

Ingeniería del Software

A close-up, low-angle shot of a hand typing on a laptop keyboard. The lighting is dramatic, with strong highlights and deep shadows, creating a professional and focused atmosphere. The background is blurred, emphasizing the hand and the keys.

Rubén Béjar

@rubejar

17 de abril de 2018

"As a working definition: software engineering is that part of computer science, which is too difficult for the computer scientist."

Friedrich L. Bauer (1971)

La frase anterior solo es un "click bait años 70" y hay que ponerla en su contexto histórico, pero encierra una verdad

La ingeniería del software consiste en resolver los problemas de la informática ...

Diseñar e implementar algoritmos correctos y eficientes que manipulan las estructuras de datos necesarias, construir interfaces de usuario realmente usables, asegurar que no se pierden cambios en tu base de datos o en la comunicación entre componentes distribuidos en una red...

... pero en el mundo real

Con restricciones de tiempo y dinero, trabajando junto a decenas o centenares de personas, en sistemas que pueden tener que ser operativos durante décadas, tolerando ambigüedades, indefiniciones y cambios, y considerando que el software tiene un entorno (mercado, usuarios, competencia, legislación...) que también es variable y al que te tienes que adaptar

"El desafío esencial de la programación es **manejar la complejidad**. Los que piensan que la dificultad está en traducir ideas a algún lenguaje de programación línea a línea no han escrito programas grandes."

Trabajo de Proyecto Software (un cuatrimestre, 7 personas): 10.000 Líneas de Código

```
static Private $char
static Private $link = null
public function Connect()
if (!$link = mysql_connect(self::$host,
throw new MySQLException("Cannot connect to
mysql_query("SET CHARACTER SET
mysql_query("SET NAMES
mysql_query("USE
```

Trabajo de Proyecto Software (un cuatrimestre, 7 personas): 10.000 Líneas de Código

App iOS media: 40.000 Líneas de Código

Trabajo de Proyecto Software (un cuatrimestre, 7 personas): 10.000 Líneas de Código

App iOS media: 40.000 Líneas de Código

Photoshop CS 6 (2012): 4,5 Millones de Líneas de Código

Las líneas de código son una medida gruesa, pero directa y automática, del tamaño del software

Hay que interpretarlas y compararlas con algunas cautelas y, por ejemplo, nunca usarlas como medida de productividad de los desarrolladores

Sin embargo, y aunque parezca sorprendente, están muy correlacionadas con métricas de complejidad que parecen mucho más sofisticadas

En el software real, más grande significa también más complejo

Mars Curiosity Rover: 5 Millones de Líneas de Código

Kernel de Linux 4.15.9 (2018): 20 Millones de Líneas de Código

Coche de gama alta (2009): 100 Millones de Líneas de Código

Considerad además que **el esfuerzo(*)** de desarrollo de software se incrementa **exponencialmente, no linealmente, con el tamaño**

Desarrollar un sistema el doble de grande, cuesta bastante más que el doble de esfuerzo

(*) El esfuerzo se mide en meses-persona, siendo un mes-persona la cantidad de trabajo que hace una persona en un mes a tiempo completo

Un coche de gama alta de 2009 lleva tanto código como **10.000** trabajos de Proyecto Software

Y haciendo una estimación muy conservadora, puede requerir **15.000 veces** más esfuerzo

"El desarrollo de software es muy difícil por un montón de razones. Tristemente, muchos textos sobre desarrollo de software proporcionan recomendaciones para situaciones simples..."

Photoshop en 1990 tiene unas **110.000** líneas de código

El kernel de Linux en 1999 tiene **1,8 millones** de líneas de código

Photoshop en 2012 tiene **4,5 millones**

El kernel de Linux en 2018 tiene **20 millones**

Las prácticas de programación de 1º de Ingeniería

Informática en 1995 son **iguales** a las de 2018

Los mismos problemas y las mismas técnicas para resolverlos

La **ingeniería del software** es la disciplina que nos ha permitido multiplicar **por diez** y en muchos casos hasta **por cien**, el tamaño, y la complejidad, del software real en los últimos 20-30 años

Con proyectos de este tamaño el trabajo en equipo es, lógicamente, un requisito esencial

Date: Tue, 22 Aug 2000 16:00:52 -0400

From: "Eric S. Raymond" <esr@thyrsus.com>

To: Linus Torvalds <torvalds@transmeta.com>

[...]

>> Linus Torvalds <torvalds@transmeta.com>:

>>> But the "common code helps" thing is WRONG. Face it. It can hurt. A lot.

[...]

You are a brilliant implementor, more able than me and possibly [...] the best one in the Unix tradition since Ken Thompson himself. As a consequence, you suffer **the curse of the gifted programmer** -- you lean on your ability so much that **you've never learned** to value certain kinds of **coding self-discipline and design craftsmanship** that lesser mortals **must** develop in order to handle the kind of problem complexity you eat for breakfast.

[...]

As Linux grows, there will come a time when your raw talent is not enough. What happens then will depend on how much discipline about coding and release practices and fastidiousness about clean design you developed **before** you needed it, back when your talent was sufficient to let you get away without.

Date: Tue, 22 Aug 2000 16:00:52 -0400
From: "Eric S. Raymond" <esr@thyrsus.com>
To: Linus Torvalds <torvalds@transmeta.com>

[...]

>> Linus Torvalds <torvalds@transmeta.com>:

>>>

[... **As Linux grows, there will come a time when your raw talent is not enough.** What happens then will depend on how much discipline about coding and release practices and fastidiousness about clean design you developed ***before*** you needed it...

[... **Conforme Linux crezca, llegará el momento en el que tu talento no será suficiente.** Lo que suceda entonces dependerá de cuánta disciplina sobre programación y prácticas de lanzamiento y diseño limpio hayas desarrollado ***antes*** de necesitarlas...

without.

Cada día se borran 2.500 líneas de código del kernel de Linux, se añaden 10.000 y se modifican 2.000

Recordad: un trabajo de cuatrimestre de Proyecto Software (7 personas) son 10.000

Chromium (en 2011) añade 800 modificaciones validadas a la semana al código

Amazon (en 2011) sube un cambio a producción (accesible para los usuarios finales) cada 11.6 segundos

La **ingeniería del software** es la disciplina que nos ha permitido multiplicar **por diez**, y en algunos casos hasta **por cien**, la velocidad a la que se entregan cambios en el software real a sus usuarios en los últimos 20 años

De sacar una o dos versiones de los productos al año, a desplegar cambios en producción varias veces al día

La esencia de la ingeniería es diseñar soluciones que transformen la realidad

El diseño de productos, procesos, sistemas y
métodos es el problema de ingeniería más
común

B. V. Koen (2003). *Discussion of the Method. Conducting the Engineer's Approach to Problem Solving*. Oxford University Press. (p. 28)

H. Petroski (2011). *An Engineer's Alphabet. Gleanings from the Softer Side of a Profession*. Cambridge University Press. (p. 66)

J. Hughes (2009). Practical Reasoning and Engineering. *Philosophy of Technology and Engineering Sciences*. Elsevier.

Jonassen, D., Strobel, J., & Lee, C. B. (2006). Everyday problem solving in engineering:

Lessons for engineering educators. *Journal of Engineering Education*, 95, 139-151.

Los problemas de diseño están entre los
peor estructurados y peor definidos

Jonassen, D. (2011). Supporting problem solving in PBL.
The Interdisciplinary Journal of Problem-Based Learning, 5, 95-119.

Todos los proyectos de
ingeniería tienen lugar siempre
sujetos a diversas **restricciones**

El triángulo de
hierro

Resultados



Coste

Plazos

El triángulo de
hierro

Resultados

Coste

Plazos



Por tanto, la tarea que caracteriza a la ingeniería informática es coger problemas complejos y mal especificados y **diseñar soluciones bajo restricciones** de costes y plazos, y con exigencias de resultados

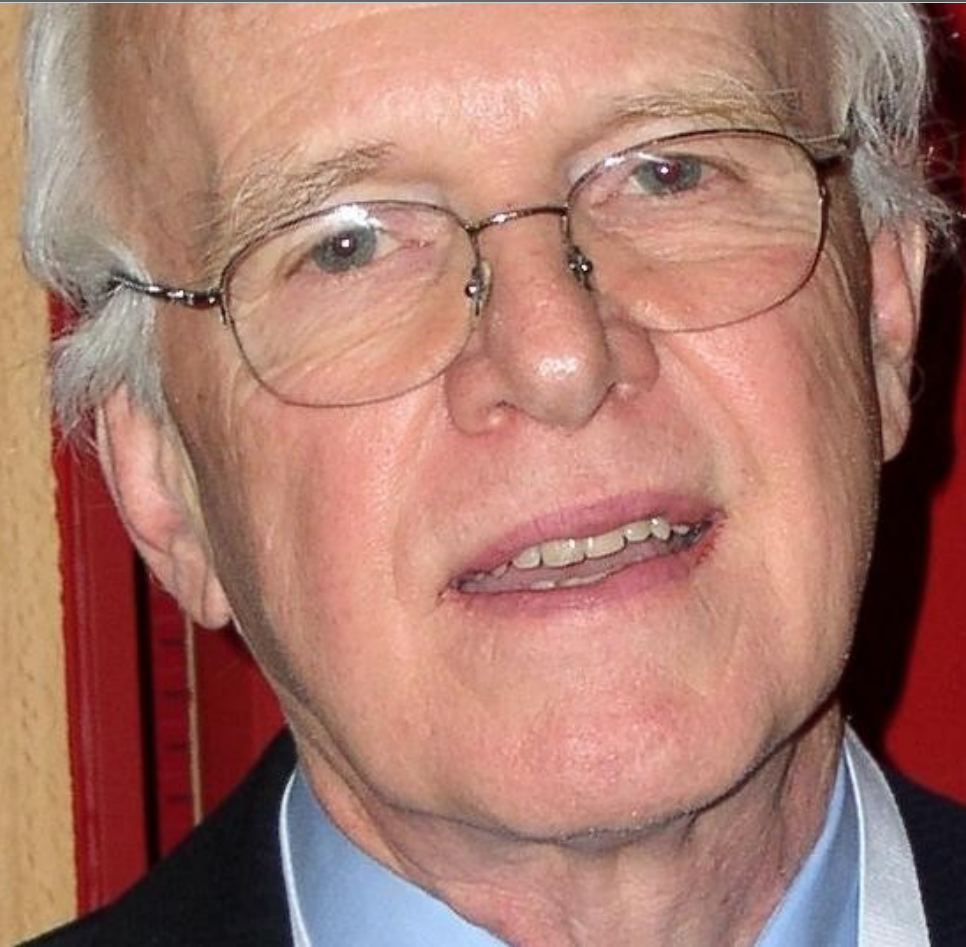
En problemas que no dejan de crecer en tamaño y complejidad

Y entregando resultados cada vez más frecuentemente

Ingeniería del Software (siglo XX vs siglo XXI)

"No puedes controlar lo que no puedes medir."

(Tom deMarco, 1982)



Siglo XX



"No puedes controlar lo que no puedes medir."

(Tom deMarco, 1982)

"era ... correcto en ese momento, es todavía relevante, y aún creo que las métricas son una necesidad...?"

Mis respuestas son no, no y no."

(Tom deMarco, 2009)

Siglo XXI

Sistemas complejos
creados a partir de
partes simples y bien
comprendidas

Structure and Interpretation of Computer Programs

Second Edition



Siglo XX

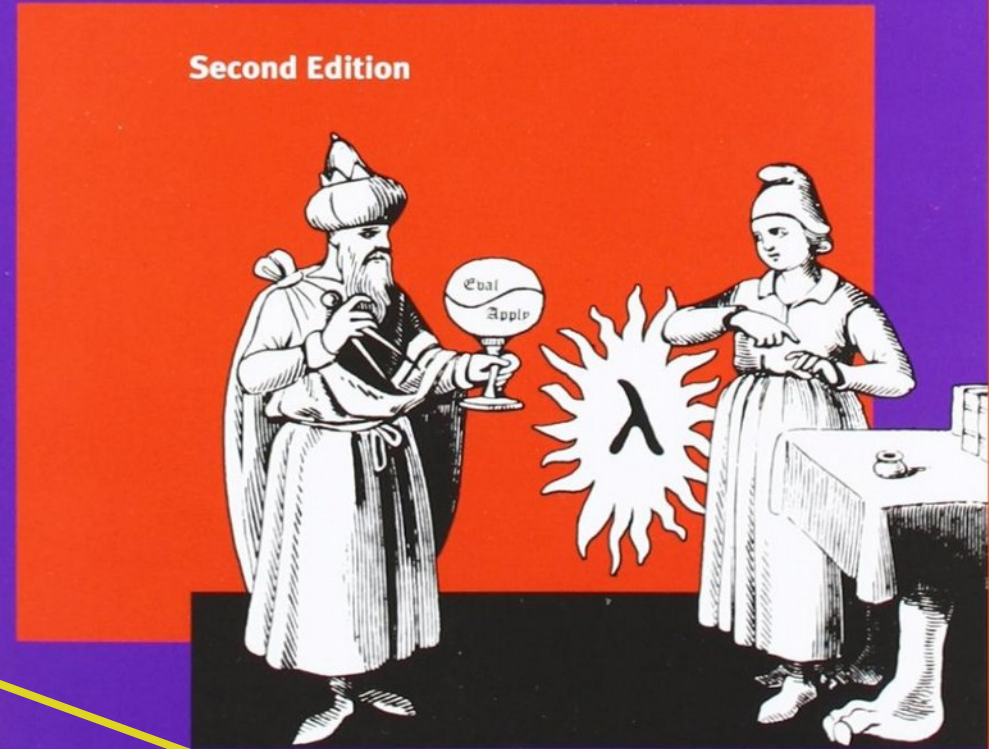
Harold Abelson and
Gerald Jay Sussman
with Julie Sussman

SICP ya no prepara a los ingenieros para la ingeniería de hoy en día.

Los ingenieros escriben rutinariamente código para hardware y software tan complicados que no se pueden entender en su totalidad.

(Gerald Sussman, 2016)

Structure and Interpretation of Computer Programs



Siglo XXI

Harold Abelson and
Gerald Jay Sussman
with Julie Sussman

Diseño de software
dirigido por
arquitectos de Power
Point

También conocidos como
arquitectos de torre de
marfil y arquitectos
astronautas

Nunca tocan el código

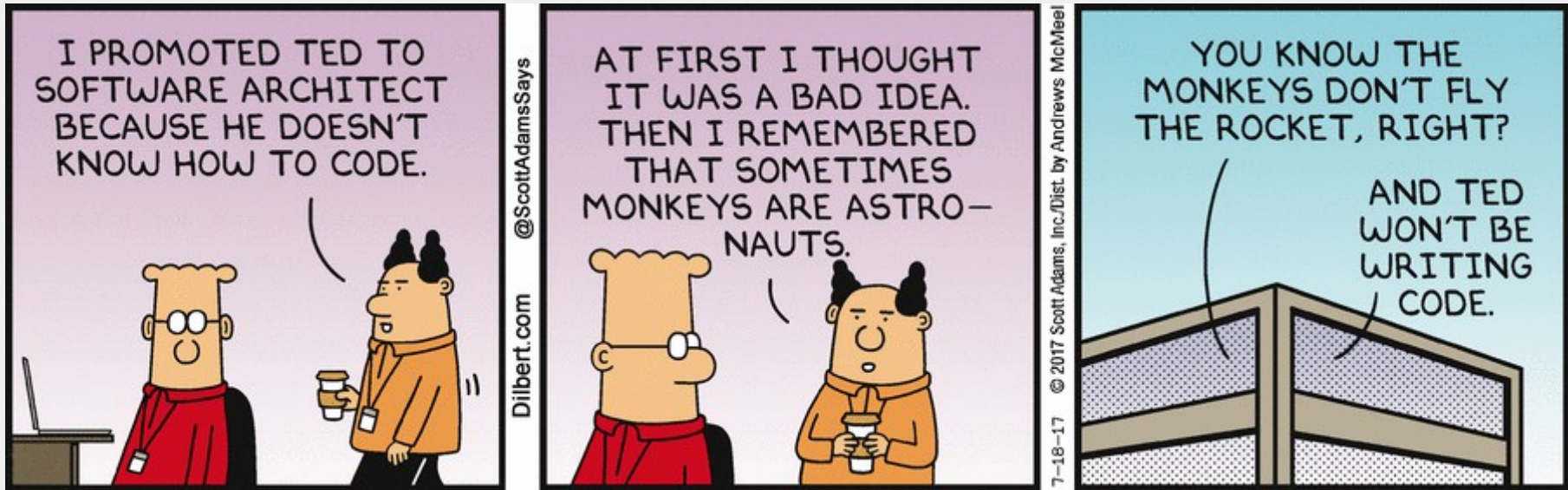


Siglo XX

Arquitecto & Maestro Programador



Siglo XXI



He ascendido a Ted a arquitecto de software, porque no sabe programar.

Primero pensaba que era mala idea. Luego me acordé que a veces los monos son astronautas.

Sabes que los monos no pilotan el cohete, ¿no?

Y Ted no escribirá el código.

Programadores que
no participan en el
análisis o el diseño
del software

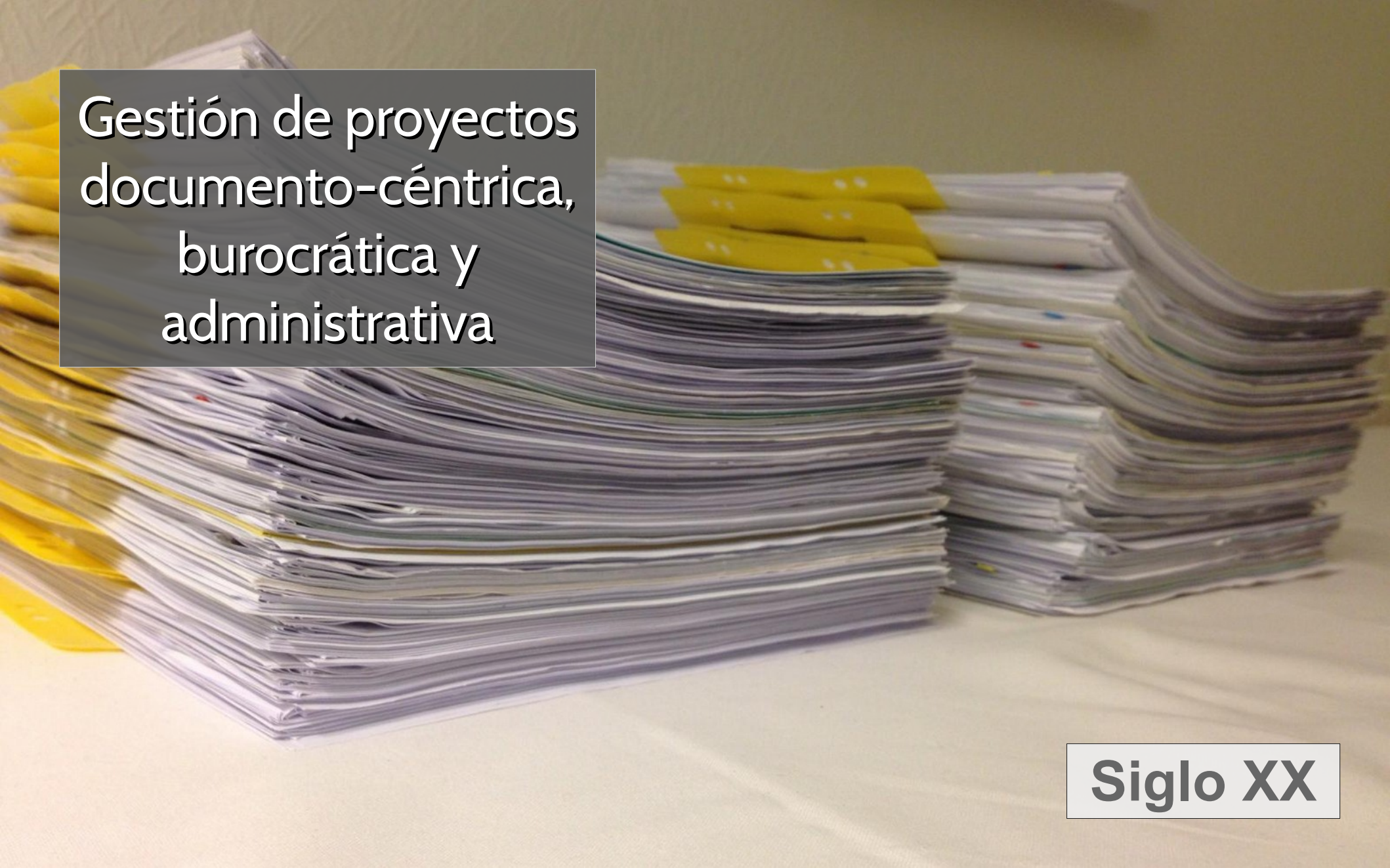
Siglo XX



Equipos pequeños y muy
cualificados, que son
responsables de todos los
aspectos de análisis y
diseño del software

Siglo XXI

This file is licensed under the Creative Commons
Attribution 2.0 Generic license. Author: Jon Lim

The image shows several large stacks of white papers and folders, some with yellow covers, arranged on a light-colored surface. The stacks are thick and appear to be organized in a way that suggests a large volume of administrative or project-related documents. A semi-transparent grey box is overlaid on the left side of the image, containing text.

Gestión de proyectos
documento-céntrica,
burocrática y
administrativa

Siglo XX

Scrum!

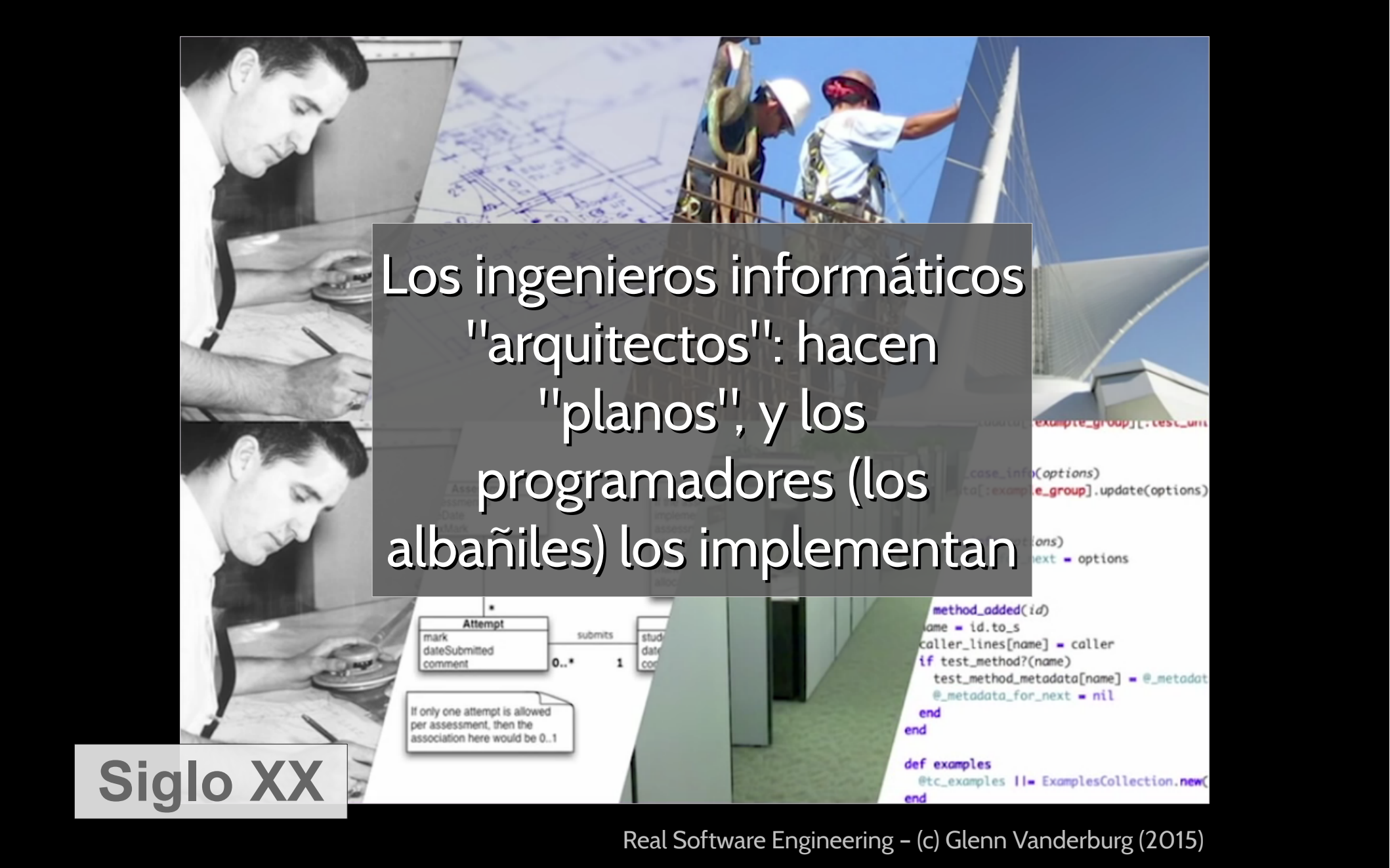
Gestión de proyectos
iterativa y ágil

Siglo XXI



Documentos para la
comunicación y la
interacción, no para
tratar de controlar

Siglo XXI




Los ingenieros informáticos
"arquitectos": hacen
"planos", y los
programadores (los
albañiles) los implementan

Siglo XX

Los ingenieros informáticos
"arquitectos": hacen
"plomos", y los
programadores (los
albañiles) los implementan

**Metáfora
Incorrecta**

Siglo XX



Los ingenieros informáticos diseñan el sistema usando lenguajes de programación y los compiladores (los albañiles) los implementan

Siglo XXI

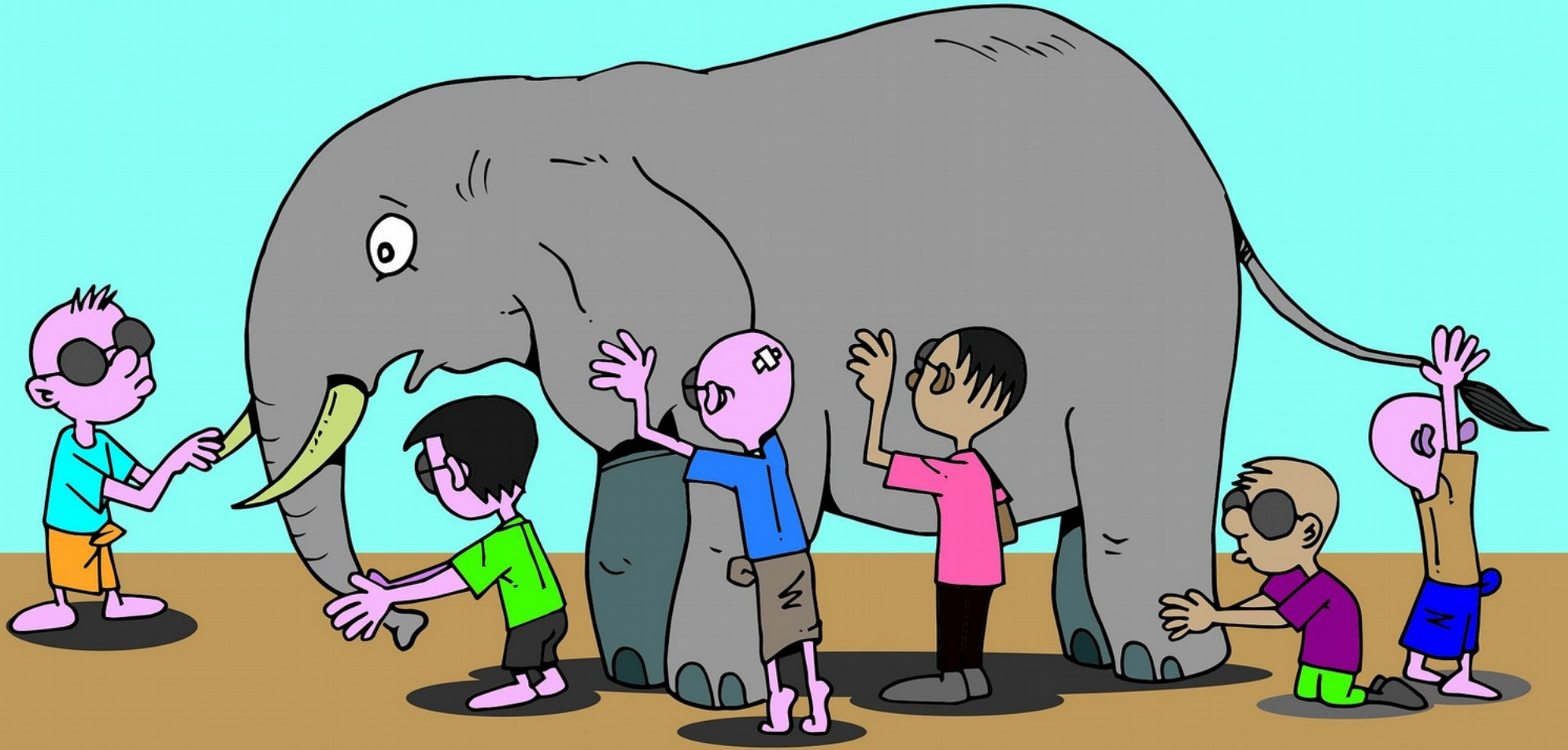
```
method_added(id)
  name = id.to_s
  caller_lines[name] = caller
  if test_method?(name)
    test_method_metadata[name] = @_me
    @_metadata_for_next = nil
  end
end

def examples
  @tc_examples ||= ExamplesCollect'
end
```



```
09 48 ..4).0),..
65 F8 >..*D..OM.
7F DF g.N.O...C
F3 22 0.....\..Z
8F B2 .....R..
9F 88 .)...../..{
28 33 i....ZF{6.
08 60 .k... ..
C5 97 .h.....
18 00 G...|...}.
11 B6 w.6u.R...
A1 17 P5 08 06 E5 76 .P.....
94 0F C6 44 AC 05 18 .._b...
A 2A FF F3 20 C4 CE 09 .....*
96 3C 7B 9E C9 2E 74 BF ...9...f<
JA 5F EE 3E 70 DD B5 73 .....i..
D3 54 46 EA 71 6E FF F3 .....}.7..T
48 44 C8 A5 A1 D5 58 C8 "...x..HHD
B DC DD BA F6 A6 00 C0 02 ...W..Z;..
5 62 F5 A9 2A D4 A8 8C 32 .....m..b.
02 0A 29 5E E8 48 41 84 08 .hF... ..)
F6 55 DF 47 F7 33 BE 03 40 .)|.m$.U.
FD 30 28 8E E4 EB BC 10 9F .4.1...0{
```


La historia de los ciegos y el elefante



Es una web

Es una red neuronal

Es un algoritmo de Montecarlo

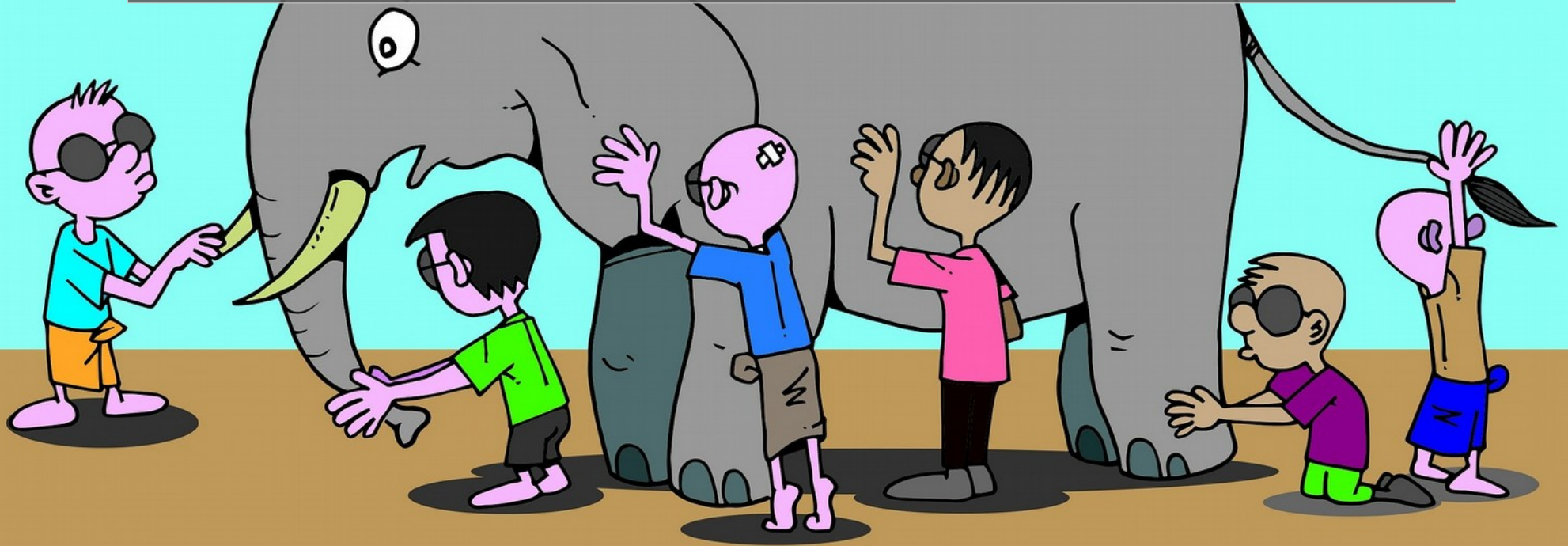
Es una BD NoSQL

Son unas máquinas virtuales en cloud

Es un sistema concurrente



Todos tenéis razón
Pero ninguno estáis viendo el sistema
completo



En 2008, se reportan diversos problemas en un proyecto de software

Inicialmente planificado para 3 años, ya lleva 12

6 millones de líneas de código, casi todo C++. Compilarlo tarda 48 horas en 32 máquinas en paralelo

Basado en CORBA, usa una base de datos de una compañía que quebró y componentes de GUI que ya no se mantienen. Posiblemente eran buenas elecciones tecnológicas cuando se empezó el proyecto, pero han pasado 12 años

En un caso se comprueba que un clic derecho tarda 45 minutos en mostrar el menú de contexto y en otro que cuesta 7 días leer un CD-ROM completo (700 MB)

El tiempo medio que un trabajador nuevo se queda en la empresa son 3 meses

~55 personas en el equipo, de las que solo 20 son desarrolladores (¿recordáis a los arquitectos de Power Point?)

Este proyecto es un fracaso sistémico, global

En proyectos grandes, las tecnologías, técnicas concretas, componentes o lenguajes de programación, importan poco

Importan, pero son el problema fácil

La diferencia entre el éxito y el fracaso la marcan el diseño del software, la arquitectura del sistema, la automatización de procesos y la gestión de los equipos y del proyecto. Es decir, la ingeniería del software

An open notebook with lined pages is shown from a top-down perspective. The text '¡GRACIAS!' is written across the right page in a large, black, serif font. To the right of the notebook, a dark-colored cup of coffee is partially visible, resting on a dark wooden surface. The lighting is warm and focused on the notebook.

¡GRACIAS!

rbejar@unizar.es

@rubejar