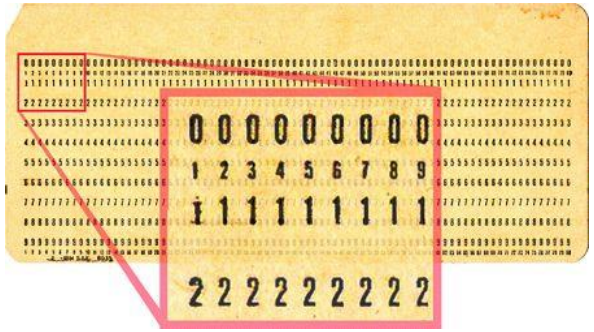
Introducción al desarrollo software

Metodologías de desarrollo software

Historia del software (1930-50).

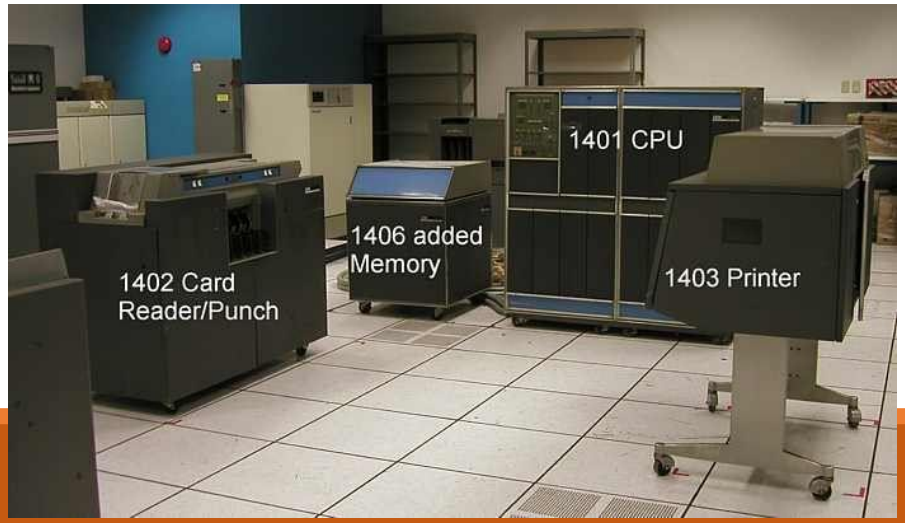
- Desde la década de 1930 hasta la década de 1950, las tarjetas perforadas se convirtieron en la fuerza motriz de las empresas, ya que se utilizaron en prácticamente todas las máquinas de contabilidad de oficina. Las tarjetas fueron creadas con lenguajes de programación como FORTRAN de IBM y COBOL del Departamento de Defensa de EEUU.
- Proyecto ENIAC.



Historia del software (1950-60).

- El término «software» se creó a finales de 1950 y pronto fue adoptado por toda la industria. Dividiendo el software en dos tipos principales: software de sistema y programa aplicaciones.
  - Software del sistema incluye los procesos generales de la ejecución del programa, tales como compiladores y sistema operativo de disco.
  - Aplicaciones del programa como pueden ser aplicaciones de oficina.
- Mainframe IBM 1401

  (1959)



Crisis del software (1960-80)

- A finales de los 60, la potencia de las maquinas empezó a aumentar de forma considerable. Empezaron a aparecer los lenguajes de programación de alto nivel, y las maquinas necesitaban programas mucho más complejos de los desarrollados hasta la época.
- En definitiva, fue un salto tremendo en cuanto a potencial de hardware, que no fue acompañado por un salto en el desarrollo de software.

Crisis del software (1960-80)

- En esta época, se empezó a concebir el Software como producto, y se empezaron a desarrollar algunos proyectos para que funcionaran en las máquinas de la época.
- Pero aparecieron importantes problemas:
  - Los productos excedían la estimación de costes.
  - Había retrasos en las entregas.
  - Las prestaciones no eran las solicitadas.
  - El mantenimiento se hacía extremadamente complicado y a veces imposible, las modificaciones tenían un coste prohibitivo…

Crisis del software

- La Crisis del software fue acuñada principios de los años 70, cuando la ingeniería de software era prácticamente inexistente.
- El término expresaba las dificultades del desarrollo de software frente al rápido crecimiento de la demanda por software, de la complejidad de los problemas a ser resueltos y de la inexistencia de técnicas establecidas para el desarrollo de sistemas que funcionaran adecuadamente o pudieran ser validados.
- Historia del Altair 8800

  (1974)

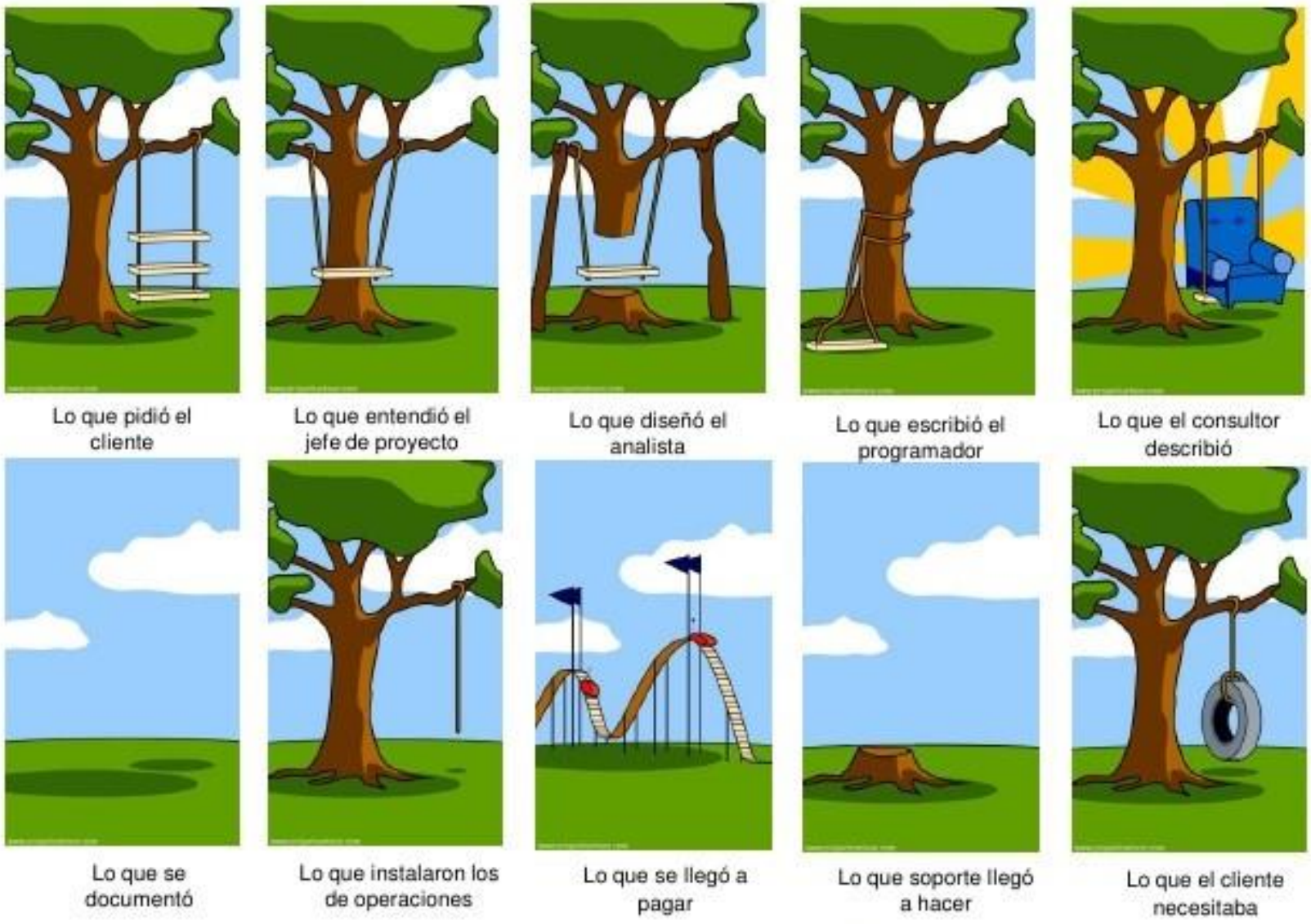
Causas de la crisis del software.

- Una de las principales causas de todo esto, si no la principal, era el enfoque dado al proceso de desarrollo de software, el cual era malo e incluso a veces era inexistente.
- En este proceso, solo ¼ del tiempo de desarrollo se dedicaba a las fases de análisis, diseño, codificación y pruebas, y más de ¾ del tiempo se dedicaba a correcciones y mantenimiento.
- Es evidente que dedicándole sol ¼ del tiempo a las primeras fases, se arrastran errores graves, sobre todo procedentes de las fases de análisis y diseño, lo que dificultaba muchísimo la implementación, produciendo constantes paradas y retrocesos para revisar este análisis/diseño.

Proyectos Fallidos en la Crisis del Software.

- **Accidente de un F-18 (1986):** En abril de 1986 un avión de combate se estrelló por culpa de un giro descontrolado atribuido a una expresión “if then”, para la cual no había una expresión “else”, debido a que los desarrolladores del software lo consideraron innecesario.
- **Muertes por el Therac-25 (1985-1987):** El Therac-25 fue una máquina de radioterapia que causó la muerte de varios pacientes en diversos hospitales de Estados Unidos y Canadá, debido a las radiaciones de alto poder aplicadas sin control, las cuales fueron atribuidas a la falta de control de calidad del software médico.
- **Sobrecosto, retraso y cancelación en el sistema del Bank of America (1988):** En el año de 1988, este banco invirtió 23 millones de dólares en un sistema computarizado llamado MasterNet, el cual servía para contabilidad y reportes de fideicomisos. No obstante, para que el sistema funcionara, se tuvo que invertir 60 millones de dólares más, por lo que finalmente el sistema fue cancelado.

Poco tiempo + mala comunicación



Nacimiento de la Ingeniería del Software.

- La primera conferencia sobre Ingeniería de Software fue allá por 1968, en Múnich, financiada por la OTAN. Allí fue donde se adoptó el término, hasta entonces prácticamente desconocido, de «ingeniería de software», y quien primero lo usó fue Fritz Bauer.
- La **Ingeniería del Software** es una disciplina que se ocupa de proporcionar un marco adecuado para la gestión de los proyectos software, y para el diseño de aplicaciones empresariales con una arquitectura software que favorezca su mantenimiento y evolución a lo largo del tiempo.

Ingeniería del software

- La ingeniería del software abarca un conjunto de tres elementos clave:
  - Métodos
  - Herramientas
  - Procedimientos
- Estos facilitan la gestión del proceso de desarrollo y suministran a los desarrolladores bases para construir de forma productiva software de alta calidad.

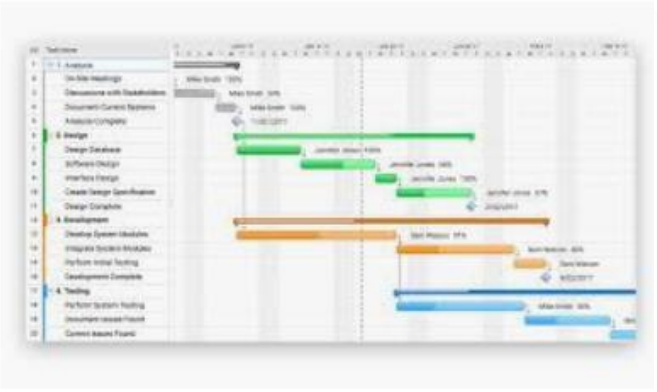
Métodos

- Indican cómo construir técnicamente el software, y abarcan una amplia serie de tareas que incluyen la planificación y estimación de proyectos, el análisis de requisitos, el diseño de estructuras de datos programas y procedimientos, la codificación, las pruebas y el mantenimiento.
- Los métodos introducen frecuentemente una notación específica para la tarea en cuestión y una serie de criterios de calidad.
- Ejemplo: Cuestionarios, entrevistas, grupos de trabajos …

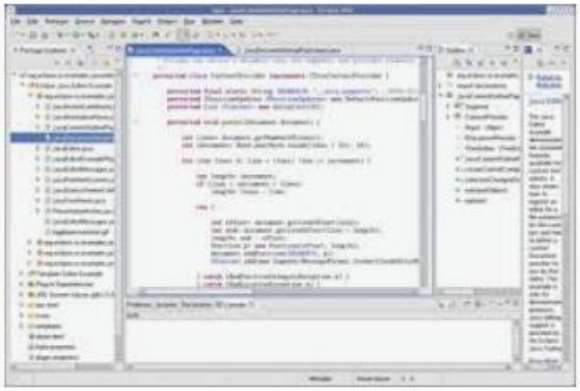
Herramientas

- Las **herramientas** proporcionan un soporte automático o semiautomático para utilizar los métodos. Existen herramientas automatizadas para cada una de las fases vistas anteriormente, y sistemas que integran las herramientas de cada fase de forma que sirven para todo el proceso de desarrollo.

MagicDraw



Microsoft Project Integration - ProjectManager.com



Eclipse desktop & web IDEs | The Eclipse Founda...

Procedimientos

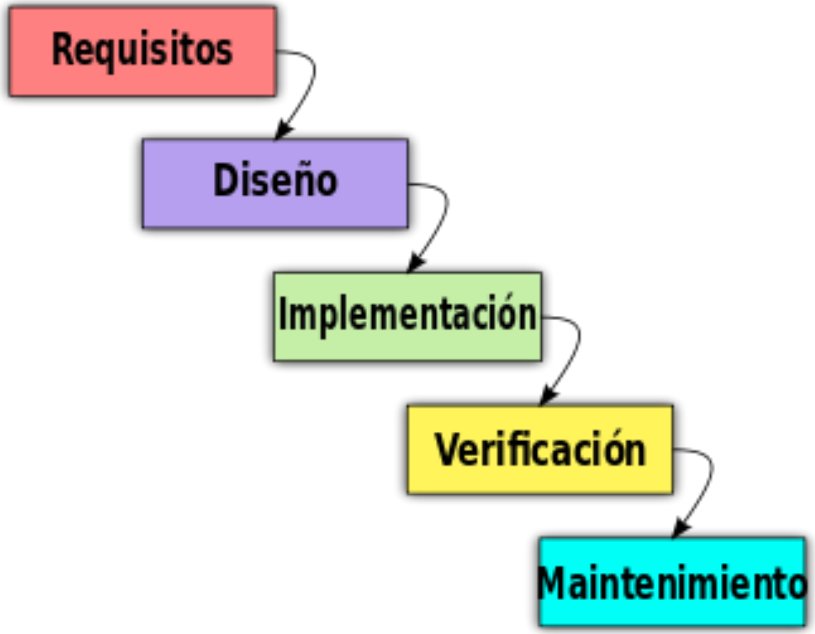
- Definen la secuencia en que se aplican los métodos, los documentos que se requieren, los controles que permiten asegurar la calidad y las directrices que permiten a los gestores evaluar los progresos.
- En los procesos se distinguen 4 fases básicas:
  - **Análisis:** Proceso de reunión de requisitos funcionales y no funcionales de un sistema.
  - **Diseño:** Se refiere ala establecimiento de las estructuras de datos, la arquitectura general de software, interfaz…
  - **Implementación:** traducimos el diseño en un forma legible por la máquina. Lo programamos.
  - **Pruebas:** Una vez generado el software comienzan las pruebas. Para demostrar que no es encuentran fallos.

Ciclos de vida

- Un ciclo de vida es el conjunto de fases por las que un sistemas software pasa desde que surge la idea hasta que muere.
- De los ciclos de vida destacan ciertos aspectos claves que son: las fases , las transiciones entre fases y las posibles entradas y salidas que surgen en cada transición.
- Ejemplos de Ciclos de Vida:
  - Ciclo de vida en Cascada
  - Ciclo de vida en V
  - Ciclo de vida Incremental.

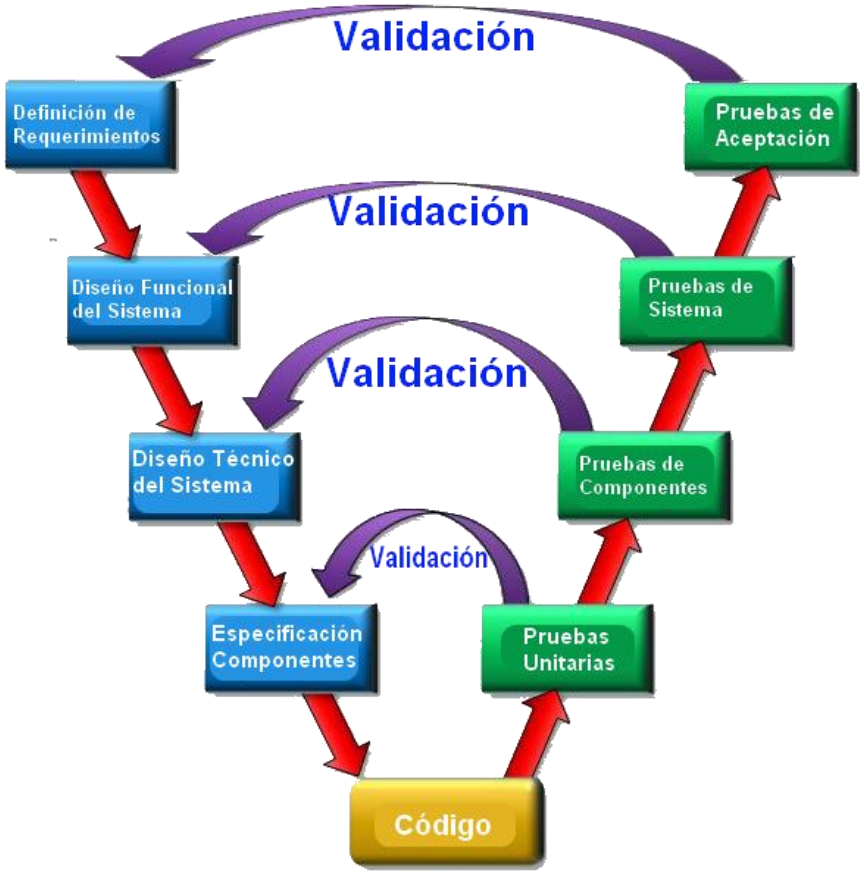
Ciclos de vida en Cascada

- Ciclo de vida en las distintas fases se van encadenado las etapas de manera consecutiva.
- Este ciclo de vida comprendía la retroalimentación entre fases y la posibilidad de volver atrás.
- Sin embargo esto no se llevó a la práctica lo que ocasionó que no se supiera reaccionar a errores ocurridos en una fases avanazadas.



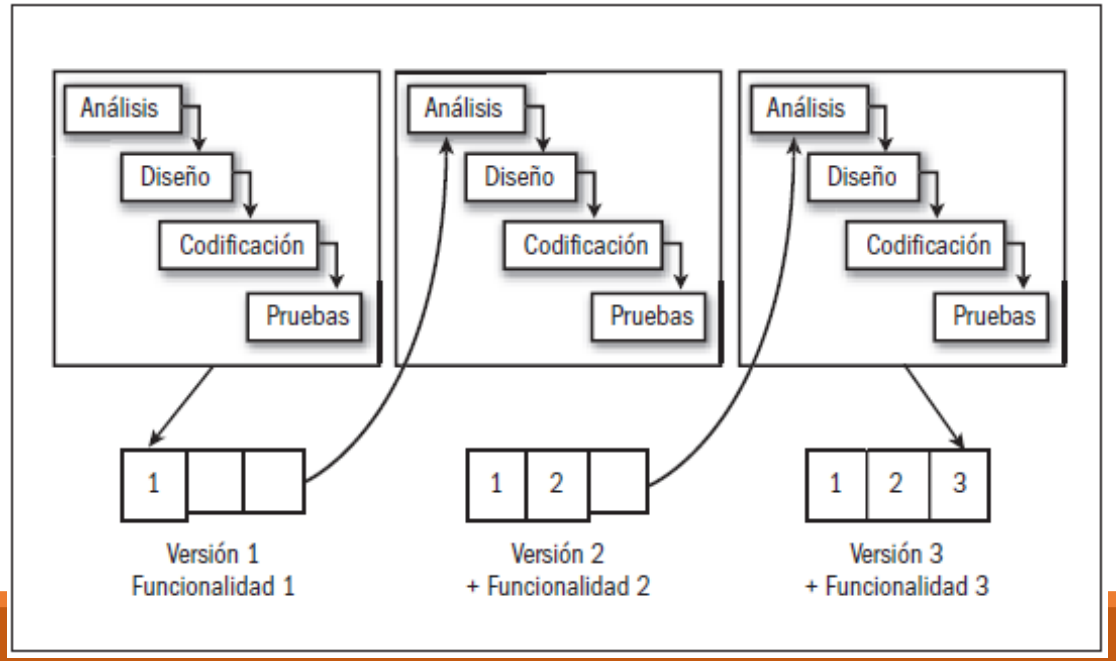
Ciclos de vida en V.

- El modelo en V viene para corregir los fallos del modelo en cascada, incluyendo explícitamente la retroalimentación.
- El problema es que los fallos detectados en fases tardías de un proyecto obligaban a empezar de nuevo, lo cual supone un gran sobrecoste.



Ciclos de vida Incremental.

- Este modelo busca reducir el riesgo que surge entre las necesidades del usuario y el producto final por malos entendidos durante la etapa de solicitud de requerimientos.
- El ciclo se compone de iteraciones. Iteraciones compuestas por las fases básicas.
- Al final de cada iteración se le entrega al cliente una versión mejorada o con mayores funcionalidades del producto.
- El cliente al finalizar cada iteración, evalúa el producto y lo corrige o propone mejoras.
- Estas iteraciones se repetirán hasta que se desarrolle un producto que satisfaga las necesidades del cliente.

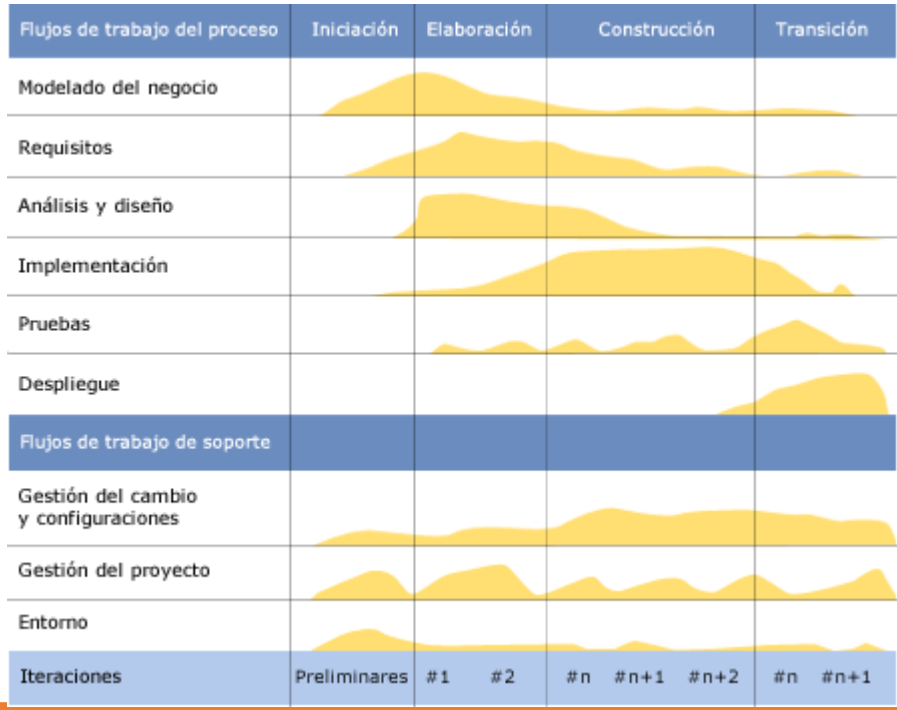


Metodologías.

- Metodología esta compuesta por modos sistemáticos de realizar, gestionar y administrar un proyecto a través de etapas y acciones, partiendo desde las necesidades del producto hasta cumplir con el objetivo para el que fue creado.
- Distinguimos:
  - Metodologías Tradicionales:
    - RUP
  - Metodologías Ágiles
    - XP
    - SCRUM

RUP (Proceso unificado de rational)

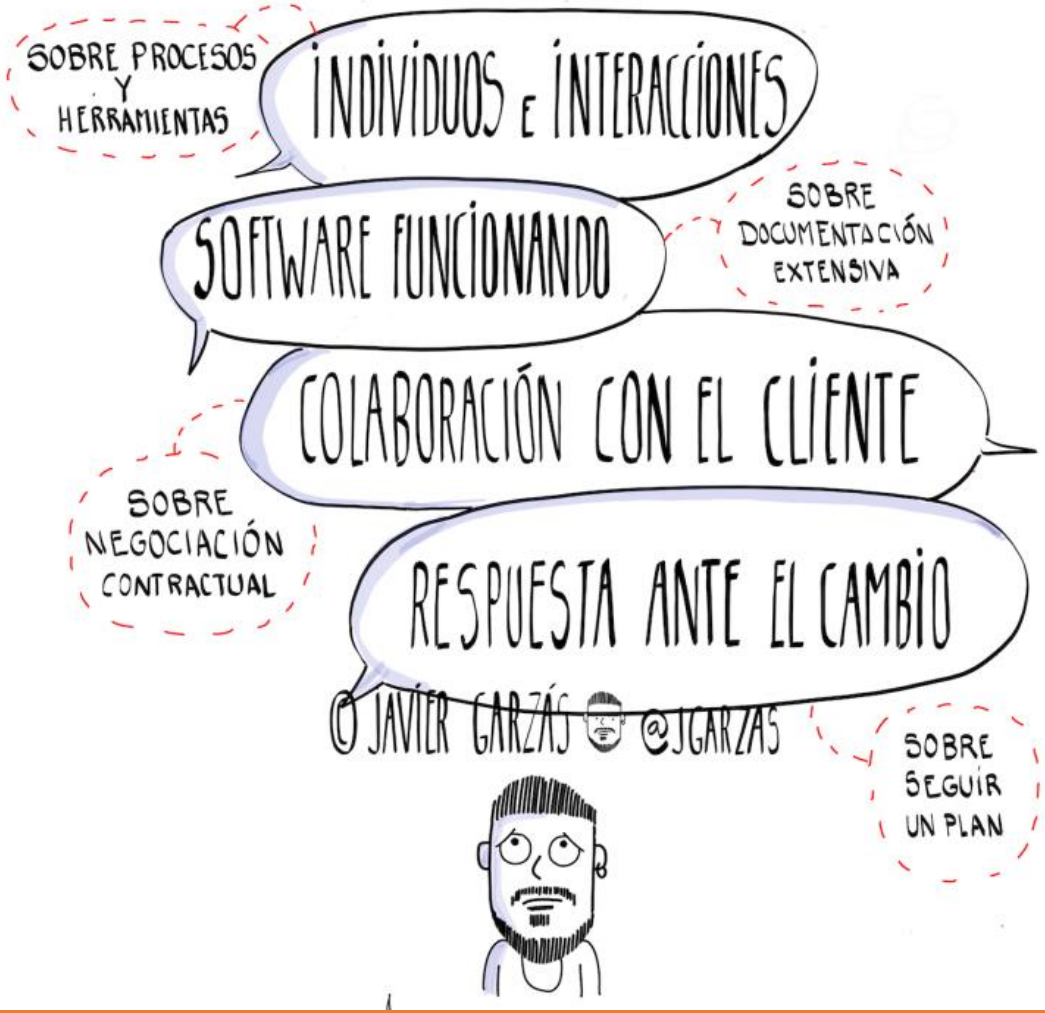
- Metodología Orientada a Objetos.
- **Dirigido por casos de uso:** los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).
- **Centrado en la arquitectura:** Se definen los Casos de Usos principales, los cimientos de la aplicación y sobre ello se va incrementando.
- **Iterativo e Incremental:** Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.
- Uso extensivo de documentación.



Manifiesto Ágil

- El 12 de febrero de 2001 diecisiete críticos de los modelos de mejora del desarrollo de software basados en procesos, convocados por Kent Beck, se reunieron en Snowbird, Utah para tratar sobre técnicas y procesos para desarrollar software.
- En la reunión se acuñó el término “Métodos Ágiles” para definir a los métodos que estaban surgiendo como alternativa a las metodologías formales (CMMI, SPICE) a las que consideraban excesivamente “pesadas” y rígidas por su carácter normativo y fuerte dependencia de planificaciones detalladas previas al desarrollo.
- Los integrantes de la reunión resumieron los principios sobre los que se basan los métodos alternativos en cuatro postulados, lo que ha quedado denominado como **Manifiesto Ágil**.

Manifiesto Ágil



Metodologías Ágiles

- Las **metodologías ágiles** son aquellas que permiten adaptar la forma de trabajo a las condiciones del proyecto, consiguiendo flexibilidad e inmediatez en la respuesta para amoldar el proyecto y su desarrollo a las circunstancias específicas del entorno.
- Metodologías ágiles:
  - XP
  - SCRUM

XP.- eXtreme Programming

- Se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad.
- Consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos.
  - El cliente no sabe lo que quiere hasta que ve lo que no quiere.
- Esta metodología lleva las buenas prácticas a niveles extremos. Por ejemplo el TDD (test driven development) fue usado en la NASA en el proyecto mercurio en los 60.

XP.- 4 Valores

- **Comunicación**: dentro en un equipo de desarrollo es fundamental para que la información entre los miembros fluya con normalidad, esto ayudará a evitar errores y agilizar las tareas compartidas
- **Retroalimentación (Feedback):**con el cliente es de gran ayuda entregar un software de calidad que cumpla las expectativas del mismo cliente. Para ello, el desarrollo debe realizarse en ciclos cortos que permitan mantener una retroalimentación constante y continua.
- **Simplicidad**: Cuando se trabaja en el diseño de una historia de usuario, es importante centrarse en esa tarea exclusivamente para mantener un código limpio y sencillo. Esto ayuda a evitar el *over-engineering*, es decir, no se debe preparar software "por si acaso"
- **Coraje** ser valiente para evitar over-engineering, para desechar un código fuente antiguo inservible, para refactorizar código que funcione o para sobreponerse de problemas que lleva tiempo sin resolverse.

XP.- Prácticas

- Planificación:
  - Historias de usuario
  - Plan de entregas
  - Plan de Iteraciones
  - Reuniones diarias
- Diseño
  - Simplicidad
  - Soluciones
  - Refactorización
  - Metáforas
- Desarrollo de código
  - Disponibilidad del cliente
  - Uso de estándares
  - Programación dirigidas por pruebas
  - Programación por pares.
  - Integración continua.
  - Propiedad colectiva del código
  - Ritmo sostenido.
- Pruebas
  - Pruebas unitarias
  - Detección y corrección de errores.
  - Pruebas de aceptación.