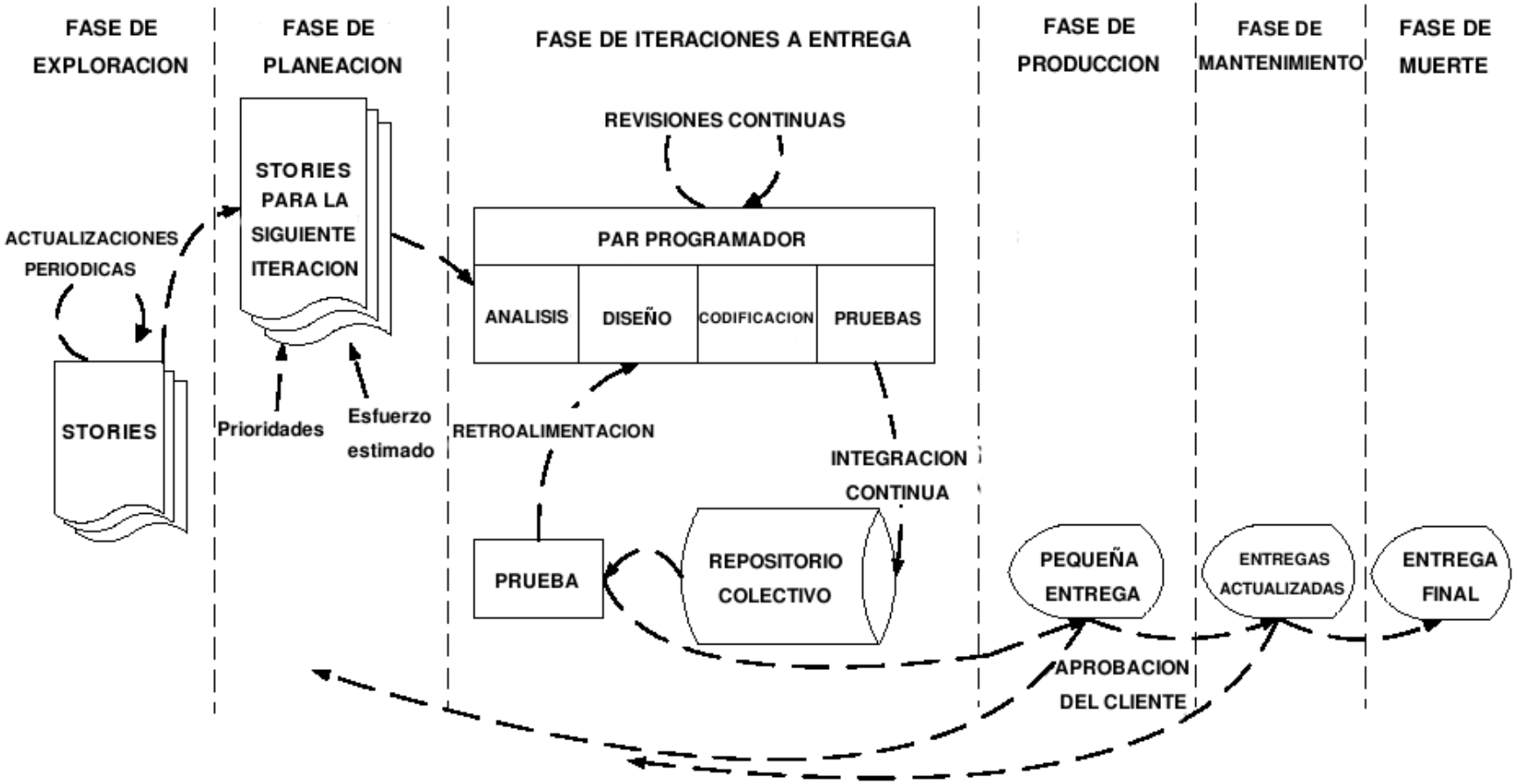
Pruebas: Nunca se puede estar seguro de algo hasta haberlo probado.

Escuchar: Escuchar los requisitos del cliente acerca del sistema a crear.

Diseño: Crear una estructura del diseño para evitar problemas.

Metodología Ágil XP

Fases de la metodología e**X**treme **P**rogramming propuesta por Kent Beck



Metodología Ágil XP

XP Extreme Programming

Se fundamenta en 12 practicas

El proceso de planificación
Define características que se incluirán en la versión siguiente combinando prioridades de negocio definidas con el cliente con estimaciones técnicas de los programadores.

Los pequeños "releases"
Un sistema simple en producción temprana, se actualiza frecuentemente en un ciclo corto

Metáfora
El sistema se define como una metáfora o conjunto de metáforas una historia compartida por clientes, managers y programadores que orienta todo el sistema, describiendo como funciona.

Diseño simple
Diseñar la solución más simple susceptible de implementarse en el momento. No implementar nada que no se necesite ahora. (Nivel de complejidad)

Recodificación
El diseño se mejora en todo el ciclo, manteniendo el software limpio, sin duplicación, simple y completo. Si funciona bien arréglelo de todos modos.

Programación en pares
La programación se hace entre dos personas (Nivelación técnica)

Metodología Ágil XP

XP Extreme Programming

Propiedad colectiva

Todo el código pertenece al grupo. Cualquiera puede cambiar cualquier parte del código en cualquier momento, siempre que escriba antes la prueba correspondiente.

Integración continua

Cada pieza se integra a la base de código apenas está lista, varias veces al día. Debe correrse la prueba antes y después de la integración. Hay un repositorio (solamente) dedicada a este propósito.

40-horas semana

Se evitan los sobretiempos excesivos y los equipos cansados. Evitar los "héroes" y eliminar el "proceso neurótico" (Salario emocional, amor a la compañía)

Cliente en sitio

El cliente debe estar presente y disponible a tiempo completo para el equipo. (Línea directa)

Estándar de codificación

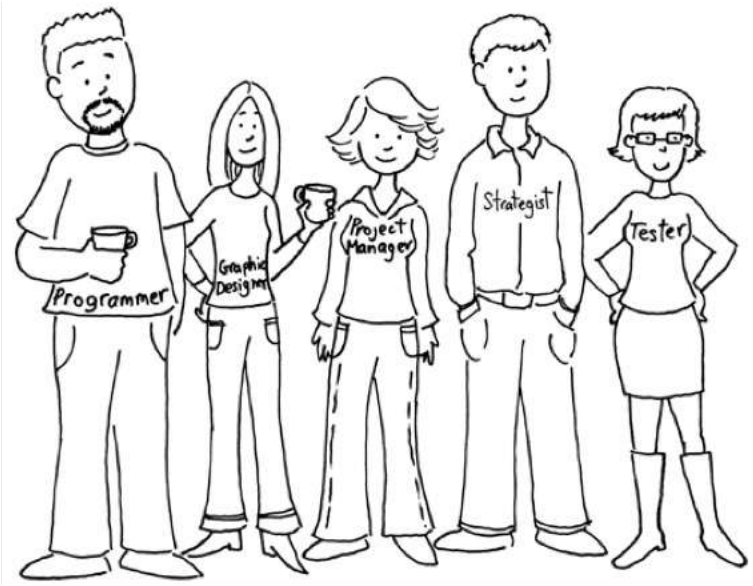
Para compartir el código la codificación debe ser similar

Prueba

Desarrollo orientado a las pruebas UT, TDD.

Metodología Ágil XP

Roles en XP



Prácticas ágiles en XP

Plan de entregas "Release Plan"

Es donde se definen las iteraciones que tendrá el proyecto, las HU a desarrollar por iteración y se acuerdan las entregas a realizar

Iteración	Historia de Usuario	Puntos de Historia		Fecha (inicio: Agosto 2-2020)	Par Programador
Iteración 1	HU1 HU2	3 4	7	** **	Juan-Maria Dani-Pedro
Iteración 2	HU3 HU4 HU6	2 4 2	8	** ** **	Juan-Maria Sofia-Juliana Dani-Pedro
Iteración 3	HU5 HU7	4 3	7	** **	Sofia-Juliana Juan-Maria

Nota: Si una HU es muy grande (EJ: 6 puntos de historia), sería más adecuado dividirla en dos historias (EJ: una de 4 y otra de 2) y distribuirlas en las iteraciones.
La velocidad de desarrollo debería ser similar para cada iteración.
** : Se calcula a partir de la fecha de inicio y depende de la experiencia del equipo de desarrollo (horas por punto)
Nota: El plan de entregas puede cambiar en cada iteración

Prácticas ágiles en XP

Velocidad del proyecto:

La velocidad del proyecto es una medida que representa la rapidez con la que se desarrolla el proyecto y se mide en puntos.

Para estimarla la velocidad de un proyecto se requiere sumar el número de puntos de cada historia de usuario que se determina implementar en una iteración.

Usando la velocidad del proyecto controlaremos que todas las tareas se puedan desarrollar en el tiempo del que dispone la iteración.

Prácticas ágiles en XP

Programación en pareja:

La metodología XP aconseja la programación en parejas pues incrementa la productividad y la calidad del software desarrollado.

El trabajo en pareja involucra a dos programadores trabajando en el mismo equipo; mientras uno codifica haciendo hincapié en la calidad de la función o método que está implementando, el otro analiza si ese método es adecuado y si está bien diseñado.

ROLES

Controlador / Piloto

Programador encargado de escribir el código.

Navegador /Copiloto

Programador encargado de supervisar y debatir entorno al código que se está escribiendo.

Intercambio Regular entre pares.

VENTAJAS

I. Los estudios han demostrado que después de entrenar para las "***habilidades sociales***" implicadas, parejas de programadores son más de **dos veces productivos** que uno para una tarea dada. Según *The Economist*.

II. *"**Laurie Williams** de la universidad de Utah en Salt Lake City ha demostrado que los programadores emparejados son solamente **15% más lentos** de dos programadores trabajando independientemente, **pero producen 15% menos errores**. Y ya que la prueba y depuración son a menudo muchas veces más costosa que la programación inicial, esto es un **resultado impresionante**"*

- ✓ Moral Mejorada.
- ✓ Propiedad colectiva del código.
- ✓ Método más eficaz de capacitación.

DESVENTAJAS

- ❖ Aumento de los costes.

Prácticas ágiles en XP

Reuniones diarias.

Es necesario que los desarrolladores se reúnan diariamente y expongan sus problemas, soluciones e ideas de forma conjunta.

Las reuniones tienen que ser fluidas y todo el mundo tiene que tener voz y voto.

- 15 Minutos
- De pie
- Sin café

Prácticas ágiles en XP

..... Diseños simples: La metodología XP sugiere que hay que conseguir diseños simples y sencillos. Hay que procurar hacerlo todo lo menos complicado posible para conseguir un diseño fácilmente entendible e impleméntable que a la larga costará menos tiempo y esfuerzo desarrollar.

..... Glosarios de términos: Usar glosarios de términos y un correcta especificación de los nombres de métodos y clases ayudará a comprender el diseño y facilitará sus posteriores ampliaciones y la reutilización del código.

..... Riesgos: Si surgen problemas potenciales durante el diseño, XP sugiere utilizar una pareja de desarrolladores para que investiguen y reduzcan al máximo el riesgo que supone ese problema (**spyke**).

Prácticas ágiles en XP

Funcionalidad extra

Nunca se debe añadir funcionalidad extra al sistema aunque se piense que en un futuro será utilizada. Típicamente sólo el 10% de la misma es utilizada, lo que implica que el desarrollo de funcionalidad extra es un desperdicio de tiempo y recursos.

De las funcionalidades desarrolladas:

7% se usan "siempre"

13% se usan "a menudo"

16% "a veces"

19% "pocas veces"

45% "NUNCA"

Fuente: Standish Group

Refactorizar

Refactorizar es mejorar y modificar la estructura y codificación (código) ya creado sin alterar su funcionalidad. Refactorizar supone revisar de nuevo el código para procurar optimizar su funcionamiento. Es muy común rehusar código ya creado que contienen funcionalidades probadas.

Prácticas ágiles en XP

Codificación.

.....La codificación debe hacerse ateniendo estándares de codificación ya creados. Programar bajo estándares mantiene el código consistente para facilitar su comprensión y escalabilidad.
https://django-best-practices.readthedocs.io/en/latest/code.html
https://www.geeksforgeeks.org/best-practices-for-professional-developer-django-framework/

.....La optimización del código siempre se debe dejar para el final. Hay que hacer que funcione y que sea correcto, más tarde se puede optimizar.
https://docs.djangoproject.com/en/4.1/topics/performance/

Prácticas ágiles en XP

Pruebas

Hacer pruebas valida el funcionamiento de la aplicación.

Pruebas Unitarias : es un método que prueba una unidad estructural de código.
Pruebas de aceptación: Se especifica por medio de criterios de aceptación.

1

Pruebas | CRITERIOS DE ACEPTACIÓN

Técnica: **Acción Reacción**

2

Pruebas | CRITERIOS DE ACEPTACIÓN

Técnica: Escenarios

Criterios de aceptación

Características de los Criterios de Aceptación

Para medir la calidad de un criterio de aceptación se utiliza el método SMART en el que se han de cumplir en lo máximo posible los siguientes criterios:

S - Specific (Específicas)
M - Measurable (Medibles)
A - Achievable (Alcanzables)
R - Relevant (Relevantes)
T - Time-boxed (Limitados en el tiempo)

Criterios de aceptación

Pruebas: Técnica acción/reacción

Son las condiciones que una historia de usuario debe satisfacer para ser aceptada por un usuario, cliente o stakeholder.

Si todas las HU son aceptadas el producto de software es aceptado.

Se presentan como un conjunto de sentencias redactadas de tal manera que conduzcan a una respuesta clara de “aceptado/rechazado”

Un criterio de Aceptación debe ser escrito en formato de Pruebas de Aceptación.

Prácticas ágiles en XP

Pruebas: Técnica acción reacción

Dada una condición, cuando ocurre un evento o acción, entonces sucederá una consecuencia.
Criterio de aceptación expresado en términos funcionales (Acción/Reacción) de uso del sistema.

Item	Criterio de acetantación
1	Un usuario no puede enviar un formulario sin completar todos los campos.
2	El login no puede ser igual al password.
3	El password debe tener al menos 6 caracteres, una letra en mayúscula, un símbolo y un número.
4	Los campos de entrada de datos se deben validar antes de ejecutarlos o guardarlos.

Prueba	Login	Password	Resultado esperado
1			Login o password vacíos
2	123456789	123456789	Login o password incorrecto
3	usuario	Secret()22	Login exitoso
4	prueba	SELECT * FROM Users WHERE Username='1' OR '1' = '1' AND Password='1' OR '1' = '1'	Login o password incorrecto

Acción

Reacción

Criterios de aceptación

Técnica de escenarios:

- Restringir la forma en que el usuario procedería y el sistema correspondería,
- Elimina toda la información innecesaria.
- Tiene la ventaja de que permite al equipo de desarrollo ir dando por hechas las funcionalidades y los casos que de ella se origina mientras las van implementando.
- Un test de aceptación puede comprobar más de un criterio de aceptación.
- Permiten usar más de un usuario para describir el escenario del criterio.
- Su mayor beneficio es que los criterios de aceptación pueden ser trasladados directamente de la historia de usuario a un test de aceptación automático.

Criterios de aceptación: técnica de escenarios

Usa la sintaxis de **Gherkin** creada específicamente para las descripciones de comportamiento de un software.

La sintaxis de gherkin es la siguiente:

```
Scenario: Some determinable business situation
  Given some precondition
    And some other precondition
   When some action by the actor
    And some other action
    And yet another action
   Then some testable outcome is achieved
    And something else we can check happens too
```

Criterios de aceptación: técnica de escenarios

(Scenario) Escenario [Número de escenario] [Titulo del escenario]:
(Given) Dado que [Contexto] **y** adicionalmente [Contexto]
(When) cuando [Evento] **y** [evento]
(Then) entonces el sistema [Resultado / Comportamiento esperado]

Número de escenario: Número (ejemplo 1, 2 o 3 ó CA1_HU1, CA2_HU1), que identifica al escenario asociado a la historia.
Título del escenario: Describe el contexto del escenario que define un comportamiento.
Contexto: Proporciona mayor descripción sobre las condiciones que desencadenan el escenario.
Evento: Representa la acción que el usuario ejecuta, en el contexto definido para el escenario.
Resultado / Comportamiento esperado: Dado el contexto y la acción ejecutada por el usuario, la consecuencia es el comportamiento del sistema en esa situación.

Criterios de aceptación: técnica de escenarios

Ejemplo: **HU:** Como cliente del banco deseo retirar dinero de un cajero automático para evitar ir al banco.

Scenario 1: La cuenta tiene saldo disponible
Given la cuenta tiene saldo disponible
and que la tarjeta es válida
and que el cajero tiene dinero disponible
When el cliente solicita dinero
Then la cuenta es debitada
and el dinero es entregado al cliente
and el cliente recupera su tarjeta

Escenario 1: La cuenta tiene saldo disponible
Dado que la cuenta tiene saldo disponible
y que la tarjeta es válida
y que el cajero tiene dinero disponible
Cuando el cliente solicita dinero
Entonces la cuenta es debitada
y el dinero es entregado al cliente
y el cliente recupera su tarjeta

Criterios de aceptación: técnica de escenarios

Ejemplo: **HU:** Como cliente del banco deseo retirar dinero de un cajero automático para evitar ir al banco.

Scenario 2: El retiro excede el saldo disponible en el banco
Given el retiro excede el saldo disponible en el banco
and que la tarjeta es válida
and que el cajero tiene dinero disponible
When el cliente solicita dinero
Then el cajero muestra un mensaje negando el retiro
and el dinero no es entregado al cliente
and el cliente recupera su tarjeta

Criterios de aceptación: técnica de escenarios

Ejemplo: **HU:** Como operador de oficina quiero ingresar al sistema para trabajar en mis actividades

Escenario 1: El usuario operador de oficina desea ingresar al sistema
Dado que el usuario operador tiene una cuanta habilitada en el sistema
y su login es prueba
y su password es tester1
Cuando el usuario operador escribe su login
y el usuario operador escribe su contraseña
Entonces el sistema le muestra un mensaje que dice “Login exitoso”
y lo redirige a la pantalla de inicial para el rol de operador

https://cucumber.io/docs/guides/10-minute-tutorial/

Mensaje final sobre metodologías

Ninguna metodología hará el trabajo por ti,
porque ninguna metodología trabaja sola.

Preguntas ?

Gracias

Universidad del Valle