Metodologías de Desarrollo Sw Ágiles

Pablo Sánchez

Dpto. Ingeniería Informática y Electrónica
Universidad de Cantabria
Santander (Cantabria, España)
p.sanchez@unican.es

Advertencia

Todo el material contenido en este documento no constituye en modo alguno una obra de referencia o apuntes oficiales mediante los cuales se puedan preparar de manera autónoma las pruebas necesarias para superar la asignatura.

Este documento contiene exclusivamente una serie de diapositivas cuyo objetivo es servir de complemento visual a las actividades realizadas en el aula.

Dicho de forma más clara, estas transparencias no son apuntes y su objetivo no es en modo alguno servir para que el alumno pueda preparar la asignatura.

Índice

Objetivos del Tema

1. Conocer y comprender los principios básicos de las metodologías de desarrollo software ágiles.
2. Conocer y saber usar los conceptos y técnicas más comúnmente utilizados por las metodologías ágiles.
3. Conocer las principales metodologías ágiles actuales, sabiendo distinguir sus características principales.
4. Ser capaz de desarrollar proyectos sw utilizando Scrum.

Bibliografía

Schwaber, K. and Sutherland, J. (2011).
The Scrum Guide - The Definitive Guide to Scrum: The Rules of the Game.

Cohn, M. (2004).
User Stories Applied,
Addison-Wesley.

Cockburn, A. (2006).
Agile Software Development: The Cooperative Game,
Addison-Wesley, 2ª edición.

Bibliografía


Poppendieck, M and Poppendieck, T. (2003).
Lean Software Development: An Agile Toolkit.
Addison-Wesley.


Anderson, D. J. and Reinertsen, D. G. (2010).
Kanban: Successful Evolutionary Change for Your Technology Business.
Blue Hole Press.


Sommerville, I. (2010).
Software Engineering.
Addison Wesley, 9ª edición.


Bourque, P. and Dupuis, R., editores (2004).
Guide to the Software Engineering Body of Knowledge.
IEEE (Institute of Electrical and Electronics Engineers).

Índice

Índice

Metodologías Fuertemente Planificadas vs Ágiles

Metodologías Fuertemente Planificadas

Se basan en la definición de un plan de trabajo al cual debemos de adherirnos y que es complejo o difícil de cambiar. Suele tener roles y procesos claramente definidos.

Metodologías Ágiles

Se basan en una organización del trabajo que permita la definición de planes de trabajo, procesos y productos fácilmente modificables en función de las demandas de los usuarios, con los cuales se debe mantener un permanente contacto, y al cual se le deben proporcionar versiones parciales y operativas del software con frecuencia.

Índice

Manifiesto Ágil

1. Personas e interacciones personales sobre procesos y herramientas.
2. Software operativo antes que extensos modelos software.
3. Colaboración con el cliente frente a renegociaciones de contrato.
4. Ser capaz de responder al cambio antes que seguir un plan fijo.

Principios del Manifiesto Ágil (1-6)

1. Satisfacción del cliente mediante entregas tempranas y continuas de software operativo.
2. Los cambios son siempre bien recibidos.
3. Entregar software operativo frecuentemente.
4. Gestores y programadores deben trabajar juntos siempre.
5. Motivar y confiar en los trabajadores.
6. La mejor forma de transmitir información es cara a cara.

Principios del Manifiesto Ágil (7-12)

7. El principal indicador del avance de un proyecto es la cantidad de software operativo entregado.
8. El desarrollo software debería ser sostenible.
9. La excelencia técnica y los buenos diseños facilitan la agilidad.
10. La simplicidad (el arte de maximizar el trabajo a no hacer) es primordial.
11. Las mejores arquitecturas y diseños emergen de equipos autoorganizados.
12. El equipo de trabajo debe reflexionar periódicamente acerca de su eficiencia y productividad.

Principales Tipos de Metodologías Ágiles

Fuente: 8th Annual State of Agile

1. Scrum
2. eXtremme Programming (XP)
3. Kanban
4. Lean
5. Feature-Driven Development (FDD)

Índice

Limitaciones de las Metodologías Ágiles

1. Necesidad de involucrar al cliente.
2. Necesidad de tener una actitud y una personalidad cooperativa.
3. Necesidad de tener unas ciertas habilidades técnicas.
4. Dificultades para identificar prioridades e incrementos sw.
5. Mantener la simplicidad frente al simple parcheo.
6. Problemas contractuales y de definición de ámbito del proyecto.

Índice

Desarrollo de Software Lean

Metodología Lean

Metodología articulada en torno a una serie de técnicas y métodos que cumplen con los siguientes principios:

1. Eliminar los desperdicios.
2. Fomentar el aprendizaje.
3. Postergar las decisiones tanto como se pueda.
4. Realizar entregas tan rápido como sea posible.
5. Delegar responsabilidades en el equipo de trabajo.
6. Fomentar la integridad del producto y del equipo.
7. Facilitar la visión global del conjunto.

Índice

Índice

Historia de Usuario (*User Story*)

Historia de Usuario

Un *historia de usuario* describe una funcionalidad del sistema que posee valor para algún *stakeholder* del sistema. Una historia de usuario se compone de tres elementos:

Conversación Corresponde con las conversaciones (efímeras) mantenidas entre las diferentes personas involucradas en el desarrollo de la historia de usuario, tales como usuarios, clientes, testadores, programadores, etc. Se puede documentar parte de estas conversaciones.

Tarjeta Cada historia de usuario se anota en una tarjeta física, que representa la historia de usuario.

Confirmación Procedimiento para verificar que la historia de usuario ha sido realizada.

Propiedades Deseables de una Historia de Usuario

1. Independiente.
2. Negociable.
3. Añade valor al producto.
4. Estimable.
5. Pequeñas.
6. Verificable (testable).

Historia Épica

Historia Épica

Una historia épica (*epic*) es una historia que por su tamaño no puede ser desarrollada en un periodo corto de tiempo, por lo que ha de ser descompuesta en historias de usuario de menor tamaño.

Índice

Sprint and Spike

Sprint

Un *sprint* es una iteración de una duración concreta de tiempo, normalmente corta, y al final de la cual se produce software potencialmente entregable.

Spike

Un *spike* es una iteración en la cual no se libera nuevo código que pueda ser instalado o probado por los usuarios de la aplicación. Un *spike* se utiliza para probar nuevos algoritmos, aprender nuevas tecnologías, generar documentación o productos de marketing, entre otros elementos.

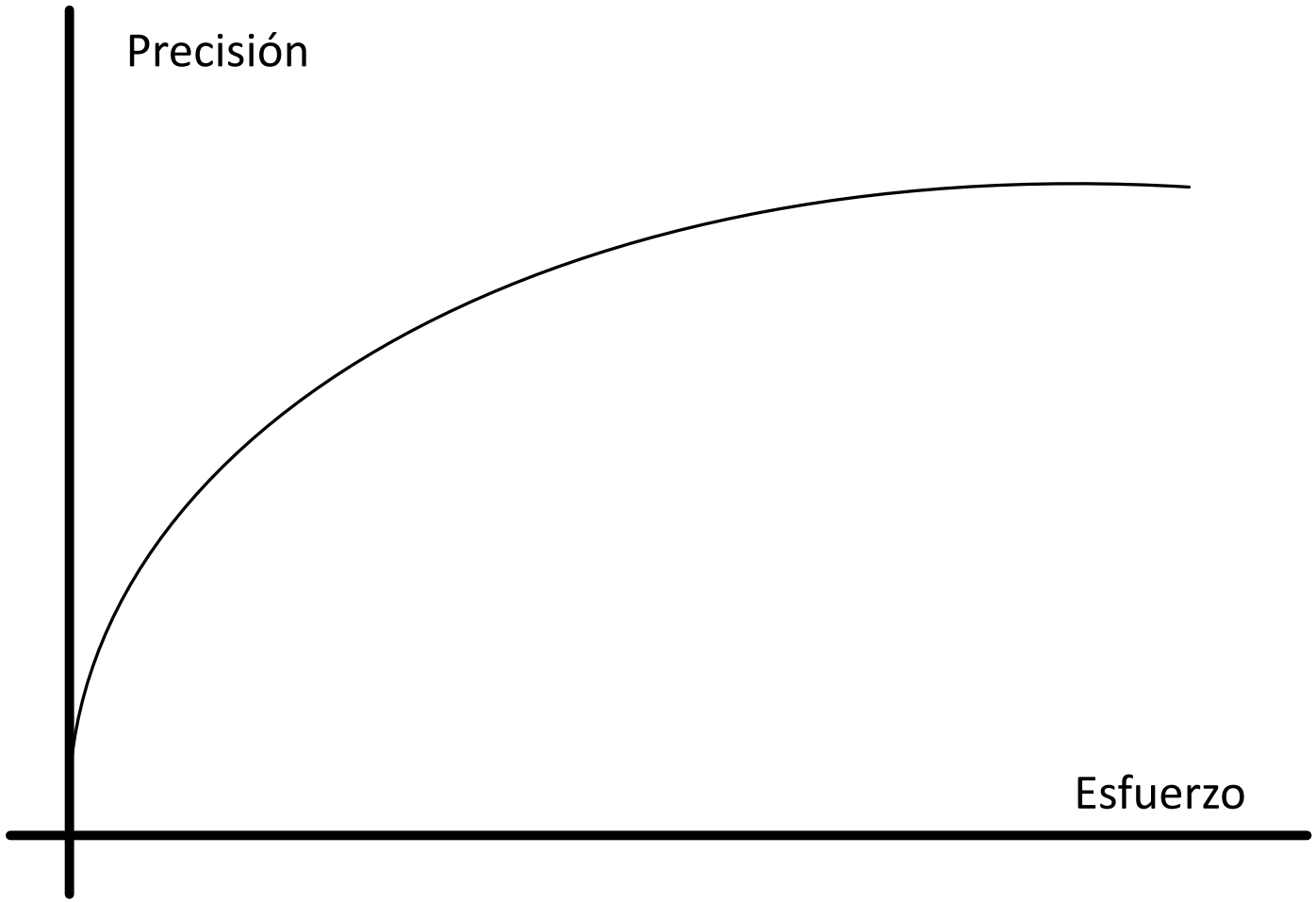
Índice

Desarrollo Dirigido por Pruebas

1. Los casos de prueba se escriben antes que el código.
2. Los casos de prueba se automatizan.
3. El código se escribe buscando satisfacer las pruebas.
4. Una nueva funcionalidad se considera completada cuando supera todos los casos de prueba.

Índice

Precisión y Esfuerzo en las Estimaciones



Principios Estimación Ágil

1. Las estimaciones son compartidas y consensuadas.
2. Utilizar una escala discreta para las estimaciones (Fibonacci).

Planning Poker

1. Se reune al equipo de desarrollo.
2. Cada miembro recibe una serie de cartas con los valores de la escala de estimación.
3. Por cada elemento a estimar, el moderador presenta el elemento a estimar.
4. Si es necesario, los diferentes miembros piden que se realicen aclaraciones sobre dicho elemento.
5. A continuación, cada miembro escoge una carta con el valor de su estimación, ocultando su valor.
6. Cuando todos los miembros han realizado su estimación, se muestran las cartas.
7. Si hay variaciones significativas en la estimación, se discute brevemente los valores dados, y se vuelve al punto 5.
8. Si no hay diferencias significativas, se selecciona el valor de la estimación consensuada.

Índice

Programación por Pares

Programación por Pares

Se codifica por parejas, donde una persona escribe el código y la otra persona supervisa y aporta ideas. Los roles se permutan con frecuencia.

1. Se reduce el número de defectos.
2. Se genera código más compacto.
3. Se detectan más refactorizaciones.
4. El par que observa actúa de revisor del código (aumenta la calidad).
5. Se evita que una pieza de código sólo pertenezca a una persona.

Índice

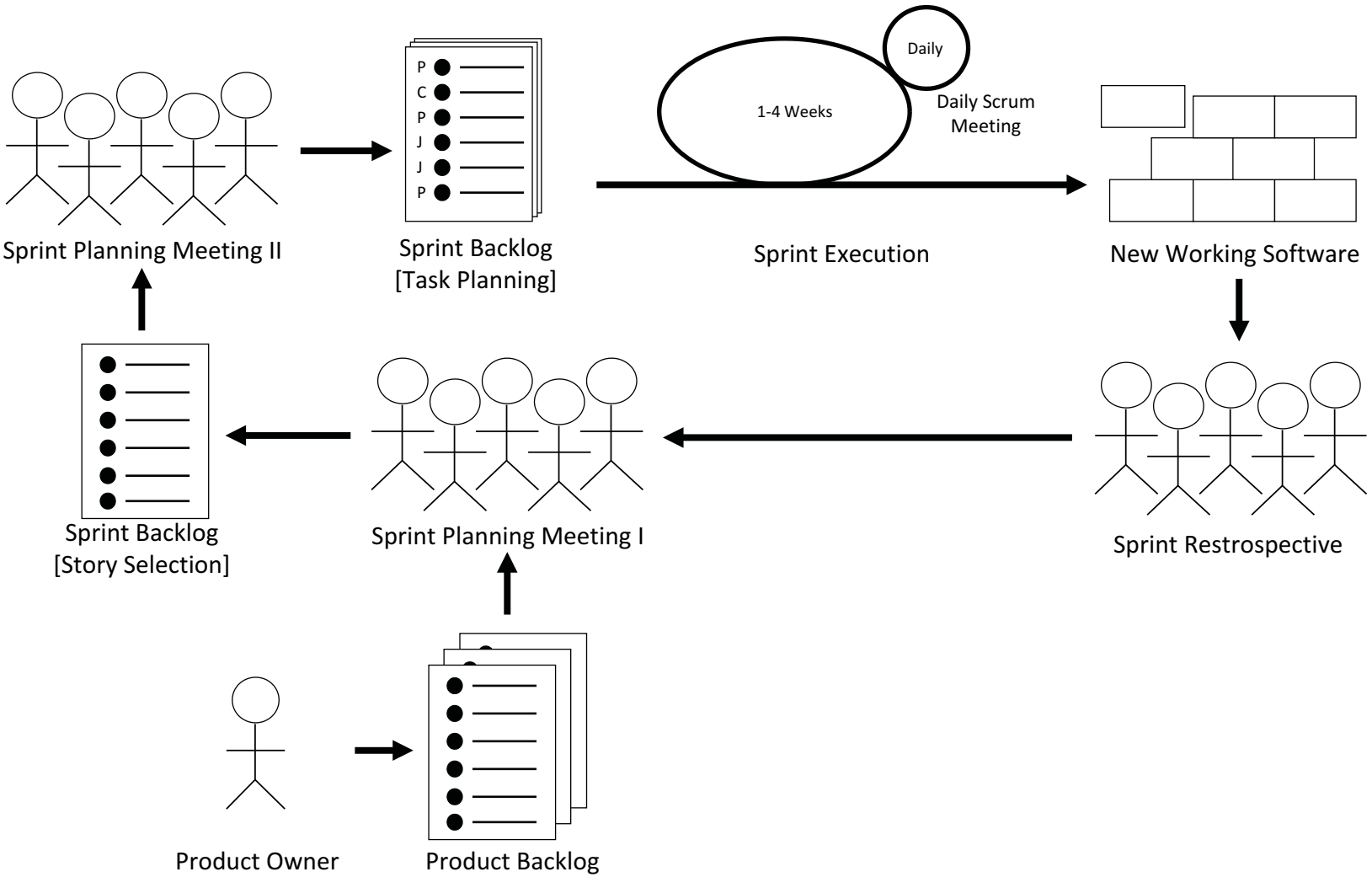
Índice

Roles en Scrum

1. Product Owner.
2. Scrum Team.
3. Scrum Master.

Índice

Proceso de Desarrollo Scrum



Índice

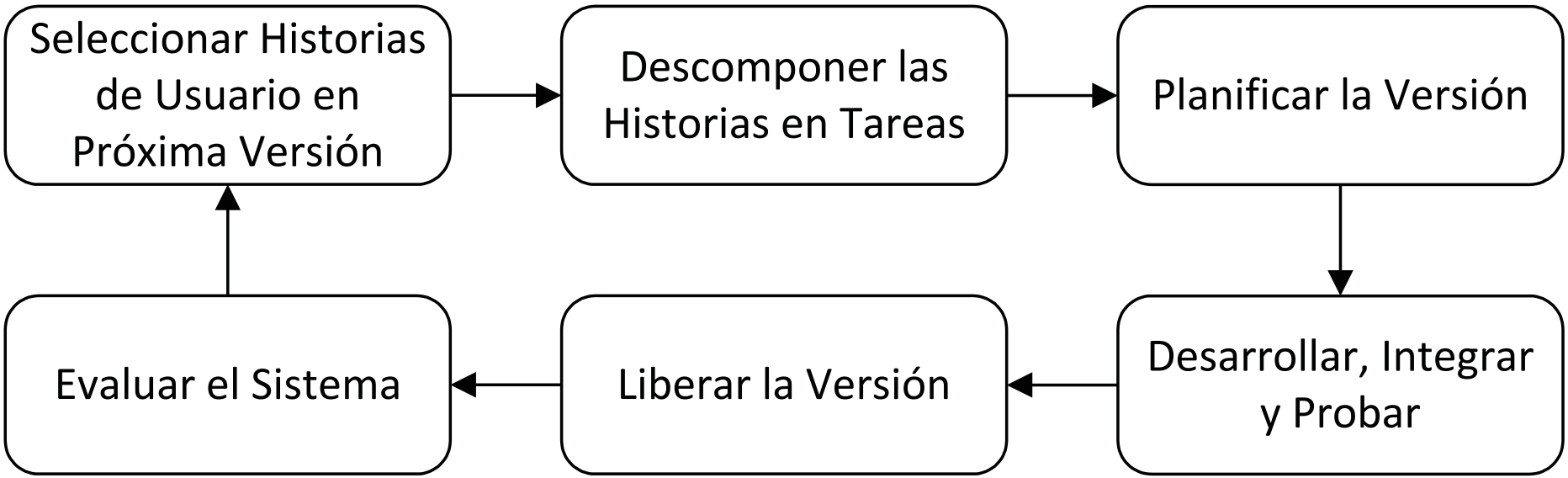
Conceptos de Scrum

1. Release backlog.
2. Velocidad del equipo.
3. Definición de Completado.
4. Sprint and Release Burndown Charts.
5. Product Backlog Refinement.

Índice

Índice

Ciclo de de Vida XP



Índice

Las Doce Prácticas de XP

1. Versiones pequeñas.
2. El juego de la planificación.
3. Refactorización.
4. Desarrollo dirigido por pruebas.
5. Desarrollo por pares.
6. Ritmo de trabajo sostenible.
7. Propiedad colectiva del código.
8. Estándares para la codificación.
9. Diseños simples (no puntos de variación anticipados).
10. Utilización de metáforas.
11. Integración continua.
12. Inclusión del cliente en el equipo de desarrollo.

Índice

Índice

Kanban

1. El objetivo encontrar un ritmo de trabajo sostenible y óptimo.
2. Aplica *Teoría de las Restricciones*.
3. Busca identificar y eliminar cuellos de botella de los procesos de trabajo.
4. Para poder identificar cuellos de botellas, debemos visualizar el proceso.

Índice

Filosofía de Kanban

Filosofía Kanban

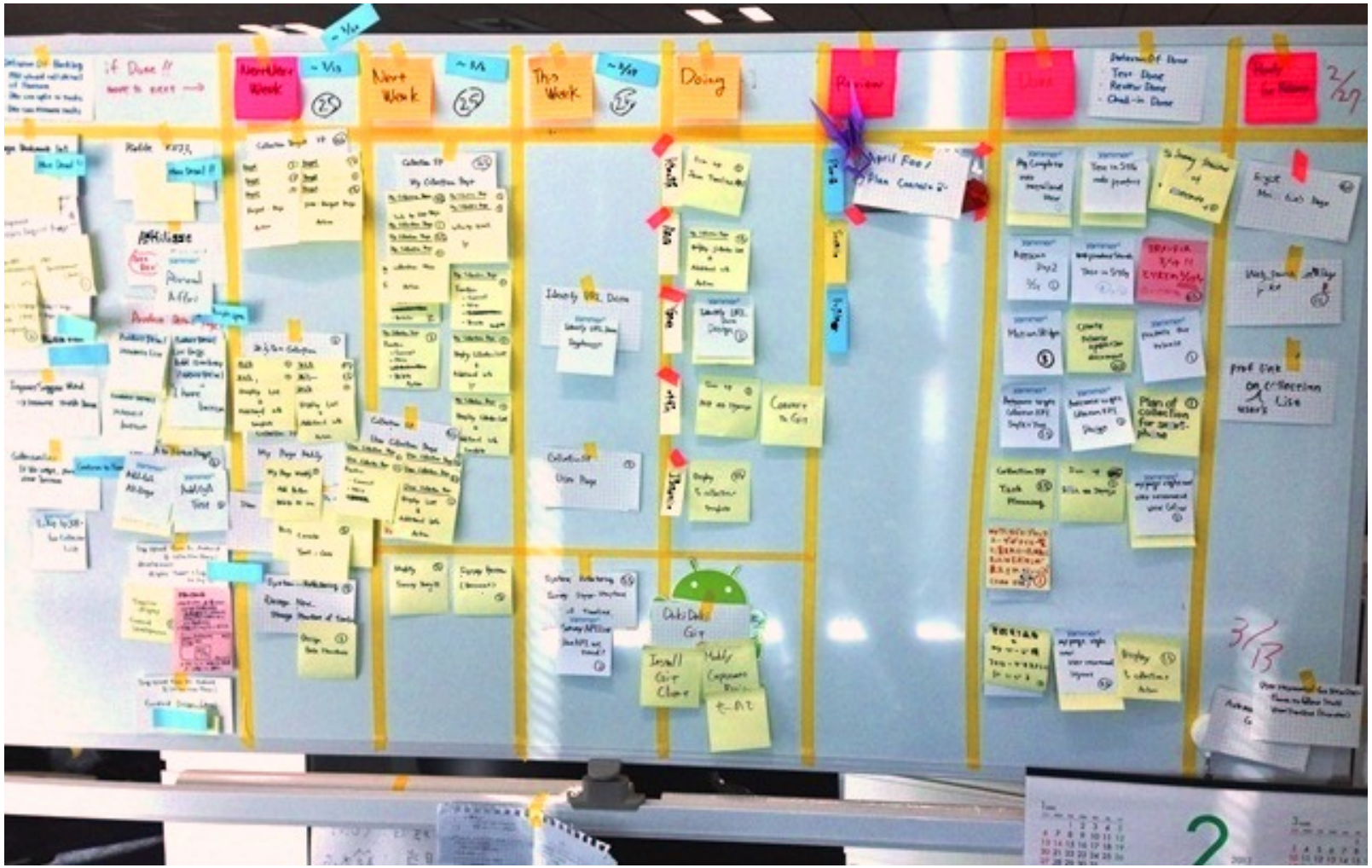
1. El equipo de trabajo tiene unos recursos.
2. Cada recurso tiene una capacidad de producción determinada.
3. Cuando un recurso tiene capacidad libre, libera tarjetas que señalan que puede admitir más trabajo.
4. Cuando un recurso no tiene capacidad libre, no habría tarjetas disponibles para solicitar más trabajo.

Índice

Método Kanban

1. Las tareas a realizar se escriben en tarjetas.
2. Cada miembro del equipo es un recurso que puede admitir un determinado número de tarjetas simultáneas.
3. Por tanto, la cantidad de trabajo que puede estar en progreso es limitada.
4. Cuando un recurso puede admitir más trabajo, debe señalarse para que se le asigne.

Método Kanban



Método Kanban

Propiedades Fundamentales de un Método Kanban

1. Limitar la capacidad del *work in progress*.
2. Visualizar el proceso y el flujo de trabajo.
3. Medir y optimizar el flujo de trabajo.
4. Hacer explícitas y visibles las reglas del proceso de trabajo.
5. Gestionar el proceso con métricas.

Índice

¿Qué Tengo que Saber de Todo Esto?

1. Saber cuáles son y comprender los principios del manifiesto ágil.
2. Saber cuáles son y comprender los principios de las metodologías *Lean*.
3. Saber discernir cuando una metodología cumple con los principios del manifiesto ágil o de la filosofía Lean.
4. Saber especificar requisitos funcionales como historias de usuario.
5. Conocer la terminología y las técnicas asociadas a la metodología de desarrollo Scrum.
6. Ser capaz de trabajar como miembro de un *Scrum Team*.
7. Ser capaz de utilizar una herramienta de gestión de proyectos Scrum como Scrum Desk.
8. Comprender los principios y organización de la metodología de desarrollo XP.
9. Comprender los principios y objetivos de la metodología de desarrollo Kanban.