


Tema 2

De la crisis del software al Manifiesto Ágil

La crisis del software





Conferencia OTAN 1968

Conferencia OTAN 1968



Propuestas de solución

- **Nuevas metodologías de análisis y diseño en base a unas fases secuenciales**
- Dedicar un tiempo razonable a **pensar en el diseño** de la solución a utilizar antes de escribir una línea de código.
- Definir unos **modelos gráficos** que permitirían al analista-programador comunicarse de manera mucho más razonable con el cliente a la hora de valorar si había comprendido realmente cuáles eran sus necesidades.
- **Documentación** de todos los aspectos que conllevaban la programación y que multiplicaba las líneas de código de programa con líneas de texto que describían cada una de las variables, el objetivo de cada una de las funciones o procedimientos.

A long-exposure photograph of a waterfall cascading over mossy rocks in a lush forest. The water appears as a series of white, silky strands falling into a pool below. The surrounding rocks and foliage are covered in vibrant green moss and ferns. In the top left corner, there is a small logo that reads '4ever.com'.

Las fases del modelo Waterfall (cascada)

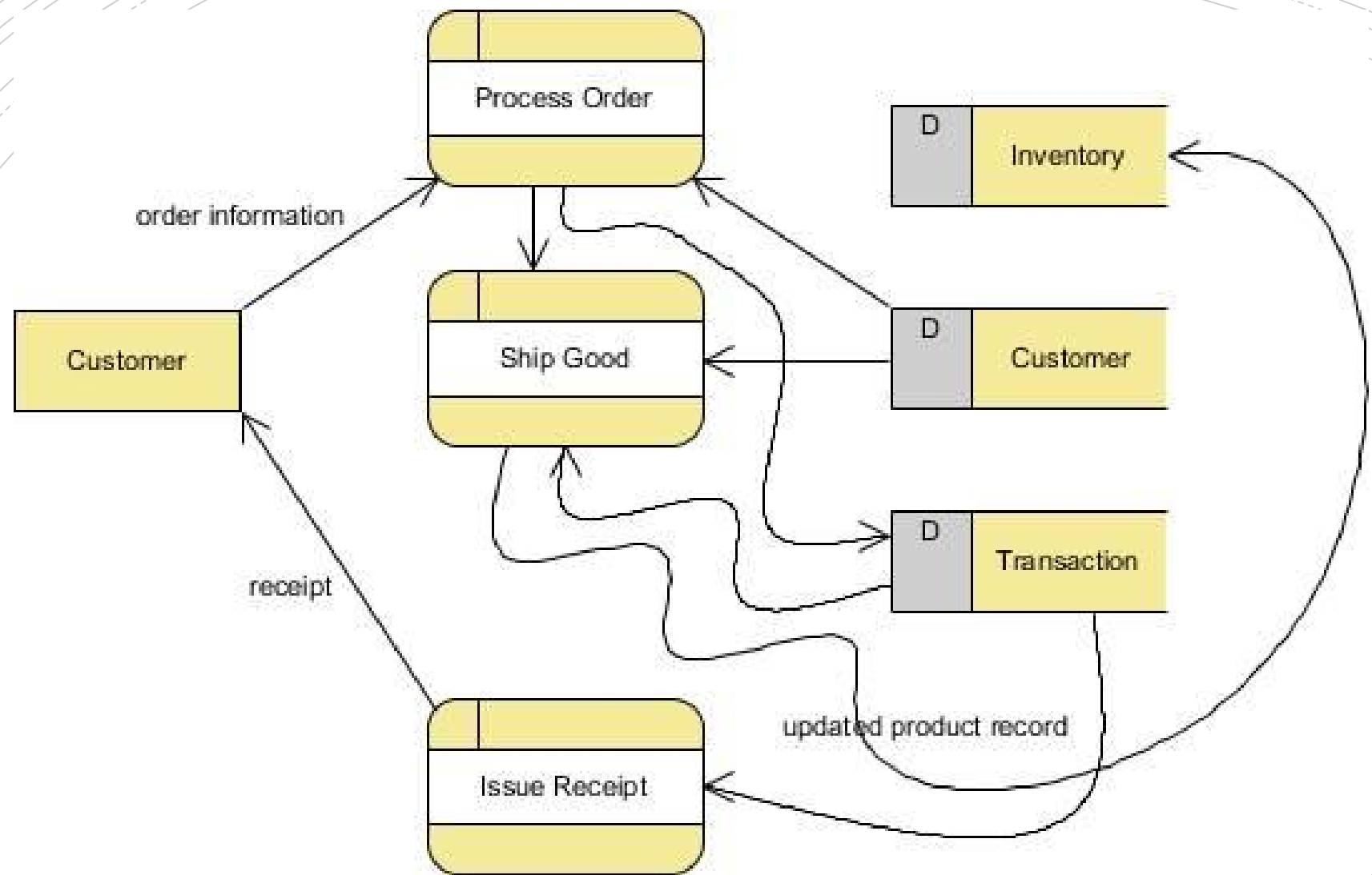
- Definición de requerimientos
- Análisis y Diseño de Software
- Implementación
- Prueba
- Implantación
- Mantenimiento

1º Definición de requerimientos



¿Qué quiere el cliente o qué es lo que yo creo que quiere el cliente?

2º Análisis y Diseño de Software



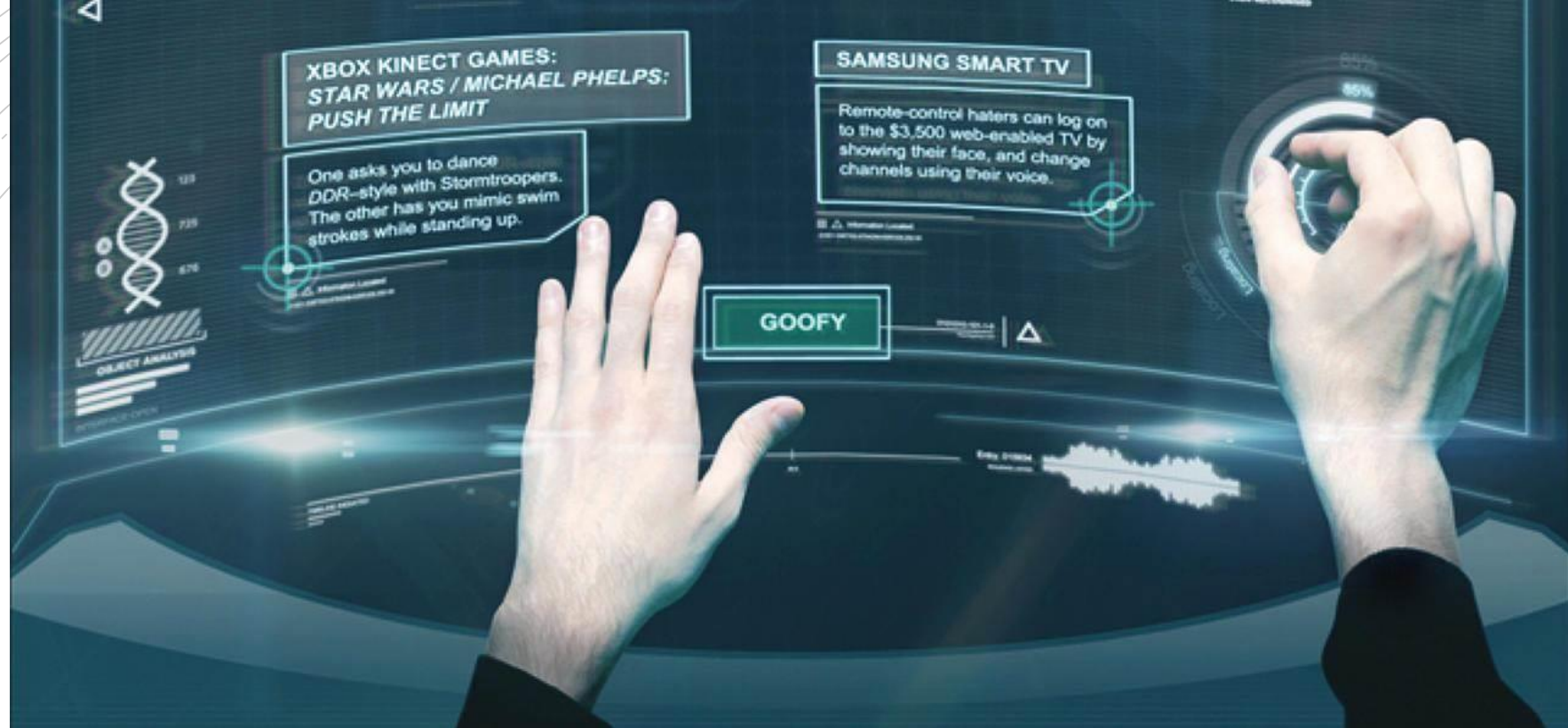
¿Qué vamos a hacer y cómo lo vamos a hacer?

3º Implementación



Hacemos lo que creo que el analista dice sobre lo que quería el cliente

4º Pruebas



Probamos que hace lo que pensaba que tenía que hacer

5º implantación



Instalamos el programa en casa del cliente

6º Mantenimiento



Solucionamos los pequeños problemillas que puedan surgir

El feliz modelo de la OTAN y las metodologías en cascada

Pero.....





Problemas



El cliente debe ser capaz de especificar sin ambigüedades los requisitos del sistema que quiere.

La incertidumbre afecta irremediablemente al resultado final.

Detección tardía de errores



Los contactos del cliente con el desarrollo se producen fundamentalmente al principio y mucho más tarde en la fase de implantación y pruebas.

La mayoría de errores se detectan en estas últimas etapas afectando notablemente a todo el proyecto.

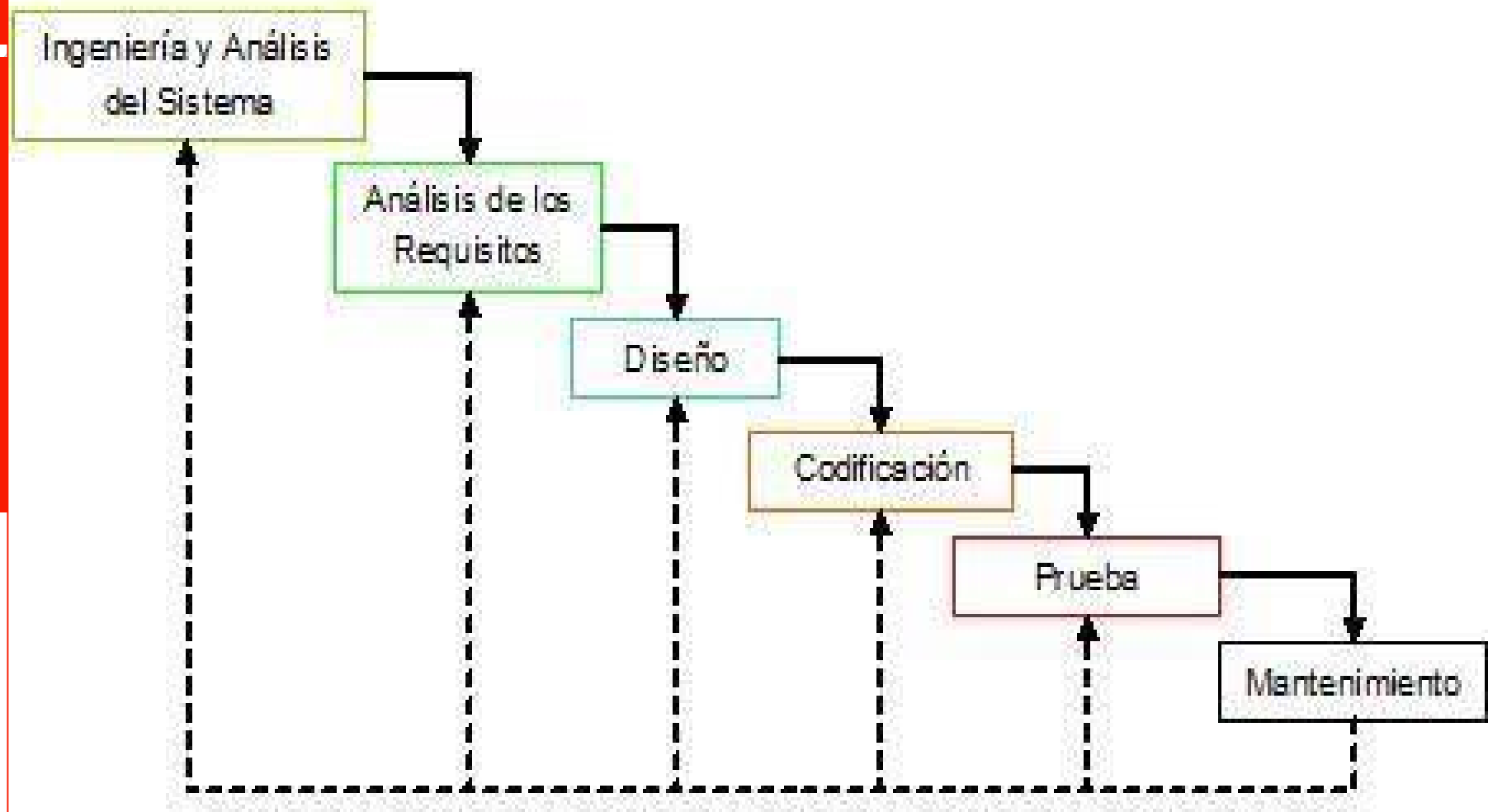
La
documentación
no se actualiza



El entorno cliente
no es el de
desarrollo



Solución : La
realimentación



Pero al final
seguimos igual

- Los proyectos no terminaban en plazo.
- Los proyectos no se ajustaban al presupuesto inicial.
- Baja calidad del software generado.
- Software que no cumplía las especificaciones.
- Código inmantenible que dificultaba la gestión y evolución del proyecto.

Problemas

- Comunicación Cliente-Analista
- Comunicación Analista-Programador
- Modelo jerárquico poco integrador del equipo
- Detección tardía de errores



Algo más cercanos



Kent Beck



Mike Beedle



Arie van Bennekum



Alistair Cockburn



Ward Cunningham



Martin Fowler



James Grenning



Jim Highsmith



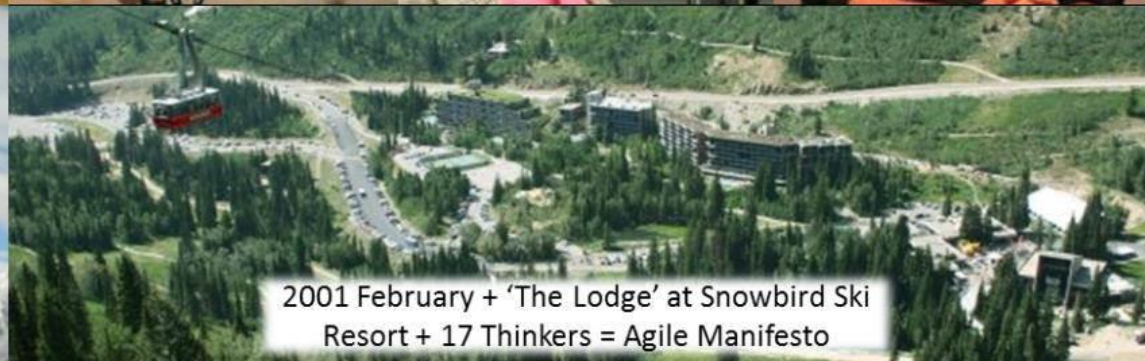
Andrew Hunt



Ron Jeffries



Jon Kern



2001 February + 'The Lodge' at Snowbird Ski Resort + 17 Thinkers = Agile Manifesto



Brian Marick



Bob Martin



Stephen Mellor



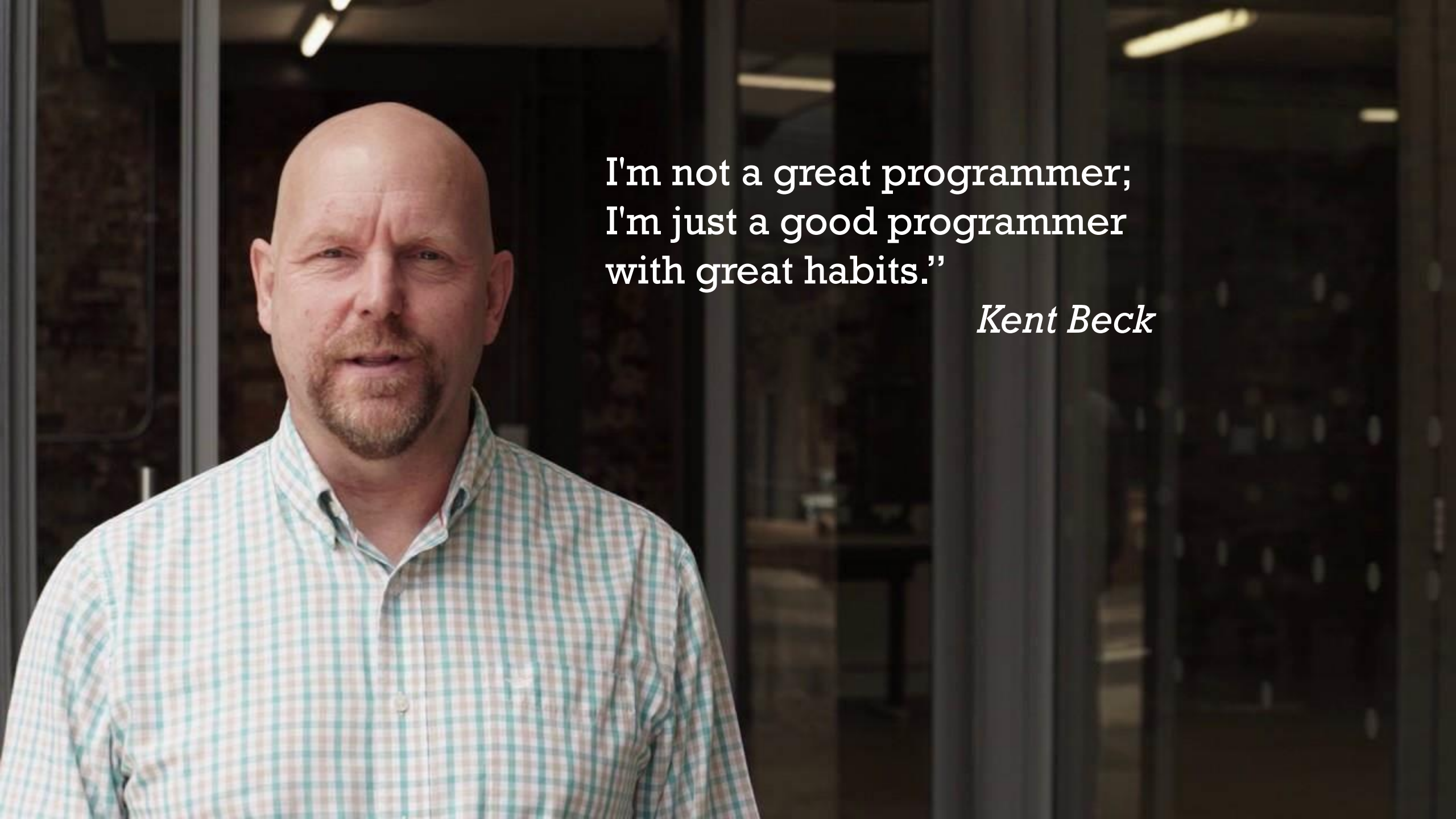
Jeff Sutherland



Ken Schwaber



Dave Thomas



I'm not a great programmer;
I'm just a good programmer
with great habits.”

Kent Beck




“La única forma de ir rápido es ir bien”
Robert C. Martín UncleBob

“A manager's most important work is helping the people doing the work. Give them a goal and let them work. Remove any impediments that get in their way. Do anything that make them more effective or productive. Then the organization can capitalize on the fruits of their work.”

Ken Schwaber



A man with a beard and balding head, wearing a red button-down shirt, is speaking and gesturing with his right hand. The background is dark and out of focus.

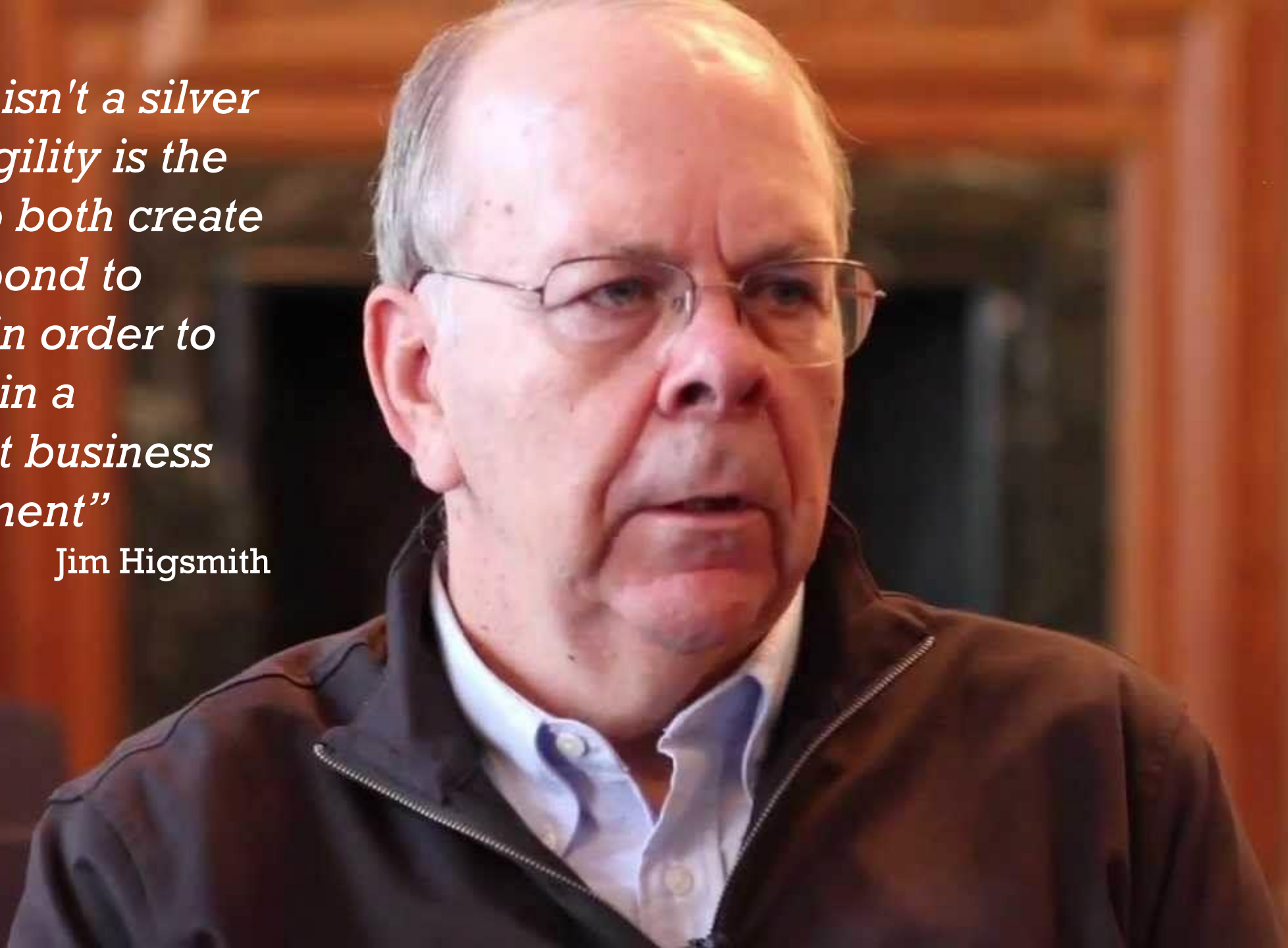
“Any fool can write code that a computer can understand. Good programmers write code that humans can understand.”

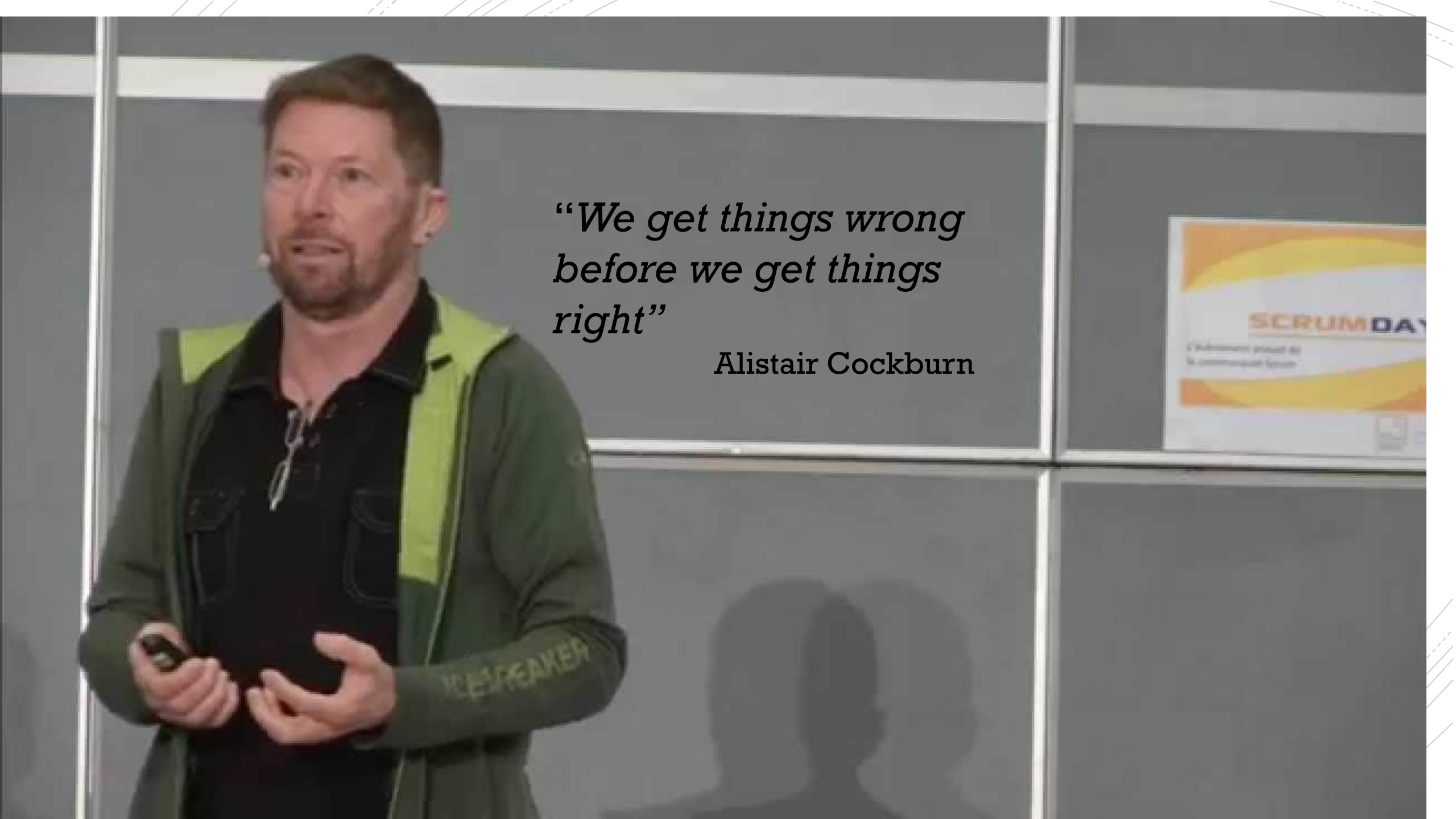
Martin Fowler

“When you feel the need to write a comment, first try to refactor the code so that any comment becomes superfluous.”

“Agility isn't a silver bullet. Agility is the ability to both create and respond to change in order to prosper in a turbulent business environment”

Jim Higsmit



A man with a beard and short brown hair is speaking at a conference. He is wearing a dark green jacket with a bright green stripe on the shoulder and a black shirt. He is holding a small object in his right hand and gesturing with his left. The background is a grey wall with a white grid pattern. To the right, there is a poster for 'SCRUM DAY' with a yellow and orange design. The quote 'We get things wrong before we get things right' is overlaid on the image in a black, italicized font.

*“We get things wrong
before we get things
right”*

Alistair Cockburn

Manifiesto Ágil

- 4 Postulados
- 12 Principios



1. Individuos e interacciones
sobre
procesos y herramientas

2. Software funcionando
sobre
documentación extensiva

3. Colaboración con el cliente
sobre
negociación contractual

4. Respuesta ante el cambio
sobre
seguir el plan

1º Principio



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

2º Principio



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

3º Principio



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

4º Principio



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

5º Principio



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.

6º Principio



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

7º Principio



El software **funcionando** es la medida principal de progreso.

8º Principio



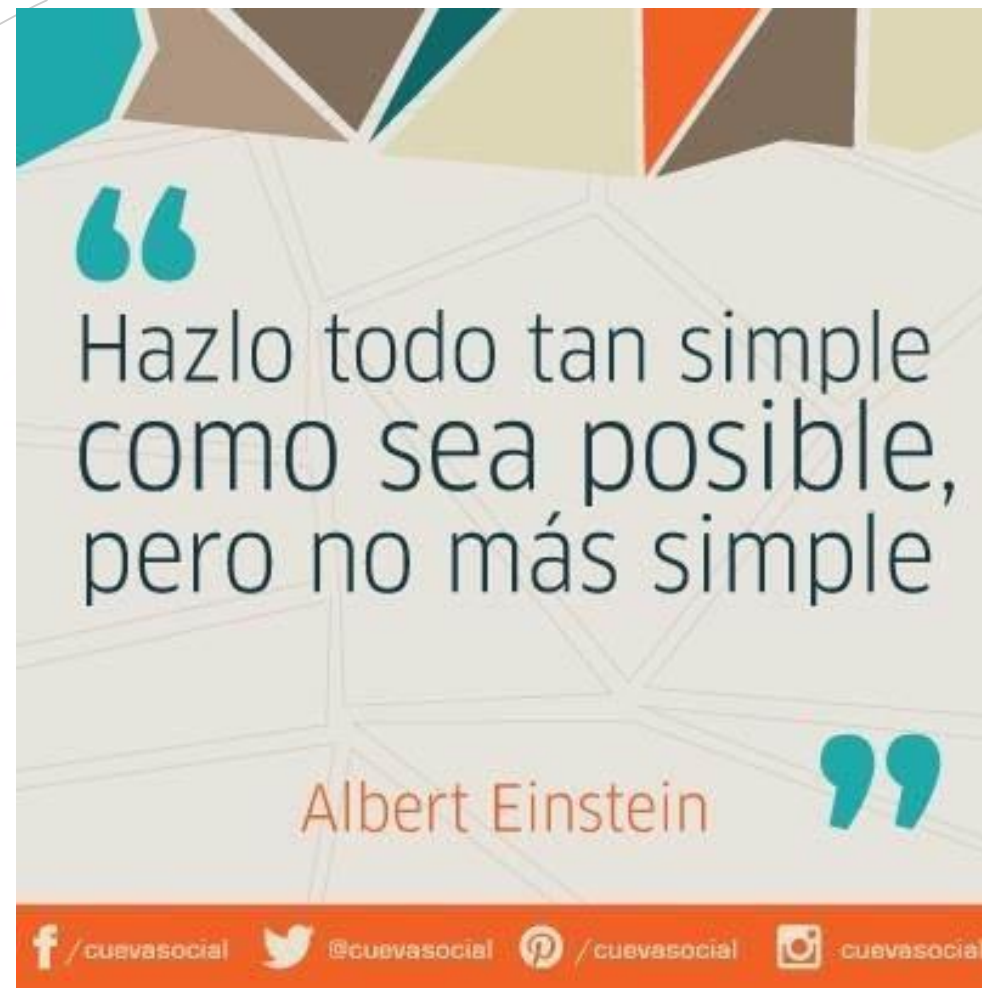
Los procesos Ágiles promueven el **desarrollo sostenible**. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un **ritmo constante** de forma indefinida.

9º Principio



La atención continua a la **excelencia técnica** y al **buen diseño** mejora la **Agilidad**.

10º Principio



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

11º Principio



Las mejores arquitecturas, requisitos y diseños emergen de equipos **auto-organizados**.

12º Principio



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



WOULD YOU LIKE TO KNOW MORE?