Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

AGILIDAD Y PROCESO

Sem - 3

Ing. Ena Mirella Cacho Chávez

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

Logro de la unidad

Al concluir la unidad, el estudiante describe los conceptos y principios de la ingeniería del software demostrando dominio del tema con claridad y precisión.

Logro de la sesión

Al finalizar la sesión, el estudiante estará en condiciones de:

- Identificar aspectos generales del desarrollo ágil
- Comparar las metodologías ágiles de software

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Tabla de Contenidos

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

1. EVALUACIÓN DE MATERIAL ASINCRÓNICO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

1.1. Actividad asincrónica

- Antes de la sesión sincrónica, el estudiante debe revisar capítulo 3 (Pressman & Maxim, 2020).

- Visualizar video (Material complementario) introducción

1.2. Evaluación de material asincrónico

- Estimado estudiante,

Luego de revisar el material brindado, responder a la interrogante:
¿Qué es agilidad?

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

2. DESARROLLO DE CONTENIDOS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

2.1. La naturaleza del software

Visualizar un corto video que lo puedes encontrar en “Material Complementario”

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

El desarrollo ágil

- **Adaptabilidad y flexibilidad.**
- **No es aplicable universalmente.**

La **agilidad**

- Permite responder a un entorno empresarial dinámico
- R**educe costos** de cambio a lo largo del proceso de software.

El desarrollo ágil

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Alistair Cockburn destaca en un libro sobre desarrollo ágil que

- Los **modelos prescriptivos** a menudo pasan por alto la **variabilidad y fragilidad** de las personas que construyen software.
- Para que los **modelos** de proceso **funcionen**, deben **ser realistas y tolerantes con la diversidad** de habilidades y estilos de trabajo de los ingenieros de software.

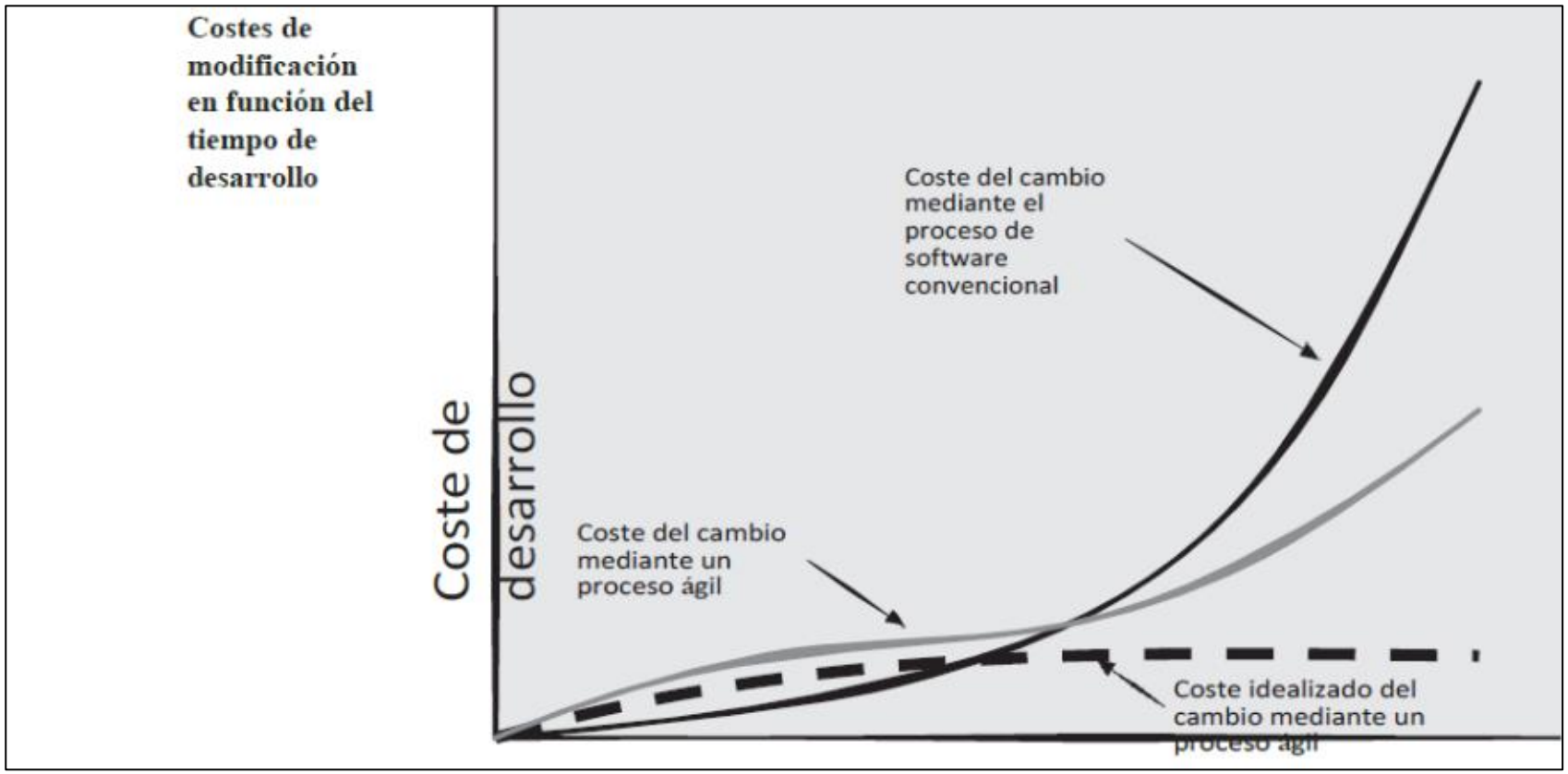
Agilidad

- Capacidad de **adaptarse** rápidamente al cambio constante.
- Promueve estructuras **de equipo** y actitudes que facilitan la **comunicación**
- Destaca la **entrega rápida** de software operativo.
- **Reduce** la importancia de los **productos intermedios**
- **Incluye al cliente** como parte del equipo y
- Busca eliminar la brecha "nosotros y ellos".
- Reconoce las **limitaciones de la planificación** en un entorno incierto y aboga por la **flexibilidad** en los planes de proyecto.
- Se enfoca en **eliminar productos de trabajo no esenciales**
- Entrega incremental para poner rápidamente a disposición del cliente.

Agilidad y el costo del cambio

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- El **costo** del cambio **aumenta no linealmente** a medida que avanza un proyecto.
- Iniciar **cambios** al recopilar requisitos al **principio** es relativamente **fácil**, pero hacerlo en **fases avanzadas**, como durante las pruebas de validación, resulta **costoso y consume tiempo** considerable.
- Los defensores de la agilidad argumentan que un **proceso ágil bien diseñado** puede "**aplanar" la curva** del costo del cambio, permitiendo que un equipo acomode cambios en fases finales del proyecto sin impactos dramáticos en costos y tiempo.
- La **entrega incremental**, junto con prácticas ágiles como pruebas unitarias continuas y programación en parejas, contribuye a **mitigar el costo** de realizar cambios. Aunque la magnitud de la reducción del costo del cambio es objeto de debate, hay evidencia que sugiere una significativa disminución con enfoques ágiles.


Costes de modificación en función del tiempo de desarrollo
Coste de desarrollo
Coste del cambio mediante el proceso de software convencional
Coste del cambio mediante un proceso ágil
Coste idealizado del cambio mediante un proceso ágil

¿Qué es un proceso ágil?

- Un proceso ágil se centra en abordar la **imprevisibilidad** en proyectos de software al reconocer la **dificultad de prever requisitos** y **cambios** en las **prioridades del cliente.**
- Se destaca la **interrelación entre diseño y construcción**, así como la no completa previsibilidad en análisis, diseño, construcción y pruebas.
- La clave para enfrentar esta incertidumbre es la **adaptabilidad** del proceso a la rápida evolución del proyecto y las condiciones técnicas.

1. ¿Qué es un proceso ágil?

- Un proceso ágil debe ser **adaptable e incremental**, incorporando retroalimentación del cliente a través de **entregas frecuentes** de prototipos o partes operativas del sistema.
- La estrategia de desarrollo incremental permite la **evaluación regular** del software entregado, recopilando **comentarios del cliente** y **ajustando el proceso** para incorporar esas observaciones. Este enfoque iterativo facilita la **adaptación continua** y el **progreso efectivo** en un entorno de desarrollo de software imprevisible.

1.1 Principios de agilidad

La Agile Alliance define 12 principios para aquellas organizaciones de software que quieran alcanzar la agilidad. Estos principios se resumen en los párrafos siguientes.

1. Satisfacer al cliente mediante la **entrega temprana y continua** de software con valor.
2. **Cambiar los requisitos**, incluso en etapas avanzadas del desarrollo, para proporcionar ventaja competitiva al cliente.
3. Entregar **software funcional** frecuentemente, con preferencia a períodos cortos.
4. **Colaborar con el cliente** a lo largo del desarrollo y responder a sus necesidades cambiantes.
5. Construir proyectos en torno **a individuos motivados**. Darles el entorno y el apoyo que necesitan, y confiar en ellos para conseguir el trabajo.
6. Utilizar **conversaciones cara a cara** como el medio más eficiente y efectivo de comunicación dentro de un equipo de desarrollo.
7. El **software funcional** es la **medida principal** de progreso.
8. Mantener un ritmo constante de trabajo de **desarrollo sostenible** para los desarrolladores.
9. Lograr la **excelencia técnica** y el buen diseño.
10. **Simplicidad:** la capacidad de maximizar la cantidad de trabajo no realizado es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen **de equipos autoorganizados.**
12. A intervalos regulares, el equipo reflexiona sobre cómo volverse más efectivo, ajusta su comportamiento en consecuencia y luego continúa mejorando.

1.2. La política del desarrollo ágil

- Existe un debate acalorado sobre las ventajas y aplicabilidad de la agilidad en el desarrollo de software frente a enfoques más convencionales. Jim Highsmith humorísticamente describe los extremos de ambas posturas, con **"agilistas" acusando a los tradicionalistas de priorizar documentación sobre sistemas efectivos, y viceversa**. Este debate corre el riesgo de convertirse en una guerra religiosa, donde las creencias, no los hechos, dirigen las decisiones.
- La verdadera cuestión es cómo lograr la agilidad de la mejor manera. Aunque el software funcional es crucial, se debe considerar la calidad en términos de fiabilidad, facilidad de uso y mantenimiento. No hay respuestas absolutas, y dentro de la escuela ágil, existen varios modelos con enfoques sutiles. Aunque algunos conceptos ágiles difieren de la ingeniería tradicional, muchos son adaptaciones positivas. **La conclusión es que se puede obtener lo mejor de ambas escuelas, sin menospreciar ninguna de ellas.**

2. CRUM

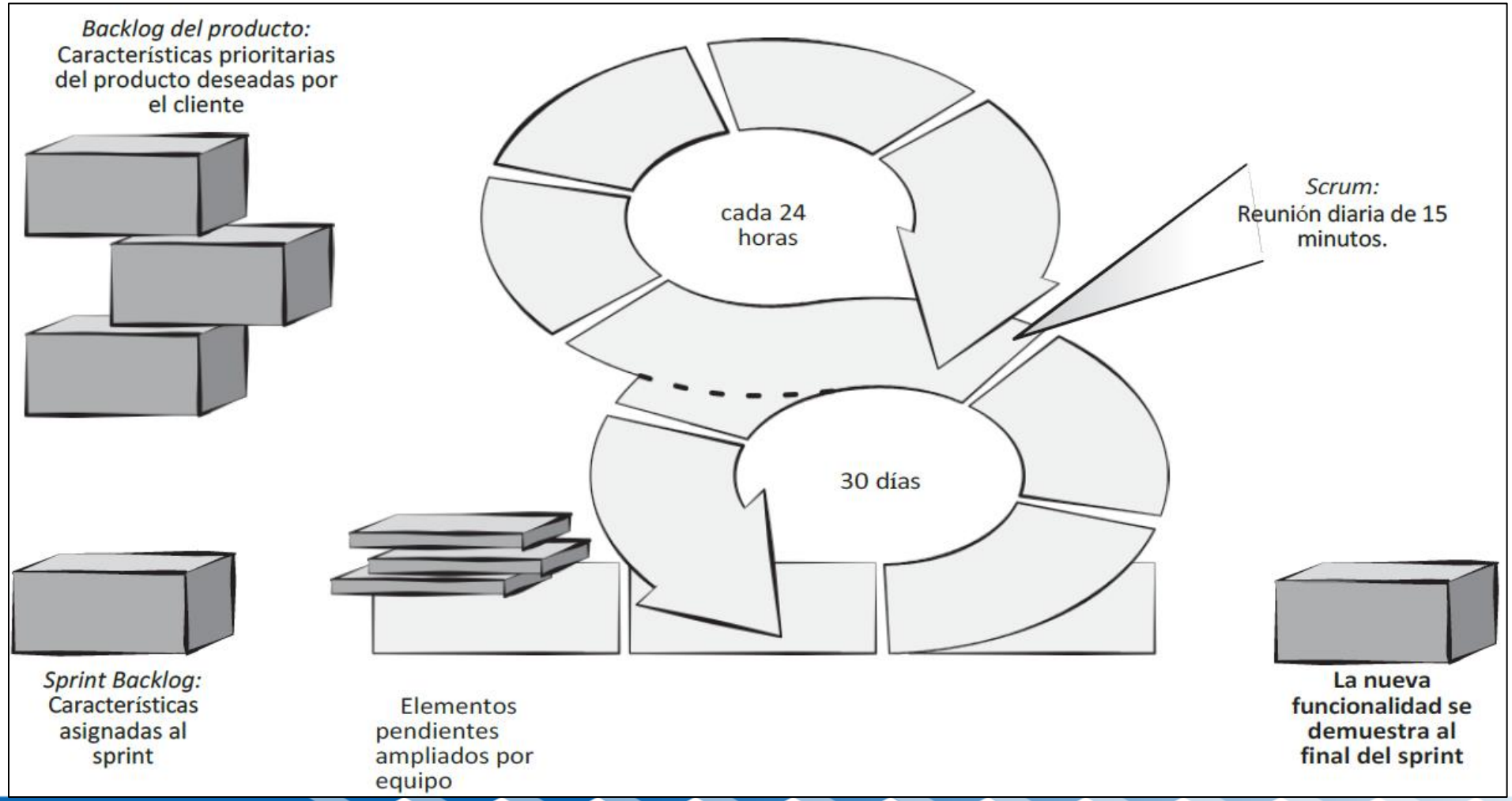
Scrum (el nombre deriva de una actividad que tiene lugar durante un partido de rugby) es un método ágil de desarrollo de software muy popular que fue **concebido por Jeff Sutherland** y su equipo de desarrollo a principios de la década de **1990.**

Schwaber y Beedle desarrollaron posteriormente los **métodos Scrum**.

Los **principios de Scrum** son coherentes con el manifiesto **ágil** y se utilizan para guiar las actividades de desarrollo dentro de un proceso que incorpora las siguientes actividades marco: **requisitos, análisis, diseño, evolución y entrega.**

Dentro de cada actividad marco, las tareas de trabajo tienen lugar en un **período de tiempo** relativamente corto llamado ***sprint.***

El **trabajo realizado** dentro de un sprint (el número de sprints necesarios para cada actividad marco variará en función del tamaño del producto y su complejidad) **se adapta al problema en cuestión** y se define y modifica a menudo en tiempo real por el equipo Scrum.


Backlog del producto:
Características prioritarias del producto deseadas por el cliente
cada 24 horas
Scrum:
Reunión diaria de 15 minutos.
30 días
Sprint Backlog:
Características asignadas al sprint
Elementos pendientes ampliados por equipo
La nueva funcionalidad se demuestra al final del sprint

SAFEHOME

Considerar el desarrollo ágil de software

La escena: La oficina de Doug Miller.

Los jugadores: Doug Miller, director de ingeniería de software; Jamie Lazar, miembro del equipo de software; Vinod Raman, miembro del equipo de software.

La conversación: (Llaman a la puerta, Jamie y Vinod entran en el despacho de Doug).

Doug, ¿tienes un minuto?

Doug: Claro Jamie, ¿qué pasa?

Jamie: H e m o s estado pensando en nuestro debate de ayer sobre el proceso... ya sabes, qué proceso vamos a elegir para este nuevo proyecto *SafeHome*.

¿Y?

Vinod: Yo estaba hablando con un amigo en otra empresa, y él me estaba hablando de Scrum. Es un modelo de proceso ágil... ¿has oído hablar de él?

Sí, algunas buenas, algunas malas.

Jamie: Bueno, a nosotros nos suena bastante bien. Te permite desarrollar software muy rápido, utiliza algo llamado sprints para entregar incrementos de software cuando el equipo decide que el producto está terminado... es bastante guay, creo.

Doug: Tiene muchas ideas realmente buenas. Me gusta el concepto de sprint, el énfasis en la creación temprana de casos de prueba y la idea de que el propietario del proceso debe formar parte del equipo.

Jamie: ¿Eh? ¿Quieres decir que marketing trabajará en el equipo del proyecto con nosotros?

Doug (asintiendo): Son partes interesadas, pero no realmente el propietario del producto. Esa sería Marg.

Jamie: Bien. Ella filtrará los cambios de marketing querrá enviar cada 5 minutos.

Vinod: Aun así, mi amigo dijo que hay formas de "aceptar" los cambios durante un proyecto ágil.

Doug: ¿Creéis que deberíamos usar Scrum?

Jamie: Sin duda merece la pena considerarlo.

Doug: Estoy de acuerdo. E incluso si elegimos un modelo incremental como nuestro enfoque, no hay ninguna razón por la que no podamos incorporar gran parte de lo que Scrum tiene que ofrecer.

Vinod: Doug, antes de decir "algunos buenos, algunos malos". ¿Qué era lo malo?

Doug: L o q u e no me gusta es la forma en que Scrum resta importancia al análisis y al diseño. . . como que dice que escribir código es donde está la acción. . .

(Los miembros del equipo se miran y sonríen).

Doug: ¿Estás de acuerdo con el enfoque Scrum?

Jamie (hablando en nombre de ambos): Se puede adaptar a nuestras necesidades. Además, ¡escribir código es lo que hacemos, jefe!

Doug (riendo): Cierto, pero me gustaría verte pasar un poco menos de tiempo codificando y luego recodificando y un poco más de tiempo analizando lo que hay q u e hacer y diseñando una solución que funcione.

Vinod: Tal vez podamos tener las dos cosas, agilidad con un poco de disciplina.

Doug: Creo que podemos, Vinod. De hecho, estoy seguro de ello.

3.4.1 Equipos Scrum y artefactos

- El **equipo Scrum** es un equipo interdisciplinario autoorganizado que consta de un **producto propietario**, un **Scrum master** y un **pequeño equipo de desarrollo** (de tres a seis personas).
- Los principales **artefactos de Scrum** son el **trabajo pendiente del producto**, el **trabajo pendiente del sprint** y el **código**.
- **Incremento.** El desarrollo procede dividiendo el proyecto en una serie de **etapas incrementales**.
- **Períodos** de desarrollo de prototipos de 2 a 4 semanas de duración llamados **sprints**.

2.1 Equipos Scrum y artefactos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

El **backlog del producto**

- Es una lista priorizada de **requisitos** o características del producto que aportan valor de negocio al cliente. **Se pueden añadir elementos** al backlog **en cualquier momento** con la aprobación del propietario del producto y el consentimiento del equipo de desarrollo. El propietario del producto ordena los elementos del backlog del producto para cumplir los objetivos más importantes de todas las partes interesadas.
- El backlog del producto nunca está completo mientras el producto evoluciona para satisfacer las necesidades de las partes interesadas.
- El **propietario del producto** es la única persona que decide si **terminar un sprint** prematuramente o ampliarlo si el incremento no es aceptado.

2.1 Equipos Scrum y artefactos

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

El backlog del sprint

- Es el **subconjunto de elementos del backlog del producto** seleccionados por el equipo de producto para ser completados como el **incremento de código** durante el sprint activo actual.
- El **incremento** es la unión de todos los elementos del backlog del producto completados en sprints anteriores y todos los elementos del backlog que se completarán en los sprints actuales.
- El **equipo de desarrollo** crea un **plan** para la entrega de un incremento de software que contiene las características seleccionadas destinadas a cumplir un objetivo importante según lo negociado con el propietario del producto en el sprint actual. La mayoría de los **sprints** se programan para completarse **en 3 o 4 semanas**.
- El equipo de desarrollo decide **cómo completar el incremento**. El equipo de desarrollo también decide **cuándo el incremento está terminado** y listo para demostrárselo al propietario del producto. **No se pueden añadir nuevas funciones al backlog del sprint a menos que se cancele y reinicie el sprint.**

El **Scrum master**

- Actúa como **facilitador** para todos los miembros del equipo Scrum.
- **Dirige la reunión** Scrum diaria y es responsable de eliminar los obstáculos.
- Identificados por los miembros del equipo durante la reunión.
- Ella entrena a los miembros del equipo de desarrollo para ayudarse mutuamente a completar las tareas del sprint cuando tienen tiempo disponible.
- Ayuda al propietario del producto a encontrar técnicas para gestionar los elementos del backlog del producto y ayuda a garantizar que los elementos del backlog se establezcan en términos claros y concisos.

SCRUM

en 6 Minutos

2.2. Sprint Planning Meeting

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- Antes de empezar, cualquier equipo de desarrollo **trabaja con el propietario del producto** y el resto de partes interesadas para **desarrollar** los elementos del **backlog del producto.**
- El **propietario del producto y el equipo** de desarrollo **clasifican** los elementos del **backlog** del producto según la importancia de las necesidades de negocio del propietario y la complejidad de las tareas de ingeniería de software (programación y pruebas) necesarias para completar cada uno de ellos. **A veces**, esto **da lugar a la identificación de las características que faltan** para ofrecer la funcionalidad requerida a los usuarios finales.
- **Antes de comenzar cada sprint**, el **propietario del producto establece su objetivo** de desarrollo para el incremento que se completará en el próximo sprint.
- El **Scrum Master y el equipo** de desarrollo **seleccionan los elementos que se moverán al backlog del sprint.**
- El **equipo de desarrollo** determina lo **que se puede entregar en el incremento** dentro de las limitaciones del plazo asignado para el sprint y, con el Scrum master, qué trabajo será necesario para entregar el incremento. El equipo de desarrollo decide qué funciones son necesarias y cómo tendrán que cubrirse.

2.3. Daily Scrum Meeting

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- La reunión Scrum diaria es un evento de 15 minutos programado al comienzo de cada día de trabajo para permitir a los miembros del equipo sincronizar sus actividades y hacer planes para el siguiente día.
- El Scrum Master y el equipo de desarrollo siempre asisten al Scrum diario. Algunos equipos permiten que el propietario del producto asista ocasionalmente. Todos los miembros del equipo se plantean y responden tres preguntas clave:
  - ¿Qué ha hecho desde la última reunión del equipo?
  - ¿Qué obstáculos encuentra?
  - ¿Qué piensa conseguir para la próxima reunión del equipo?

2.3. Daily Scrum Meeting

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- El Scrum Master dirige la reunión y evalúa las respuestas de cada persona. La reunión Scrum **ayuda al equipo a descubrir posibles problemas** lo antes posible. Es tarea del **Scrum master despejar los obstáculos** que se presenten antes de la siguiente reunión Scrum, si es posible. No se trata de reuniones de resolución de problemas, éstas tienen lugar fuera de línea y sólo implican a las partes afectadas. Además, estas reuniones diarias conducen a la "**socialización del conocimiento**“ y por lo tanto promueven una estructura de equipo auto-organizado.
- Algunos equipos utilizan estas reuniones para **declarar que los elementos del backlog del sprint están completos o hechos**. Cuando el equipo considera que todos los elementos del backlog del sprint están completos, puede decidir programar una demostración y una revisión del incremento completado con el propietario del producto.

2.4 Sprint Review Meeting

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- La revisión del sprint tiene lugar **al final** del sprint, cuando el equipo de desarrollo considera que el **incremento se ha completado**.
- La revisión del sprint es a menudo time-boxed como una reunión de 4 horas para un sprint de 4 semanas.
- El Scrum Master, el equipo de desarrollo, el propietario del producto, y las partes interesadas seleccionadas asisten a esta revisión.
- La **actividad principal** es una **demostración del incremento de software** completado durante el sprint. Es importante tener en cuenta que la demostración puede no contener toda la funcionalidad planificada, sino más bien aquellas funciones que debían ser entregadas dentro del plazo definido para el sprint.
- El propietario del producto puede aceptar el incremento como completo o no. Si no se acepta, el propietario del producto y las partes interesadas aportan comentarios para permitir que se lleve a cabo una nueva ronda de planificación del sprint. Este es el momento en el que **pueden añadirse o eliminarse nuevas funciones** del backlog del producto.
- Las nuevas características pueden afectar a la naturaleza del incremento desarrollado en el siguiente sprint.

2.5. Sprint Retrospective

- Idealmente, antes de comenzar otra reunión de planificación del sprint, el Scrum master programará una **reunión de 3 horas** (para un sprint de 4 semanas) con el equipo de desarrollo llamada *retrospectiva del sprint*. Durante esta reunión el equipo discute:
- · Lo que fue bien en el sprint
- · Qué podría mejorarse
- · Lo que el equipo se compromete a mejorar en el próximo sprint
- El Scrum Master dirige la reunión y anima al equipo a mejorar sus prácticas de desarrollo para ser más eficaces en el siguiente sprint. El equipo planifica formas de mejorar la calidad del producto adaptando su definición de "hecho". Al final de esta reunión, el equipo debe tener una buena idea acerca de las mejoras necesarias en el próximo sprint y estar listo para planificar el incremento en la próxima reunión de planificación del sprint.

3. Otras metodologías ágiles

- La historia de la ingeniería del software ha experimentado la aparición y desvanecimiento de numerosos procesos, metodologías y tecnologías obsoletas. El movimiento ágil, al introducir varios marcos de procesos ágiles, sigue este patrón histórico. Entre ellos, **Scrum** destaca como uno de los más utilizados, pero también existen otros como **Extreme Programming (XP), Kanban y DevOps**, cada uno **compitiendo por la aceptación** en la comunidad de desarrollo de software.

3.1. El marco XP: Programación Extrema

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Uno de los enfoques más utilizados para el desarrollo ágil de software. Kent Beck escribió el trabajo seminal sobre XP.

La programación extrema engloba un conjunto de normas y prácticas que se desarrollan en el contexto de cuatro actividades marco:

- Planificación
- Diseño
- codificación y
- pruebas.

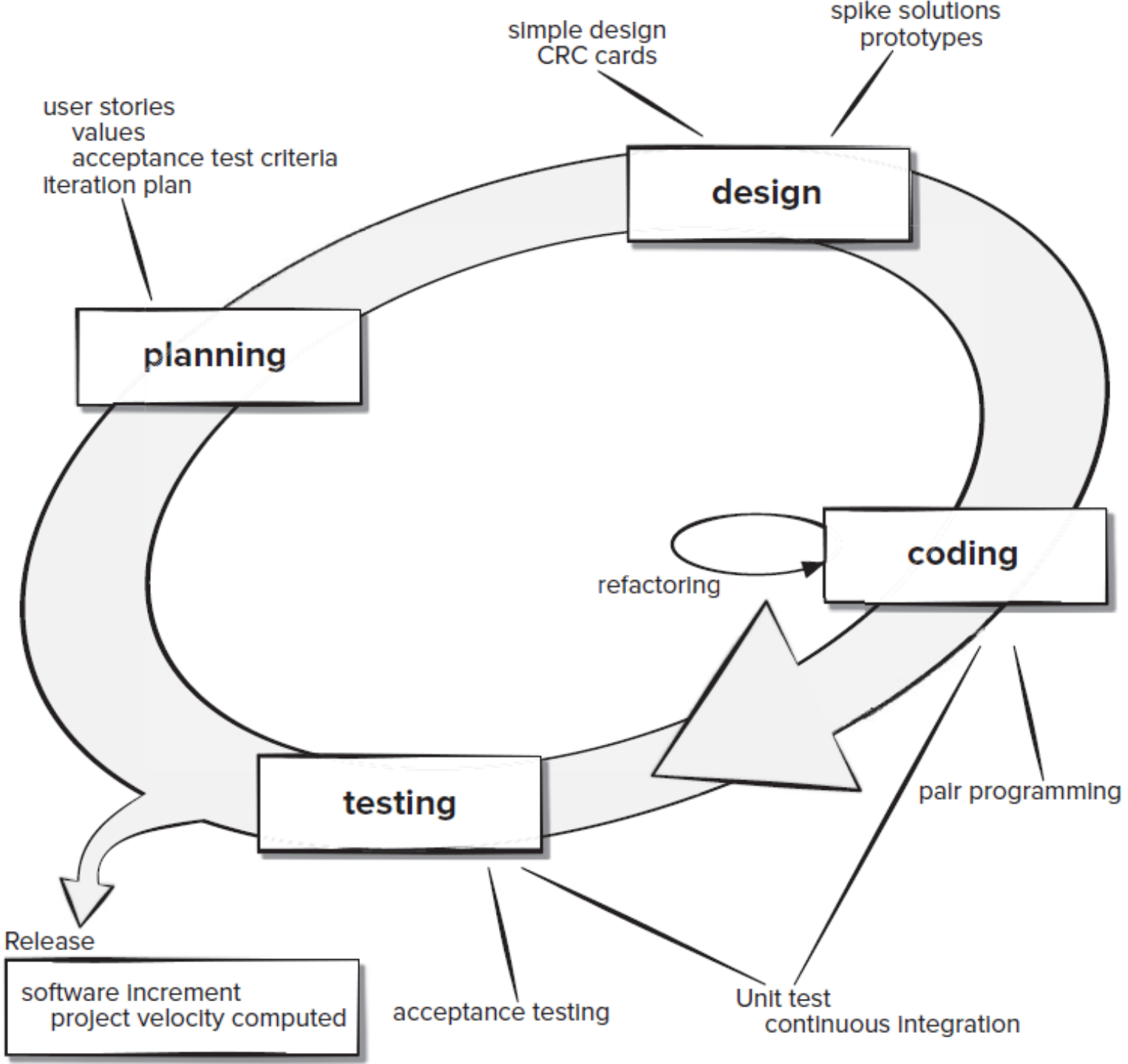
3.1. El marco XP: Programación Extrema

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Planificación

- La actividad de planificación (también llamada juego de planificación) comienza con una **actividad de requisitos** llamada **escucha**.
- La escucha conduce a la creación de un conjunto de "**historias**" (también llamadas historias de usuario) que **describen los resultados, las características y la funcionalidad** necesarios para el software que se va a construir.
- **Cada historia de usuario la escribe el cliente** y se coloca en una **ficha**.
- El cliente asigna un valor (es decir, una **prioridad**) a la historia basándose en el valor de negocio global de la característica o función.
- A continuación, los **miembros del equipo** XP evalúan cada historia y le **asignan un coste** -medido en semanas de desarrollo.
- Es importante señalar que **se pueden escribir nuevas historias** en cualquier momento.
- Los **clientes y los desarrolladores trabajan juntos** para decidir cómo agrupar las historias en la próxima versión (el siguiente incremento de software) que desarrollará el equipo de XP.
- Una vez que se establece un compromiso básico (acuerdo sobre las historias que se incluirán, fecha de entrega y otros asuntos del proyecto) para una versión, el equipo de XP ordena las historias que se desarrollarán de una de estas tres maneras:
  - (1) todas las historias se implementarán inmediatamente (en unas pocas semanas)
  - (2) las historias con mayor valor se adelantarán en el cronograma y se implementarán primero
  - (3) las historias más riesgosas se adelantarán en el cronograma y se implementarán primero.

The Extreme Programming process



Planificación

- Tras la entrega de la **primera versión** del proyecto (también llamada **incremento de software**), el equipo de XP calcula la velocidad del proyecto.
- En pocas palabras, *la* ***velocidad del proyecto*** es el **número de historias de clientes** implementadas durante la primera versión.
- La **velocidad del proyecto** se puede utilizar para ayudar a **estimar las fechas de entrega** y el calendario de las siguientes versiones. El equipo de X P modifica sus planes en consecuencia.

3.1. El marco XP: Programación Extrema

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Diseño.

- El diseño XP sigue rigurosamente el principio KIS **(keep it simple)**. Se desaconseja el diseño de funcionalidad extra (porque el desarrollador asume que será necesaria más adelante).
- XP fomenta el uso de **tarjetas CRC** como un mecanismo efectivo para pensar sobre el software en un contexto orientado a objetos.
- Las **tarjetas CRC (classresponsibility-collaborator)** identifican y organizan las clases orientadas a objetos que son relevantes para el incremento de software actual.
- Las tarjetas CRC son el **único producto de trabajo de diseño** producido como parte del proceso XP.
- Si se encuentra un **problema de diseño** difícil como parte del diseño de una historia, **XP recomienda la creación inmediata de un prototipo** operativo de esa parte de la historia.

3.1. El marco XP: Programación Extrema

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- **Codificación.**
  - Una vez desarrolladas las historias de usuario y realizado el trabajo de diseño preliminar, el equipo no pasa a codificar, sino que desarrolla una serie de **pruebas unitarias** que ejercitarán cada una de las historias que se incluirán en la versión actual (incremento de software).
  - Una vez creada la prueba unitaria, el desarrollador puede centrarse mejor en lo que debe implementarse para superar la prueba. Una vez completado el código, puede someterse inmediatamente a pruebas unitarias, lo que proporciona información instantánea a los desarrolladores.
  - Un concepto clave durante la actividad de codificación (y uno de los aspectos más comentados de XP) es la **programación por parejas**. XP recomienda que dos personas trabajen juntas en un ordenador para crear el código de una historia. Esto proporciona un mecanismo para la resolución de problemas en tiempo real (**dos cabezas piensan mejor que una**) y la garantía de calidad en tiempo real (el código se revisa a medida que se crea). A medida que los programadores en parejas completan su trabajo, el código que desarrollan se integra con el trabajo de los demás. Esta estrategia de "integración continua" ayuda a detectar pronto errores de compatibilidad e interconexión.

3.1. El marco XP: Programación Extrema

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Pruebas

- Las pruebas unitarias que se creen deben **implementarse utilizando un marco** de trabajo que **permita automatizarlas** (por lo tanto, pueden ejecutarse fácil y repetidamente).
- Esto anima a implementar una estrategia de pruebas de regresión cada vez que se modifica el código (lo que ocurre a menudo, dada la filosofía de refactorización de XP).
- Las ***pruebas de aceptación* de XP**, también llamadas ***pruebas de cliente,*** son especificadas por el cliente y se centran en las características generales del sistema y la funcionalidad que son visibles y **revisables por el cliente**. Se derivan de las historias de usuario que se han implementado como parte de una versión de software.

3.2. Kanban

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

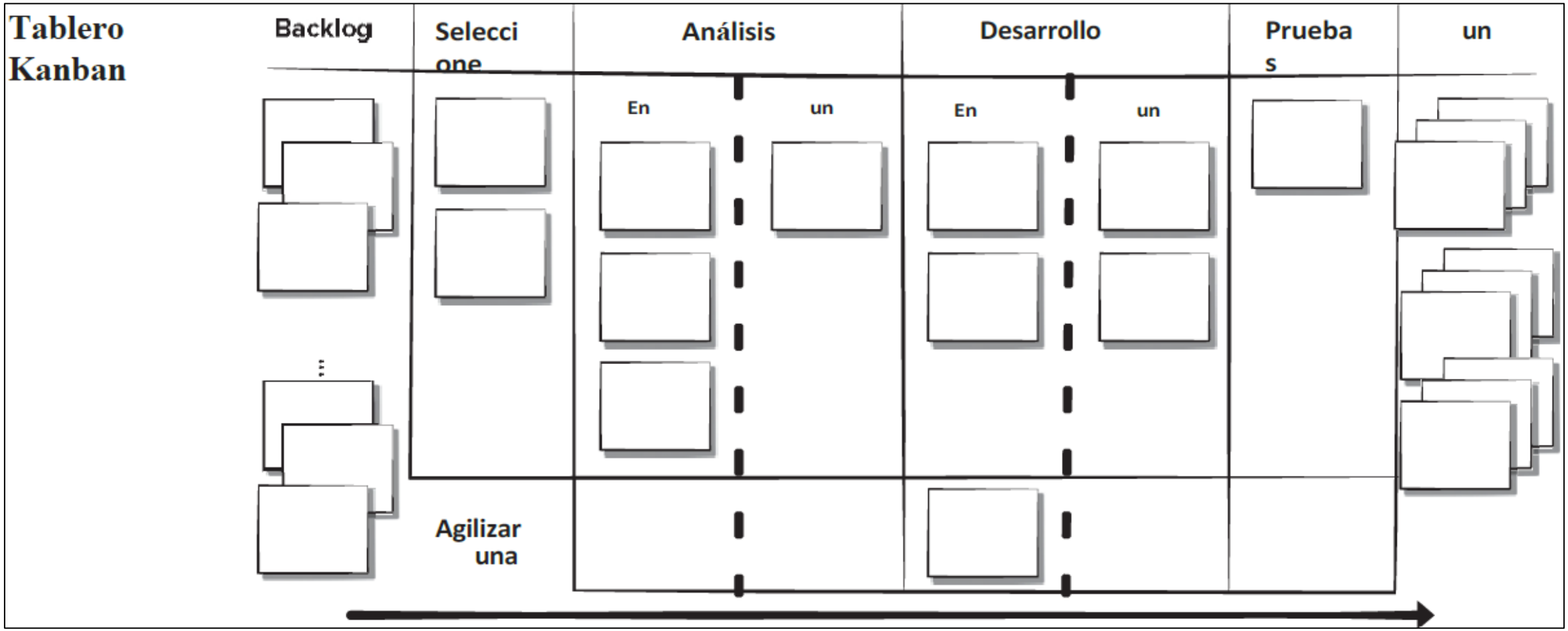
El método *Kanban* es una **metodología lean** que describe **métodos para mejorar cualquier proceso** o flujo de trabajo.

Kanban se centra en la **gestión de cambios y la prestación de servicios**.

La **gestión del cambio** define el proceso mediante el cual **un cambio solicitado se integra en un sistema basado en software.**

La **prestación de servicios** se fomenta centrándose en la **comprensión de las necesidades** y **expectativas** del cliente. Los miembros del equipo gestionan el trabajo y tienen libertad para organizarse para completarlo. Las políticas evolucionan en función de las necesidades para mejorar los resultados.

3.2. Kanban


Tablero Kanban
Backlog
Selecci one
Análisis
Desarrollo
Prueba s
un
En
un
En
un
Agilizar una

3.2. Kanban

- Kanban se originó en Toyota como un conjunto de prácticas de ingeniería industrial y fue adaptado al desarrollo de software por David Anderson. Kanban en sí depende de seis prácticas básicas:
  - **1. Visualización del flujo de trabajo mediante un tablero Kanban**. El tablero Kanban se organiza en columnas que representan la fase de desarrollo de cada elemento de la funcionalidad del software. Las tarjetas del tablero pueden contener historias de usuario individuales o defectos descubiertos recientemente en notas adhesivas y el equipo las haría avanzar de "por hacer" a "haciendo" y "hecho" a medida que avanzara el proyecto.

3.2. Kanban

- **2. Limitar la cantidad de trabajo** en curso (WIP) en un momento dado. Se anima a los desarrolladores a **terminar su tarea actual antes de empezar otra**. Esto reduce el tiempo de espera, mejora la calidad del trabajo y aumenta la capacidad del equipo para ofrecer funcionalidad de software con frecuencia a sus partes interesadas.
- **3. Gestionar el flujo de trabajo** para **reducir el despilfarro** comprendiendo el flujo de valor actual, analizando los lugares en los que se estanca, definiendo los cambios y aplicándolos a continuación.

3.2. Kanban

- **4. Hacer explícitas las políticas del proceso** (por ejemplo, anotar las **razones por las que se seleccionan los elementos** en los que se va a trabajar y los criterios utilizados para definir "hecho").
- **5. Centrarse en la mejora continua** mediante la creación de bucles de retroalimentación en los que los cambios se introducen basándose en los datos del proceso y los efectos del cambio en el proceso se miden después de realizar los cambios.
- **6. Realice cambios** en los procesos **de forma colaborativa** e implique a todos los miembros del equipo y otras partes interesadas según sea necesario.

3.2. Kanban

- Las **reuniones de equipo** para Kanban son **como las del marco Scrum**.
- Si se introduce Kanban en un proyecto existente, no todos los elementos empezarán en la columna de backlog. Los desarrolladores deben colocar sus tarjetas en la columna del proceso de equipo preguntándose:
  - ¿Dónde están ahora?
  - ¿De dónde vienen?
  - ¿Hacia dónde se dirigen?
- La base de la reunión Kanban standup diaria es una tarea llamada **"recorrer el tablero".** El liderazgo de esta reunión rota diariamente. Los miembros del equipo identifican los elementos que faltan en el tablero en los que se está trabajando y los añaden al tablero.

3.2. Kanban

- El equipo trata de **avanzar** todos los elementos que puede hasta "hecho".
- El **objetivo** es avanzar primero en los elementos de mayor valor empresarial.
- El equipo examina el flujo y trata de identificar cualquier impedimento para la finalización, analizando la carga de trabajo y los riesgos.
- Durante la reunión retrospectiva semanal se examinan las mediciones del proceso.
- El equipo estudia dónde puede ser necesario mejorar el proceso y propone cambios para su aplicación. Kanban puede combinarse fácilmente con otras prácticas de desarrollo ágil para añadir un poco más de disciplina al proceso.

3.3. DevOps

- DevOps fue creado **por Patrick DeBois** para **combinar desarrollo y operaciones**.
- DevOps intenta aplicar principios de desarrollo ágiles y ajustados en toda la cadena de suministro de software.
- El enfoque DevOps implica varias etapas que se repiten continuamente hasta que se obtiene el producto deseado:

3.3. DevOps

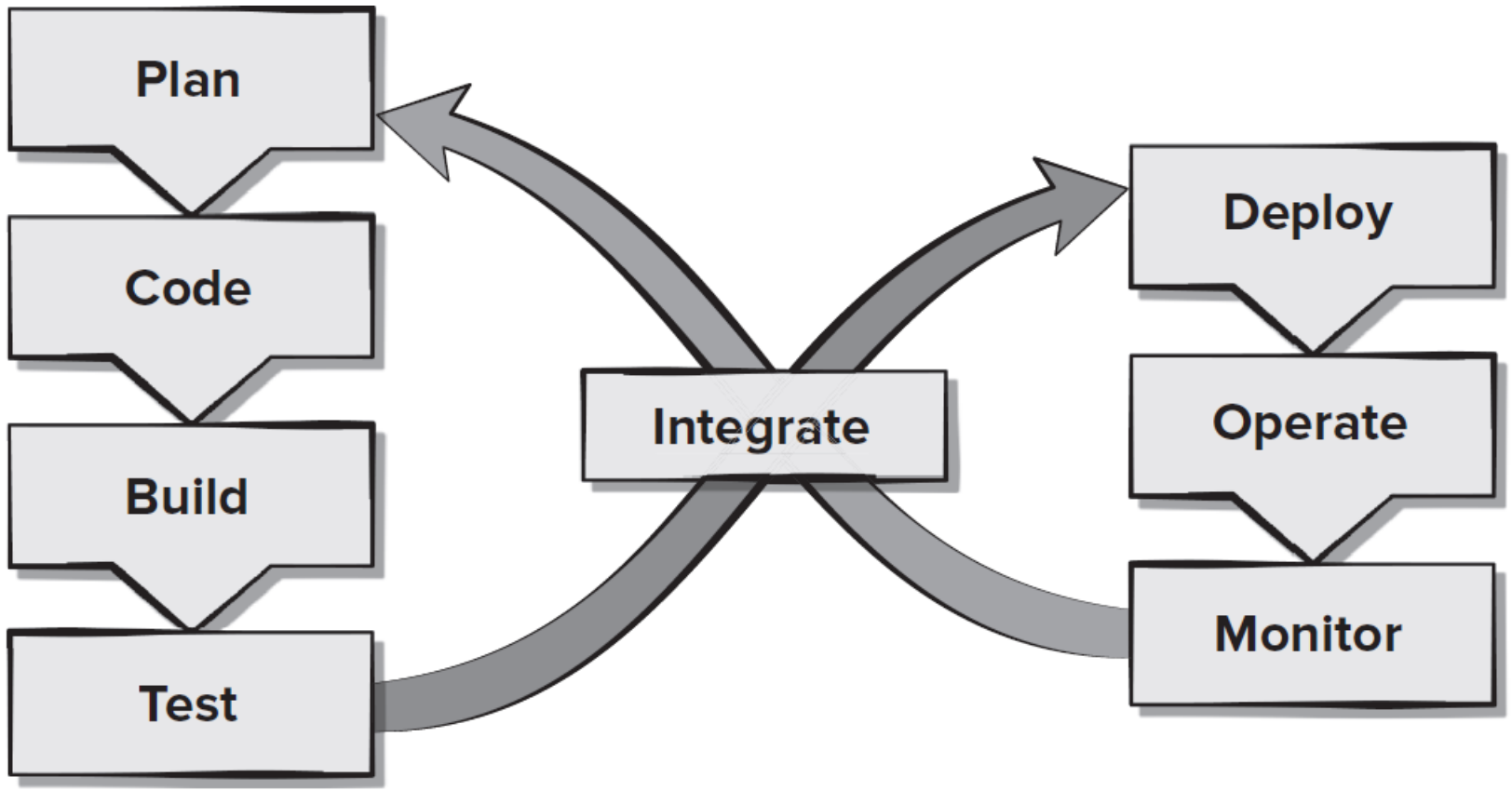
- **Desarrollo continuo. Los** entregables de software se desglosan y desarrollan en **varios sprints** y los incrementos se entregan a los miembros del equipo de desarrollo encargados de garantizar la calidad para que los prueben.
- **Pruebas continuas. Las** herramientas de pruebas automatizadas se utilizan para ayudar a los miembros del equipo a probar múltiples incrementos de código al mismo tiempo para asegurarse de que están libres de defectos antes de la
- integración.
- **Integración continua.** Las piezas de código con nuevas funcionalidades se añaden al código existente y al entorno de ejecución y, a continuación, se comprueba que no haya errores tras el despliegue.
- **Despliegue continuo.** En esta fase, el código integrado se despliega (instala) en el entorno de producción, que puede incluir varios sitios en todo el mundo que deben prepararse para recibir la nueva funcionalidad.
- **Supervisión continua.** El personal de operaciones que forma parte del equipo de desarrollo ayuda a mejorar la calidad del software supervisando su rendimiento en el entorno de producción y buscando proactivamente posibles problemas antes de que los usuarios los detecten.

3.3. DevOps

- DevOps mejora las experiencias de los clientes reaccionando rápidamente a los cambios en sus necesidades o deseos. Esto puede aumentar la fidelidad a la marca e incrementar la cuota de mercado. Los enfoques Lean como DevOps pueden proporcionar a las organizaciones una mayor capacidad de innovación reduciendo la repetición de tareas y permitiendo el cambio a actividades de mayor valor empresarial.
- Los productos no generan dinero hasta que los consumidores tienen acceso a ellos, y DevOps puede proporcionar un tiempo de despliegue más rápido a las plataformas de producción.

3.3. DevOps

DevOps



Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

3. TRABAJO COLABORATIVO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Trabajo colaborativo

- *En grupos, elaborar un cuadro comparativo de las metodologías estudiadas*

Receso

- *Para que todos los participantes pueden despejarse y regresen a la sesión con mayor predisposición para continuar, se brinda un pequeño receso de (10 minutos)*

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

4. EVALUACIÓN DE LA SESIÓN

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

Evaluación de la sesión

- *Estimado estudiante, vamos a ingresar a una herramienta de gamificación para responder a 10 preguntas que servirán para reforzar lo aprendido en la sesión.*

conker
FORMATIVES IN A FLASH

ACTIVIDAD

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

- Realiza un cuadro comparativo las metodologías ágiles de software.

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

5. CONCLUSIONES

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

5.1. Conclusiones

- Los profesionales deben enfocar la ingeniería de software de una manera que les permita seguir siendo ágiles, para definir procesos maniobrables, adaptables y ajustados que puedan acomodarse a las necesidades de la empresa moderna.
- Una filosofía ágil para la ingeniería de software hace hincapié en cuatro aspectos clave: la importancia de los equipos autoorganizados que tienen el control sobre el trabajo que realizan, la comunicación y colaboración entre los miembros del equipo y entre los profesionales y sus clientes, el reconocimiento de que el cambio representa una oportunidad y el énfasis en la entrega rápida de software que satisfaga al cliente.
- Los modelos de procesos ágiles se han diseñado para abordar cada una de estas cuestiones.

Universidad Nacional de Cajamarca

"Norte de la Universidad Peruana"

6. AVISO O ANUNCIO

Ingeniería de Sistemas

Universidad Nacional de Cajamarca

6.1. Aviso o Anuncios

- *Para la siguiente sesión, debes revisar "Capítulo 4"* (Pressman & Maxim, 2020).

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

7. REFERENCIAS BIBLIOGRÁFICAS

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Referencias Bibliográficas

- *Pressman, R., & Maxim, B. (2020). Software engineering: A practitioner's approach (Ninth edition). McGraw-Hill Education.*

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

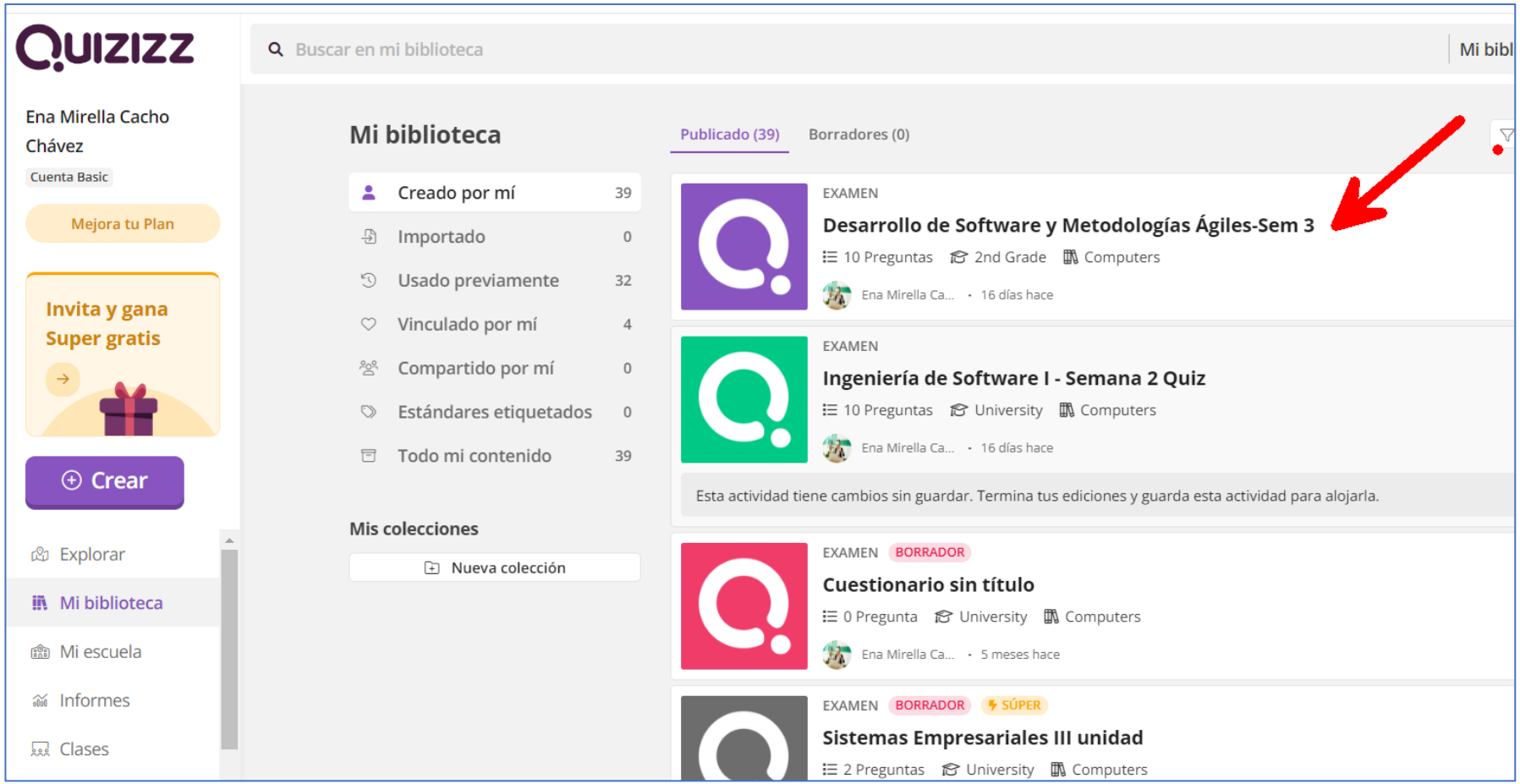
8. MATERIAL COMPLEMENTARIO

Ingeniería de Sistemas
Universidad Nacional de Cajamarca

Material complementario

- *Software e Ingeniería de software (Pressman & Maxim, 2020)*
- *Videos*

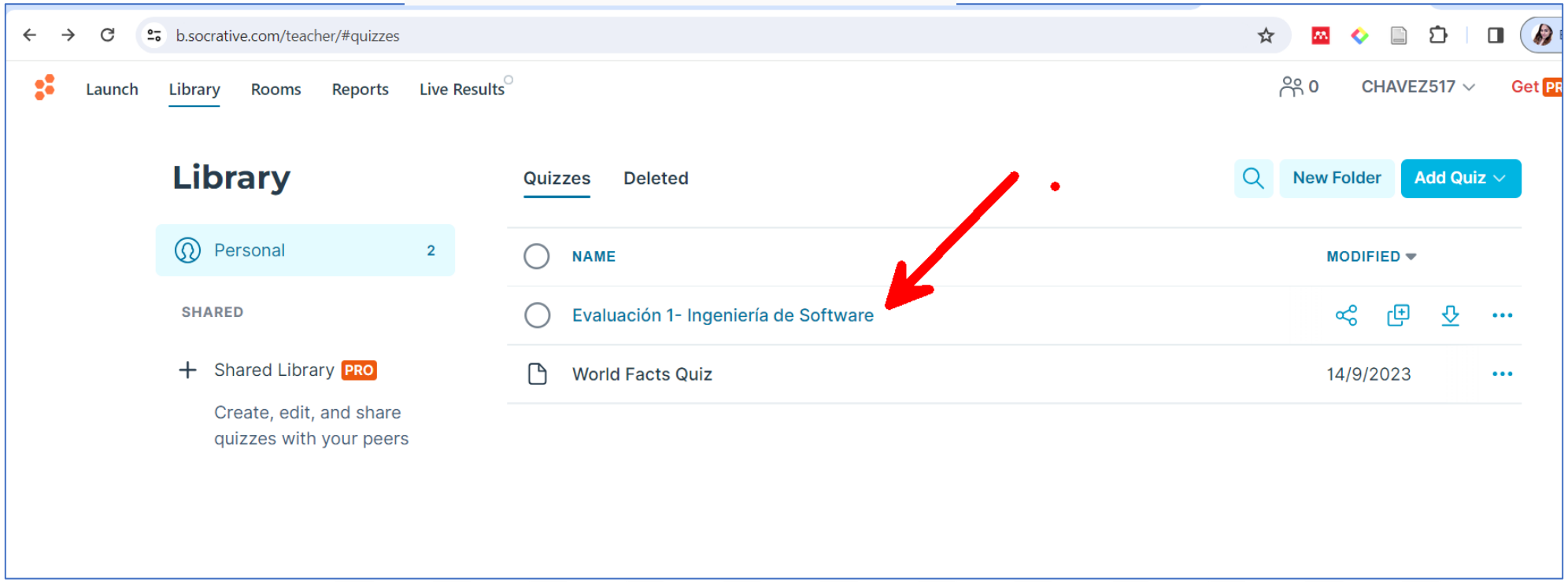
CUESTIONARIO QUIZIZ


QUIZIZZ
Buscar en mi biblioteca
Mi bibl
Ena Mirella Cacho Chávez
Cuenta Basic
Mejora tu Plan
Invita y gana Super gratis
Crear
Explorar
Mi biblioteca
Mi escuela
Informes
Clases
Mi biblioteca
Creado por mí 39
Importado 0
Usado previamente 32
Vinculado por mí 4
Compartido por mí 0
Estándares etiquetados 0
Todo mi contenido 39
Mis colecciones
Nueva colección
Publicado (39)
Borradores (0)
EXAMEN
Desarrollo de Software y Metodologías Ágiles-Sem 3
10 Preguntas 2nd Grade Computers
Ena Mirella Ca... · 16 días hace
EXAMEN
Ingeniería de Software I - Semana 2 Quiz
10 Preguntas University Computers
Ena Mirella Ca... · 16 días hace
Esta actividad tiene cambios sin guardar. Termina tus ediciones y guarda esta actividad para alojarla.
EXAMEN BORRADOR
Cuestionario sin título
0 Pregunta University Computers
Ena Mirella Ca... · 5 meses hace
EXAMEN BORRADOR SÚPER
Sistemas Empresariales III unidad
2 Preguntas University Computers

1ERA EVALUACIÓN

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

socrative



GRACIAS

Universidad Nacional de Cajamarca
"Norte de la Universidad Peruana"

Ingeniería de Sistemas
Universidad Nacional de Cajamarca