DESARROLLO DE SOFTWARE

Clase II - Metodologías Ágiles

HISTORIA

1980

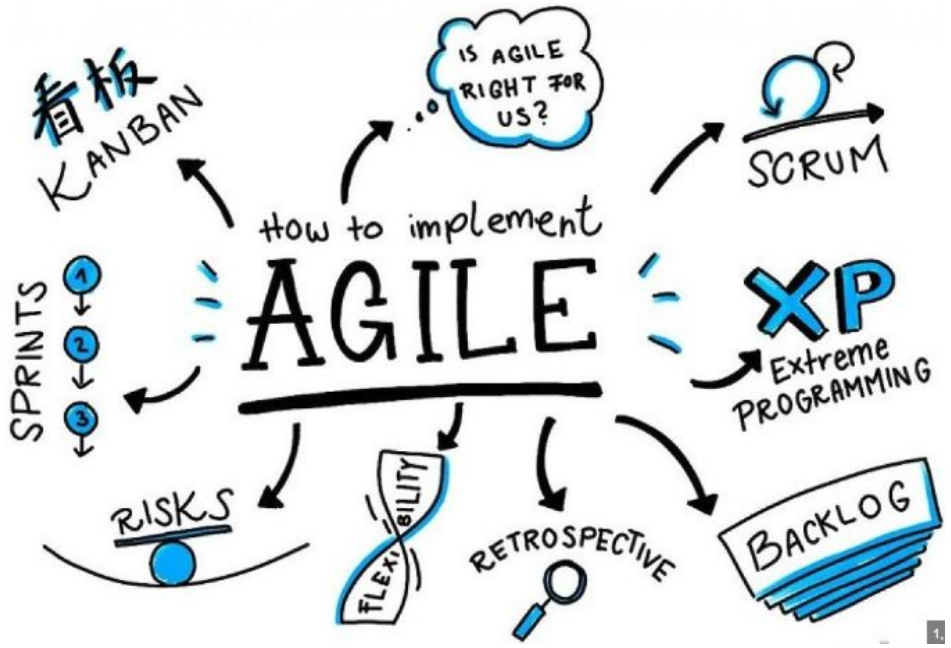
Especificaciones muy largas, de 10 años para poder implementar (Sistemas de control de un avión)

1990

Demasiada burocracia para desarrollar software, nacen las metodologías ágiles de la mano de los desarrolladores de software. Enfoque Incremental

En 2001 un grupo de 17 desarrolladores de software (Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas)

MANIFIESTO ÁGIL



El Manifiesto Ágil es un documento que se centra en los **4 valores y 12 principios** del desarrollo de software con metodologías ágiles. Lo publicaron, en febrero de 2001, 17 desarrolladores de software que necesitaban contar con una alternativa al proceso de desarrollo de productos más lineal y orientado a procesos.

4 VALORES

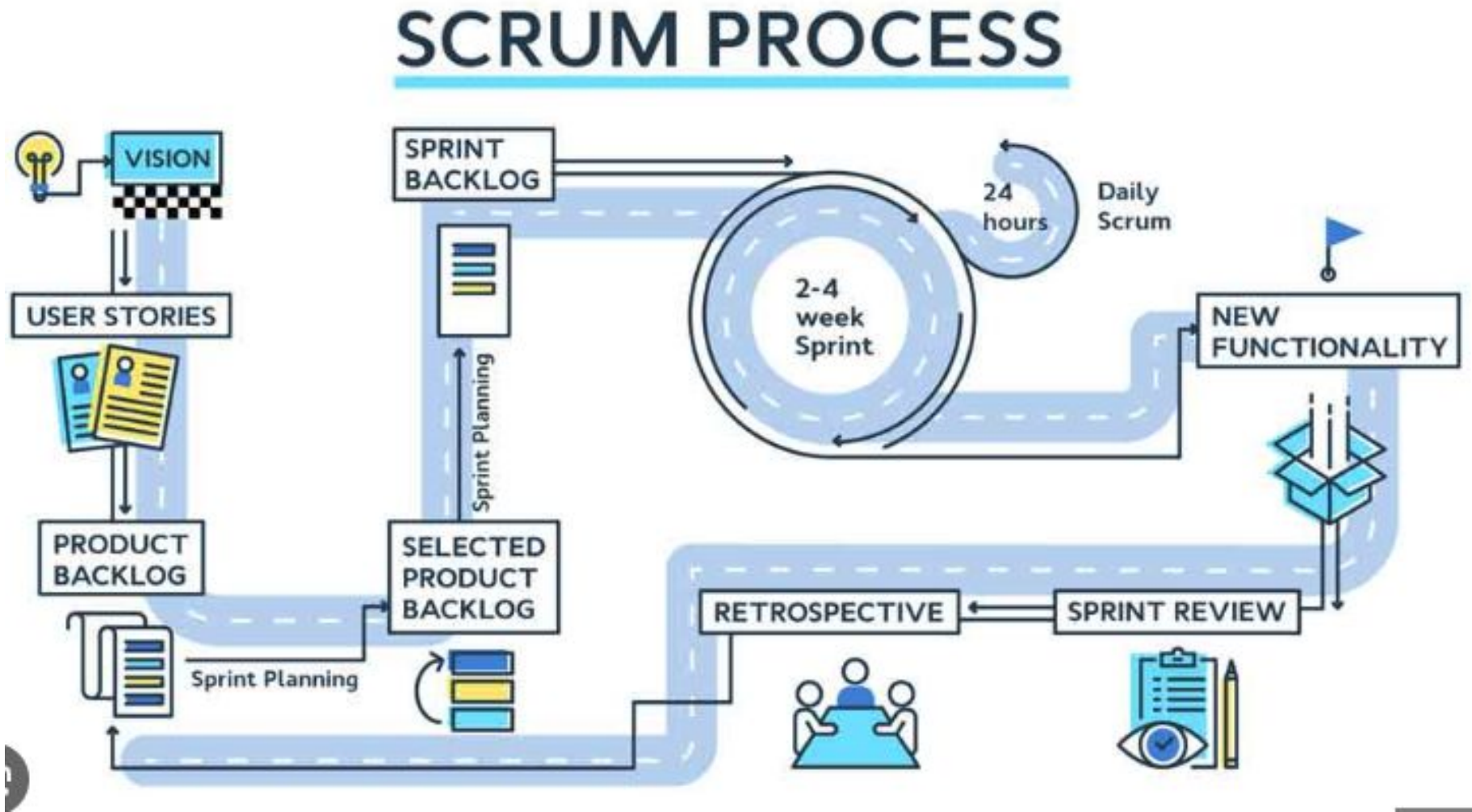
- Se buscan opciones para documentar más “ágiles”
- Conocer a tu equipo de trabajo, empatizar, conectar.
- Contratos flexibles (SOW)
- **COMUNICACIÓN**

12 PRINCIPIOS

12 Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

SCRUM


SCRUM PROCESS
VISION
USER STORIES
PRODUCT BACKLOG
Sprint Planning
SELECTED PRODUCT BACKLOG
SPRINT BACKLOG
Sprint Planning
2-4 week Sprint
24 hours
Daily Scrum
NEW FUNCTIONALITY
SPRINT REVIEW
RETROSPECTIVE

SCRUM

- El desarrollo incremental de los requisitos del proyecto en bloques temporales cortos y fijos.
- Se da prioridad a lo que tiene más valor para el cliente.
- El equipo se sincroniza diariamente y se realizan las adaptaciones necesarias.
- Tras cada iteración (un mes o menos entre cada una) se muestra al cliente el resultado real obtenido, para que este tome las decisiones necesarias en relación con lo observado.
- Se le da la autoridad necesaria al equipo para poder cumplir los requisitos.
- Fijar tiempos máximos para lograr objetivos. (Time To Market)
- Equipos pequeños (de 3 a 9 personas cada uno).

SCRUM MEETINGS (REUNIONES)

Daily Meeting:
Diariamente de no más de 15 minutos, en que estas? Algo que te esté bloqueando? Solo participan los desarrolladores, scrum master y opcional el Product Owner

Planning Meeting:
Se planifica de las tareas que hay en el backlog, se seleccionan, se refinan,se ajustan, se “miden” para que entren en 1 sprint. (Poker Planning, TShirt)

Retrospective Meeting:
Terminado el sprint revisamos si se llegó o no, cuáles fueron las causas, para poder corregirlo en el siguiente sprint.

SCRUM

Product Owner:
representa a los stakeholders del producto y los intereses del cliente, conoce el mercado y las prioridades del proyecto. Será el último responsable de la definición del backlog de producto y de la planificación y calendarización de las iteraciones

Scrum Master:
Responsable del equipo, quien supervisa los avances del desarrollo y garantiza que el equipo cuenta con las herramientas necesarias para el desempeño de su actividad. Sprint backlog. Organiza también las reuniones de trabajo

Development Team:
Formado por developers, testers, customers, etc.-. Se trata de un equipo multifunción y capaz de autogestionarse y de adaptarse rápidamente a los cambios.

KANBAN (Señal Visual en Japonés)

Enfoque visual para la aplicación de las prácticas ágiles. Los equipos usan herramientas en línea con tableros Kanban para representar ciertas tareas dentro del proceso de desarrollo. Las tareas se ven representadas en forma de tarjetas dentro de un tablero y las etapas, en columnas. A medida que los distintos miembros del equipo trabajan con esas tareas, las “tarjetas” pasan de la columna de trabajo pendiente a la que representa a la nueva etapa en la que se encuentra la tarea.

El método es muy bueno para que los equipos puedan identificar obstáculos y visualizar cuánto trabajo se está llevando a cabo.

Se utiliza mucho el tablero físico

KANBAN

Kanban es un método para “empezar con lo que haces ahora”. Esto quiere decir que no tienes que cambiar radicalmente lo que estás haciendo para empezar a usar kanban. El método kanban presupone tres cosas:

Que entendemos los procesos actuales, tal y como se llevan a cabo en la práctica, y que respetas las funciones, responsabilidades y cargos actuales.

Que te comprometes a perseguir la mejora continua a través del cambio evolutivo.

Que fomentan los actos de liderazgo en todos los niveles, desde cada uno de los colaboradores hasta la dirección sénior.

KANBAN y SCRUM

Los sprints de scrum tienen fechas de inicio y de finalización, mientras que kanban es un proceso continuo.

Las funciones del equipo están definidas claramente en scrum (propietario del producto, equipo de desarrollo y experto en scrum), mientras que kanban no tiene ninguna función formal. Ambos equipos están autoorganizados.

Un tablero de kanban se utiliza durante todo el ciclo de vida de un proyecto, mientras que un tablero de scrum se borra y se recicla después de cada sprint.

Un tablero de scrum tiene un número determinado de tareas y una fecha límite estricta para completarlas.

Los tableros de kanban son más flexibles en cuanto a tareas y plazos. Las tareas se pueden volver a jerarquizar por orden de prioridad, reasignar o actualizar según convenga.

XP - eXtreme Programming

Centra en la velocidad y la simplicidad con ciclos de desarrollo cortos y con menos documentación. La estructura del proceso está determinada por 5 valores fundamentales, 5 reglas y 12 prácticas de XP.

Se divide en sprints de trabajo. Los marcos ágiles siguen un proceso iterativo, en el que se completa y revisa el marco al final de cada sprint, refinando para adaptarlo a los requisitos cambiantes y alcanzar la eficiencia máxima. Al igual que otros métodos ágiles, el diseño de la programación extrema permite a los desarrolladores responder a las solicitudes de los clientes, adaptarse y realizar cambios en tiempo real. Sin embargo, la programación extrema es mucho más disciplinada; realiza revisiones de código frecuentes y pruebas unitarias para realizar cambios rápidamente. Además es muy creativa y colaborativa, ya que promueve el trabajo en equipo durante todas las etapas de desarrollo.

XP - REGLAS

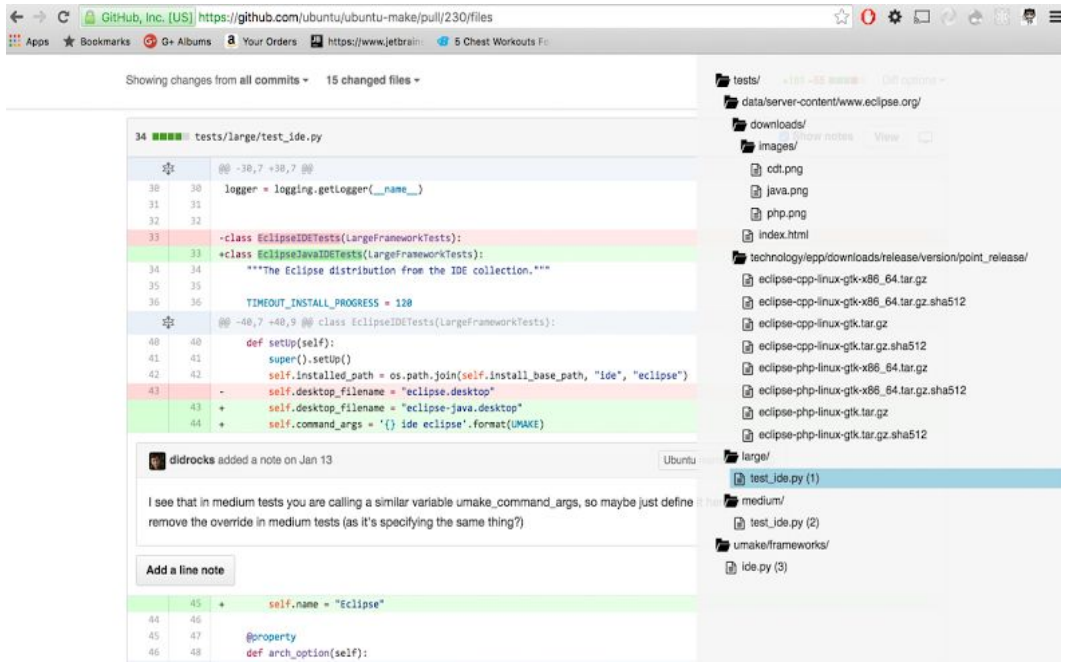
Planificación - Historias de usuarios simples, cliente disponible

Gestión - Espacio de trabajo - Ir a la oficina??

Diseño - CRC - Class Responsibility Collaboration

Codificación - Pair Programming (estándares de código)

Pruebas - Pruebas Unitarias!!!



XP - PRÁCTICAS

El juego de planificación: La planificación XP se usa para guiar el trabajo.

Pruebas de clientes: Cuando finalices una función nueva, el cliente desarrollará una prueba de aceptación para determinar si has cumplido con la historia de usuario original.

Pequeñas entregas: La programación extrema realiza entregas pequeñas y periódicas para obtener información importante durante todo el proceso. A menudo, las entregas se envían directamente a los clientes, aunque también pueden enviarse internamente.

Diseño simple: El sistema XP está diseñado para ser simple, producirá solo lo necesario y nada más.

Programación en parejas: Toda la programación la realizan simultáneamente dos desarrolladores que se sientan físicamente uno al lado del otro. No hay trabajo individual en la programación extrema.

Desarrollo guiado por pruebas (TDD): Debido a que la programación extrema se basa en los comentarios, se requieren pruebas exhaustivas. A través de ciclos cortos, los programadores realizan pruebas automatizadas para luego reaccionar de inmediato.

Refactorización: Aquí es donde se deberá prestar especial atención a los detalles más finos del código base, para eliminar los duplicados y asegurarse de que el código sea coherente. De esta manera obtendrás diseños simples y de alta calidad.

Propiedad colectiva: Cualquier par de desarrolladores puede modificar el código en cualquier momento, independientemente de que lo hayan desarrollado o no. En la programación extrema, la codificación se realiza en equipo, y el trabajo de todos se lleva a cabo según los estándares colectivos más altos.

Integración continua: Los equipos de XP no esperan a que se completen las iteraciones, sino que se integran constantemente. A menudo, un equipo de XP se integrará varias veces al día.

Ritmo de trabajo sostenible: La intensidad de los trabajos de XP requiere que se establezca un ritmo de trabajo sostenible

Metáfora: somos hormigas trabajando en colectivo para construir el hormiguero.

Estándares de codificación: Los equipos de XP siguen un estándar.

LEAN

Just in Time (JIT) - Ahorrar recursos y entregar funcionalidad cuando se necesita

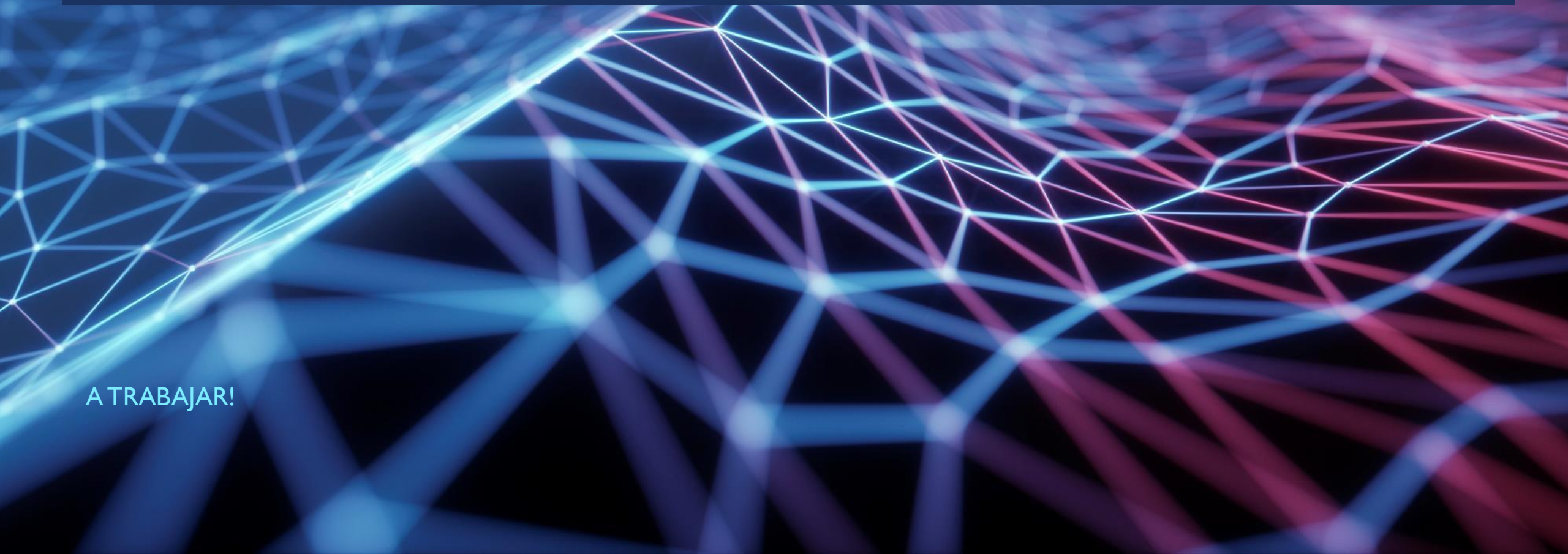
Lean fue creado y es usado por Toyota

Mejora continua - Optimizar procesos de manera proactiva.

Respeto

1. Identificar el valor
2. Mapa de flujo de valor
   a. Identificar problema y el equipo
   b. limitar y crear mapa del proceso
   c. Recopilar datos
   d. Evaluar y hacer ajustes
3. Crear estado de fluidez
4. Sistema de incorporación de requerimientos.
5. Buscar la perfección.

TRELLO - PROYECTOS



A TRABAJAR!