DESARROLLO DE SOFTWARE

Unidad Didáctica 1

Contenidos

Bloques:

I. El software del ordenador
   A. Clasificación del Software
   B. Software de Programación

II. **Fases de desarrollo de una aplicación informática**
   A. Ciclo de vida del software
   B. **Modelos de ciclo de vida**

Ciclo de vida

El estándar ISO/IEC 12207-1 define **ciclo de vida del software como..**

- Un marco de referencia que contiene los procesos, las actividades y las tareas involucradas en el desarrollo, la explotación y el mantenimiento de un producto software, abarcando la vida del sistema desde la definición de los requisitos hasta la finalización de su uso

Modelos de ciclo de vida

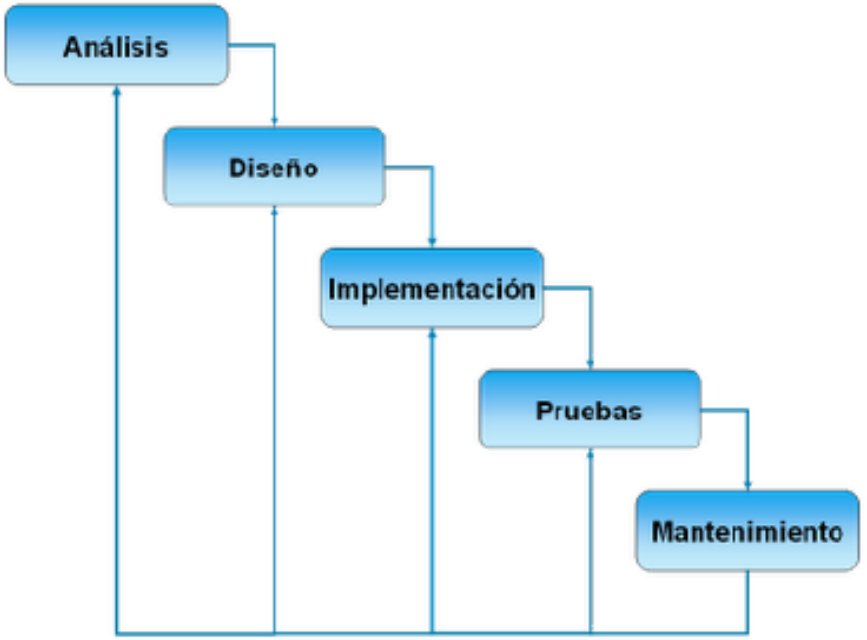
- Existen varios modelos de ciclo de vida, es importante tener en cuenta las características del proyecto software para elegir un modelo u otro
- Los modelos más importantes son:
  - Cascada
  - Iterativo Incremental
  - Espiral
  - Métodos Ágiles

https://es.wikiversity.org/wiki/Procesos_de_desarrollo_software

Ciclo de vida en cascada

- En este modelo las etapas para el desarrollo del software tienen un orden, de tal forma que para empezar una etapa es necesario finalizar la etapa anterior

- Después de cada etapa se realiza una revisión para comprobar si se puede pasar a la siguiente



Ciclo de vida en cascada

- El modelo permite hacer iteraciones:
  - si el cliente en la etapa mantenimiento solicita un cambio en el diseño, se volvería a la etapa de diseño y se pasaría de nuevo por codificación y pruebas

Ventajas	Inconvenientes
✓Fácil de comprender, planificar y seguir ✓La calidad de producto resultante es alta ✓Permite trabajar con personal poco cualificado	✓Necesidad de tener todos los requisitos definidos al principio (normalmente el cliente no sabe que quiere al principio) ✓Difícil vuelta atrás si hay errores ✓El producto no está disponible para su ✓uso hasta que no está terminado

Ciclo de vida en cascada

Se recomienda cuando:

- El proyecto es similar a alguno que se haya realizado con éxito anteriormente
- Los requisitos son estables y están bien comprendidos
- Los clientes no necesitan versiones intermedias

https://es.wikipedia.org/wiki/Desarrollo_en_cascada

Modelos Evolutivos de Ciclo de Vida

- Es usual que mientras se desarrolla el software los requisitos de los usuarios y del producto cambien
- Actualmente la gran competencia en el mercado y la urgencia de los usuarios para disponer del software obliga a desarrollar versiones del software no completas:
  - Versiones que posteriormente se irán perfeccionando y completando su funcionalidad
- En el modelo en cascada debemos entregar un producto completo, en cambio **el modelo evolutivo nos permite generar versiones cada vez más complejas** hasta llegar al producto final
- Los modelos evolutivos más conocidos son
  - **Iterativo incremental**
  - **Prototipos**
  - **Espiral**

Ciclo de Vida Iterativo Incremental

- Está basado en varios ciclos en cascada realimentados aplicados repetidamente
- Entrega el software en partes pequeñas pero utilizables, llamadas **incrementos**
- Cada incremento se construye sobre aquel que ya ha sido entregado

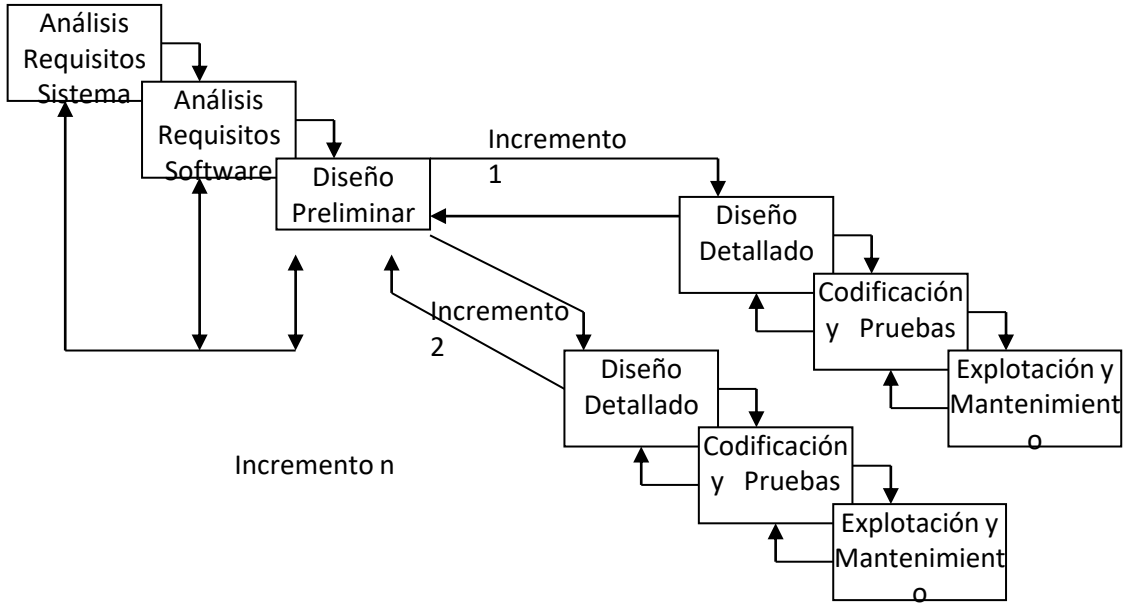


Figura 2.Modelo incremental

Ciclo de Vida Iterativo Incremental

Ventajas	Inconvenientes
✓No se necesitan conocer todos los requisitos al comienzo ✓Permite la entrega temprana al cliente de partes operativas del software ✓Las entregas facilitan la realimentación de los próximos entregables	✓Es difícil estimar el esfuerzo y el coste final necesario ✓Se tiene el riesgo de no acabar nunca ✓No recomendable par desarrollo de sistemas de tiempo real, de alto nivel de seguridad, de procesamiento distribuido y/o de alto índice de riesgos

Se **recomienda cuando**

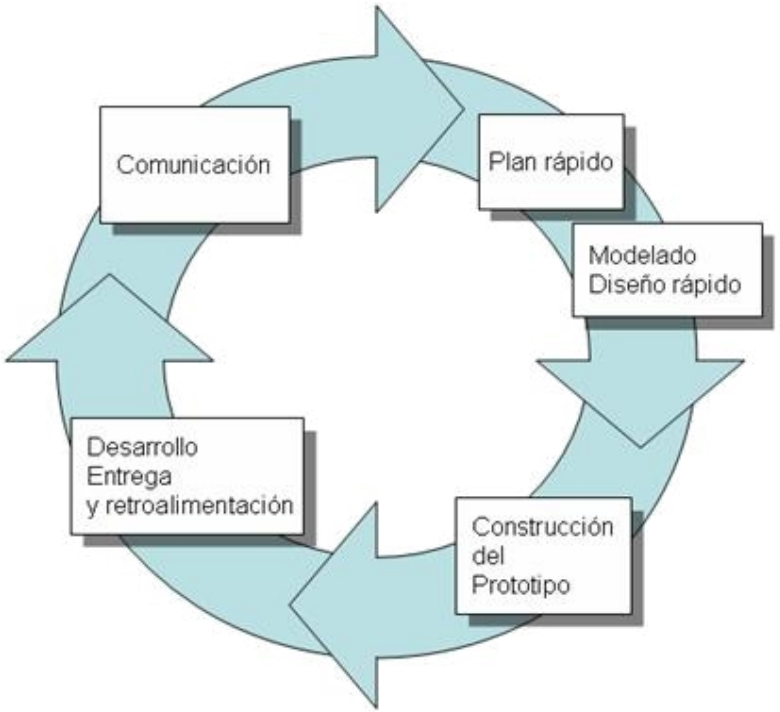
- Los requisitos o el diseño no están completamente definidos y es posible que haya grandes cambios
- Se están probando o introduciendo nuevas tecnologías

https://es.wikipedia.org/wiki/Desarrollo_iterativo_y_creciente

Construcción de prototipos

- **Prototipos:** software con funcionalidad parcial o incompleta

- La construcción de prototipos se suele usar como parte de otros modelos



Construcción de prototipos

- Tipos de prototipos:
  - **Prototipos rápidos**
    - El prototipo puede estar desarrollado usando otro lenguaje y/o herramientas
    - Finalmente el prototipo se desecha
  - **Prototipos evolutivos**
    - El prototipo está diseñado en el mismo lenguaje y herramientas del proyecto
    - El prototipo se usa como base para desarrollar el proyecto

Construcción de prototipos

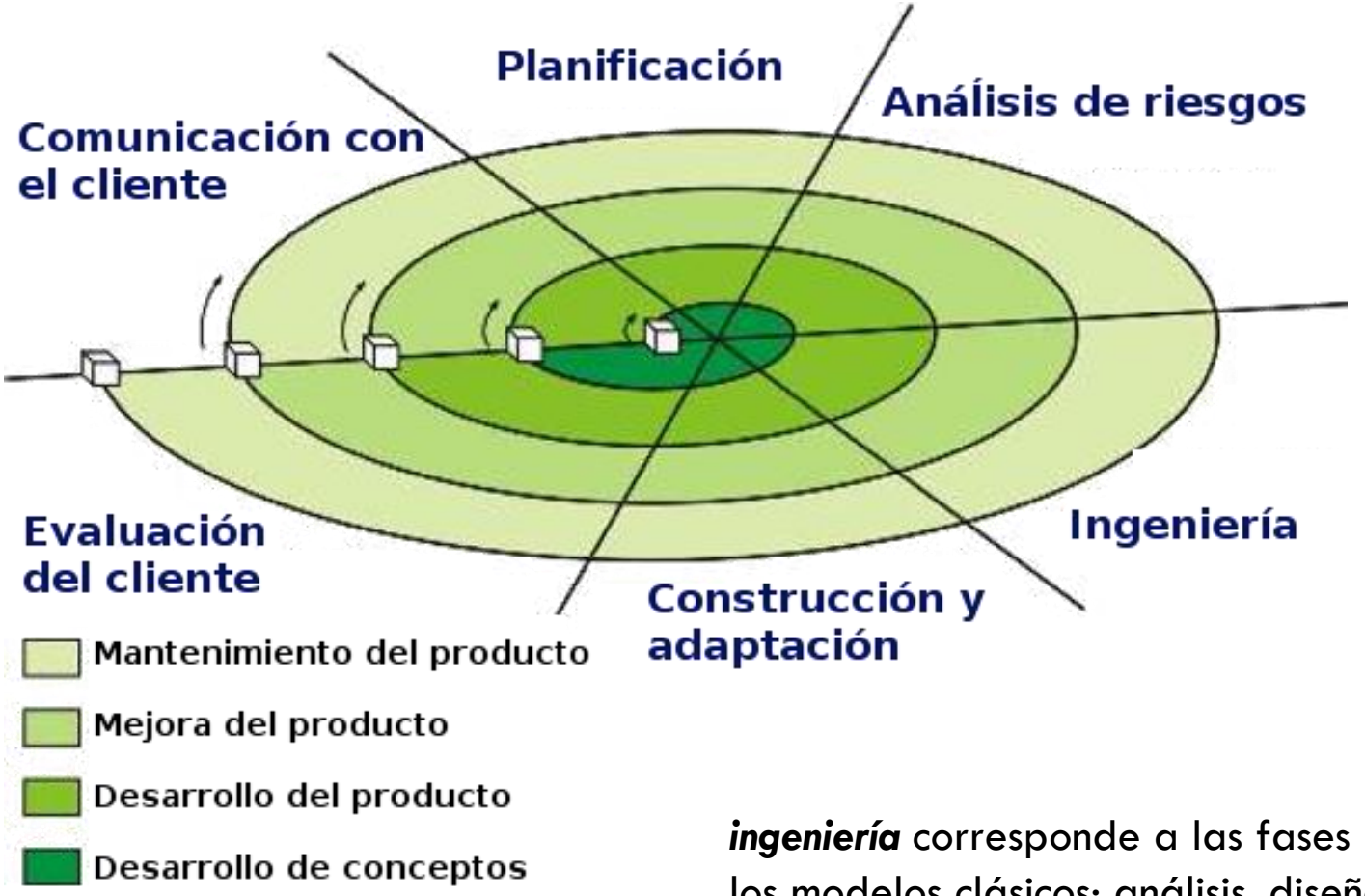
Problemas:

- Con demasiada frecuencia el prototipo pasa a ser parte del sistema final sin estar previsto en la planificación previa:
  - conduce a que éste contenga numerosos errores latentes, sea ineficiente, poco fiable, incompleto o difícil de mantener
  - se puede perder toda la documentación existente sobre los requisitos del sistema (si únicamente estaban recogidos en el prototipo y no existía otro tipo de documentación)

Ciclo de Vida en Espiral

- Este modelo combina el Modelo en Cascada con el modelo Iterativo con construcción de prototipos
- El proceso de desarrollo se representa como una espiral, donde en cada ciclo de la espiral se desarrolla una parte del mismo
- **Cada ciclo** está formado por fases y cuando termina **produce una versión incremental** del software con respecto al ciclo anterior
- En este aspecto se parece al Modelo Iterativo Incremental con la diferencia de que en cada ciclo se tiene en cuenta el **análisis de riesgos**

Ciclo de Vida en Espiral



ingeniería corresponde a las fases de los modelos clásicos: análisis, diseño, codificación, ...

Ciclo de Vida en Espiral

Para cada ciclo, los desarrolladores siguen estas fases:

- **Determinar objetivos:** cada ciclo de la espiral comienza con la identificación de los objetivos, alternativas para alcanzar los objetivos (diseño A, diseño B, reutilización, compra, etc.) y las restricciones impuestas
- **Analizar riesgos:** hay que evaluar las alternativas en relación con los objetivos y limitaciones. En este proceso se identifican los riesgos involucrados y (si es posible) la manera de solucionarlos
  - Un riesgo puede ser cualquier cosa: requisitos no comprendidos, mal diseño, errores en la implementación, etc.
- **Desarrollar y probar:** desarrollar la solución al problema en este ciclo y verificar que es aceptable
- **Evaluar y Comunicar:** revisar y evaluar todo lo que se ha hecho, y con ello decidir si se continúa
  - Si es así, hay que planificar las fases del ciclo siguiente

Ciclo de Vida en Espiral

Ventajas	Inconvenientes
✓No requiere una definición completa de los objetivos para empezar a funcionar ✓Análisis de riesgo en todas las etapas ✓ Reduce riesgos del proyecto ✓Incorpora objetivos de calidad	✓Es difícil evaluar los riesgos ✓El costo del proyecto aumenta a medida que la espiral pasa por sucesivas Iteraciones ✓El éxito del proyecto depende en gran medida de la fase de análisis de riesgos

Se **recomienda cuando**

- Proyectos de gran tamaño y que necesitan constantes cambios
- Proyectos donde sea importante el factor riesgo

https://es.wikipedia.org/wiki/Desarrollo_en_espiral

Modelos de Desarrollo Ágiles

Modelos Ágiles

- Son una respuesta a los fracasos y las frustraciones del modelo en cascada
- Contempla un enfoque para la toma de decisiones y la forma de organización en los proyectos de software, basándose en **los modelos de desarrollo iterativo e incremental, con iteraciones cortas (semanas) y sin que dentro de cada iteración tenga que haber fases lineales**
- En el 2001, 17 representantes de las nuevas tecnologías y el desarrollo de software se juntaron para discutir sobre el desarrollo de software. De esa reunión surgió el Manifiesto Para el Desarrollo Ágil:

Manifiesto por el Desarrollo Ágil de Software

Valoramos:

- … **al individuo y las interacciones en el equipo de desarrollo** más que a las actividades y las herramientas
- … **desarrollar software que funciona** más que conseguir una buena documentación
- … **la colaboración con el cliente** más que la negociación de un contrato
- … **responder a los cambios** más que seguir estrictamente una planificación

http://agilemanifesto.org/iso/es/manifesto.html

Principios de las Metodologías Ágiles (1/4)

1. La prioridad principal es satisfacer al cliente mediante tempranas y continuas entregas de software utilizable.
2. Dar la bienvenida a los cambios. Los procesos ágiles aplican los cambios para que el cliente sea competitivo.
3. Entregar el software desarrollado frecuentemente con el menor intervalo de tiempo posible entre una entrega y la siguiente.

Principios de las Metodologías Ágiles (2/4)

4. La gente de negocios y los desarrolladores trabajan juntos a través de un proyecto.

5. Construir proyecto empujados por motivaciones personales. Dar el entorno que necesitan las personas y confiar en ellos.

6. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.

Principios de las Metodologías Ágiles (3/4)

7. Desarrollar software es la primera medida de progreso.
8. Los procesos ágiles promueven un desarrollo llevadero. Los patrocinadores, desarrolladores y usuarios son capaces de mantener una paz constante.
9. La atención continua a la calidad técnica y al buen diseño incrementa la agilidad.

Principios de las Metodologías Ágiles (4/4)

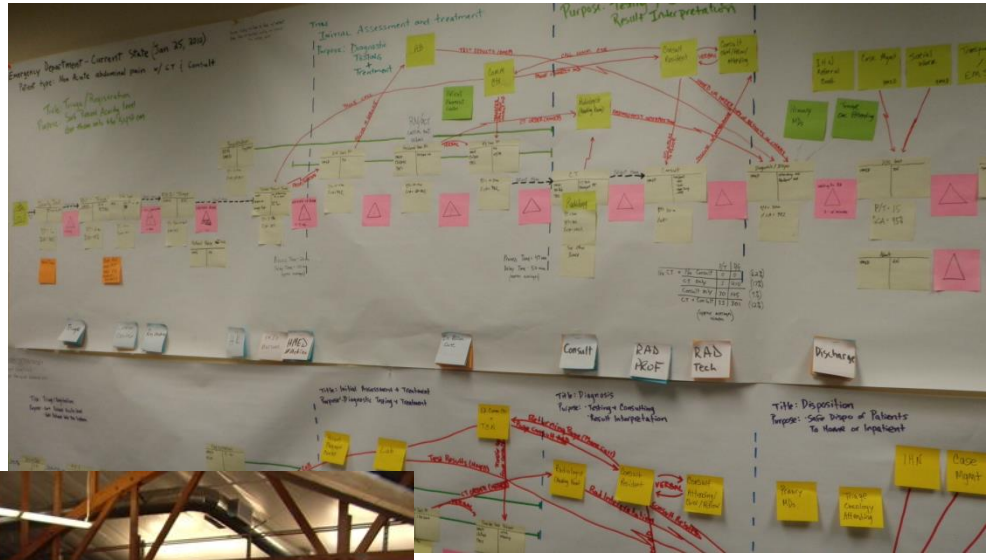
10. La simplicidad es esencial.

11. Las mejores arquitecturas, requisitos y diseños surgen de la propia organización del equipo.

12. En intervalos regulares, el equipo reflexiona en como llegar a ser más efectivo, sincronizar y ajustar su comportamiento.

http://agilemanifesto.org/iso/es/principles.html

Un día en la vida de un proyecto ágil



Estrategias de las Metodologías Ágiles

- Habilidad para adaptarse a los cambios
  - La superación de obstáculos imprevistos tiene prioridad sobre las reglas generales de trabajo preestablecidas.
  - Es decir, se para la forma habitual de trabajo para resolver un problema
- Los equipos deben ser pequeños, típicamente menores de siete personas
  - Todo el equipo se reúne periódicamente, incluidos usuarios
  - Las reuniones tienen hora de comienzo y de final y son breves
- Todo el equipo trabaja unido, y el cliente es parte de él. No es un oponente
  - La jerarquía clásica (director técnico, analista de negocio, arquitecto, programador senior, junior …) pierde sentido y los roles se disponen sobre un eje horizontal
  - La estrategia ya no es el control sino la colaboración y la confianza
  - Cada cual cumple su cometido, pero sin estar por encima ni por debajo de los demás

Estrategias de las Metodologías Ágiles

- Existen las fases de análisis, desarrollo y pruebas pero, en lugar de ser consecutivas, están solapadas y se suelen repetir en cada iteración
  - Las iteraciones suelen durar de dos a seis semanas
- En cada iteración se habla con el cliente para analizar requerimientos, se escriben pruebas automatizadas, se escriben líneas de código nuevas y se mejora código existente (Refactorización)
  - Al cliente se le enseñan los resultados después de cada iteración para comprobar su aceptación e incidir sobre los detalles que se estimen oportunos
- En lugar de trabajar por horas, se trabaja por objetivos y se usa el tiempo como un recurso más
  - PERO existen fechas de entrega para cada iteración
- La aplicación se ensambla y se despliega a diario, de forma automatizada
  - Las baterías de test se ejecutan varias veces al día
  - Los desarrolladores envían sus cambios al repositorio de código fuente al menos una vez al día (commit)

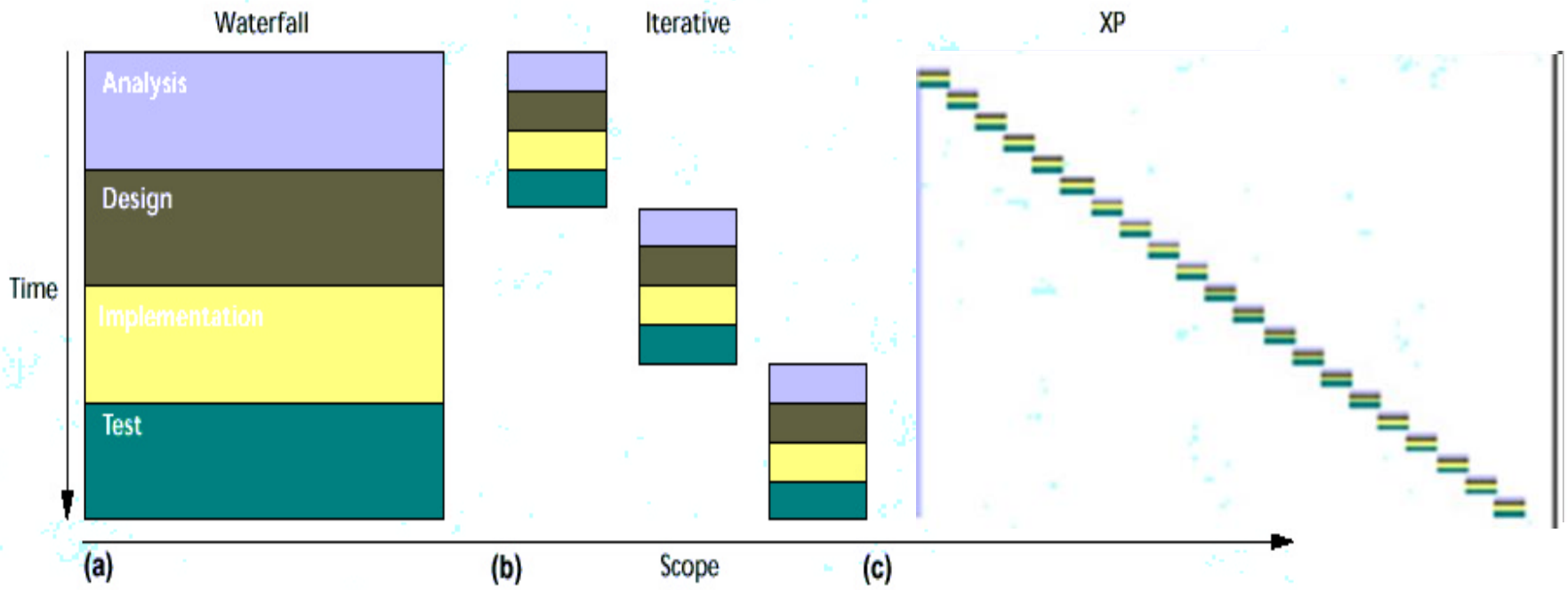
Metodologías Ágiles vs. Clásicas (1/2)

Metodología Ágil	Metodología No Ágil
El cliente es parte del equipo de desarrollo (además *in-situ*)	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (< 10 integrantes) y trabajando en el mismo sitio	Grupos grandes
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura	La arquitectura es esencial

Metodologías Ágiles vs. Clásicas (1/2)

Metodología Ágil	Metodología No Ágil
Tolerante a los cambios	Resistente a los cambios
Impuestas internamente (equipo)	Impuestas externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe un contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado

Metodologías Ágiles vs. Clásicas

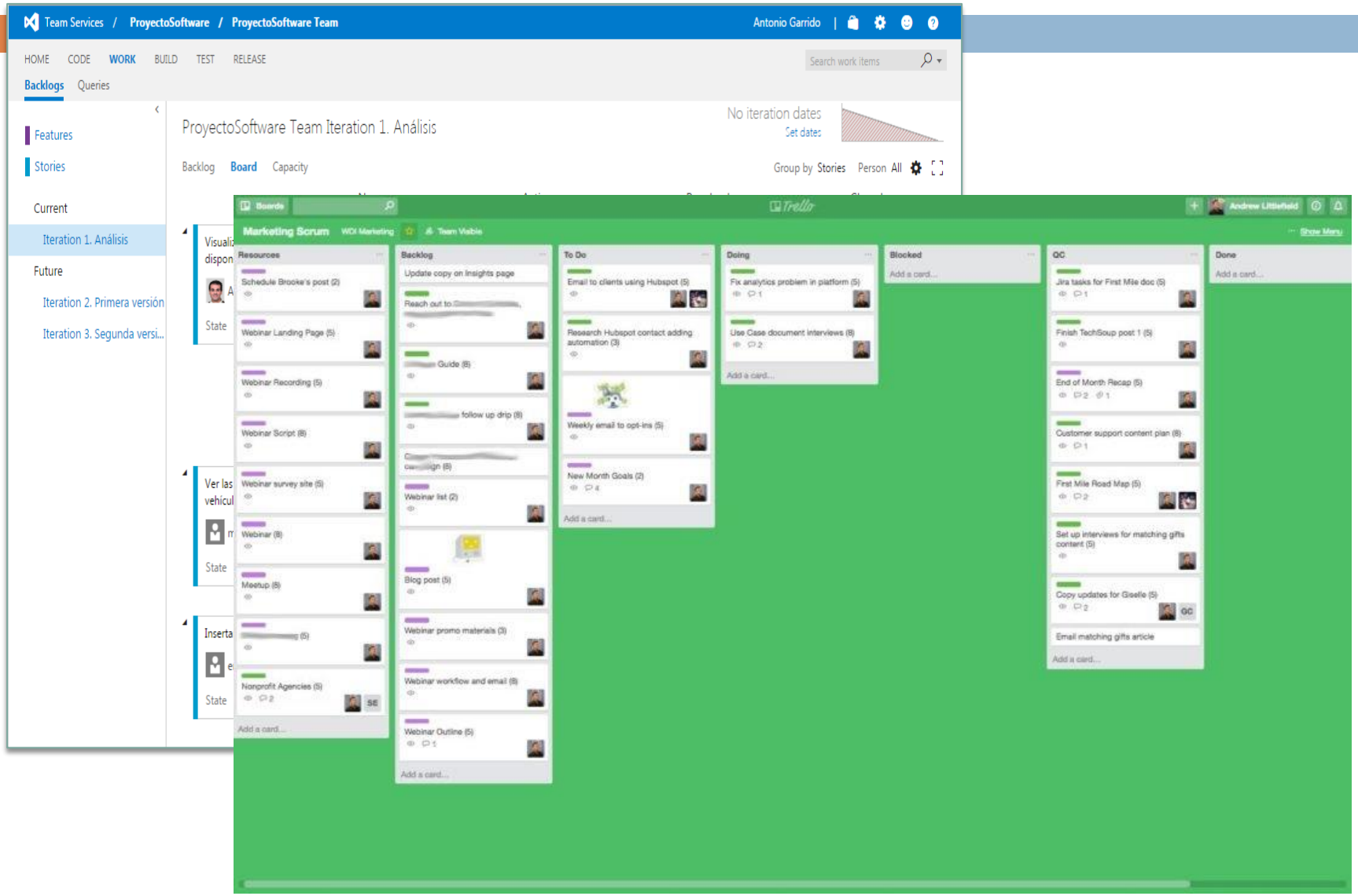


Metodologías Ágiles

- Las metodologías más conocidas son:
  - Kanban
  - Scrum
  - XP (eXtreme Programming)

- Herramientas:
  - Trello
  - KebanFlow
  - …

Herramientas



Desarrollo Ágil de Software

Documéntate sobre el tema:

- **https://proyectosagiles.org/fundamentos-de-scrum/**
- https://es.wikiversity.org/wiki/Scrum
- https://es.wikiversity.org/wiki/Extreme_Programming