

metodologías ágiles y el manifiesto ágil

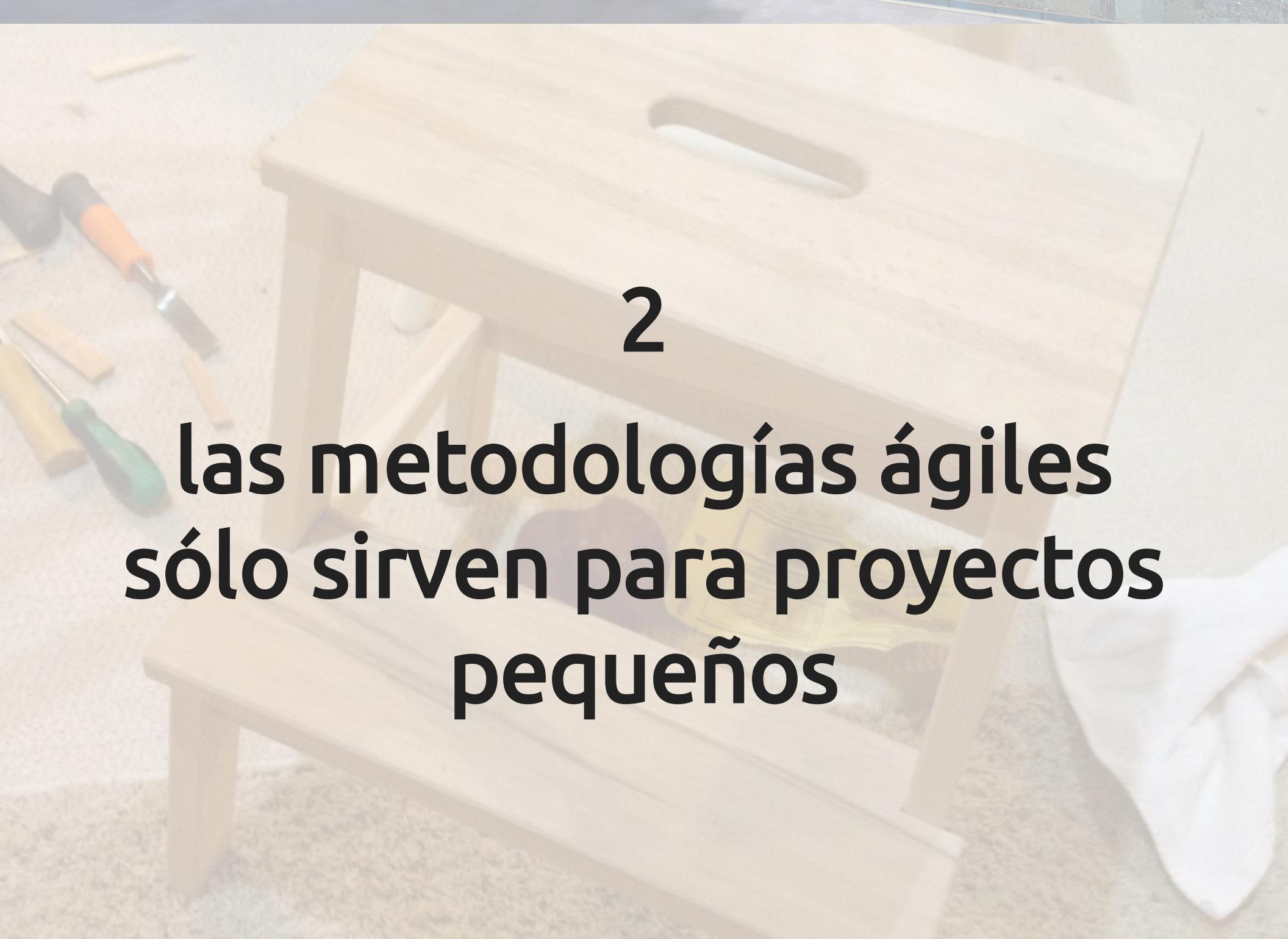
**¿qué sabemos sobre las
metodologías ágiles?**

diez mitos sobre las metodologías ágiles

The background features a dense, futuristic city with towering skyscrapers, some with glowing green panels. A large, circular flying vehicle with orange and silver accents is positioned in the upper right, hovering over the city. The sky is a light blue with wispy clouds.

**las metodologías ágiles son
algo nuevo**

1

A photograph of a light-colored wooden workbench. On the bench, there are several tools: a chisel with a black handle and silver blade, a screwdriver with an orange handle and silver blade, and a ruler. The background is slightly blurred.

2

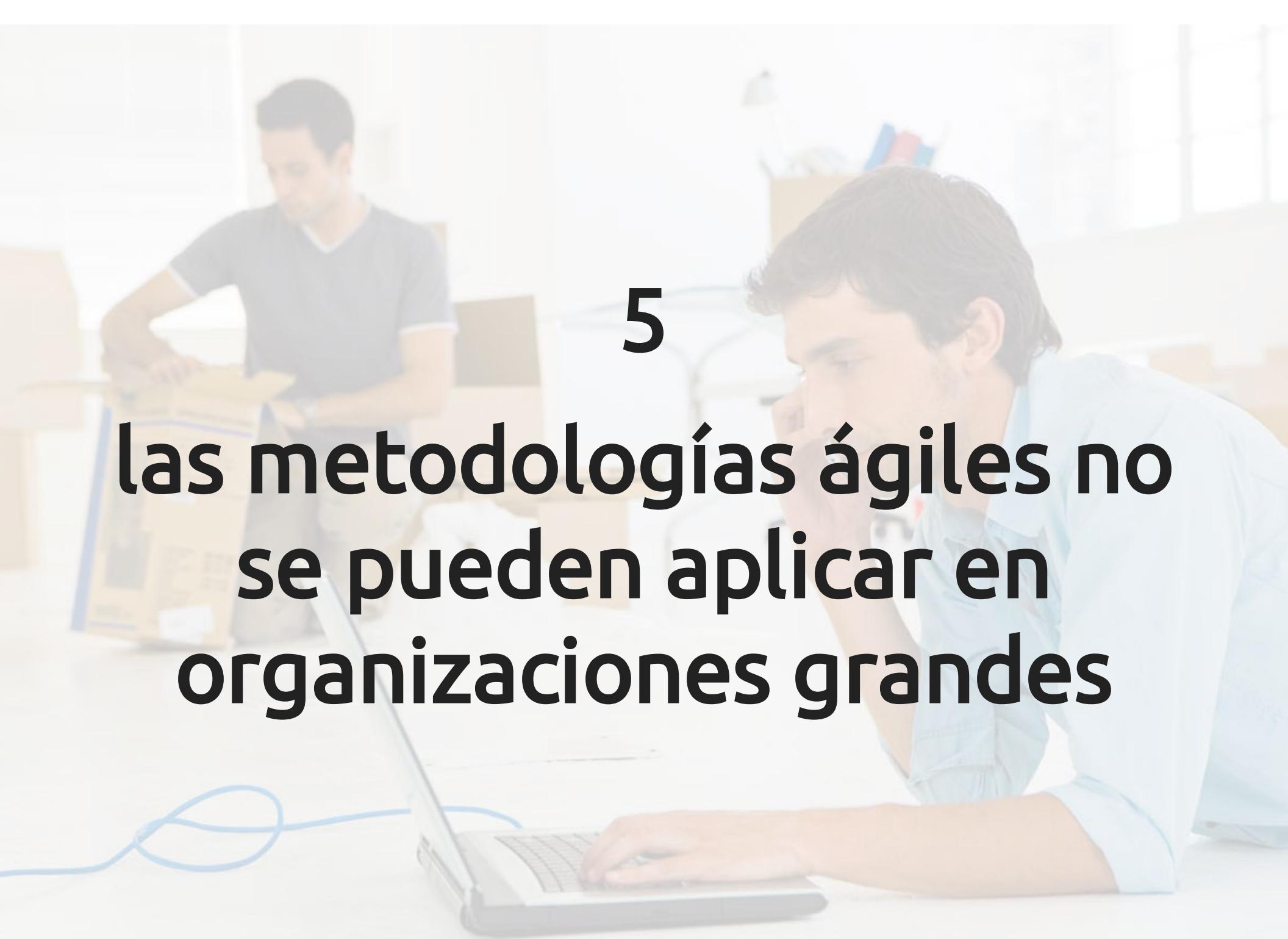
las metodologías ágiles
sólo sirven para proyectos
pequeños

A photograph of two men sitting at a light-colored wooden table outdoors. The man on the left is seen from behind, wearing a dark t-shirt. The man on the right is facing slightly towards the camera, smiling, and also wears a dark t-shirt. They appear to be engaged in a conversation or a meeting. The background consists of dense green foliage and trees.

3 las metodologías ágiles no capturan requisitos

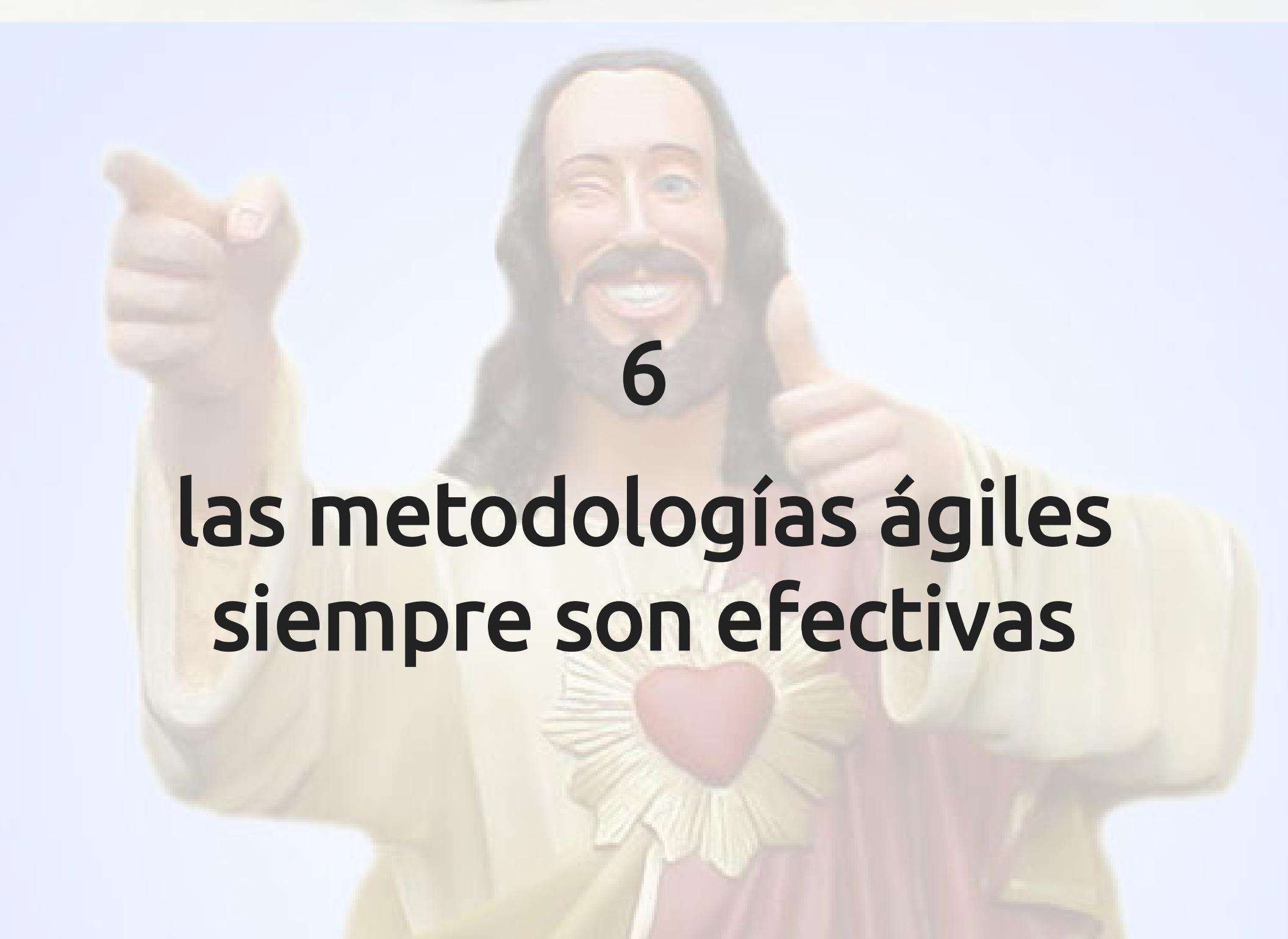
4

las metodologías ágiles no
generan documentación

A blurred background image showing two men in an office environment. One man is in the foreground, focused on a laptop screen. Another man is visible in the background, also working at a desk. The overall atmosphere is professional and focused.

5

**las metodologías ágiles no
se pueden aplicar en
organizaciones grandes**



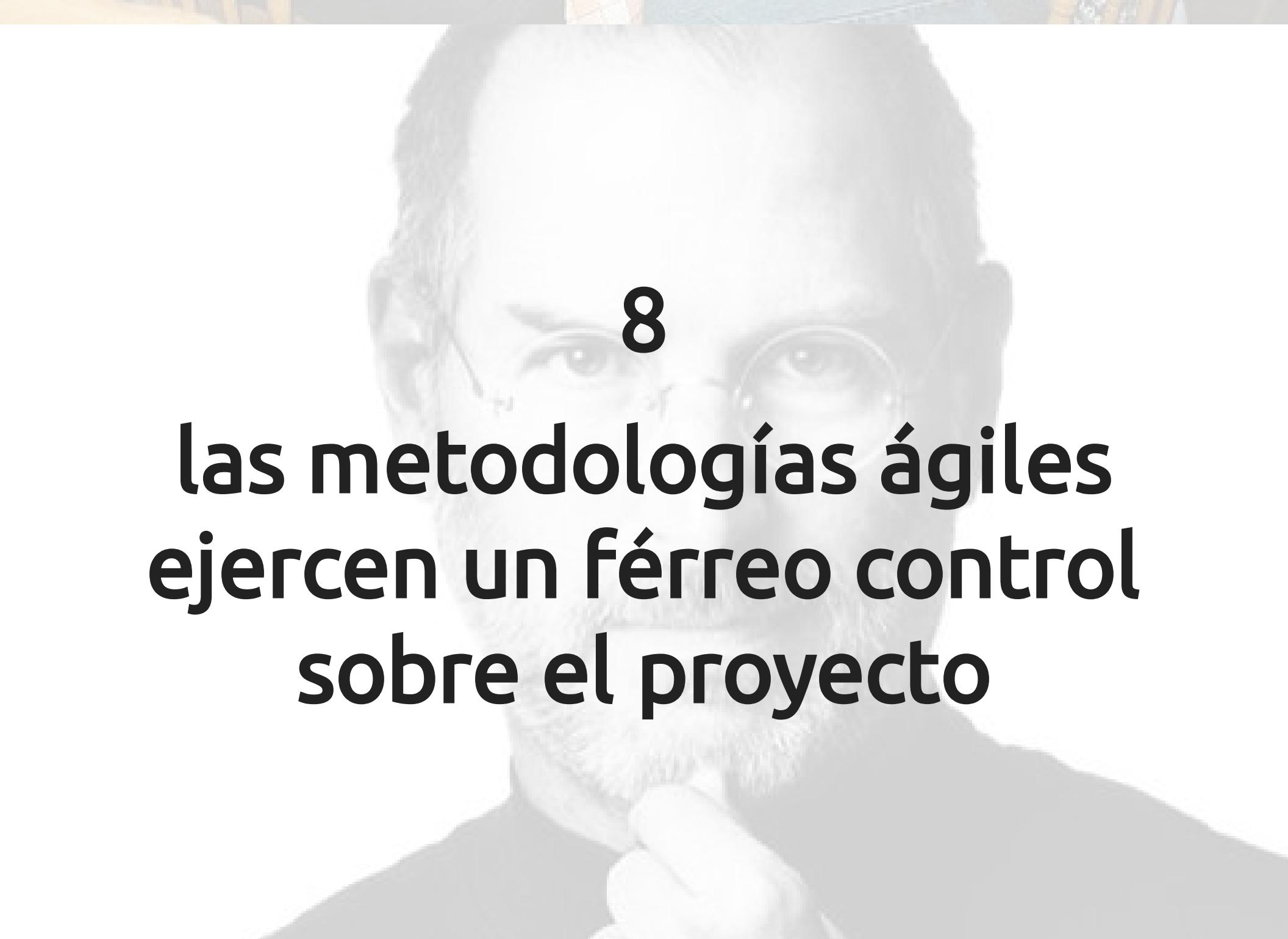
6

**las metodologías ágiles
siempre son efectivas**

A photograph of four people, three men and one woman, sitting around a round table covered with a white and red checkered cloth. They are all looking towards the camera. The man on the far left wears glasses and a light-colored shirt. The woman in the center wears a dark t-shirt. The man on the right wears a dark t-shirt and glasses. The man on the far right wears a patterned sweater and glasses.

7

las metodologías ágiles no son cosa seria



8

**las metodologías ágiles
ejercen un férreo control
sobre el proyecto**

9

las metodologías ágiles
sólo se pueden aplicar al
desarrollo de software

10

¿las metodologías ágiles no
se pueden aplicar en la
administración pública?

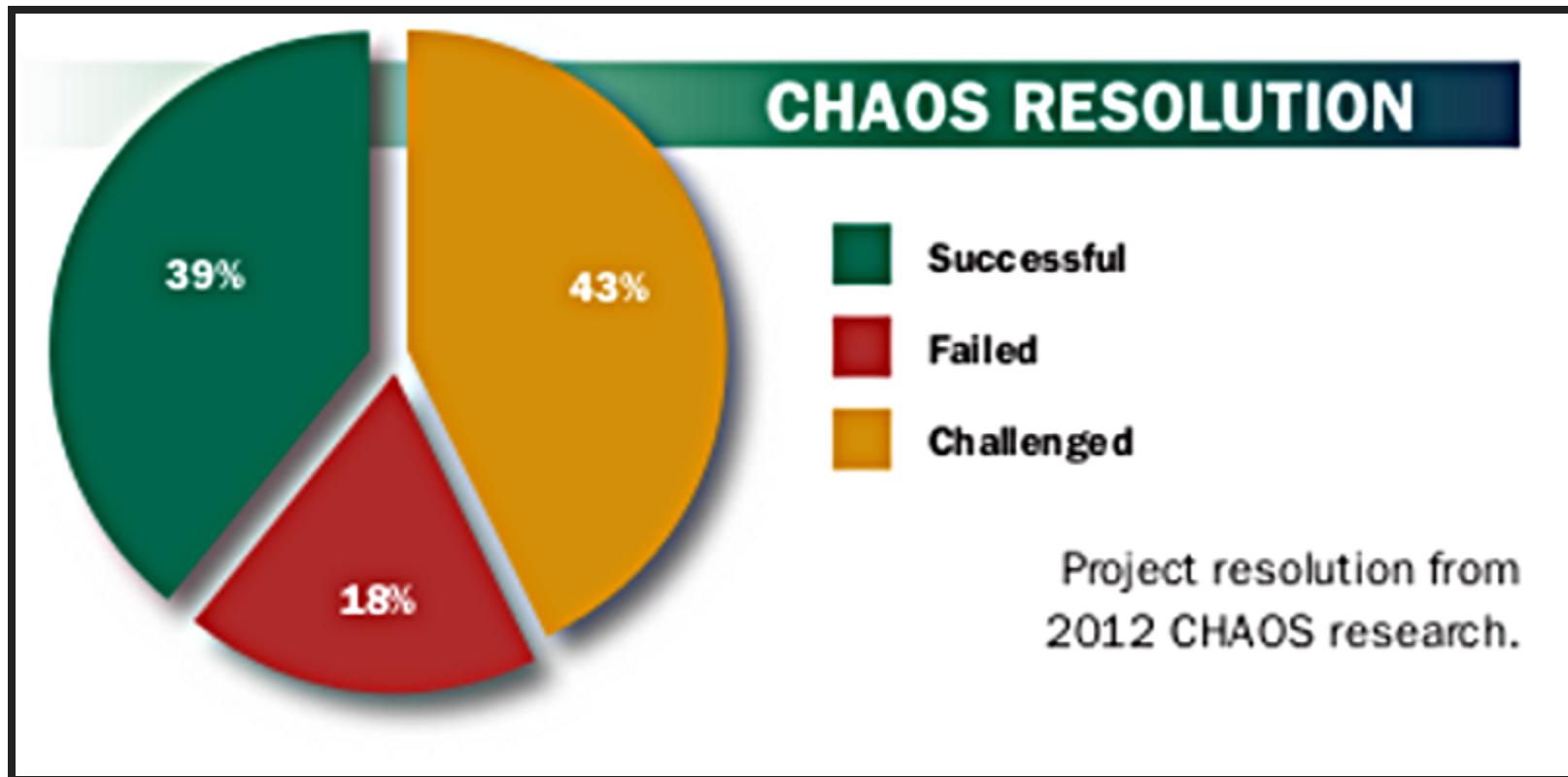
LA CRISIS DEL SOFTWARE

(cortesía del [informe CHAOS](#))

- base de datos de ~50k proyectos
- 18 años de datos; la nueva BD elimina desde 1994 hasta 2002
- 60% USA, 25% Europa, 15% resto
- 50% empresas en el Fortune 1000

pero tomémoslo con cautela

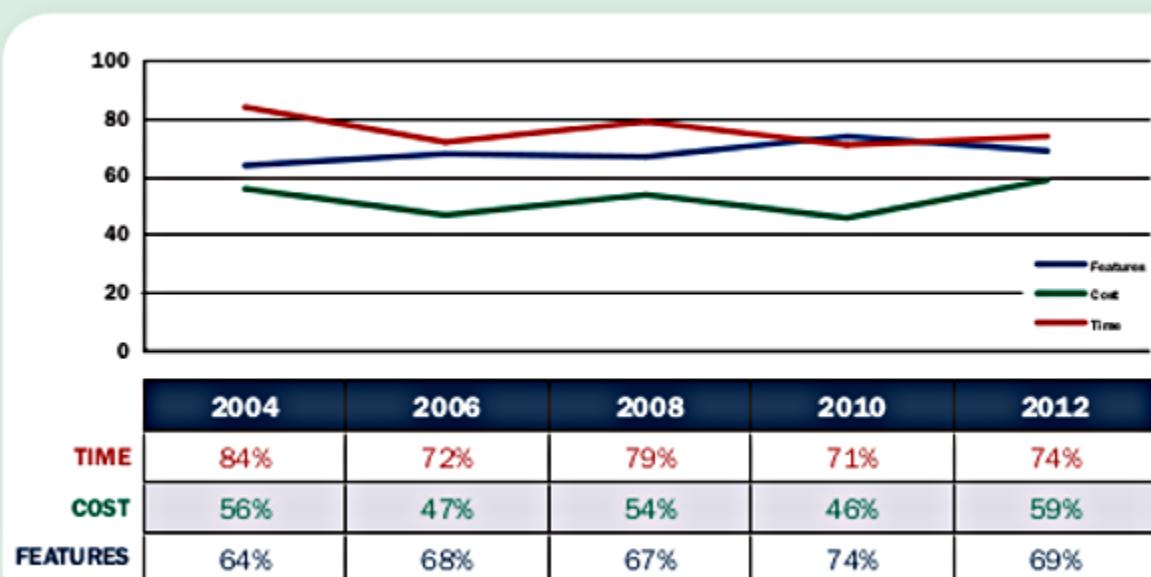
resolución de proyectos



Fuente: [Informe CHAOS 2013](#)

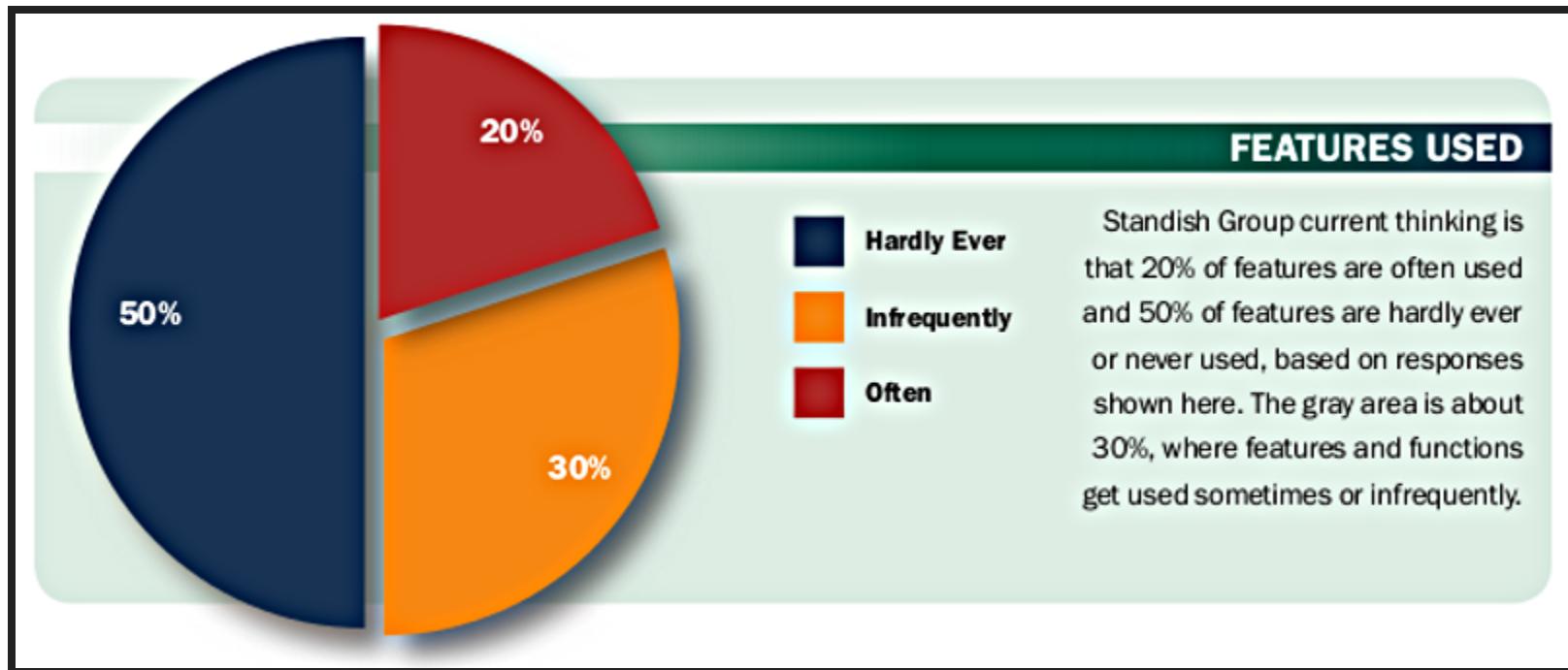
sobrecostes

Time and cost overruns, plus percentage of features delivered from CHAOS research for the years 2004 to 2012.



Fuente: [Informe CHAOS 2013](#)

funcionalidad



fuente: Informe CHAOS 2011

causas

top 5 razones de fracaso

- falta de participación de los **usuarios** (13%)
- especificaciones y **requisitos** incompletos (12%)
- **cambios** en los requisitos y especificaciones (12%)
- falta de habilidades técnicas (7%)
- falta de apoyo de la Dirección (6%)
- Otros (50%)

Fuente: [Informe CHAOS 1994](#)

hay más datos...

- otro estudio de defectos por categoría encontró que la más amplia era la de requisitos (41%)

[Reliability Measurement from Theory to Practice](#), Sheldon, F., Kavi, K. et al.

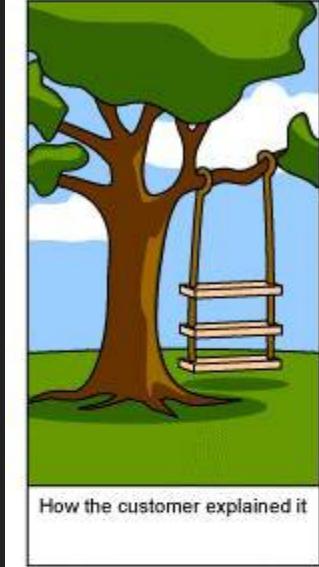
- estudios muestran que los requisitos son susceptibles de cambiar un 25% o más

[Understanding and Controlling Software Costs](#), Boehm et al.

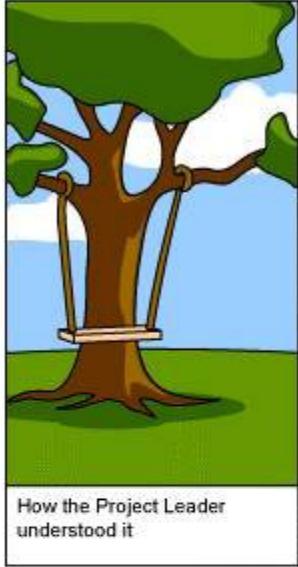
Bohem mostró que el coste de corregir un defecto se incrementa de manera no lineal a medida que transcurren las fases de un proyecto

queremos que los
requisitos sean
estables...

pero no lo son



How the customer explained it



How the Project Leader understood it



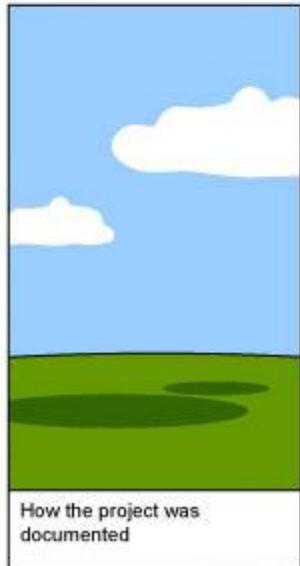
How the Analyst designed it



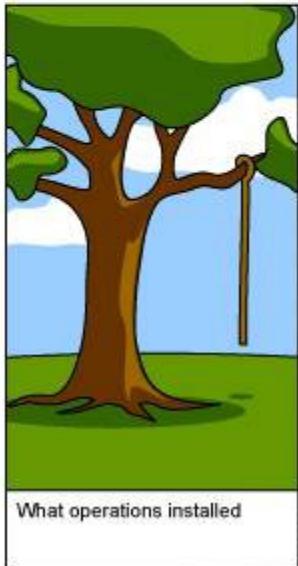
How the Programmer wrote it



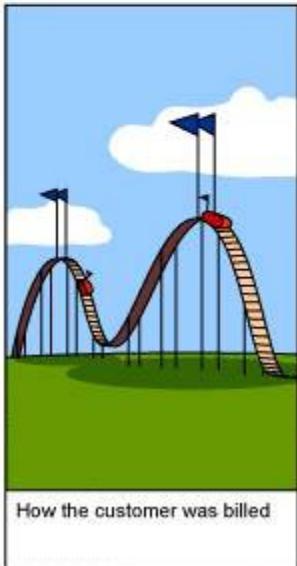
How the Business Consultant described it



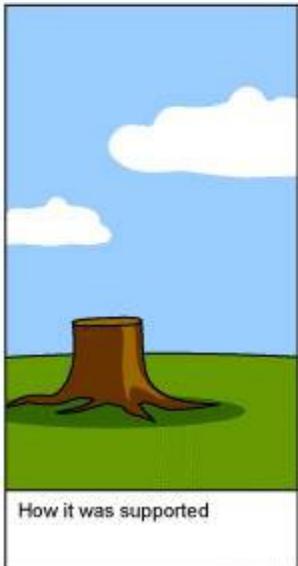
How the project was documented



What operations installed



How the customer was billed



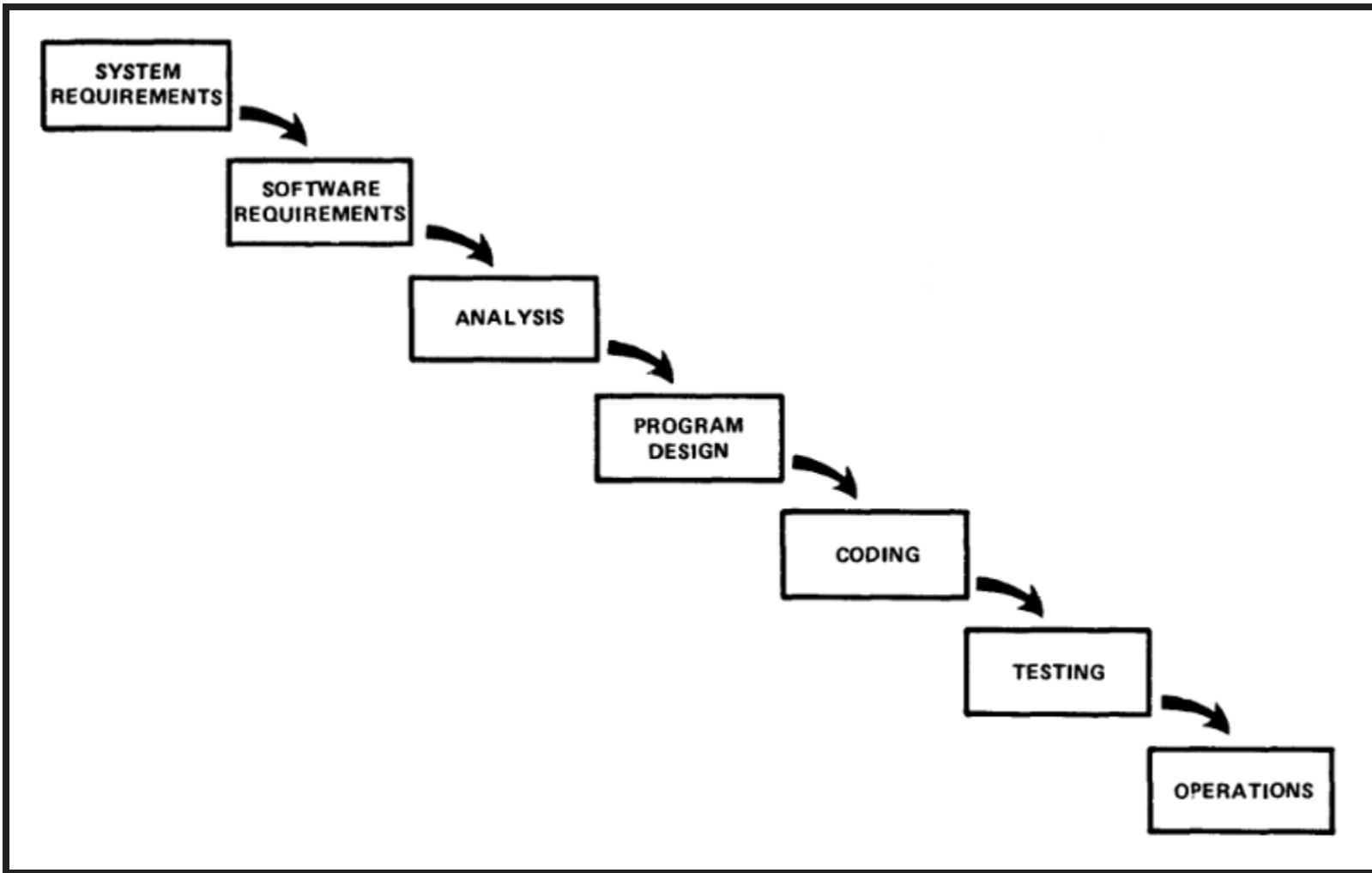
How it was supported



What the customer really needed

enfoque tradicional

desarrollo en cascada



consiste en...

- gestión predictiva de proyectos llevada al software
- se toma como modelo el resto de ingenierías
- intenta llenar el vacío del "code & fix"
- cada fase se realiza una única vez
- cada fase produce un entregable que será entrada de la siguiente
- los entregables no son, en principio, modificables

es decir:
separa diseño de construcción

(o creatividad de repetición)

Taylorismo vs Toyota Production System

- 1 BMW cada 57 horas, 78.7 defectos cada 100 vehículos
- 1 Lexus cada 17 horas, 34 defectos por cada 100 vehículos

Fuente: [Scrum: the art of doing twice the work in half the time](#), Jeff Sutherland (2014)

pero...

con respecto a otras ingenierías...

- los "planos": diagramas UML
- pesos de diseño y construcción invertidos
- la creatividad no es fácilmente predecible
- el software es un dominio de cambio y alta inestabilidad

motivación para la agilidad

desarrollos tradicionales

- muchos requisitos especulativos o vagos y diseño detallado por adelantado
- fuertemente asociados con las **tasas de fallo** más altas en proyectos
- promovidos históricamente **por creencia** más que por evidencia estadística significativa

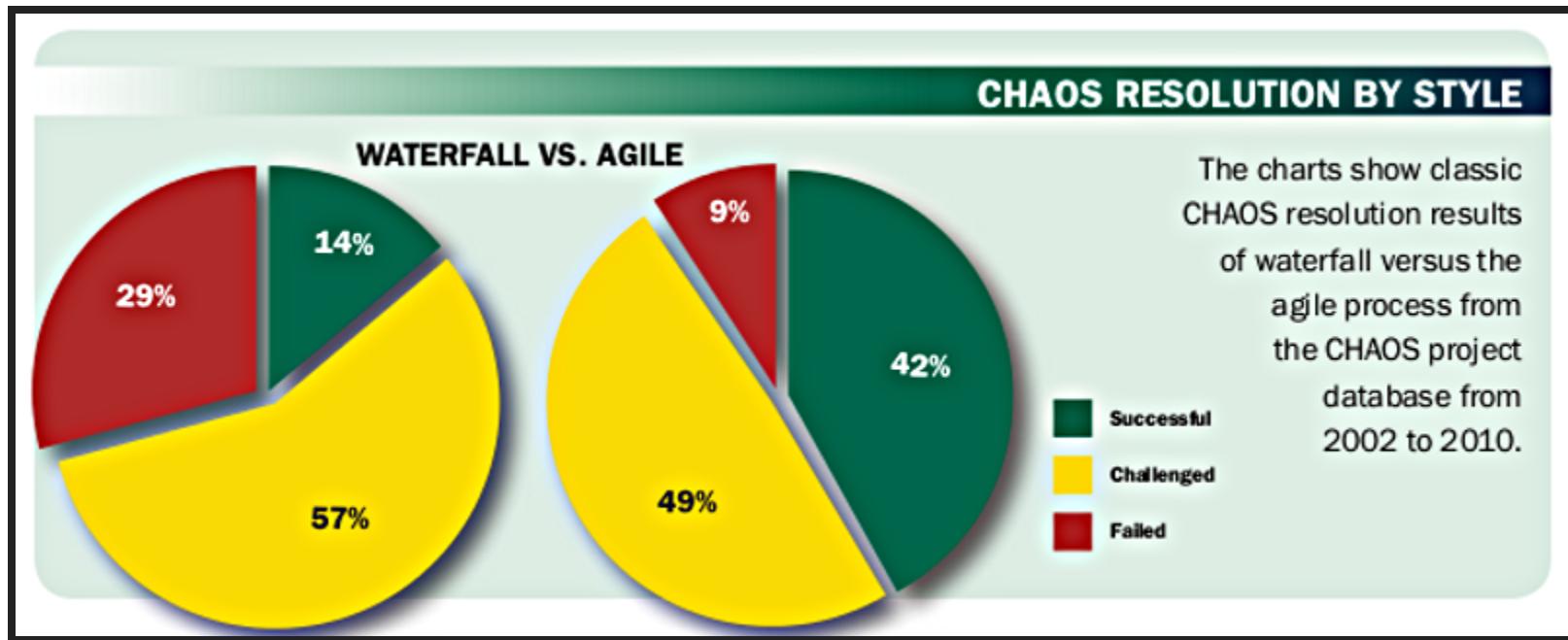
Fuente: [Applying UML and patterns](#), Craig Larman

CON QUE...

"EVIDENCIA ESTADÍSTICA"

memegenerator.es

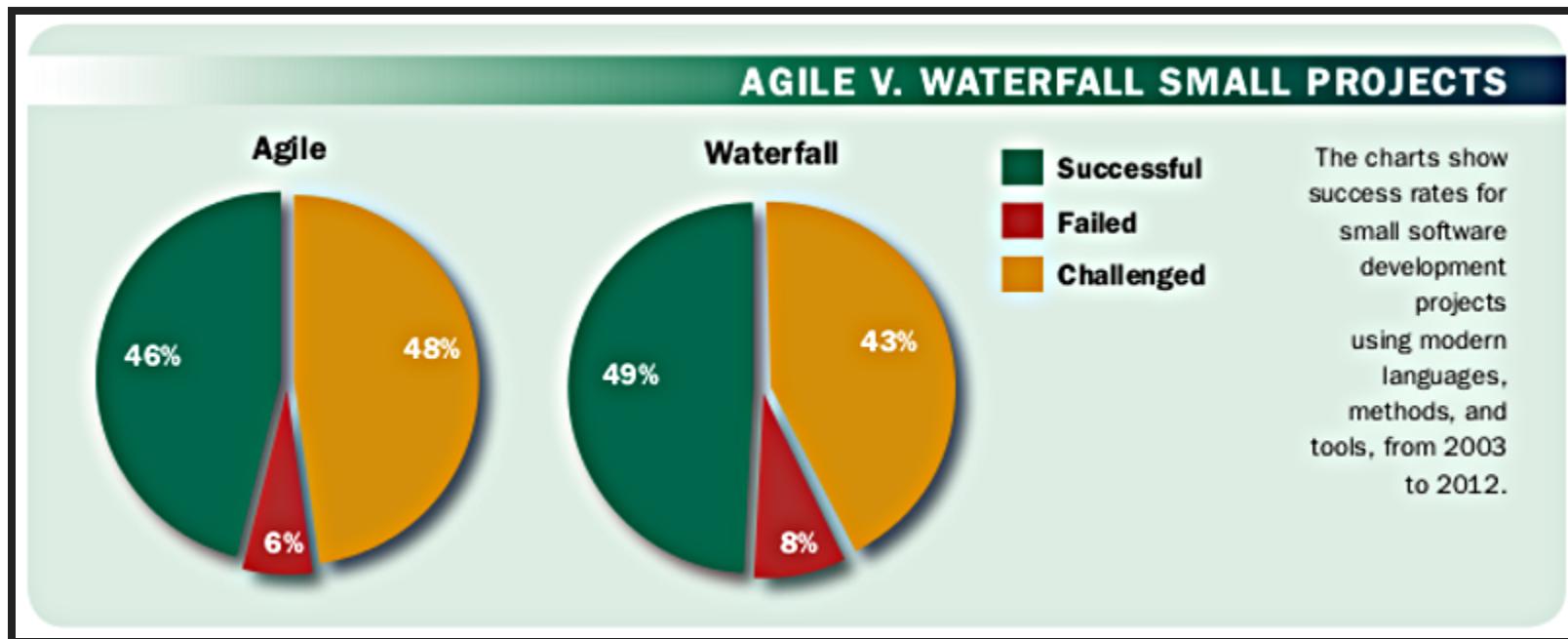
agile vs waterfall



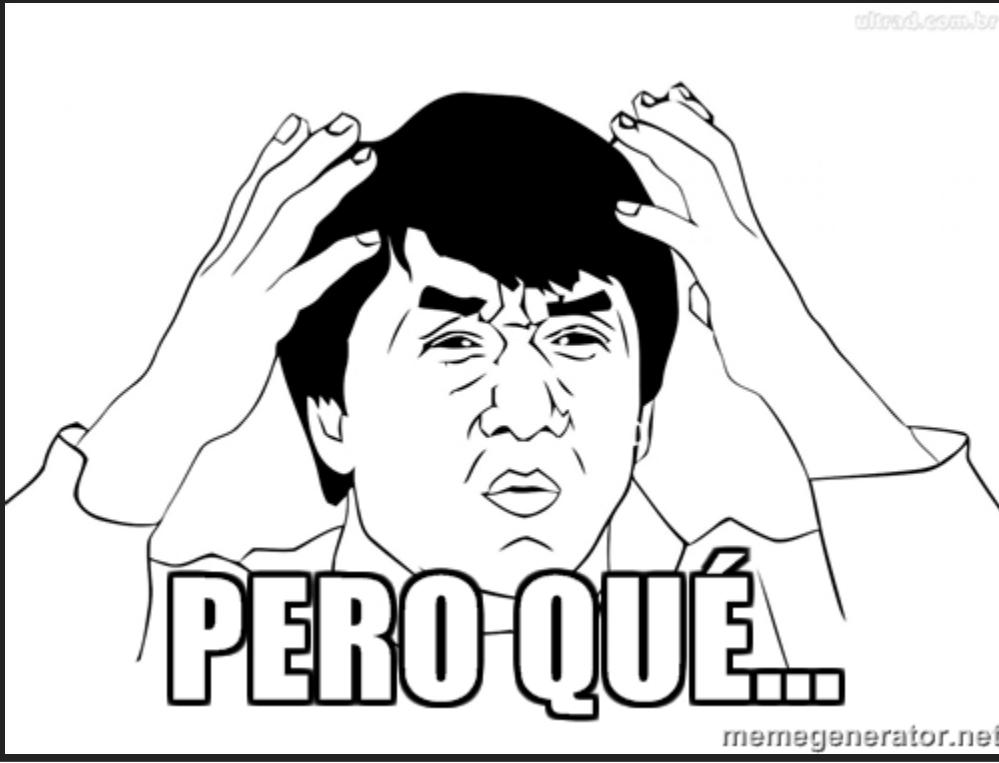
Fuente: Informe CHAOS 2011

agile vs waterfall

proyectos pequeños



Fuente: [Informe CHAOS 2013](#)



ultrad.com.br

PERO QUÉ...

memegenerator.net

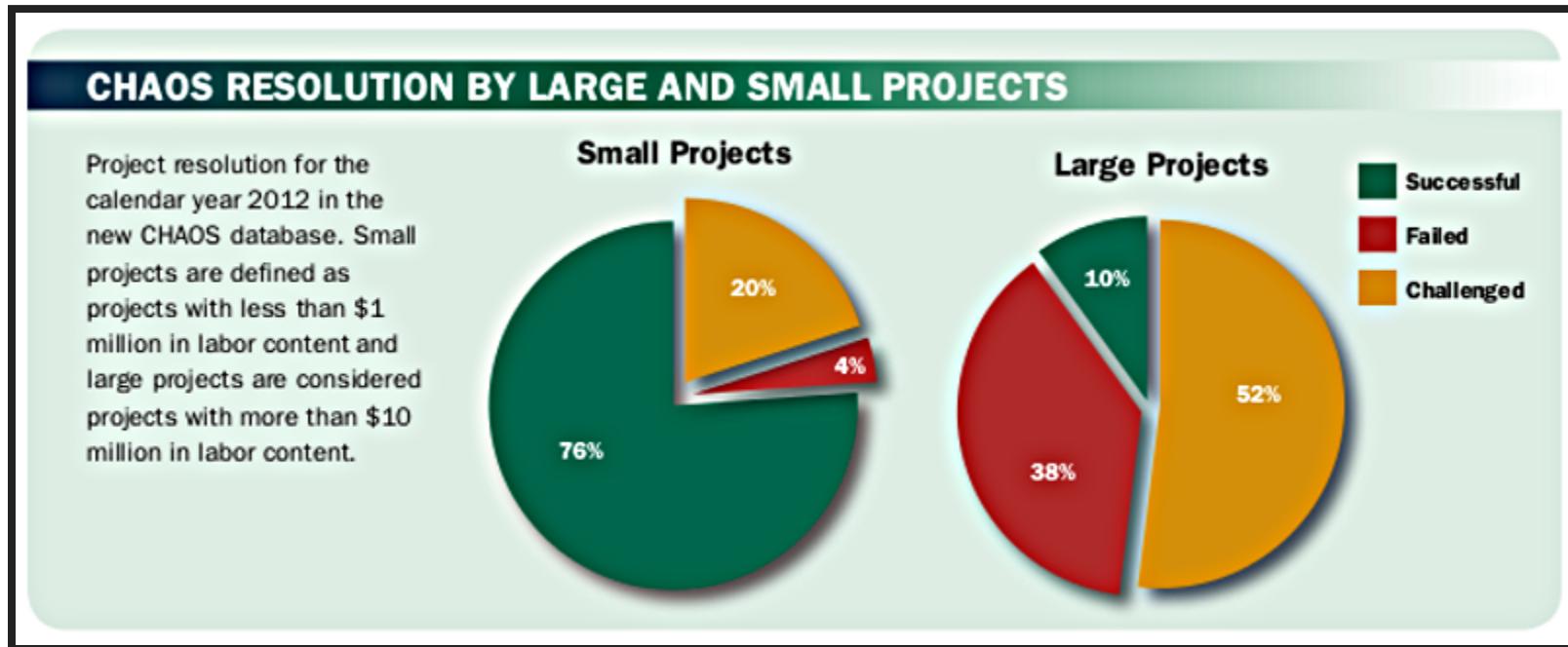
para proyectos pequeños (~700.000€), las metodologías en cascada tienen el **mismo grado de éxito** que las metodologías ágiles.

entonces, ¿son las metodologías ágiles un factor diferenciador en el caso de grandes proyectos?

NO

según el informe CHAOS, los grandes proyectos simplemente están abocados al fracaso.

grandes vs pequeños



Fuente: [Informe CHAOS 2013](#)

aunque...

según el informe CHAOS de 2015:

Los proyectos ágiles tienen un 350% más de probabilidades de tener éxito que los de cascada

“Los resultados globales muestran claramente que los proyectos cascada no escalan bien, mientras que los proyectos ágiles escalan mucho mejor.”

Fuente: [Erik Weber Consulting sobre Informe CHAOS 2015](#)

UNA PREGUNTA...

¿NO HAY MÁS EVIDENCIA?

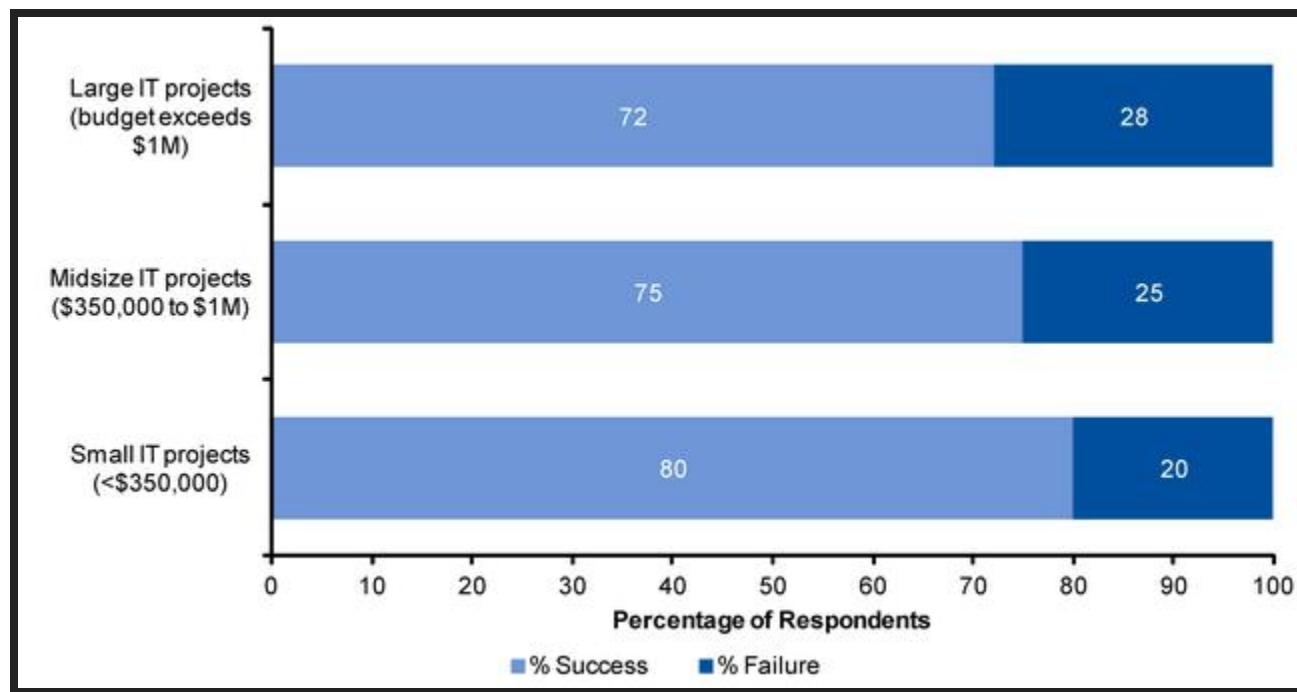
imgflip.com

encuesta con 173 muestras en Noviembre/Diciembre 2013:

Paradigma	Éxito	Reparos	Fallido
Lean	72%	21%	7%
Agile	64%	30%	6%
Iterativo	65%	28%	7%
Ad hoc	50%	35%	15%
Tradicional	49%	32%	18%

Fuente: [Dr. Dobb's Journal article The Non-Existent Software Crisis: Debunking the Chaos Report](#), Scott W. Ambler

encuesta con 154 organizaciones de varios países en 2011:



Fuente: [Gartner](#)

estudio con 38 organizaciones y 1027 proyectos:

- sólo 130 (el 12,7%) tuvieron éxito
- el factor con mayor contribución para el fracaso: el **desarrollo en cascada**, citado en el 82% de los proyectos como causa principal del mismo, con una influencia global ponderada del 25%

Fuente: [BCS Computer Bulletin](#), Thomas M.

el caso del DoD

el estándar DOD-STD-2167 obligaba a usar un ciclo de vida en cascada
un informe concluyó que el 75% de los proyectos fracasaron o nunca
se utilizaron

el grupo de trabajo, liderado por Brooks, concluyó:

“DOD-STD-2167 necesita una revisión radical para reflejar mejor las prácticas modernas. En la década posterior al desarrollo del modelo en cascada, nuestra disciplina ha llegado a reconocer que dicho desarrollo requiere iteraciones entre los diseñadores y los usuarios. El desarrollo evolutivo es mejor técnicamente y ahorra tiempo y dinero.”

Fuente: [Report of the Defense Science Board Task Force on Military Software](#), Oct. 1987. USA DoD. Brooks, F., et al.

estudio sobre 400 proyectos orientados a cascada a lo largo de 15 años en 2001:

- menos del 5% del código desarrollado fue útil o usado en algún momento
- sólo se desplegó el 10% del código desarrollado

Fuente: Improving Software Investments through Requirements Validation, Cohen, D., Larson, G., and Ware, B. 2001



por tanto...

think big, act small

- descomponer grandes proyectos en otros más pequeños
- dar prioridad al valor de negocio directo
- disponer de equipos multifuncionales siguiendo un enfoque ágil.

factores de éxito

proyectos pequeños

Factors of Success	Points
Executive management support	20
User involvement	15
Optimization	15
Skilled resources	13
Project management expertise	12
Agile process	10
Clear business objectives	6
Emotional maturity	5
Execution	3
Tools and infrastructure	1

Fuente: [Informe CHAOS 2013](#)

Y ENTONCES...

¿EL PROCESO ÁGIL?

memegenerator.net

el proceso ágil:

- facilita llevar a cabo proyectos pequeños
- facilita dividir proyectos grandes en subproyectos pequeños
- fomenta la interacción entre el equipo y los usuarios
- provoca que los inevitables cambios en requisitos sucedan en fases tempranas del proyecto

¿SUS SECRETOS?

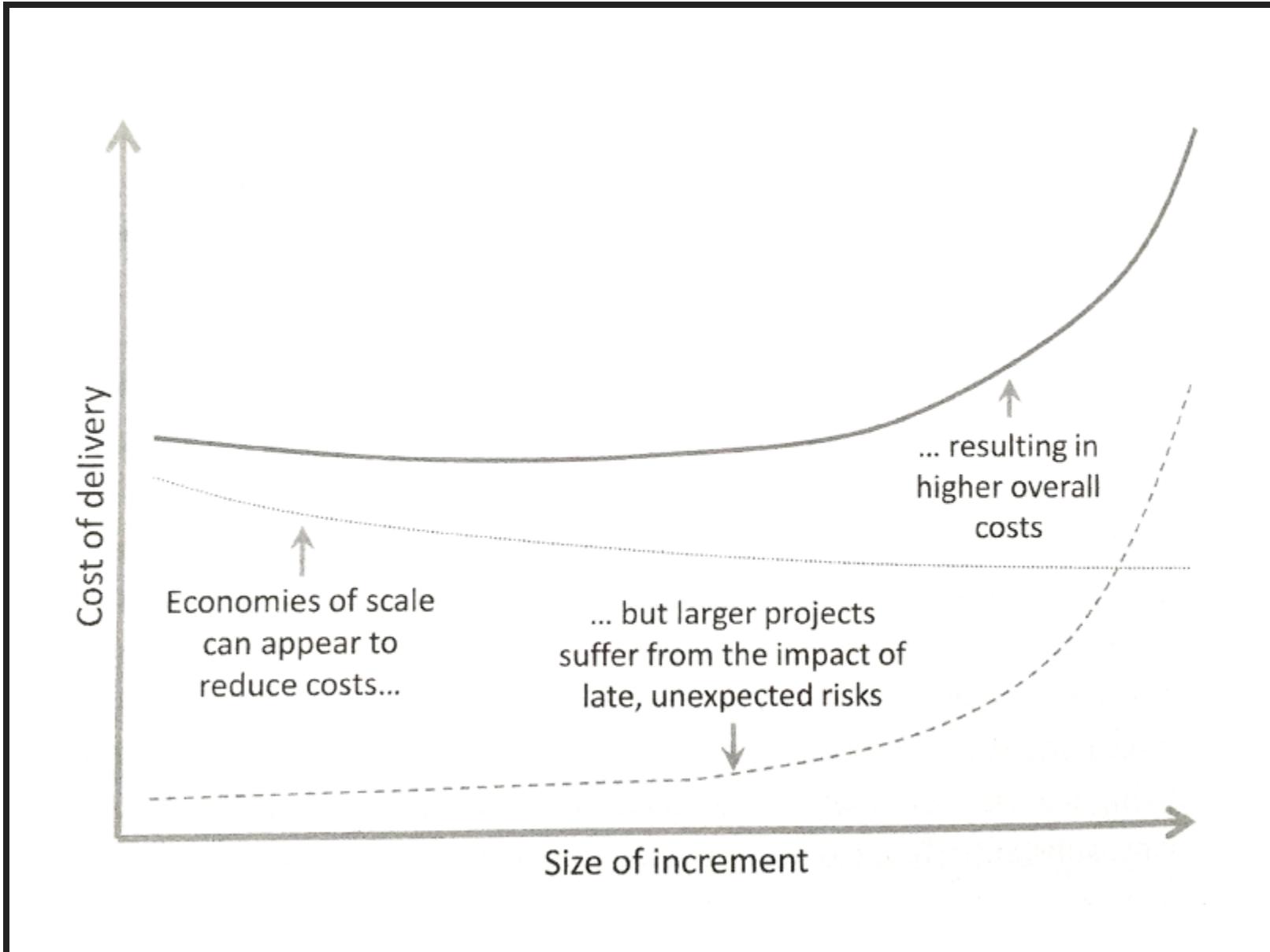
- entregas frecuentes y ciclos prueba/error
- desarrollo en pasos pequeños (incrementales) e iterativos por equipos también pequeños
- flexibilidad para la gestión del cambio
- limitación en el tiempo o time-boxing (Google)
- interacción humana

incremental e iterativo (años 50)

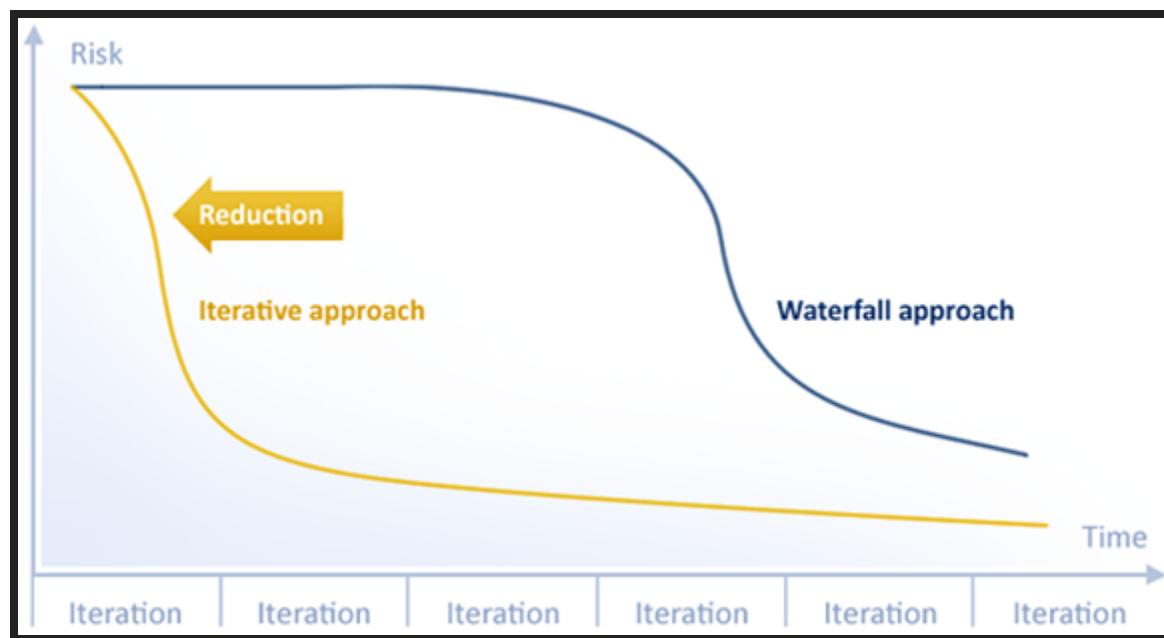
- se trabaja sobre subconjuntos de funcionalidad (features)
- incremental: añadir funcionalidad al producto (ayuda a mejorar el proceso)
- iterativo: revisar, rediseñar, re-trabajar el producto (ayuda a mejorar el producto)

¿y el desarrollo en cascada?

- su rigidez incrementa el riesgo de fracaso, pospuesto hasta las fases finales del proyecto
- asume que las especificaciones son predecibles, estables y completas
- pospone integración y tests hasta fases tardías
- se basa en estimaciones y planificación "fiables"



Fuente: [Agile project management for Government](#)



Fuente: [PLAUT](#)

**¿qué opinan los
expertos?**

Royce: autor del artículo más citado sobre el desarrollo en cascada

- su mensaje fue malentendido: irónicamente, defendía el desarrollo iterativo
- en su artículo describió el modelo en cascada como la "*descripción más simple*" que **sólo funcionaría para los proyectos más sencillos**
- su artículo no estaba basado en evidencia, sino que se trataba de su visión personal

Fuente: [Managing the development of large software systems](#), Winston Royce, a través de [Agile and Iterative Development: a manager's guide](#)', Craig Larman.

Brooks: manager de IBM OS/360, autor de [The Mythical Man-Month](#) critica el modelo en cascada como no deseable

“Gran parte de los procedimientos actuales de adquisición de software se basa en la suposición de que es posible especificar previa y satisfactoriamente los requisitos de un sistema, obtener ofertas para su construcción, construirlo e instalarlo. Creo que esta suposición está fundamentalmente equivocada, y que muchos de los problemas de la adquisición de software surgen de esta falacia.”

Fuente: No silver bullet, Frederick Brooks a través de [Agile and Iterative Development: a manager's guide](#)', Craig Larman.

Boehm: creador de modelos en espiral y COCOMO

*"Una fuente de **dificultad** importante con el modelo en cascada ha sido su énfasis en usar **documentos completamente elaborados como criterio de aceptación** para requisitos tempranos y fases de diseño. Para algunos tipos de software, como compiladores o sistemas operativos seguros, esta es la forma más efectiva de proceder.*

Pero no funciona bien para muchas clases de software, particularmente para aplicaciones interactivas con el usuario final."

Fuente: [A Spiral Model of Software Development and Enhancement](#), Barry Boehm.

Yourdon: fundador del análisis estructurado

- junto con DeMarco, impulsó en los 70 el análisis estructurado con énfasis en el modelo en cascada
- a mediados de los 80, pasó a ser promotor del desarrollo iterativo

*“Las nuevas herramientas han llevado a muchas organizaciones a utilizar un **enfoque de desarrollo iterativo** que generalmente ofrece resultados más exitosos que el **enfoque en cascada**, en parte necesitado por la antigua generación de herramientas orientadas al mainframe y a lotes.”*

Fuente: "The Future of Software: Best of Times, Worst of Times," IEEE Software, Enero 1998. Ed. Yourdon.

**¿y por qué se sigue
promoviendo el
desarrollo en cascada?**

inercia

en la historia de la ciencia, es normal que las primeras ideas en torno a un campo, incluso en ausencia de resultados, se conviertan en predominantes

desconocimiento

lo cierto es que pocos han leído el artículo original de Royce sobre el modelo en cascada

incluso los diagramas que supuestamente describen su modelo, no se corresponden con el diagrama iterativo original de Royce.

comodidad

el modelo en cascada de una "sola pasada" es más sencillo de explicar
que el iterativo

de hecho, la propuesta original de Royce en dos iteraciones fue
evolucionada por otros autores hacia un modelo con un único paso.

paradigma equivocado

en la mayor parte de los casos, el software se asemeja más al
paradigma de desarrollo de un nuevo producto que al de fabricación
predecible.

ilusión de control

muchos jefes de proyecto han impulsado los ciclos en cascada por la sensación de ofrecer un proceso predecible, ordenado y medible con hitos basados en entrega de documentación

promoción por parte de ciertos grupos

ingeniería de requisitos lo sigue promoviendo como adecuado, quizá por no estar familiarizados con la evidencia o con los modelos evolutivos

estándares

- CMMI, en sus inicios, se acercaba más al paragidma en cascada
- los primeros contenidos de PMI - PMBOK eran más consistentes con la planificación y fabricación predecibles



**KEEP
CALM
AND
EMBRACE
CHANGE**

manifiesto por el desarrollo ágil de software

Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

individuos e interacciones sobre procesos y herramientas
software funcionando sobre documentación extensiva
colaboración con el cliente sobre negociación contractual
respuesta ante el cambio sobre seguir un plan
esto es, aunque valoramos los elementos de la derecha, valoramos
más los de la izquierda.

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

principios del manifiesto ágil

seguimos estos principios:

nuestra mayor prioridad es satisfacer al cliente mediante la entrega
temprana y continua de software con valor

principios del manifiesto ágil

seguimos estos principios:

aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente

principios del manifiesto ágil

seguimos estos principios:

entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible

principios del manifiesto ágil

seguimos estos principios:

los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto

principios del manifiesto ágil

seguimos estos principios:

los proyectos se desarrollan en torno a individuos motivados; hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo

principios del manifiesto ágil

seguimos estos principios:

el método más eficiente y efectivo de comunicar información al equipo
de desarrollo y entre sus miembros es la conversación cara a cara

principios del manifiesto ágil

seguimos estos principios:

el software funcionando es la medida principal de progreso

principios del manifiesto ágil

seguimos estos principios:

los procesos ágiles promueven el desarrollo sostenible; los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida

principios del manifiesto ágil

seguimos estos principios:

la atención continua a la excelencia técnica y al buen diseño mejora la agilidad

principios del manifiesto ágil

seguimos estos principios:

la simplicidad, o el arte de maximizar la cantidad de trabajo no
realizado, es esencial

principios del manifiesto ágil

seguimos estos principios:
las mejores arquitecturas, requisitos y diseños emergen de equipos
auto-organizados

principios del manifiesto ágil

seguimos estos principios:

a intervalos regulares el equipo reflexiona sobre cómo ser más efectivo
para a continuación ajustar y perfeccionar su comportamiento en
consecuencia

malas interpretaciones

- ausencia de documentación
- ausencia de planificación
- el cliente hace todo el trabajo
- se puede modificar la metodología sin justificación

tradicional vs ágil

característica	tradicional	ágil
proceso	altamente controlado	ligeramente controlado
contrato	cerrado	flexible
cliente	reuniones	forma parte del equipo
tamaño del equipo	grandes	pequeños
documentación	exhaustiva	estrictamente

necesaria



[Acceso al repositorio del curso](#)

[Miguel Expósito Martín](#)