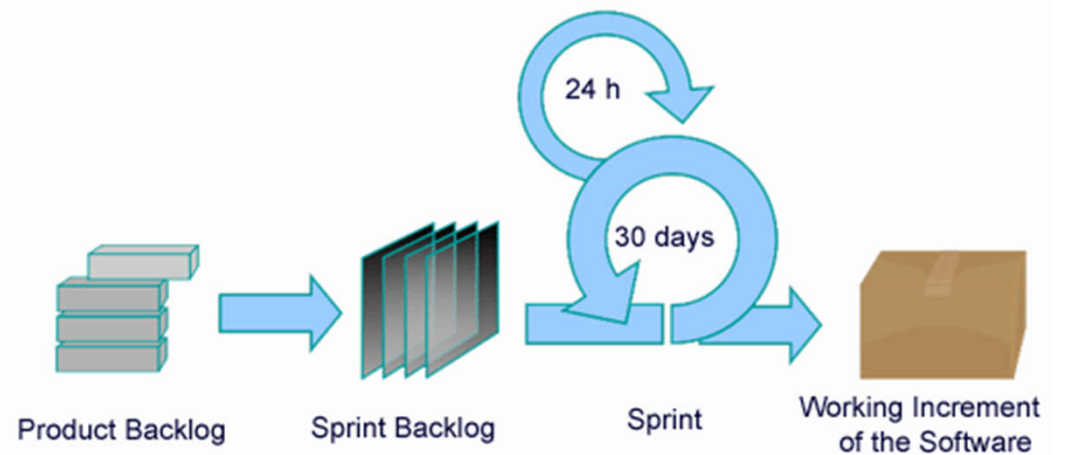




# **Desarrollo de software Metodología**

Carlos Fontela  
cfontela@fi.uba.ar

# Desarrollo y Metodología



# Temario

Disciplinas del desarrollo

Problemas de los proyectos de desarrollo de software

Metodología y tipos de métodos

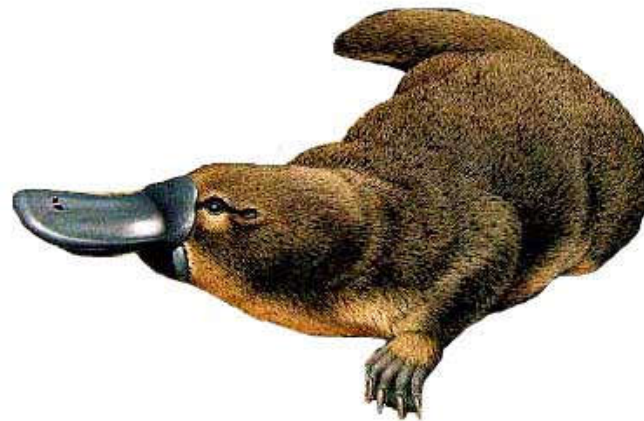
... “puente” entre áreas: Programación e Ingeniería de Software



# ¿Mitos o hechos?

Desarrollo de software es lo mismo que programación

Entre las ingenierías, la de software es la de peor reputación en cuanto a la satisfacción de los clientes, al cumplimiento de plazos y presupuestos



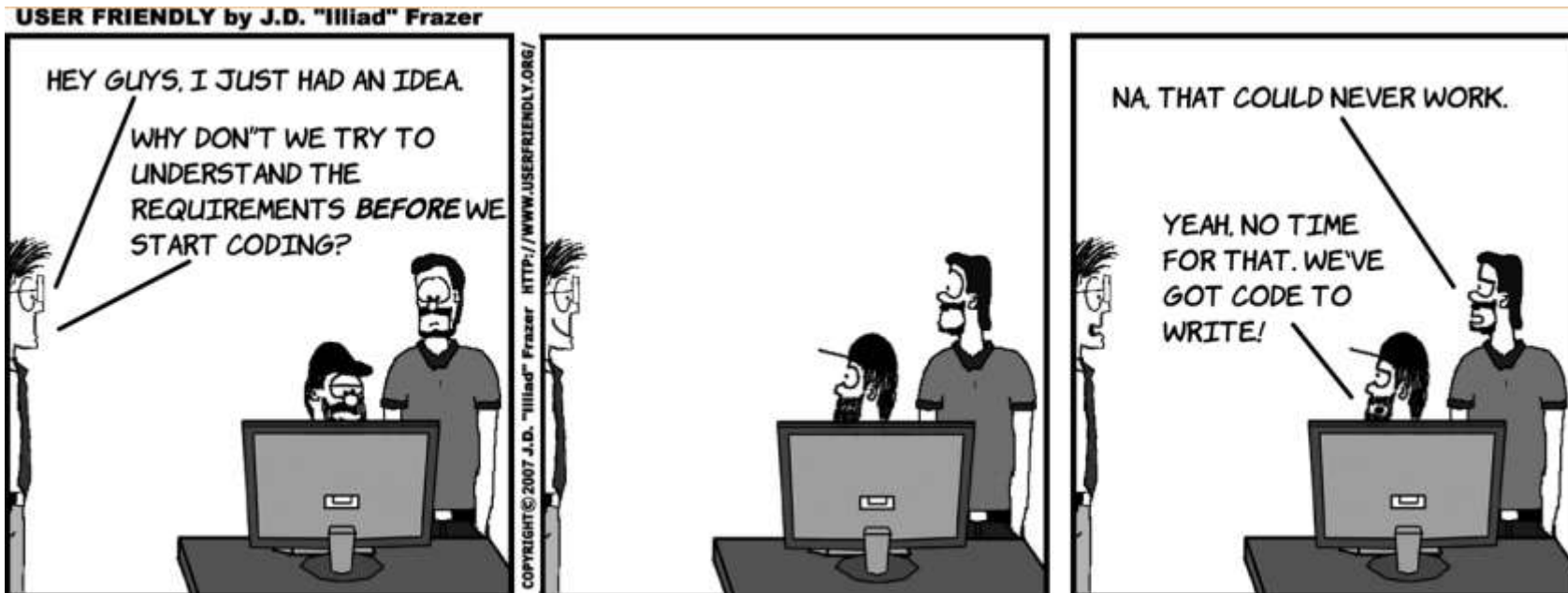
# Desarrollo de software

Principal ocupación de los egresados de  
carreras informáticas de FIUBA

Desarrollo != Programación

Desarrollo de software incluye a la Programación

Pero también a otras disciplinas



# Disciplinas del desarrollo de software

No todo es programación en el desarrollo de software





# Disciplinas del desarrollo (operativas)

Captura de requisitos: qué necesita el cliente

Análisis: qué vamos a construir

Diseño: cómo

Construcción o implementación

Pruebas: verificación y validación

Despliegue (en hardware)



# Disciplinas del desarrollo (soporte)

Administración del proyecto, incluyendo  
seguimiento y control de tiempos y costos

Gestión de cambios

Administración de la configuración

Gestión de los recursos humanos y la  
comunicación

Gestión de riesgos

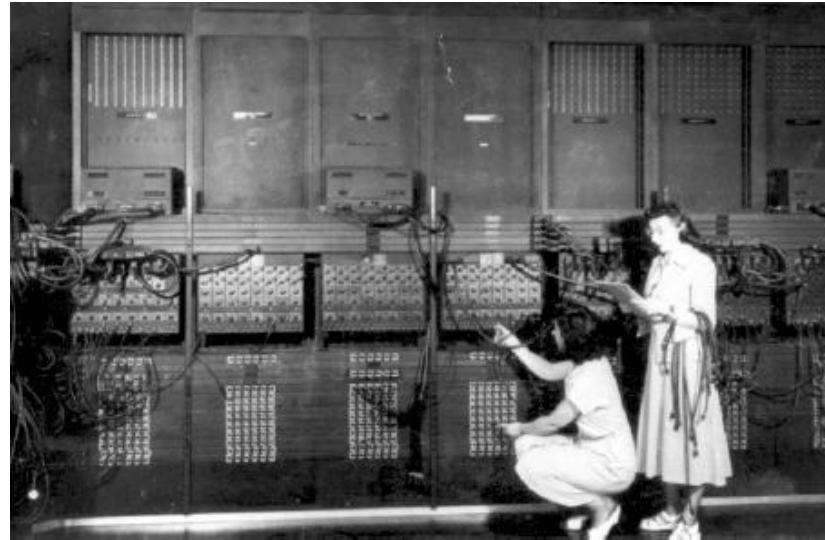
Gestión del proceso





# “Programar” = $f(t)$

- 1945: ENIAC, programas cableados
- 1954: lenguajes de programación y grandes máquinas
- 1970s/80s: PC aisladas o en red o software embebido
- 1990s: redes amplias e Internet
- 2000s/10s: celulares, automóviles, tablets...
- Historia hacia la computación ubicua => impactos en programación



fiuba

algo3

# “Desarrollo” = $f(t)$

Hasta 1970: sólo programación

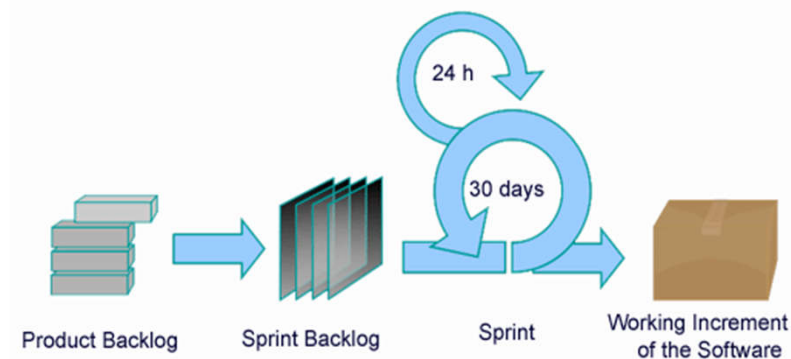
1970s: búsqueda de procesos

1980s: aumento de complejidad  
=> paradigmas de desarrollo

1990s: métodos iterativos e  
incrementales

2000s: métodos ágiles

2020s: desarrollo =>  
“no proyectos”



# Mini-historia de carreras argentinas

1961: llega Clementina a la FCEN

1960s: Computador Científico en FCEN

Departamento de Matemática

Muy centrada en programación

1970s: Analista Universitario de Sistemas en FIUBA

Centrada en problemas ingenieriles y cuestiones organizacionales

1990s: Ingeniería Informática en FIUBA

Y cambio de nombres de las carreras antiguas: licenciaturas

Siglo XXI: diversidad de carreras y facultades

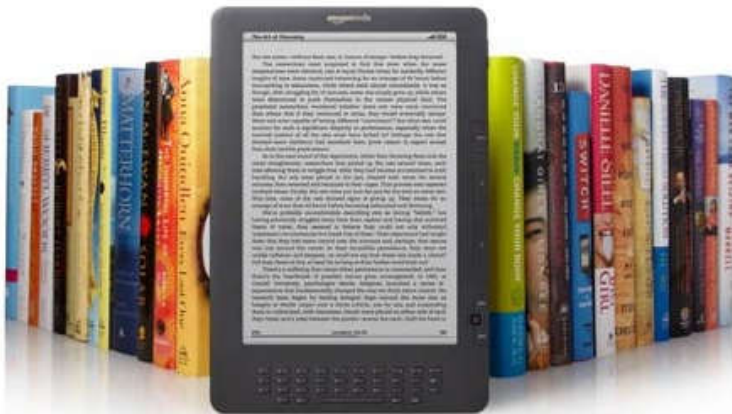
Campo cada vez más diverso



# Software hoy

“El software se está comiendo al mundo”

Marc Andreessen, 2011



fiuba  
algor3



# Características del software

Intangible

Maleable: posibilidad de cambio

Se desarrolla por proyectos o se mantiene en el tiempo como producto

Diferencia con manufactura industrial

Alto contenido intelectual

Disperso y difícil de reunir

El software no se “fabrica”

Se construye o se desarrolla

Mantenimiento constante desde su construcción



# Fracasos del desarrollo de software (1)

## Proyectos que no terminan a tiempo

Aeropuerto de Denver: sistema de administración de equipajes

agosto 1994 => diciembre => marzo => mayo

pérdidas de U\$S 1 M por día de atraso

## Proyectos que cuestan más de lo estimado

PPARS (proyecto de administración de personal, Irlanda)

Estimado en € 8,8 M => € 140 M





# Fracasos del desarrollo de software (2)



## “Accidentes”

### Software del Ariane 5

Explota a poco de salir por pérdida total de información de guiado y altitud

Origen: uso de software del Ariane 4

## Productos que no cumplen lo que el solicitante quiere

45% del software contratado nunca es usado

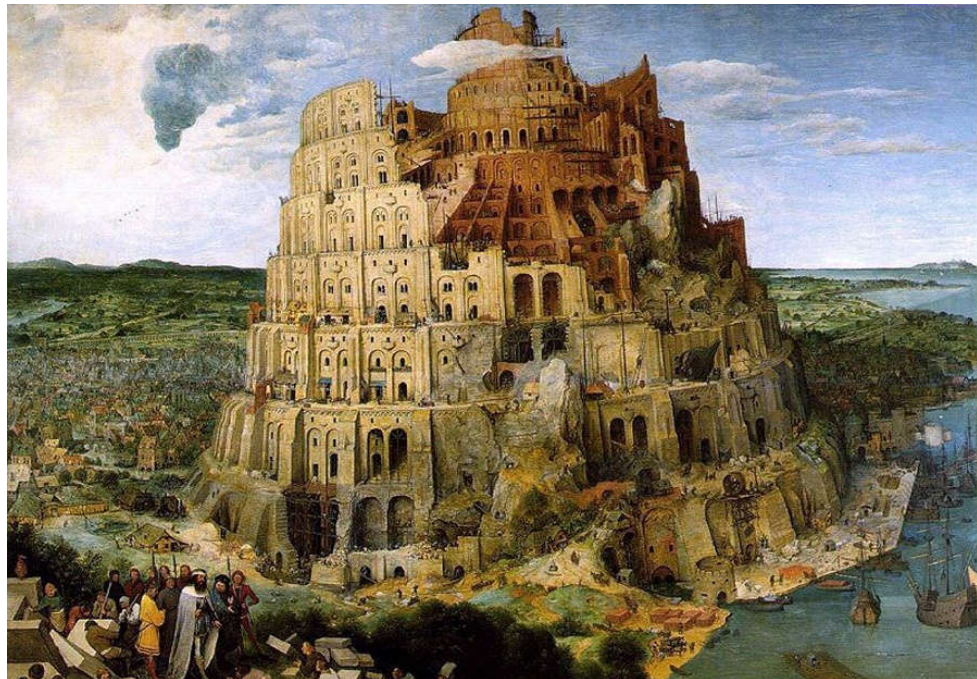
Fuente: CHAOS Report, The Standish Group

# Algunas reflexiones

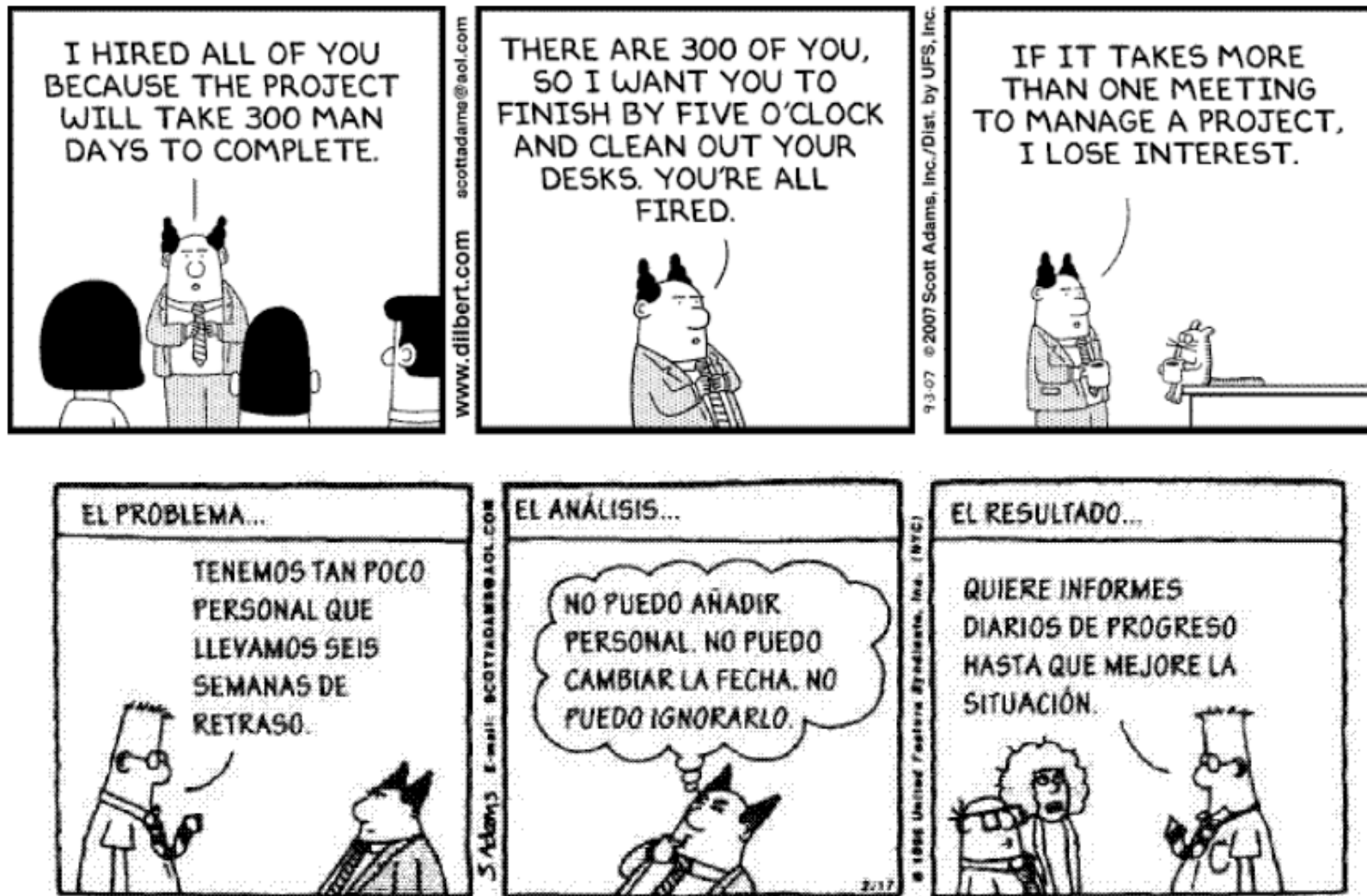
Los problemas del desarrollo no son sólo tecnológicos

Ley de Brooks: agregar gente a un proyecto atrasado lo atrasa más

Cuidar la comunicación



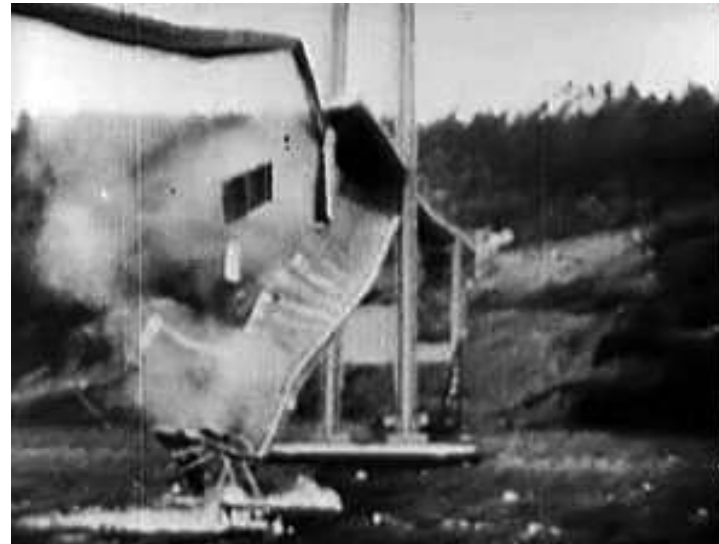
# Trabajar con personas



fiuba  
algo3

# Problemas en ingenierías de proyectos (1)

Tacoma - Narrows  
Colapsó en 1940



Puente de Aviñón  
Construido en 1171  
Destruído varias veces  
Último intento 1660





# Problemas en ingenierías de proyectos (2)

Accidentes en Three-Mile Island, Chernobyl, Fukushima  
Yacyretá-Apipé

US\$ 11.000 M en 15 años

Puente Chaco-Corrientes

Fallas de diseño (1973)

Big-Dig, Boston, EEUU

2,8 MM => 14 MM

Catedral de Colonia, Alemania

1248-1880

Hoy: reparación permanente



# Analizar

¿Qué tienen en común los casos anteriores con el desarrollo de software?





# Atención

Desarrollo de software es una actividad  
... de construcción de conocimiento  
... empírica

Dos posibilidades:

- Se desarrolla en proyectos con principio y fin
- Se desarrolla en forma continua



# Recapitulación

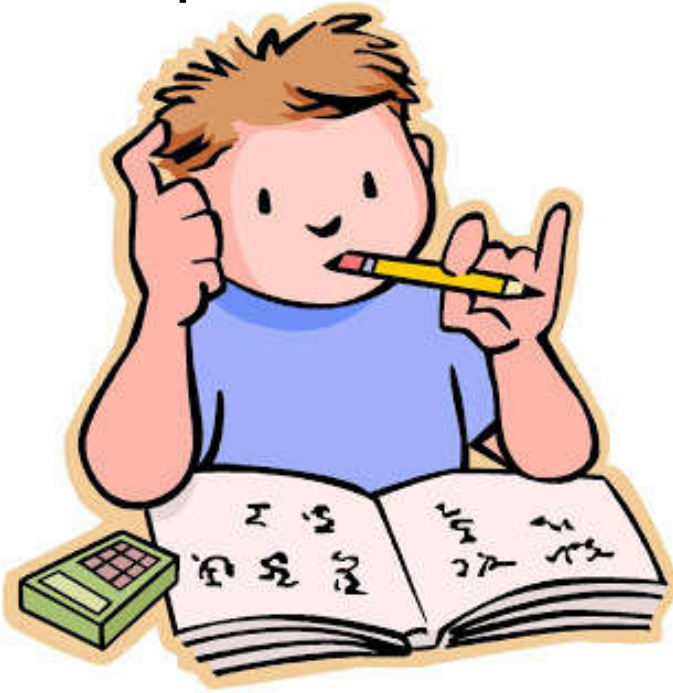


## Recapitulación: preguntas

¿Por qué decimos que el software hace útil al hardware?

¿Por qué decimos que el software no se fabrica?

¿Qué verbo usamos para no decir “fabricar”?



# Metodología



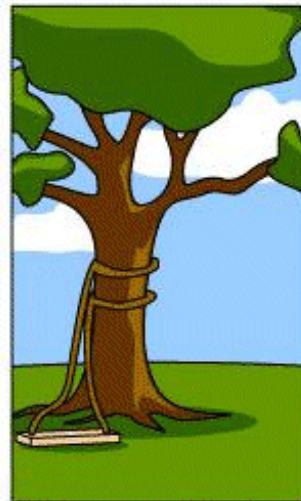
La solicitud del usuario



Lo que entendió el líder del proyecto



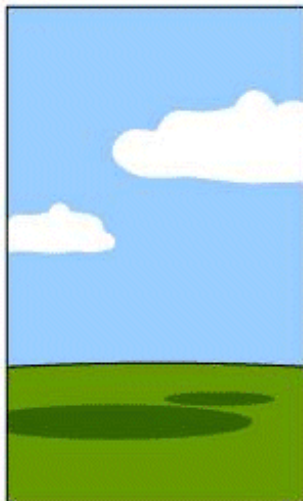
El diseño del analista de sistemas



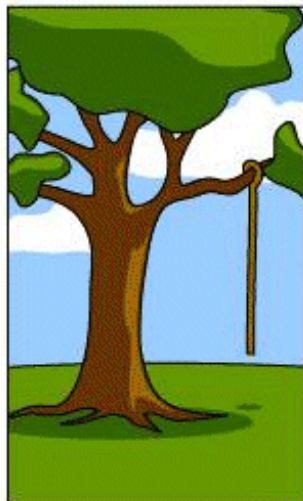
El enfoque del programador



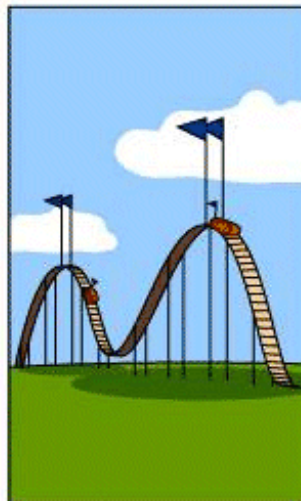
La recomendación del consultor externo



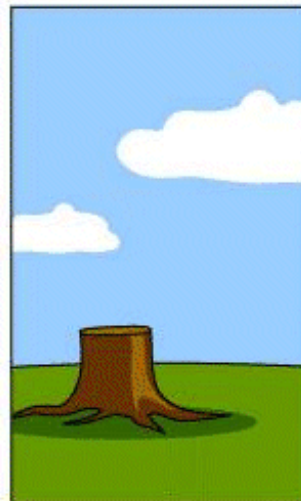
La documentación del proyecto



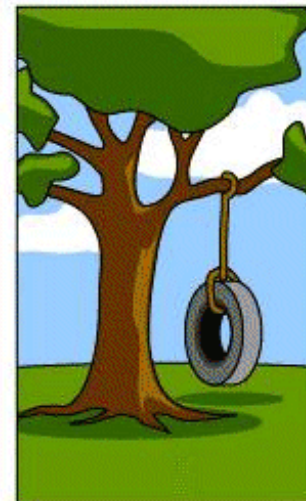
La implantación en producción



El presupuesto del proyecto



El soporte operativo



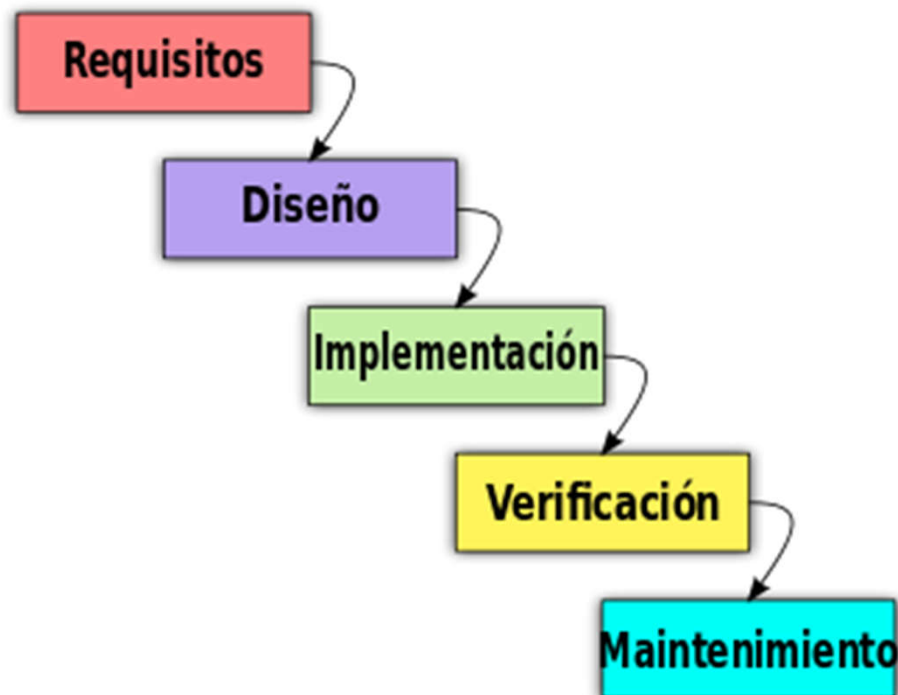
Lo que el usuario realmente necesitaba

fiuba  
algo3

# Ciclos de vida de desarrollo de software: lineal

Proyecto => serie de pasos que implican actividades distintas

“Cascada”: no permite volver atrás





# Ciclos de vida de desarrollo de software: incremental

Se llega a una solución

- Cada vez más refinada

- O que incluya más características

- O ambas

Se puede partir de una visión aproximada

Argumentos: no se conocen de entrada...

- Los “requerimientos”

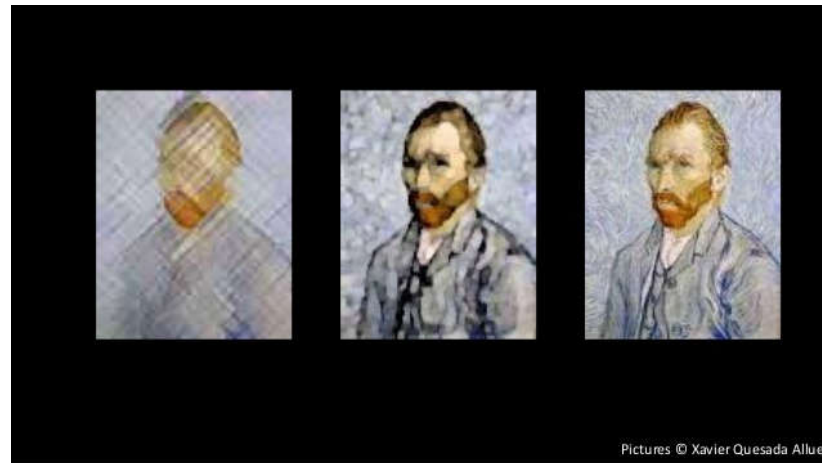
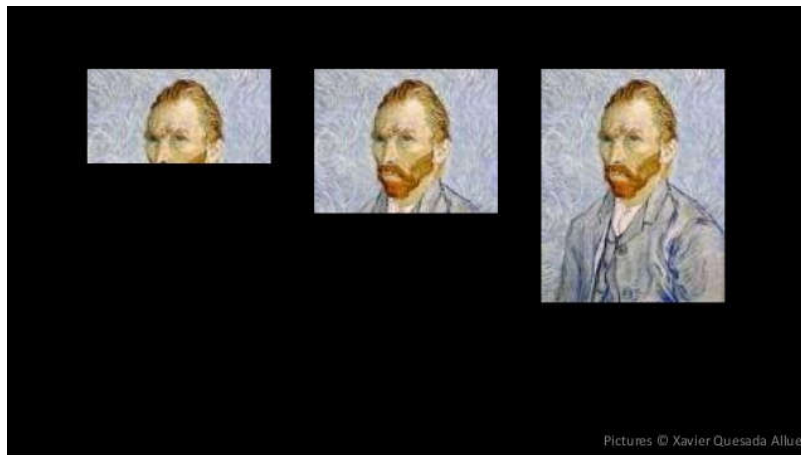
- La “tecnología”

- Las necesidades tecnológicas que cubren los requerimientos



# Hoy no se discute

El desarrollo de software debe ser incremental



... salvo programas muy chicos y de juguete

# ¿Por qué incremental?

Recordar que estamos

Experimentando

Construyendo conocimiento

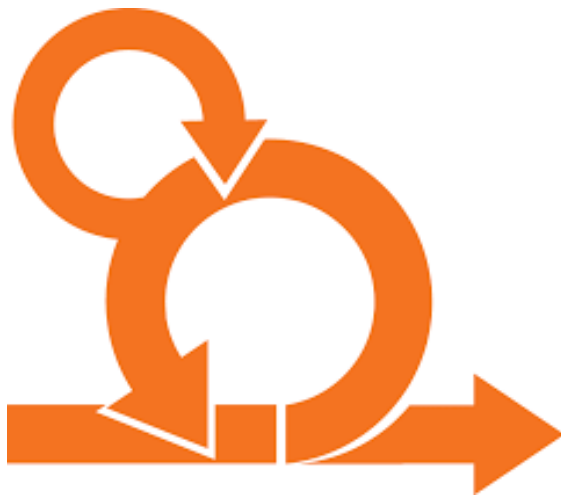
El software evoluciona mediante la interacción  
durante su construcción



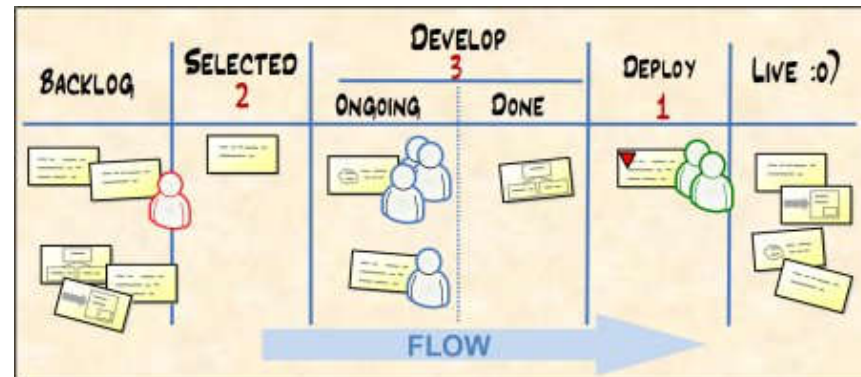
## Sí se discute...

¿Iterativo?: por iteraciones con comienzo y fin

¿Continuo?: desarrollo en base a funcionalidades sin hitos definidos



VS.



Hay situaciones para ambos modelos

# Método o proceso

Sirve para estructurar, planificar, desarrollar y controlar el desarrollo de software

Determina

Fases

Roles

Actividades

Artefactos

Etc.



# Categorías de métodos

## Procesos predictivos (¿basados en planes?)

- Planificación más detallada y rígida

- Pensados para grandes proyectos: inaceptablemente pesados para sistemas pequeños o medianos

- Destacan el Proceso Unificado (UP), TSP, Cleanroom

## Métodos ágiles o adaptables

- Más abiertos a los cambios

- Permiten organizar desarrollos sin caer en burocracia inútil

- Nacieron como alternativa a carecer de metodología

- Destacan Extreme Programming (XP) y Scrum



# Hoy todavía se discute pero...

## Evidencia

El desarrollo de software es inherentemente cambiante

=> Aceptar el cambio, no trabarlo

## Variables a controlar

calidad, costo, tiempo de desarrollo, alcance

=> ajustar cualquier variable (¿alcance?) menos la calidad

## Métodos ágiles para

Bajar el riesgo

Permitir cambios de especificaciones durante el desarrollo

Favorecer la comunicación con el cliente

Hay un “Manifiesto ágil” =>





# Manifiesto ágil

(<http://www.agilemanifesto.org/iso/es/>)

Individuos e  
interacciones

Procesos y herramientas

Software funcionando

Documentación  
extensiva

Colaboración con el  
cliente

Negociación contractual

Respuesta ante el  
cambio

Seguir un plan



... aunque valoramos los elementos de la derecha,  
valoramos más los de la izquierda

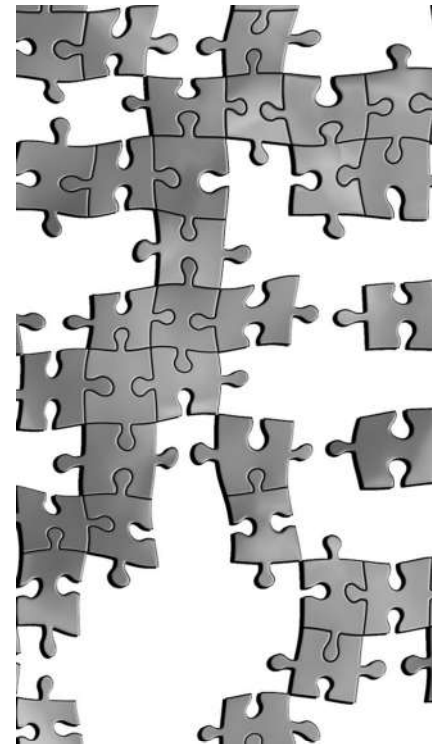
# Software (características) => Agilismo

Maleable => no impedir los cambios

Particionable => desarrollar en forma incremental

Extensible => pensar en evolución permanente

Software es materialización del conocimiento => interacción colaborativa



# Software (equipos) => Agilismo

Los que desarrollan son personas =>  
potenciar capacidades sociales

Las personas deben comunicarse  
efectivamente => reuniones cara a  
cara y abiertas

Tendemos a querer ser parte de  
equipos exitosos =>  
autoorganización

Únicos animales que podemos  
reflexionar sobre nuestros errores  
=> retrospectivas frecuentes



# Algunos métodos ágiles

## Extreme programming (XP)

de Kent Beck y la comunidad Smalltalk

Lleva al extremo las buenas prácticas

Lo analizamos acá, en un curso de Programación

## Scrum

de Ken Schwaber y Mike Beedle

Provee roles y artefactos centrados en seguimiento y control del proyecto

Lo van a analizar en materias de Administración de Proyectos

Otros: ASD, Crystal Clear, FDD, DSDM, MSF for Agile

# Nuestro TP2

Por equipos de pares

Docente/tutor como cliente

... y como docente

Iterativo

Con entregas semanales

Incremental

Cada entrega va a ir mejorando y completando el producto

Con prácticas ágiles

TDD, Refactorización, Integración continua: todas de XP



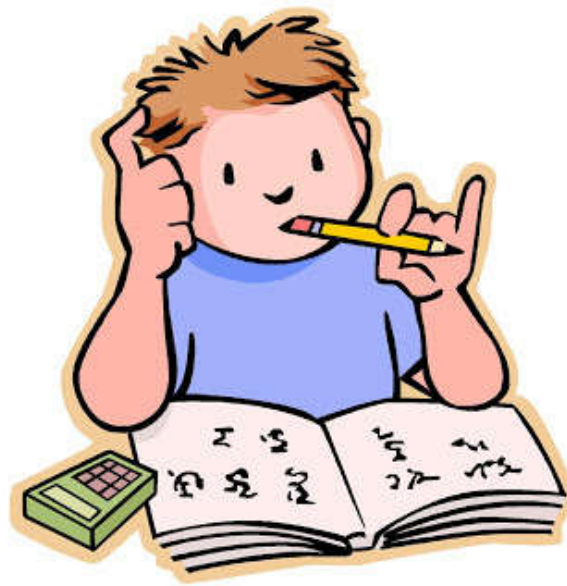


# Recapitulación



# Recapitulación: preguntas

- ¿Qué características del software impulsan el uso de métodos ágiles?
- ¿Por qué tiene sentido construir de manera incremental?
- ¿Pueden describir cómo vamos a hacer el TP2?



# Claves

Problemas desarrollo >> Problemas tecnológicos

Desarrollo => construcción de conocimiento

Naturalmente, el software se construye de manera incremental

Los métodos ágiles se adaptan mejor a las características del software

Elegir y adaptar los métodos de desarrollo  
Esa es tarea de un ingeniero

# Lectura obligatoria

Carlos Fontela, “Estado del arte y tendencias en  
Test-Driven development”

[http://web.fi.uba.ar/~cfontela/Fontela\\_EstadoDel  
ArteTDD\\_UNLP\\_EIS.pdf](http://web.fi.uba.ar/~cfontela/Fontela_EstadoDelArteTDD_UNLP_EIS.pdf)

(ojo que es largo)

# Lecturas opcionales (1)

Artículo de Wayt Gibbs en Scientific American, “Software’s Chronic Crisis” en:  
<http://www.cis.gsu.edu/~mmoore/CIS3300/handouts/SciAmSept1994.html>

Paper de Fred Brooks, “The Mythical Man-Month”: buscar en la Web

Gojko Adzic, “Bridging the Communication Gap. Specification by example and agile acceptance testing”

Dan North, “Introducing BDD”, buscar en web

Rick Mugridge y Ward Cunningham, “Fit for Developing Software...”

Lasse Koskela, “Test Driven: TDD and Acceptance TDD for Java Developers”



## Lecturas opcionales (2)

Básicos sobre métodos ágiles:

<http://agilemanifesto.org/iso/es/>

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

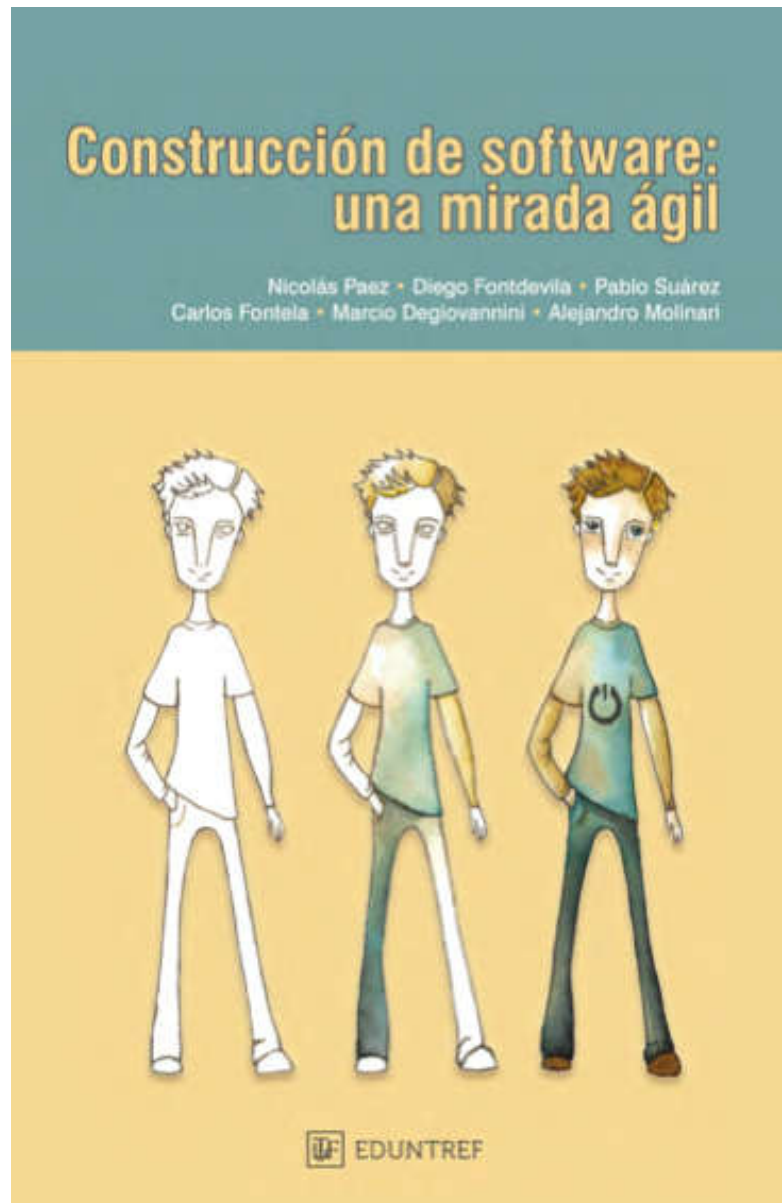
<http://www.mountangoatsoftware.com/topics/scrum>

<http://xprogramming.com/xpmag/whatisxp>

Más avanzado:

<http://www.proyectalis.com/wp-content/uploads/2008/02/scrum-y-xp-desde-las-trincheras.pdf>

# Libro de 2014



fiuba  
algo3

# Qué sigue

Prácticas: Concurrencia, persistencia,  
versionado, integración continua

Teóricas: RTTI, reflexión, temas de  
lenguajes

TP2

