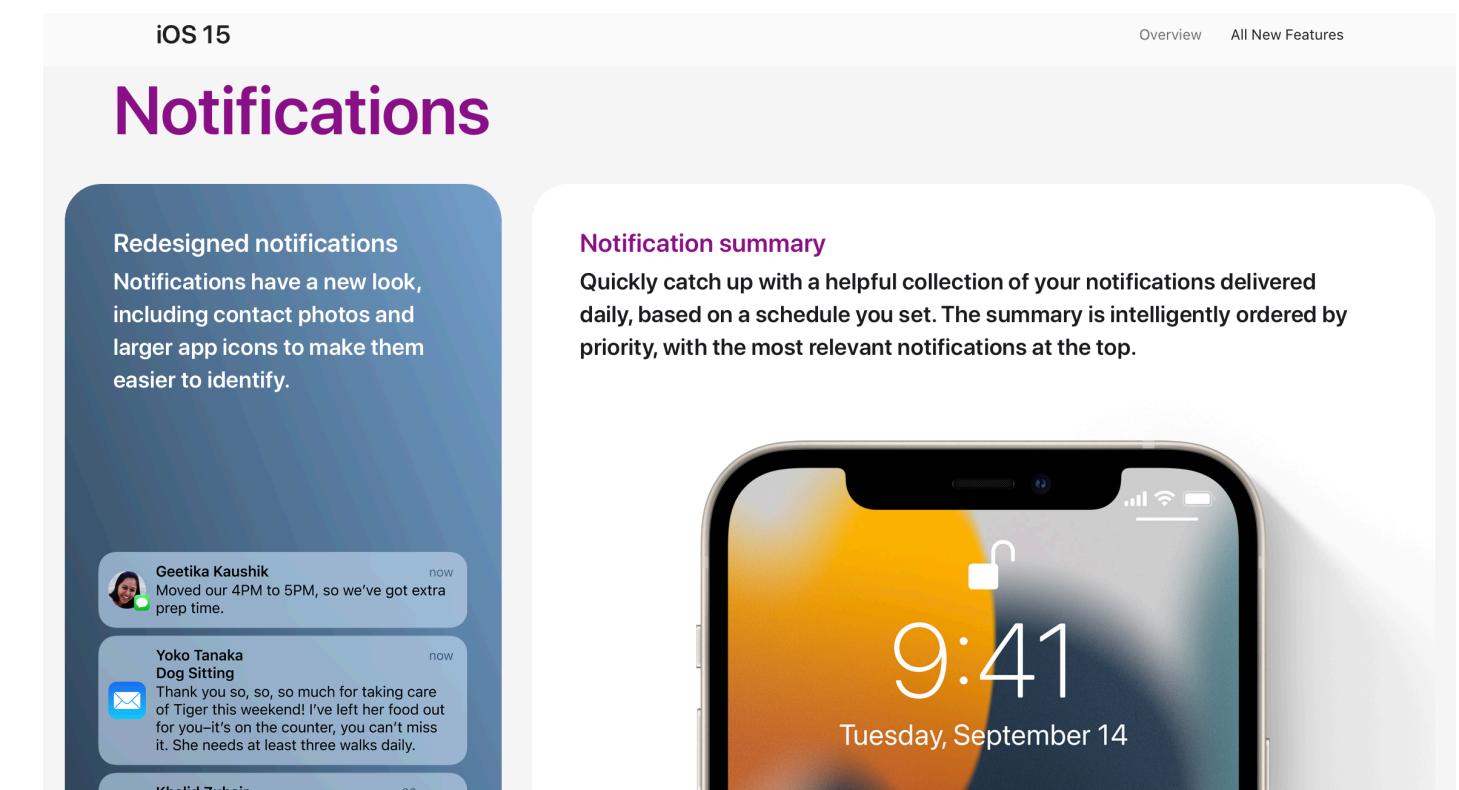
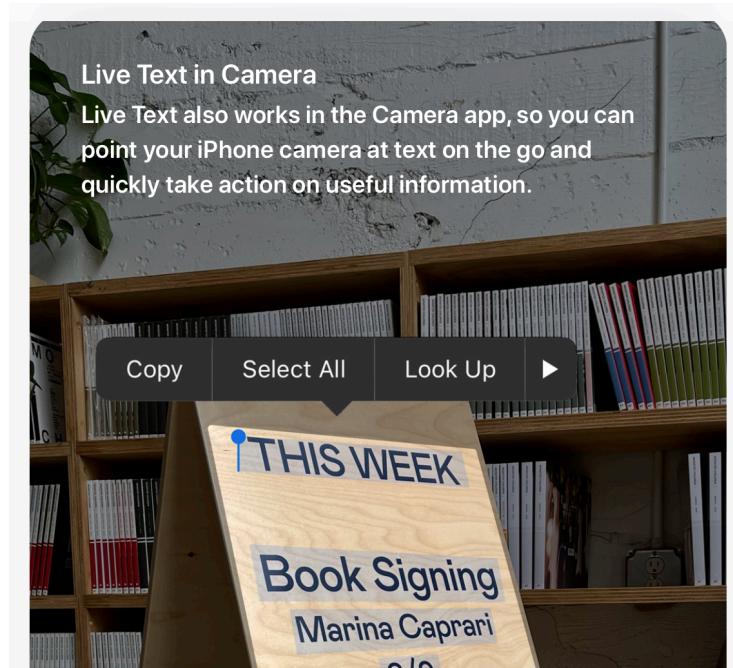


MADS

S2: Desarrollo de software

Los tweets de la semana



Los tweets de la semana

Eneko Alonso (@eneko)

Very impressive list of proposals in this Swift 5.5. release. Amazing, many of these are going to be game changers.

swift.org/blog/swift-5-5...

Swift Evolution proposals

A number of changes went through the Swift Evolution process for inclusion in Swift 5.5:

- SE-0291 Package Collections
- SE-0293 Extend Property Wrappers to Function and Closure Parameters
- SE-0295 Codable synthesis for enums with associated values
- SE-0296 Async/await
- SE-0297 Concurrency Interoperability with Objective-C
- SE-0298 Async/Await: Sequences
- SE-0299 Extending Static Member Lookup in Generic Contexts
- SE-0300 Continuations for interfacing async tasks with synchronous code
- SE-0304 Structured concurrency
- SE-0306 Actors
- SE-0307 Allow interchangeable use of CGFloat and Double types
- SE-0308 if for postfix member expressions
- SE-0310 Effectful Read-only Properties
- SE-0311 Task Local Values
- SE-0313 Improved control over actor isolation
- SE-0314 AsyncStream and AsyncThrowingStream
- SE-0316 Global actors
- SE-0317 async let bindings
- SE-0319 Conform Never to Identifiable

5:15 AM · Sep 21, 2021 · Twitter Web App

Swift Language (@SwiftLang)

Swift 5.5 is now officially released! Swift 5.5 is a massive release that includes newly-introduced language capabilities for concurrency, including `async/await`, `structured concurrency`, and `Actors`.

2:29 AM · Sep 21, 2021 · Twitter Web App

331 Retweets 37 Quote Tweets 1,201 Likes

Índice

1. Software
2. Metáforas
3. El desarrollo de software no es una ingeniería tradicional
4. El desarrollo de software es una actividad creativa

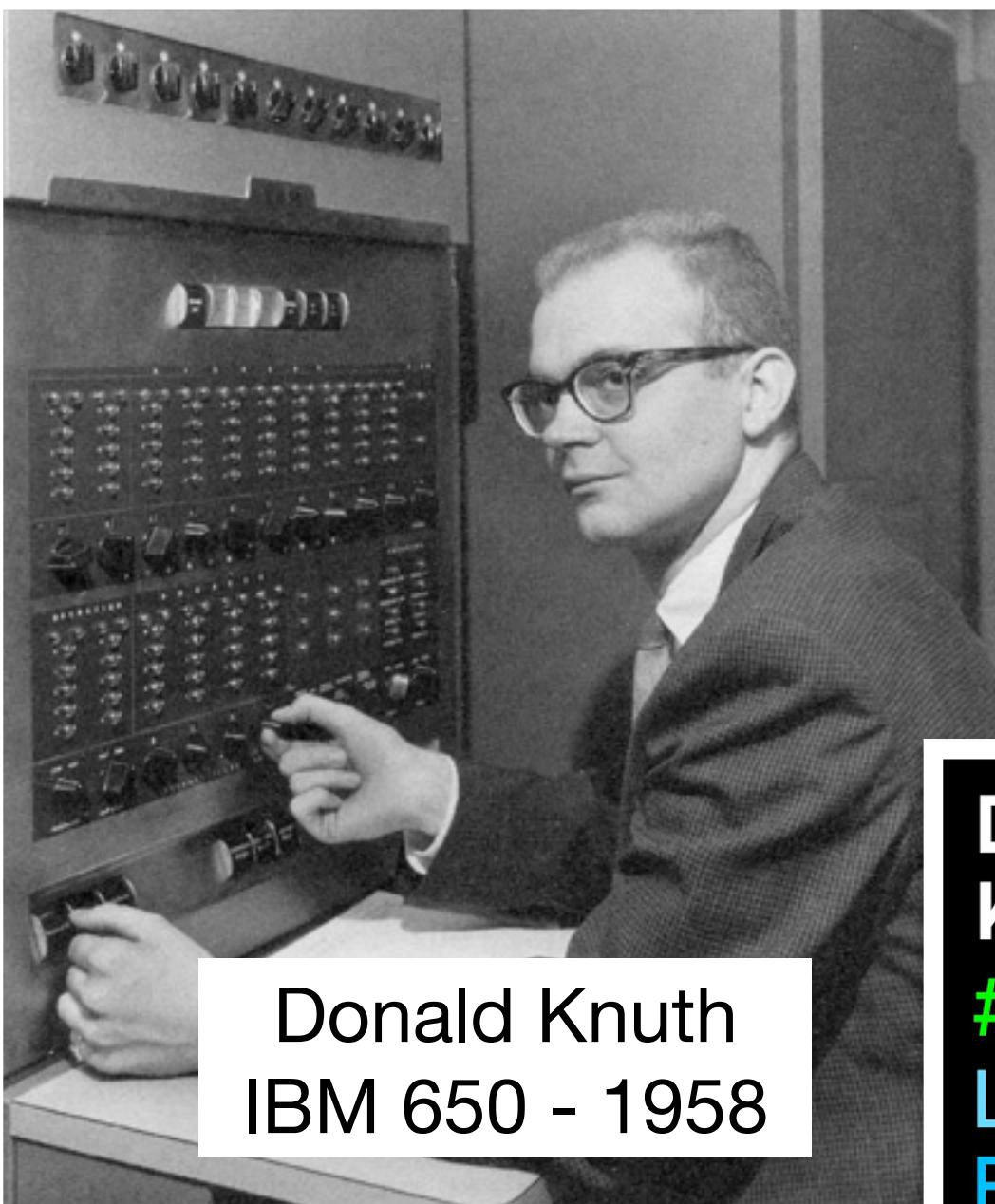
1. Software

BASIC PROGRAMMING

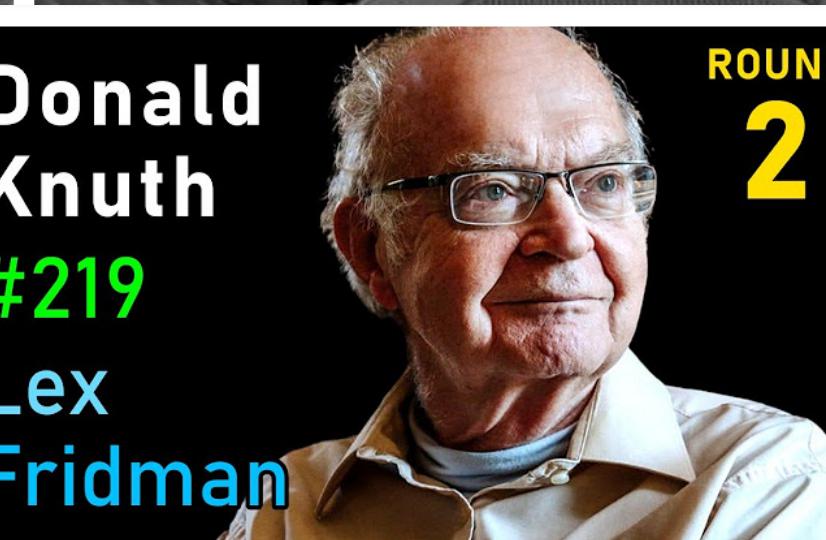


UNIVAC®
Data Automation System

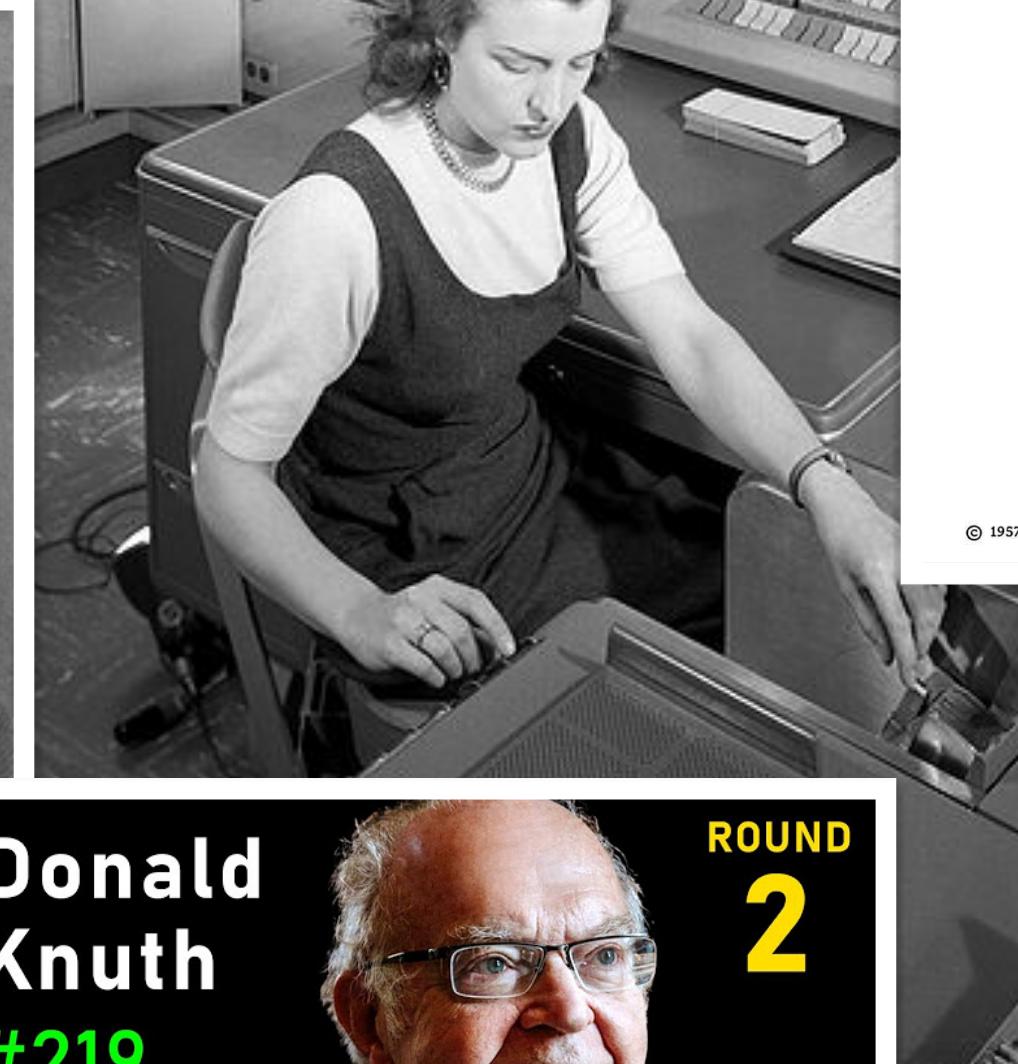
Programmer's Primer for
FORTRAN
Automatic Coding System
for the IBM 704



Donald Knuth
IBM 650 - 1958



Donald
Knuth
#219
Lex
Fridman

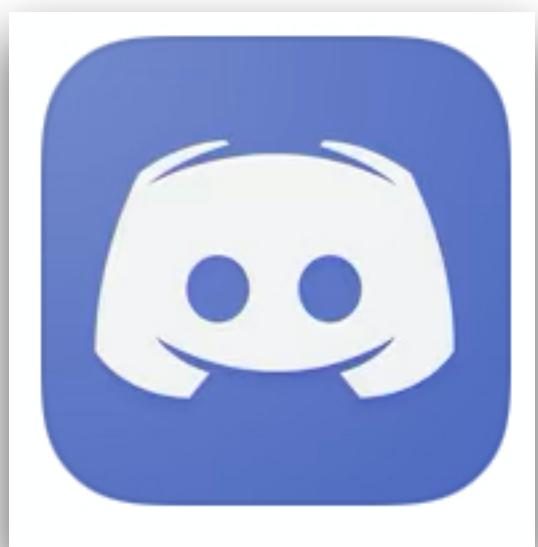


LISP I
PROGRAMMER'S MANUAL

March 1, 1960

COMPUTATION CENTER
and
RESEARCH LABORATORY OF ELECTRONICS
Massachusetts Institute of Technology
Cambridge, Massachusetts







- [CartoDB](#). Software español para representación visual de datos geográficos.
- [Guice](#). Framework de inyección de dependencias en Java.
- [swift-nio](#). Framework asíncrono de entrada-salida en Swift.
- [Spring Boot](#). Framework web en Java
- [Swift](#). Compilador y librería stándar de Swift. Escrito en C++ y Swift.

2. Metáforas

metáfora.

(Del lat. *metaphōra*, y este del gr. μεταφορά, translación).

1. f. *Ret.* Tropo que consiste en trasladar el sentido recto de las voces a otro figurado, en virtud de una comparación tácita; p. ej., *Las perlas del rocío. La primavera de la vida. Refrenar las pasiones.*

2. f. Aplicación de una palabra o de una expresión a un objeto o a un concepto, al cual no denota literalmente, con el fin de sugerir una comparación (con otro objeto o concepto) y facilitar su comprensión; p. ej., *el átomo es un sistema solar en miniatura.*

~ continuada.

1. f. *Ret.* Alegoría en que unas palabras se toman en sentido recto y otras en sentido figurado.

Diccionario de la lengua española (2001)

Real Academia Española © Todos los derechos reservados

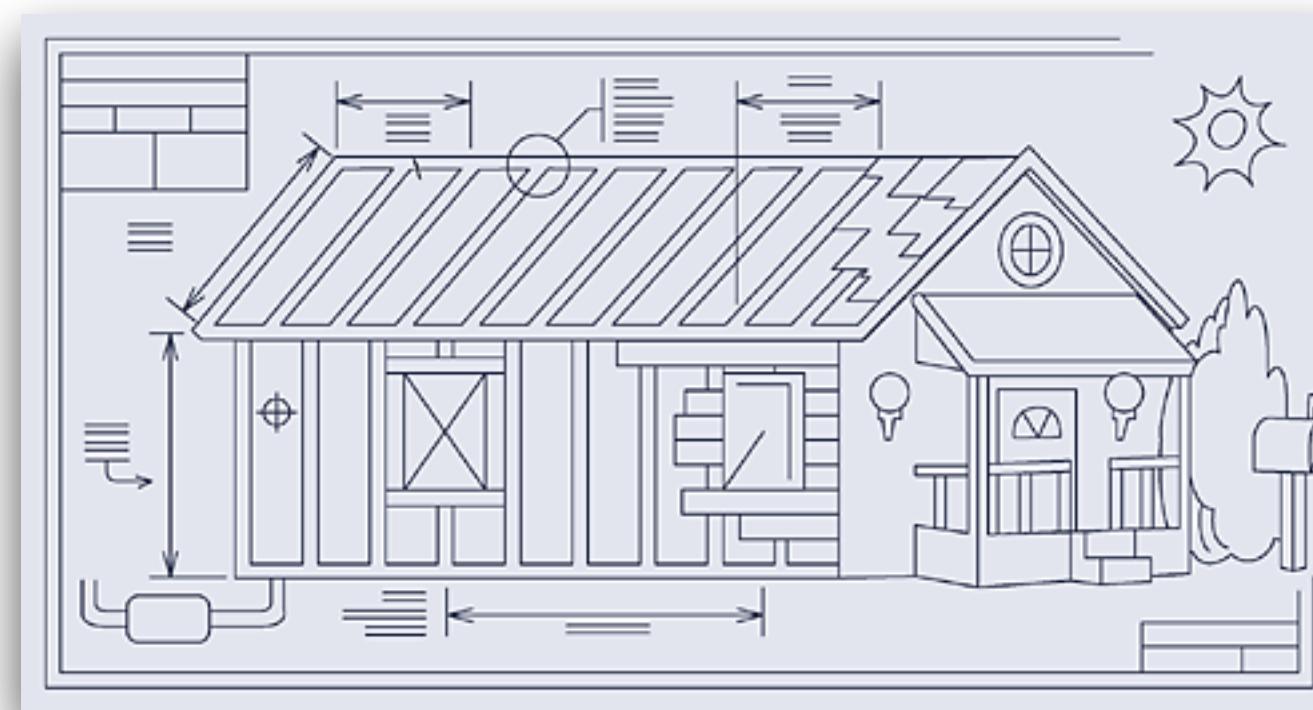
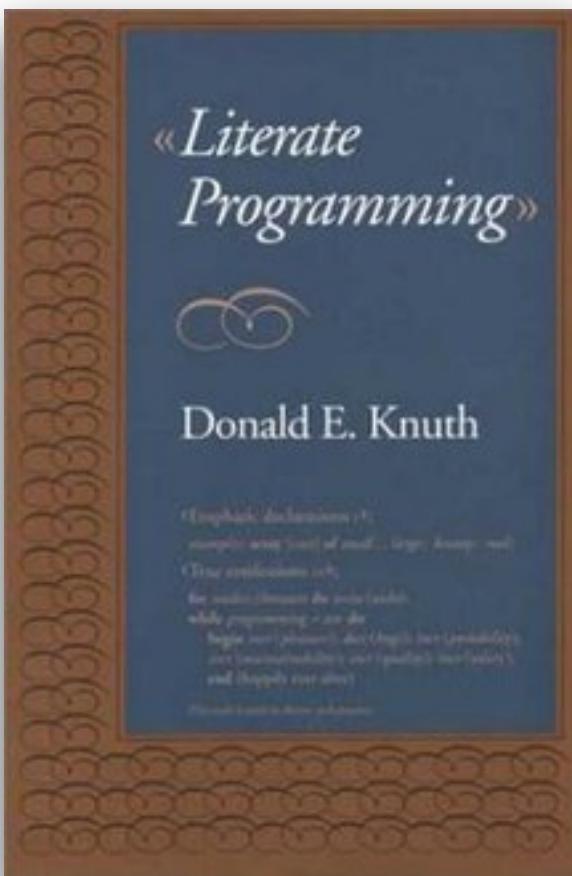
"Hemos ganado este combate"
"Nos lo jugamos todo en esta campaña"
"Es una oportunidad de vida o muerte"

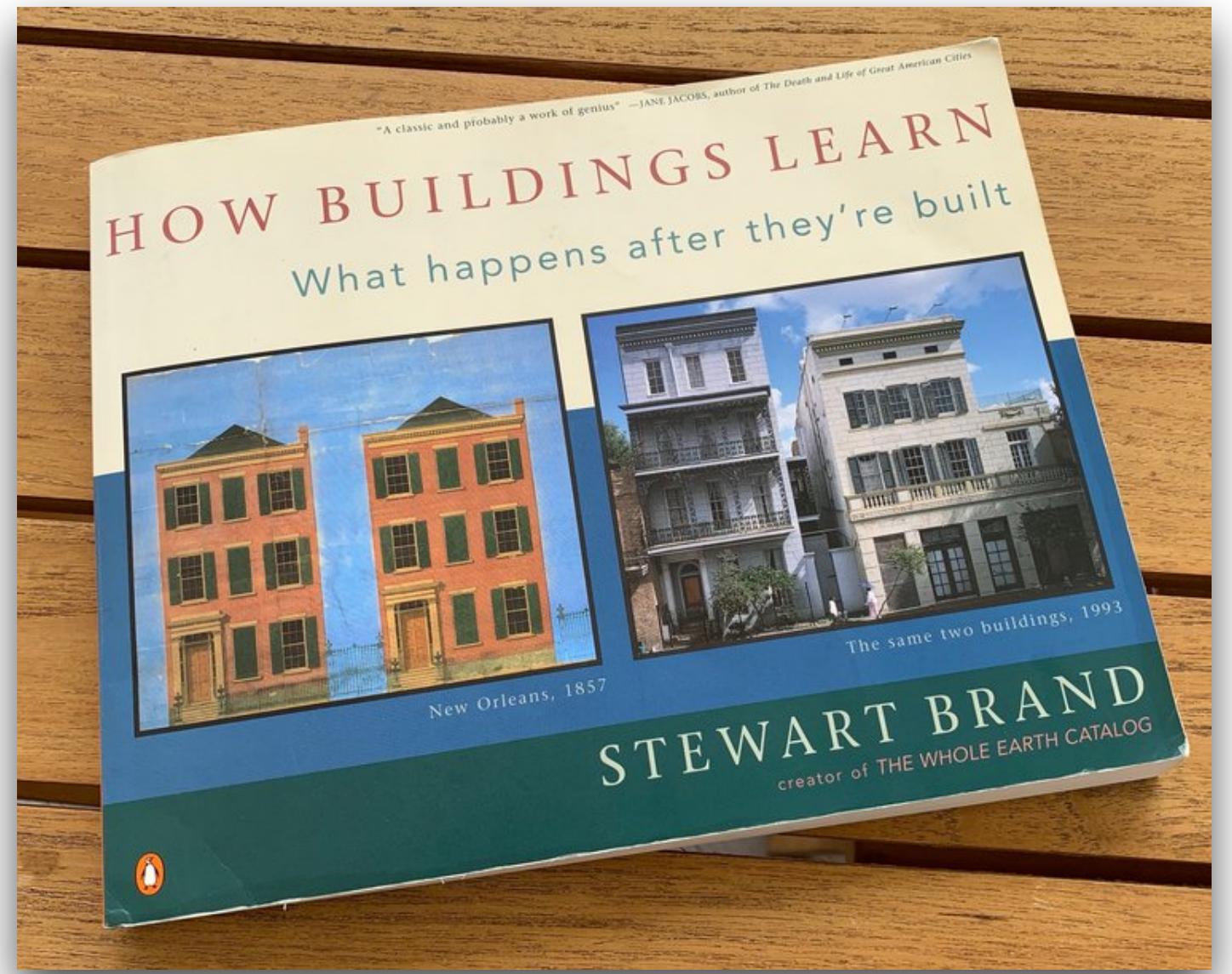
Vs.

"Hemos coreografiado perfectamente la puesta en marcha del producto"
"Todos debemos participar en este viaje"
"Nuestro ecosistema premia la lealtad de nuestros clientes"

Metáforas del desarrollo de software

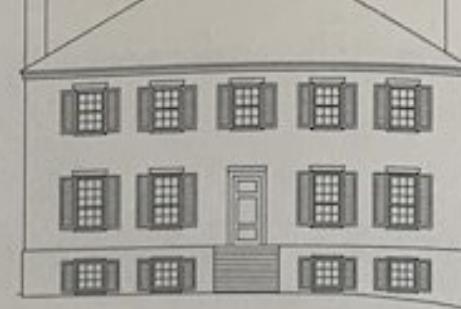
- El desarrollo de software es como ...



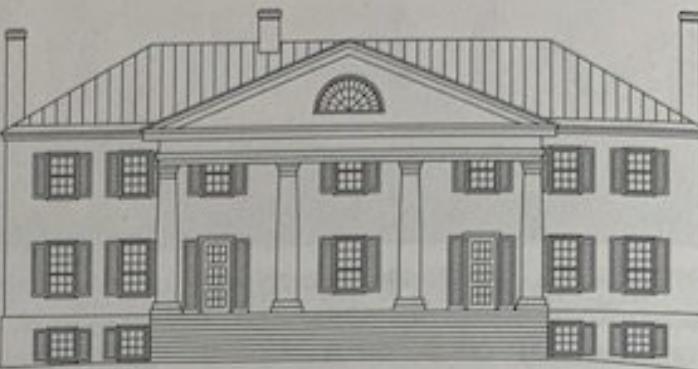


George Washington's
MOUNT VERNON

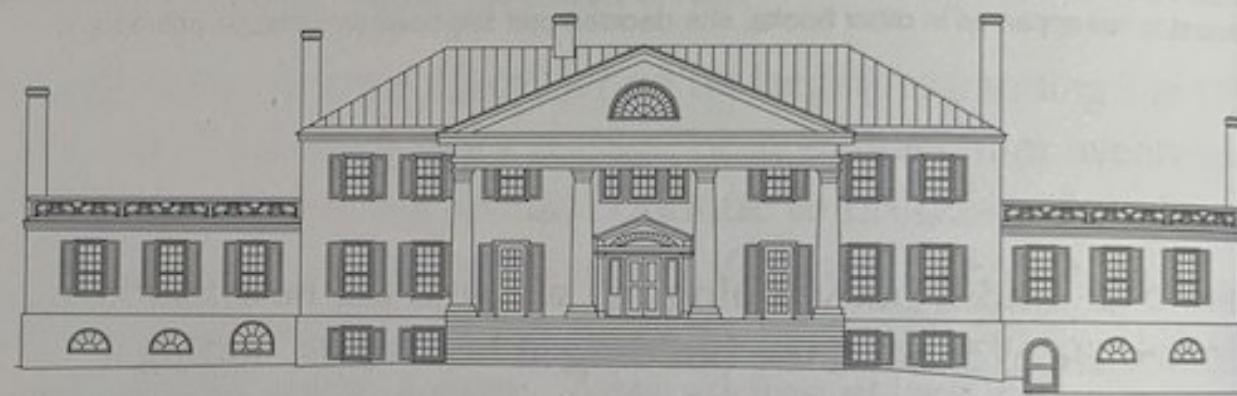
James Madison's
MONTPELIER



1765



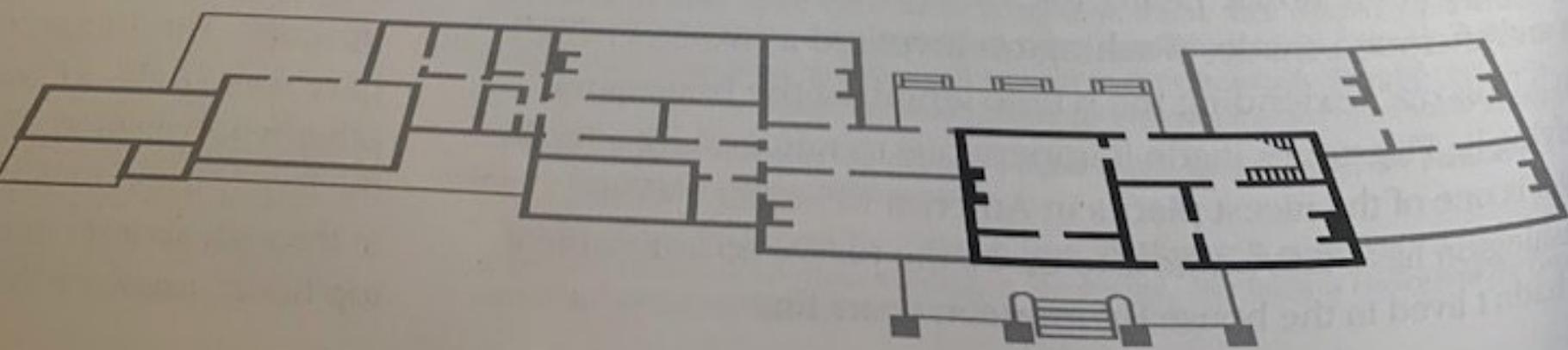
1798



1812

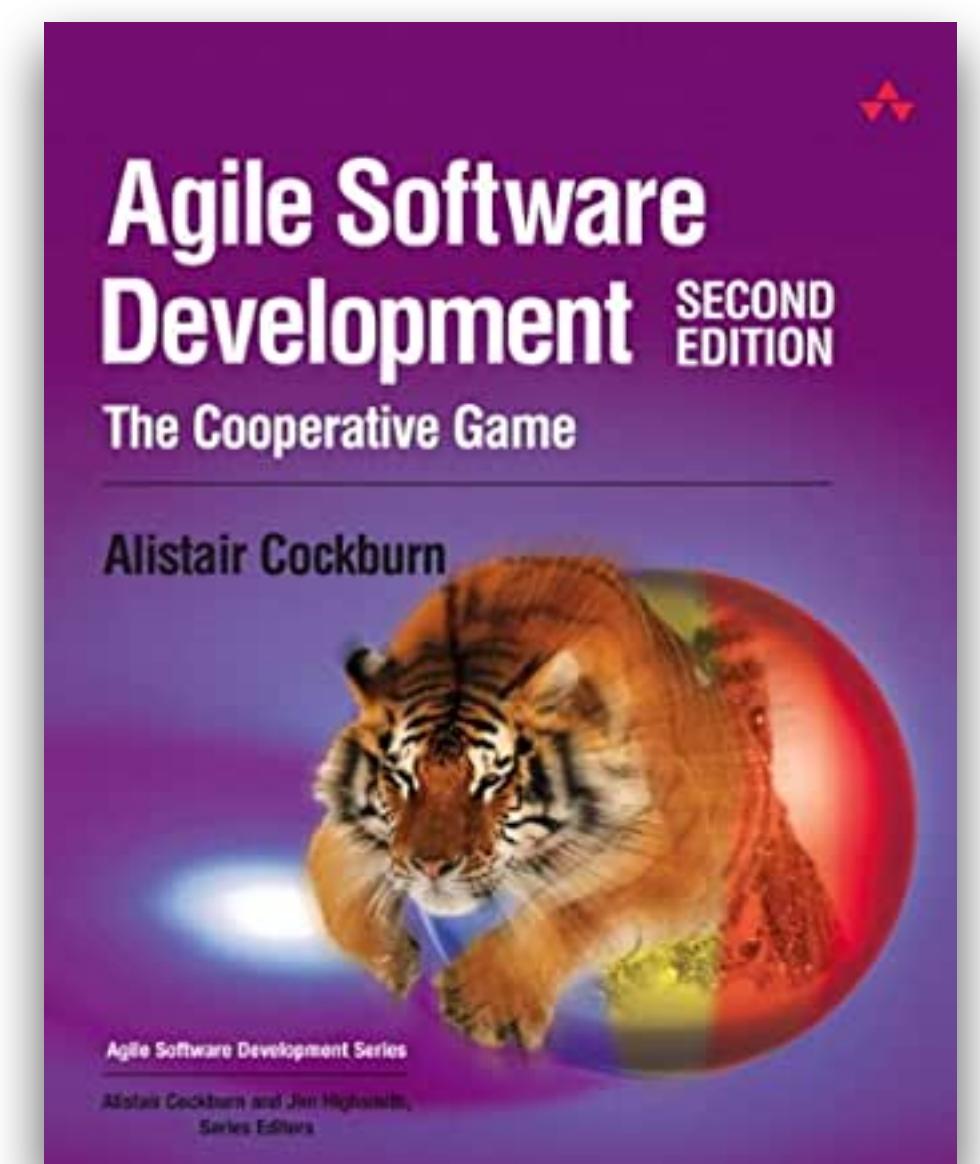
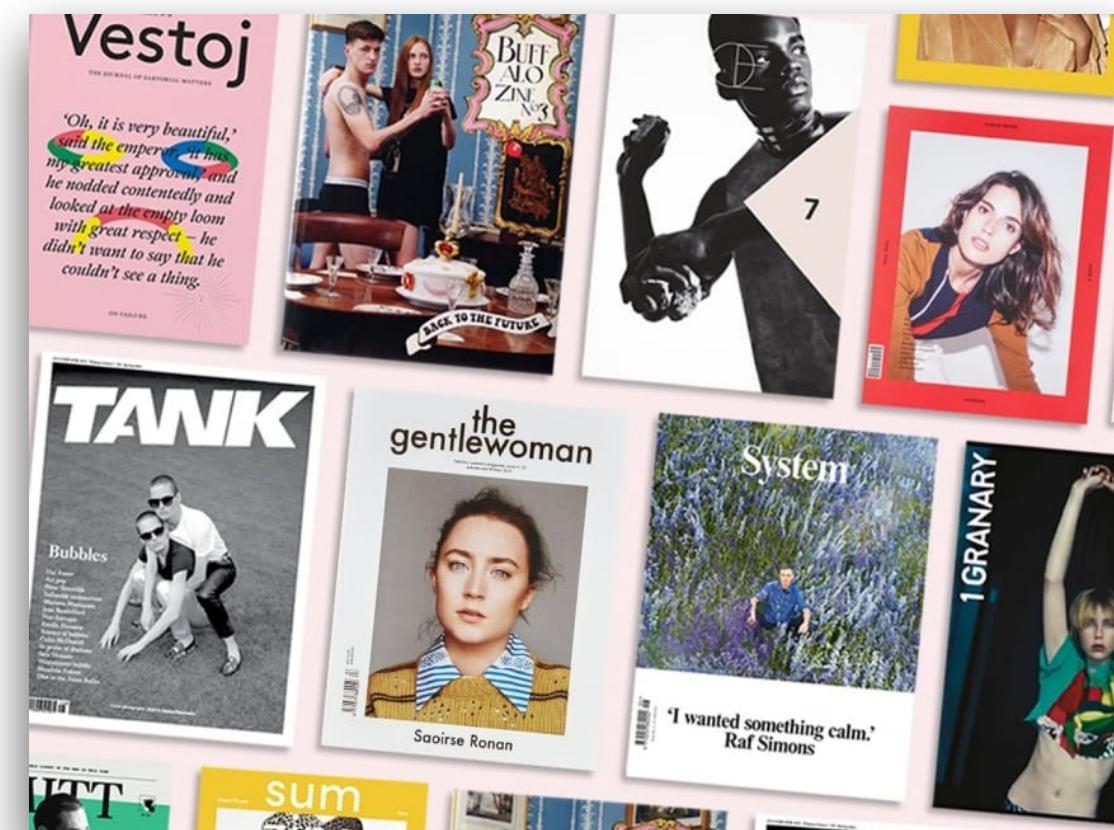


1857



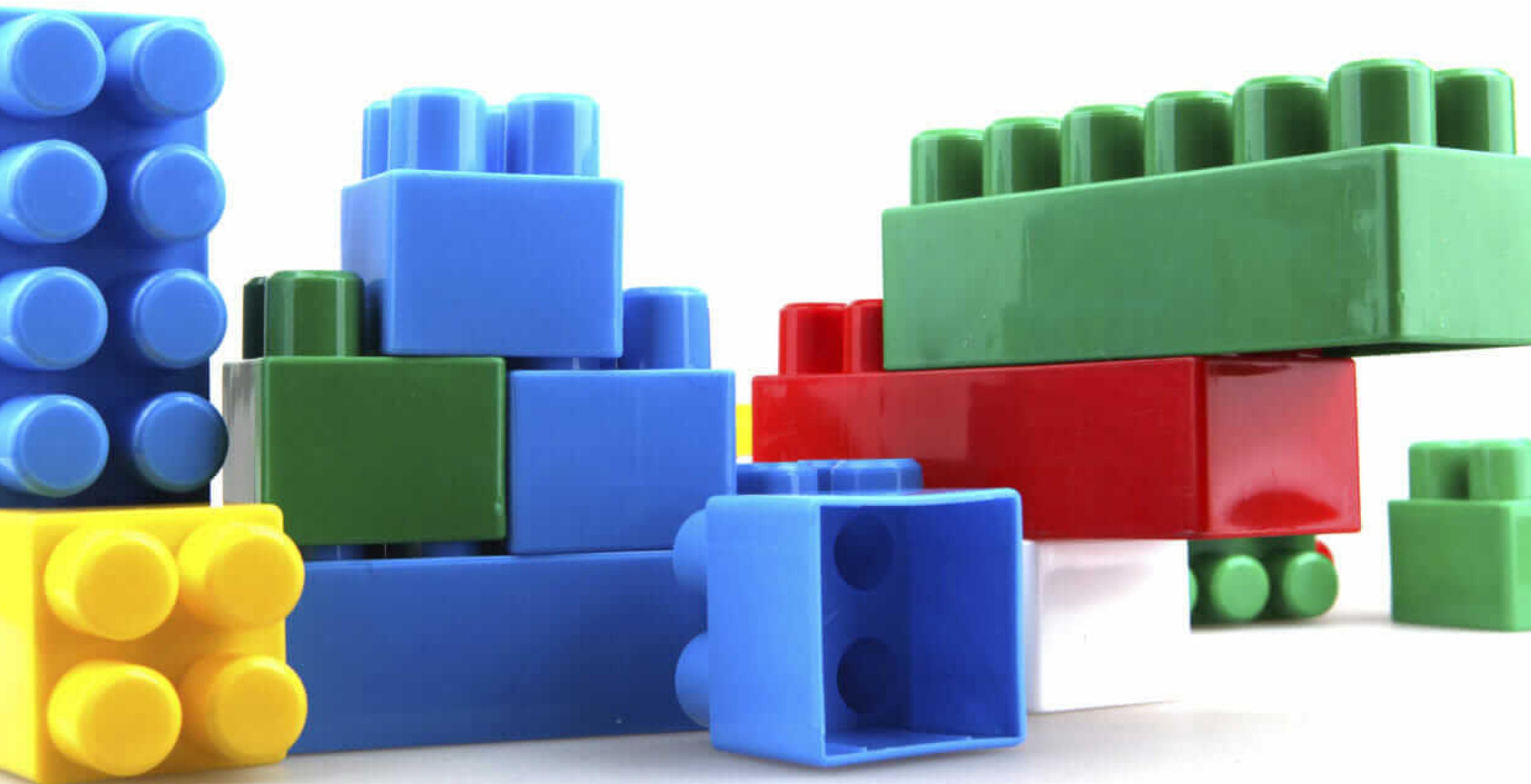
15

- El desarrollo de software es como ...



El software es único









Extreme Programming *Explained*

EMBRACE CHANGE

KENT BECK

WITH

CYNTHIA ANDRES

Foreword by Erich Gamma

Second Edition

Continued Learning: The Beauty of Maintenance

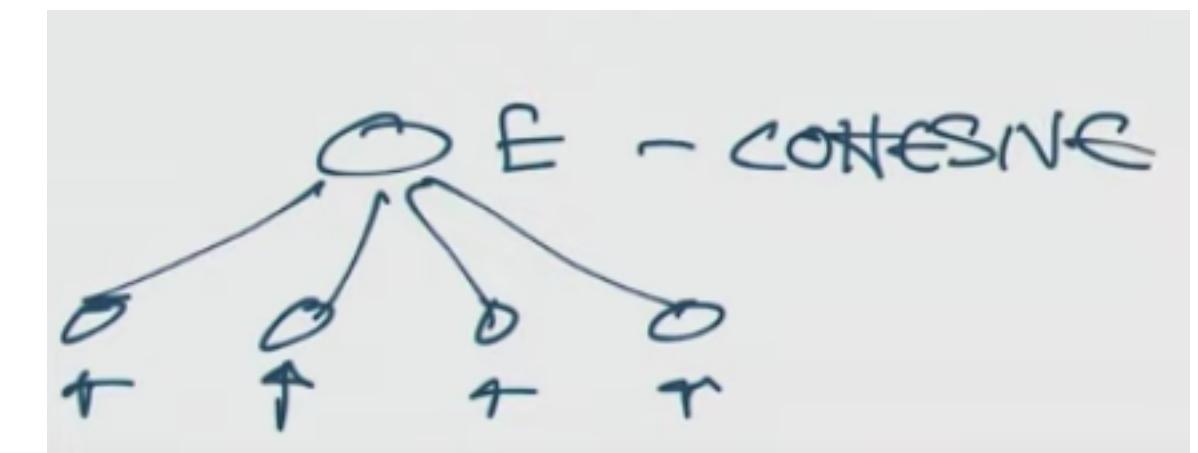
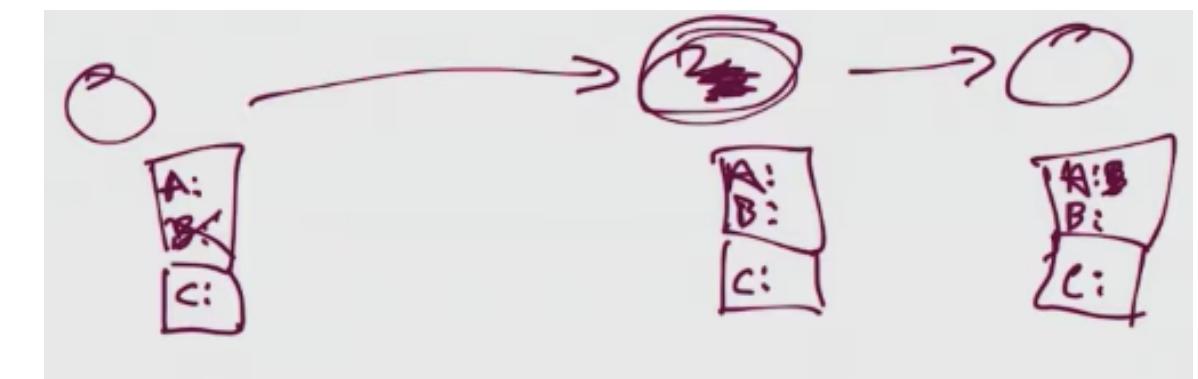
Kent Beck - DDD Europe 2020



ELEMENT

$$\begin{array}{c} A \xrightarrow{\quad} B \\ \cong \qquad \cong \end{array}$$

COUPLED(A, B, A) :: $\Delta A \rightarrow \Delta B$



$\text{COST} \approx \text{COST}_{\text{CHANGE}} \approx \text{COST}_{\text{BIG CHANGE}}$

3. El desarrollo de software no es una ingeniería tradicional

The New Methodology

In the past few years there's been a blossoming of a new style of software methodology - referred to as agile methods. Alternatively characterized as an antidote to bureaucracy or a license to hack they've stirred up interest all over the software landscape. In this essay I explore the reasons for agile methods, focusing not so much on their weight but on their adaptive nature and their people-first orientation.

13 December 2005



Martin Fowler

AGILE

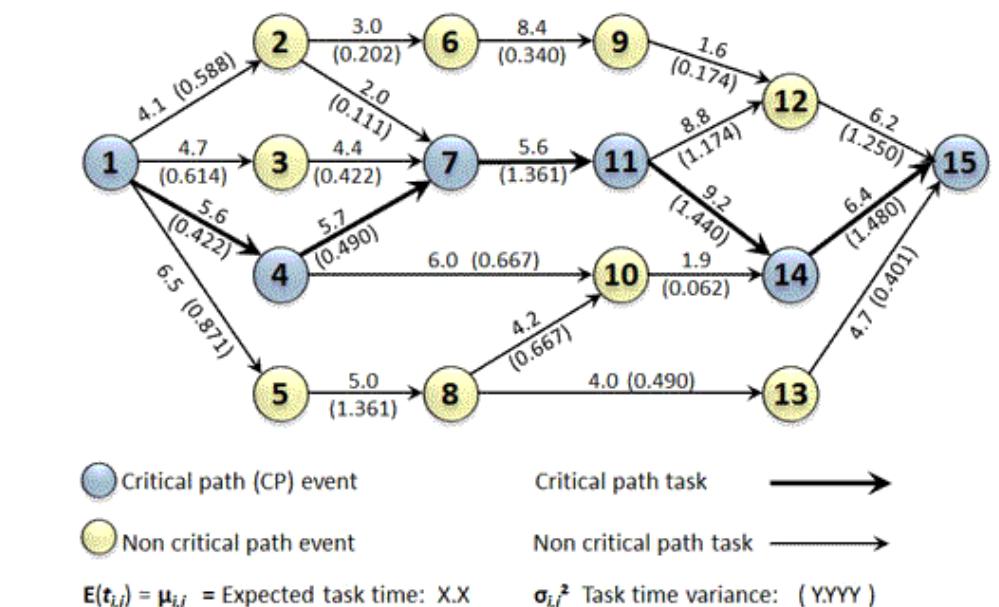
PROCESS THEORY

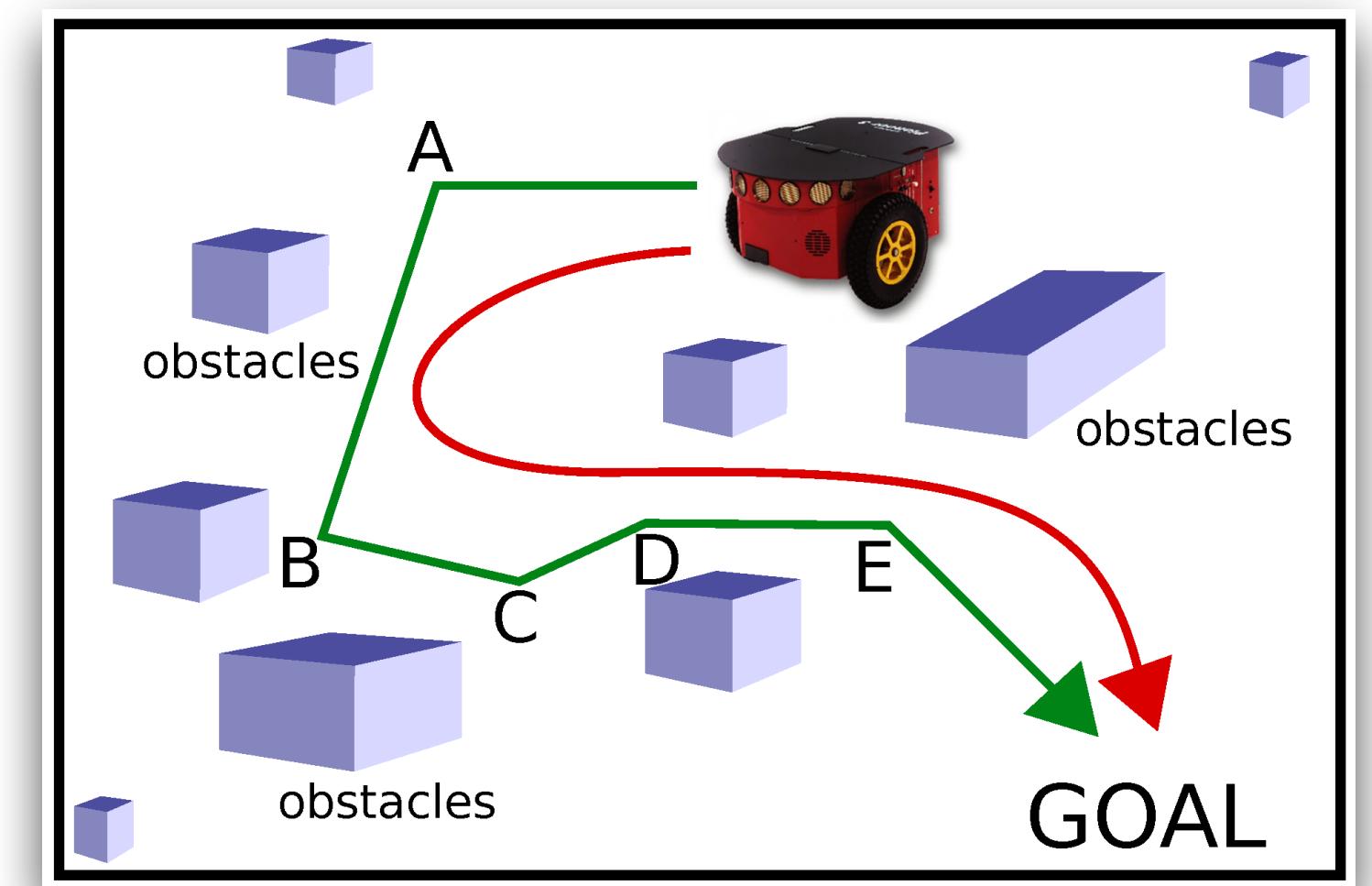
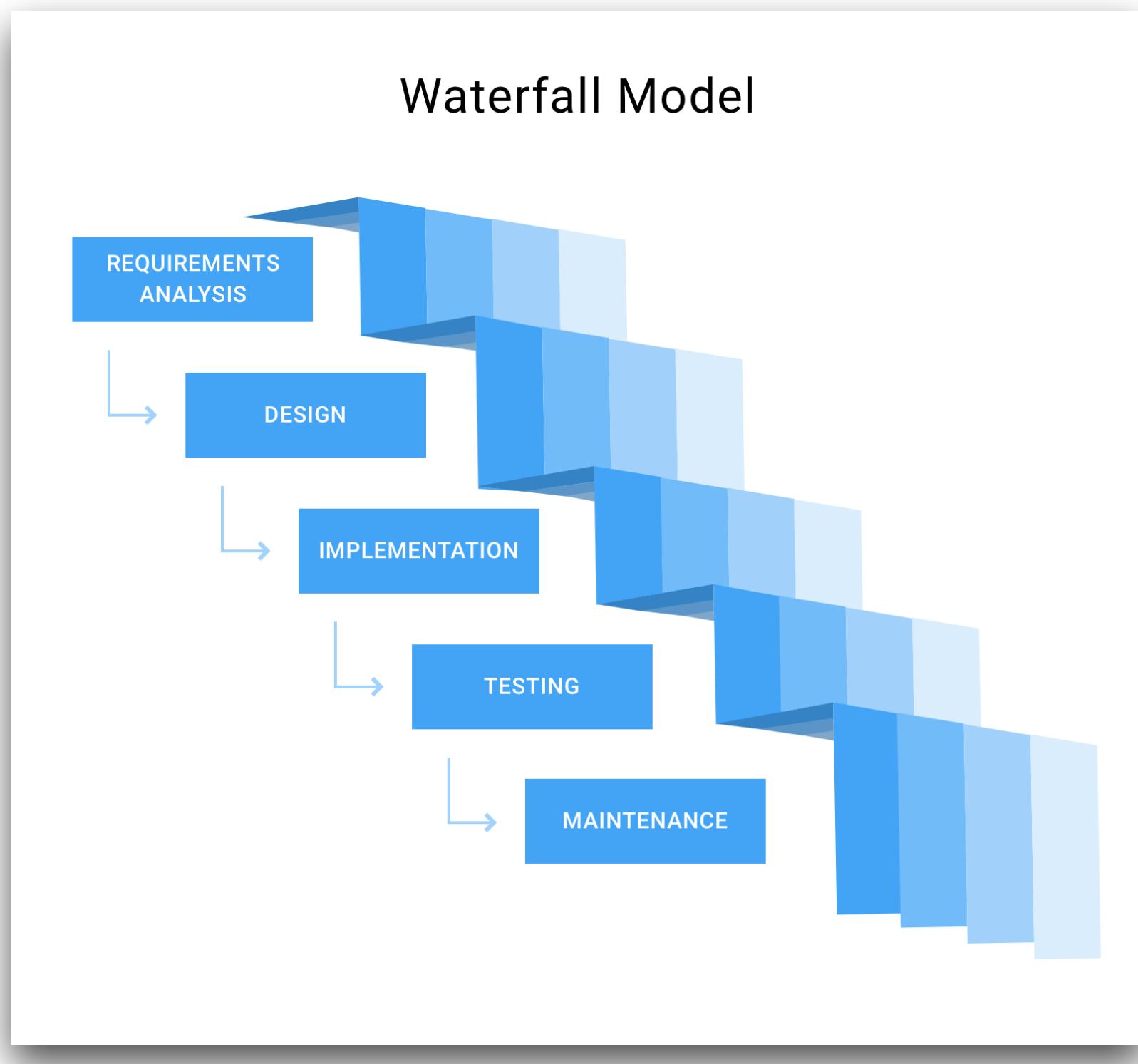
CONTENTS

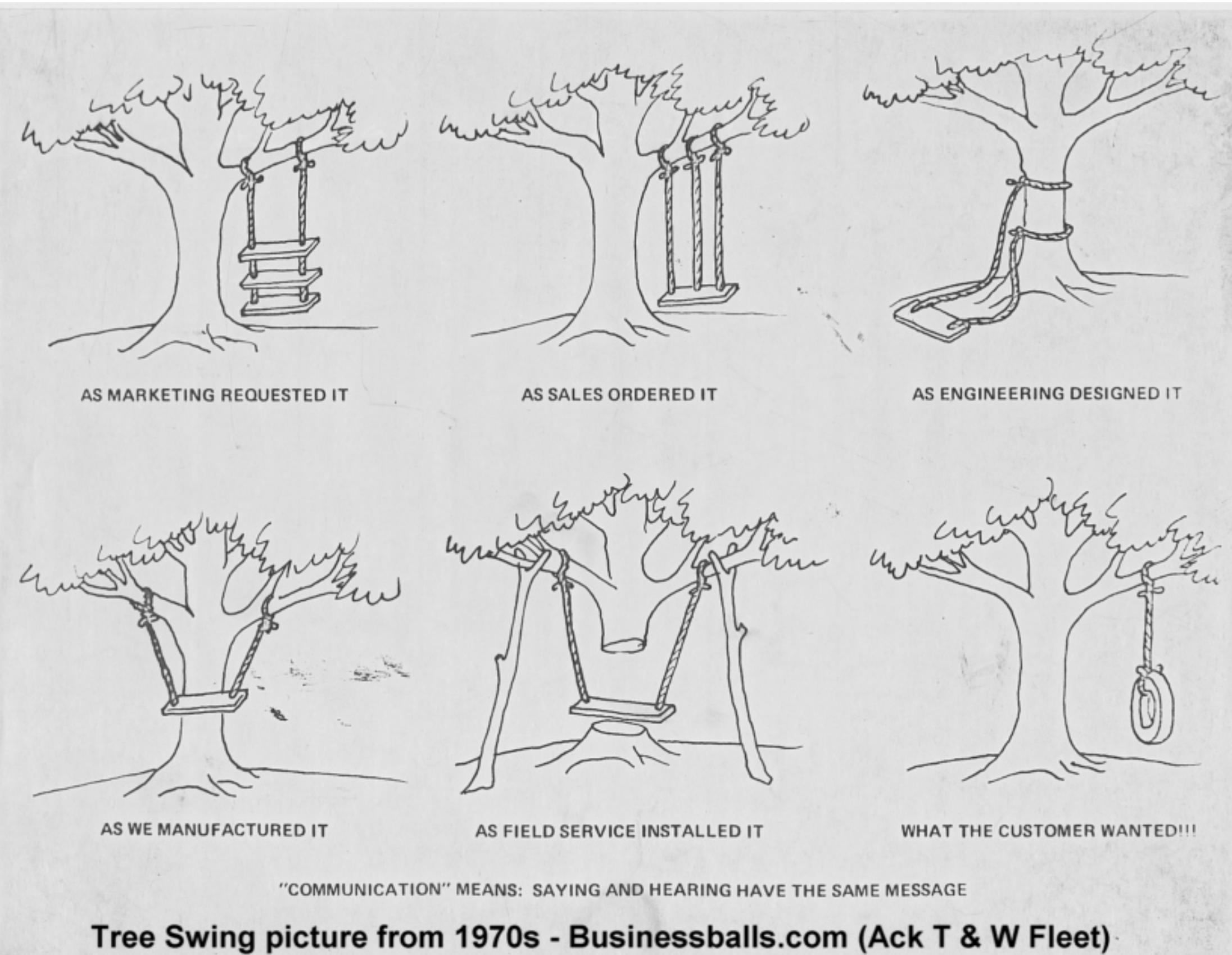
- [From Nothing, to Monumental, to Agile](#)
- [Predictive versus Adaptive](#)
- [Separation of Design and Construction](#)
- [The Unpredictability of Requirements](#)
- [Is Predictability Impossible?](#)
- [Controlling an Unpredictable Process - Iterations](#)
- [The Adaptive Customer](#)
- [Putting People First](#)
- [Plug-Compatible Programming Units](#)
- [Programmers are Responsible Professionals](#)
- [Managing a People Oriented Process](#)



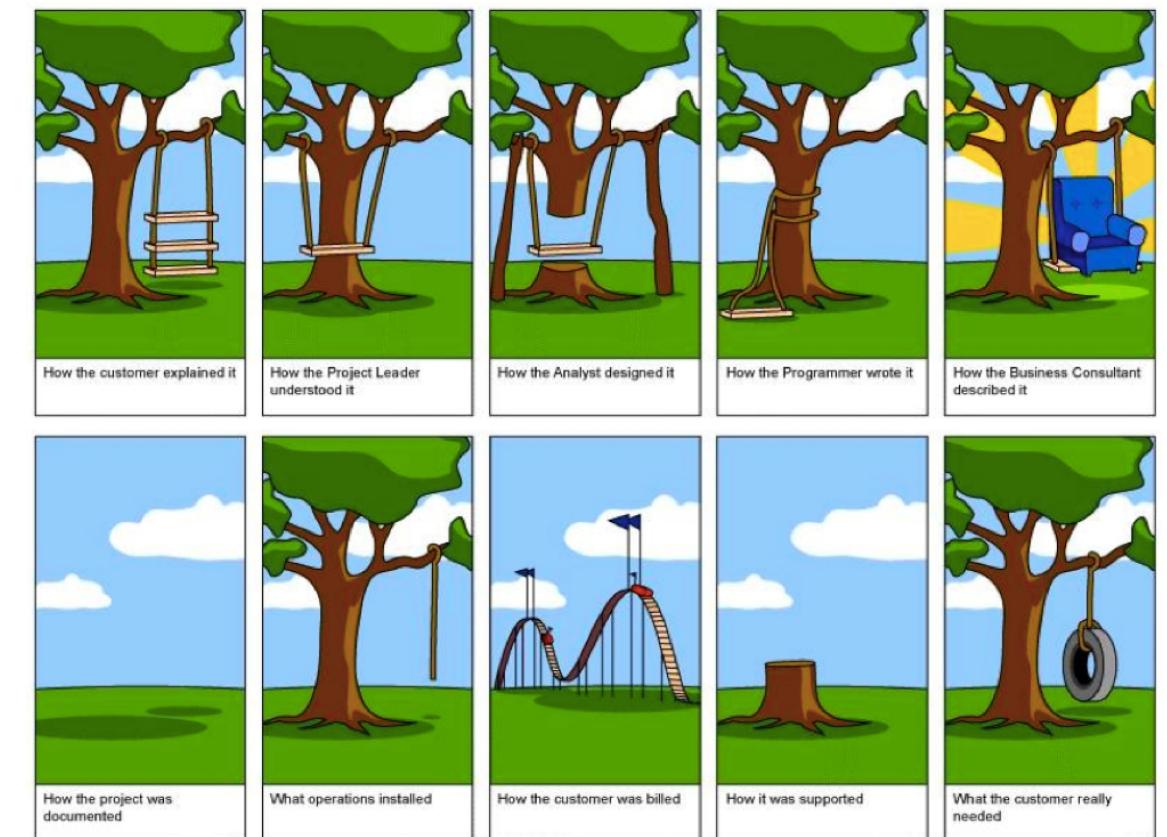
#	Resource Name	Overallocated, h	Allocated, h	2010			2011			2011	
				Sep	Oct	Nov	Dec	Jan	Feb	Mar	
1	Andrew Anderson	0.0	160.0								14 Sep 2010 - 16 Mar 2011
2	Barbara Taylor	4.0	168.0								
3	Charles Lewis	0.0	396.0								
4	David Harris	0.0	72.0								
5	Donna Hall	0.0	324.0								
6	Helen Clark	48.0	704.0								
7	James Smith	38.9	236.6								
8	John Brown	334.8	924.6								
9	Joseph Allen	120.0	608.0								
10	Karen Martin	457.6	781.2								
11	Laura Rodriguez	0.0	88.0								
12	Linda Davis	86.9	764.6								
13	Mark Robinson	120.0	608.0								
14	Mary Williams	554.1	915.5								
15	Michael Smith	64.0	152.0								
16	Nancy Garcia	0.0	96.0								
17	Patricia Jones	62.9	212.5								
18	Paul King	0.0	296.0								

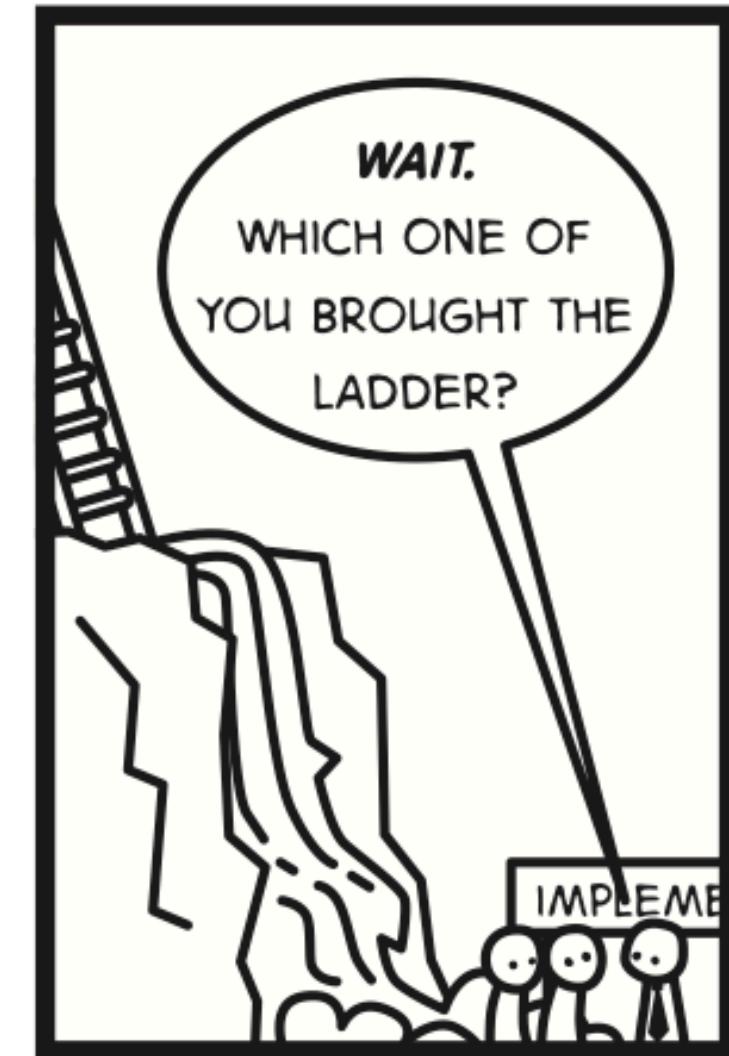
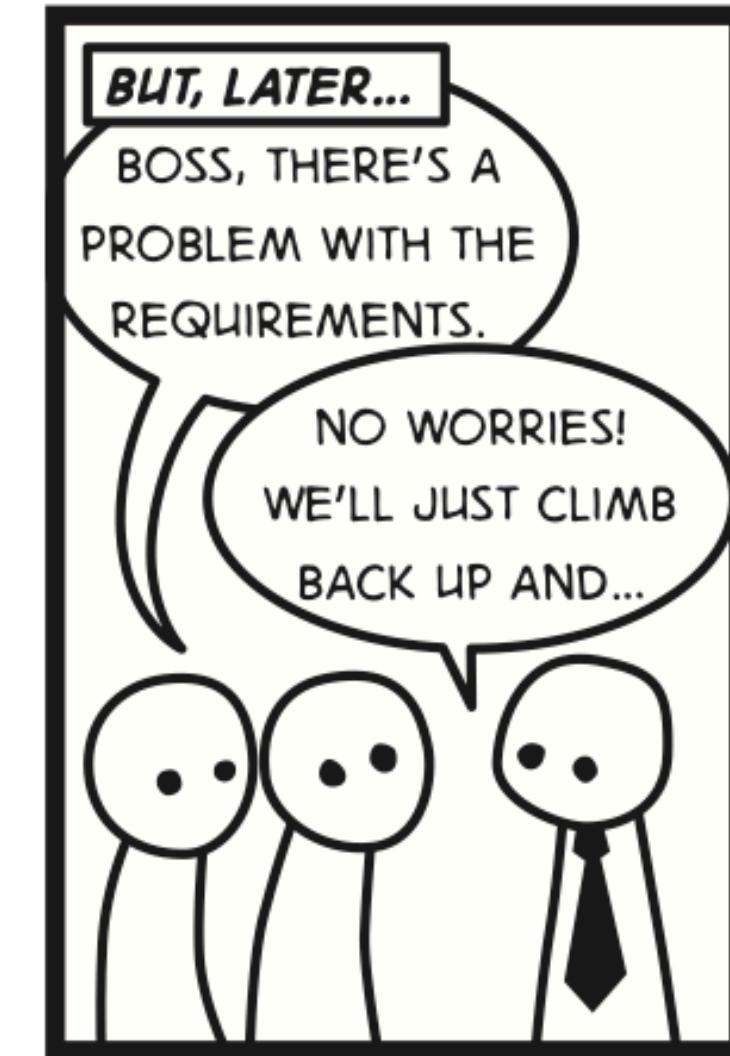
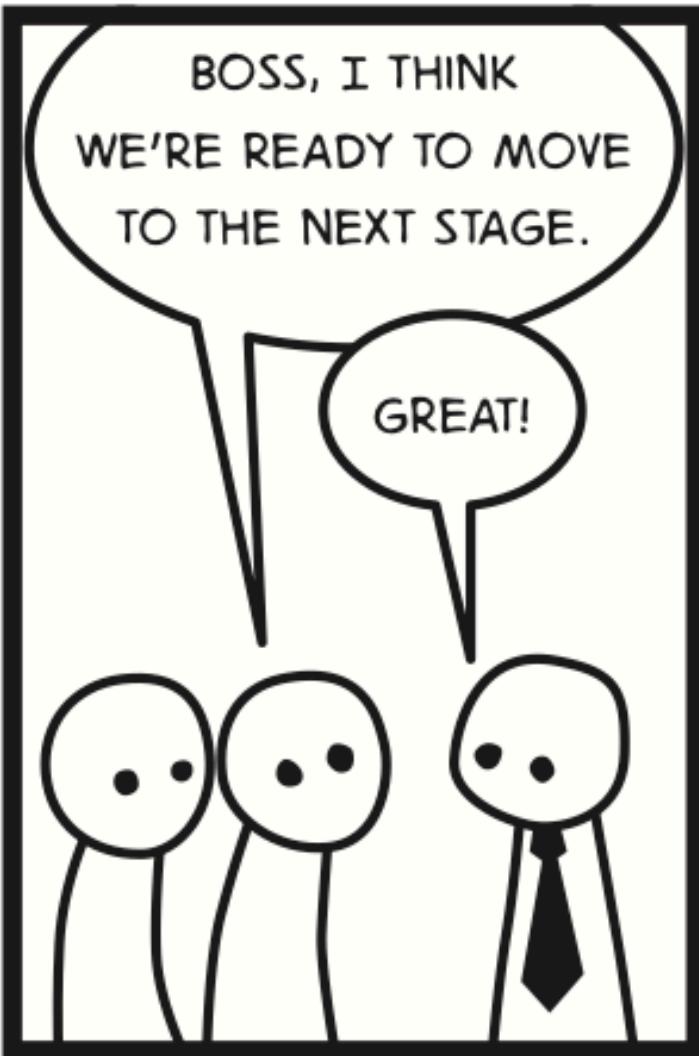






Tree Swing picture from 1970s - Businessballs.com (Ack T & W Fleet)

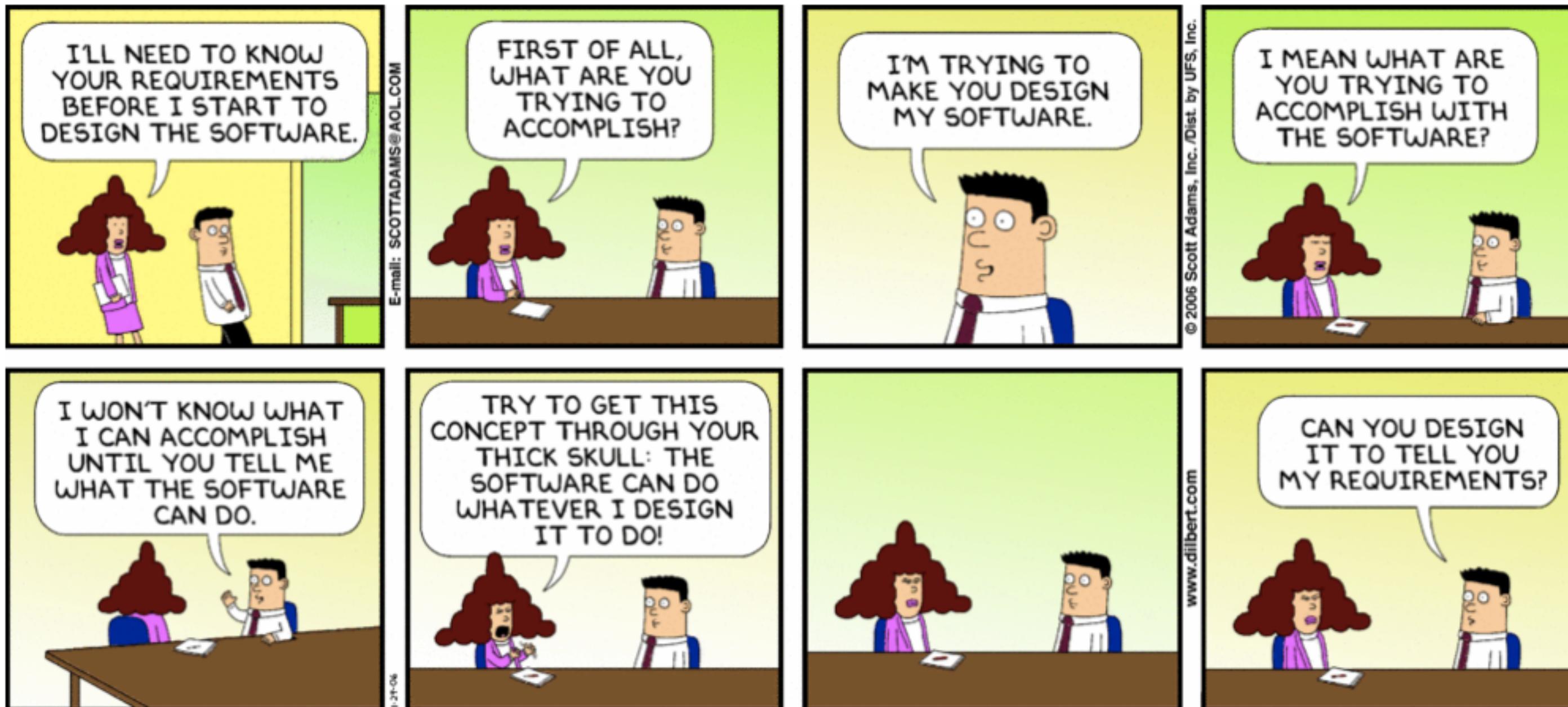




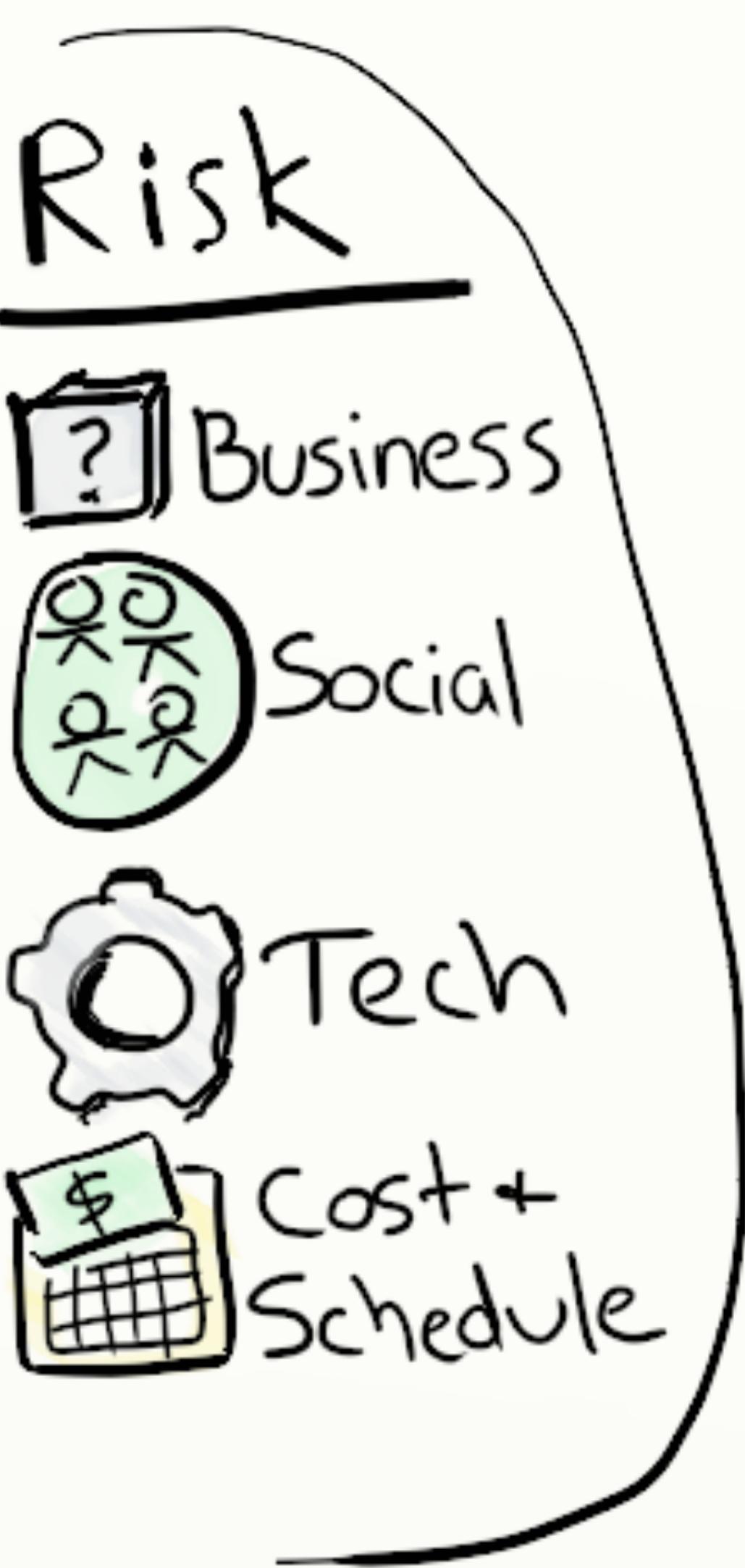
No es una ingeniería tradicional

- En el software la parte de construcción es muy barata.
- En el software todo el esfuerzo es diseño y requiere, por tanto, de personas creativas y con talento.
- Los procesos creativos no se pueden planificar fácilmente, por lo que la predictividad es un objetivo imposible.

¿Los requisitos son predecibles?

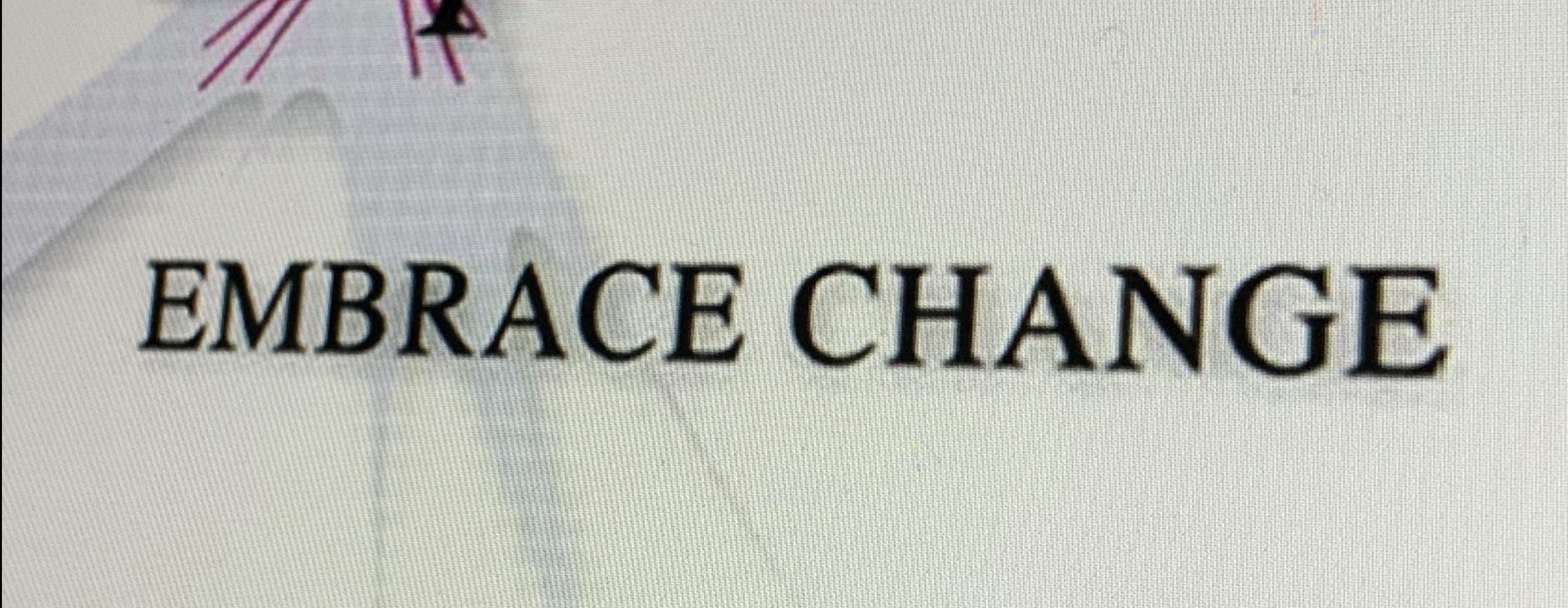


- **Lo que nos gustaría**
 - Los clientes saben lo que quieren
 - El equipo sabe cómo construirlo
 - Nada cambiará en el camino
 - Tenemos mucho tiempo y dinero para hacerlo
- **La realidad**
 - Los clientes descubren lo que necesitan
 - Los desarrolladores descubren cómo hacerlo
 - Muchas cosas cambian en el camino
 - Siempre hay más cosas qué hacer que tiempo y dinero disponible



NEW LAWS

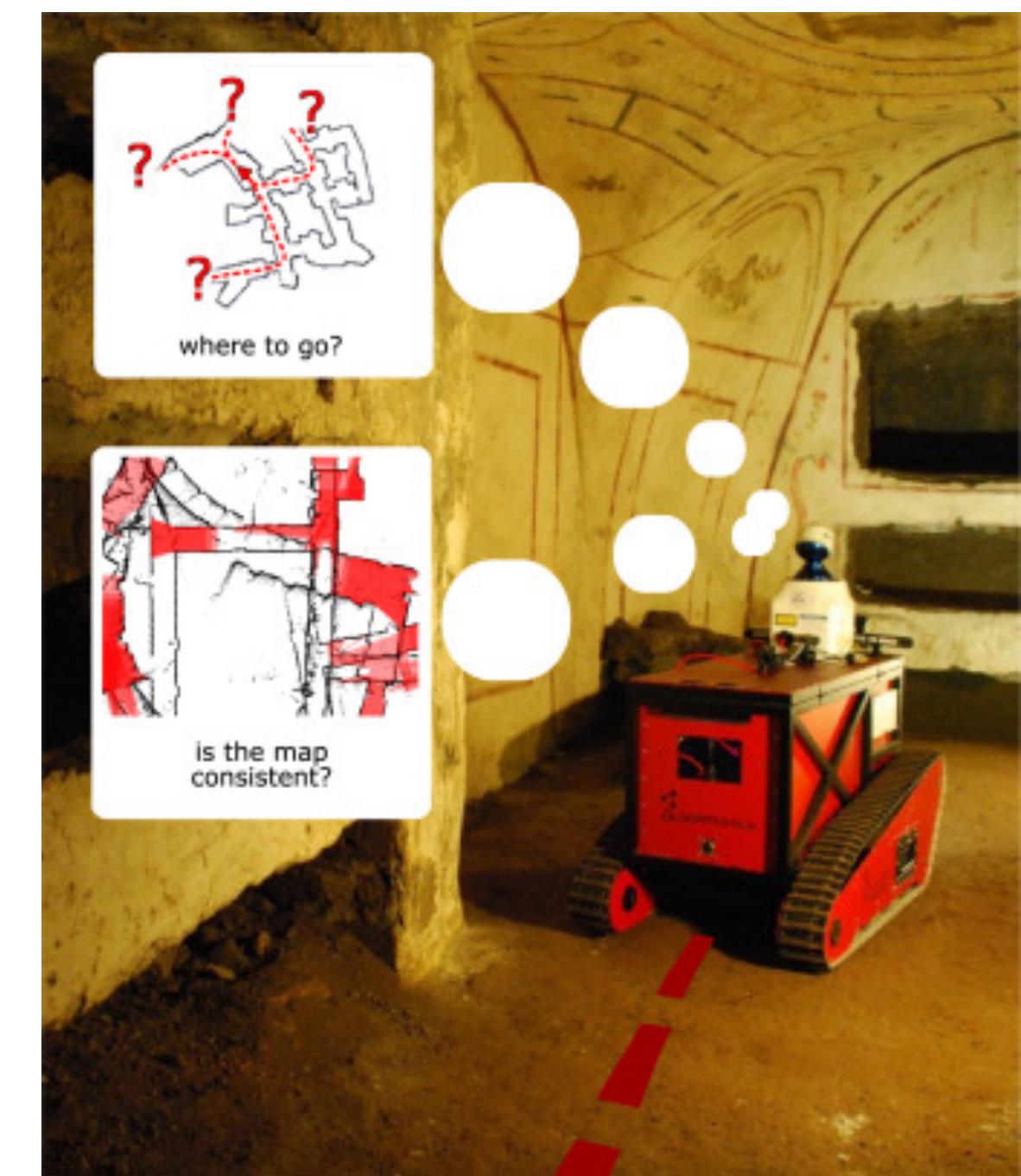


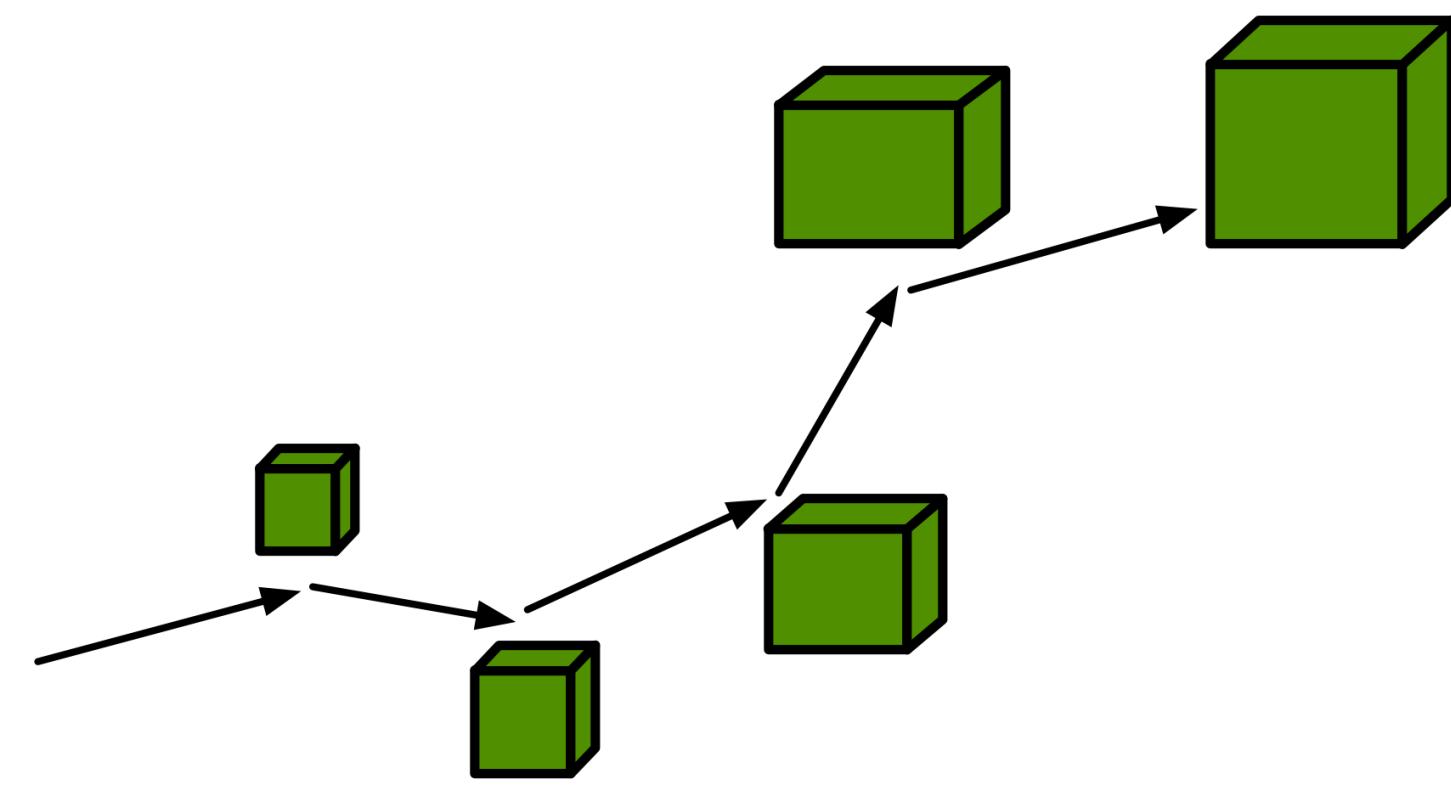
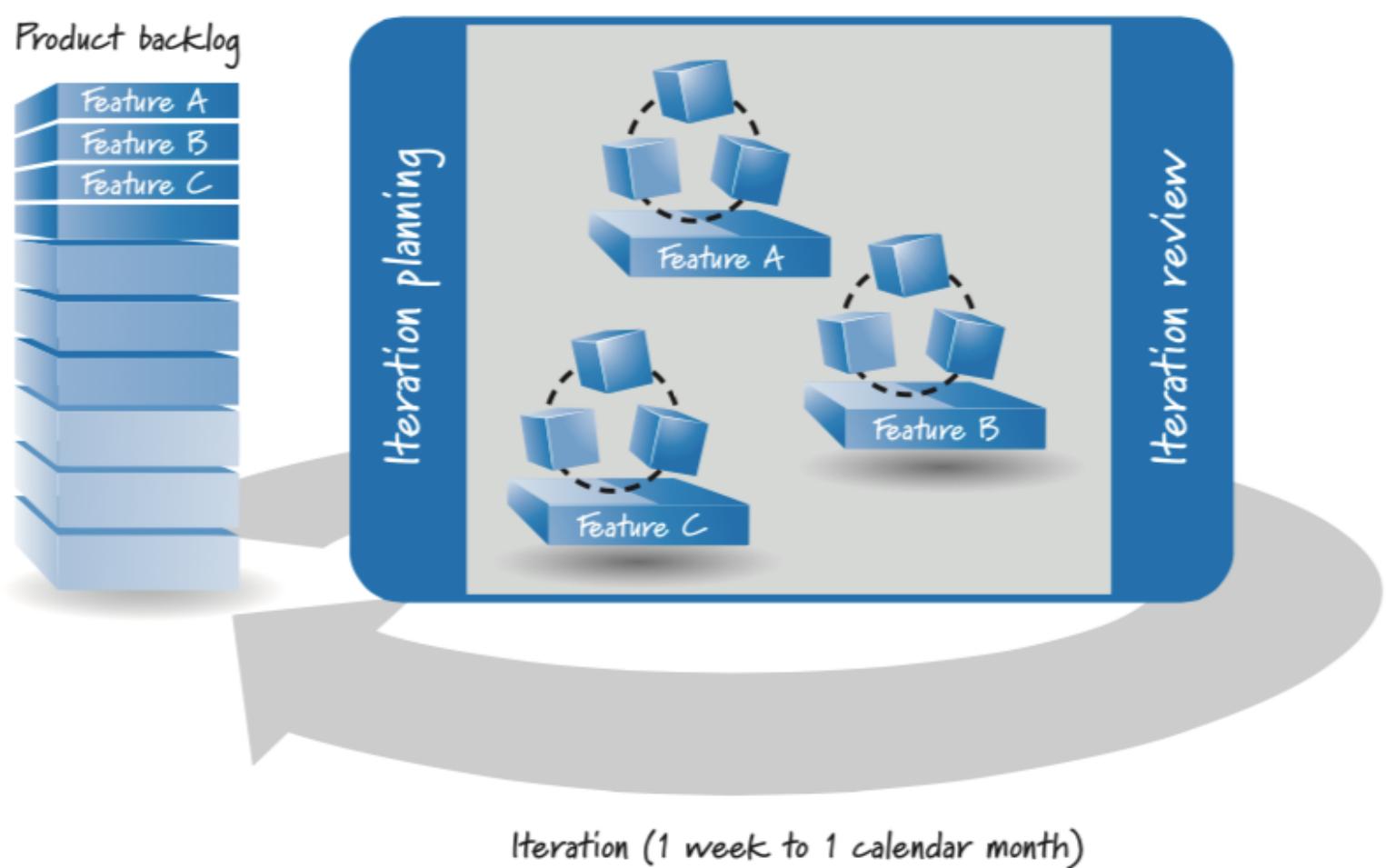


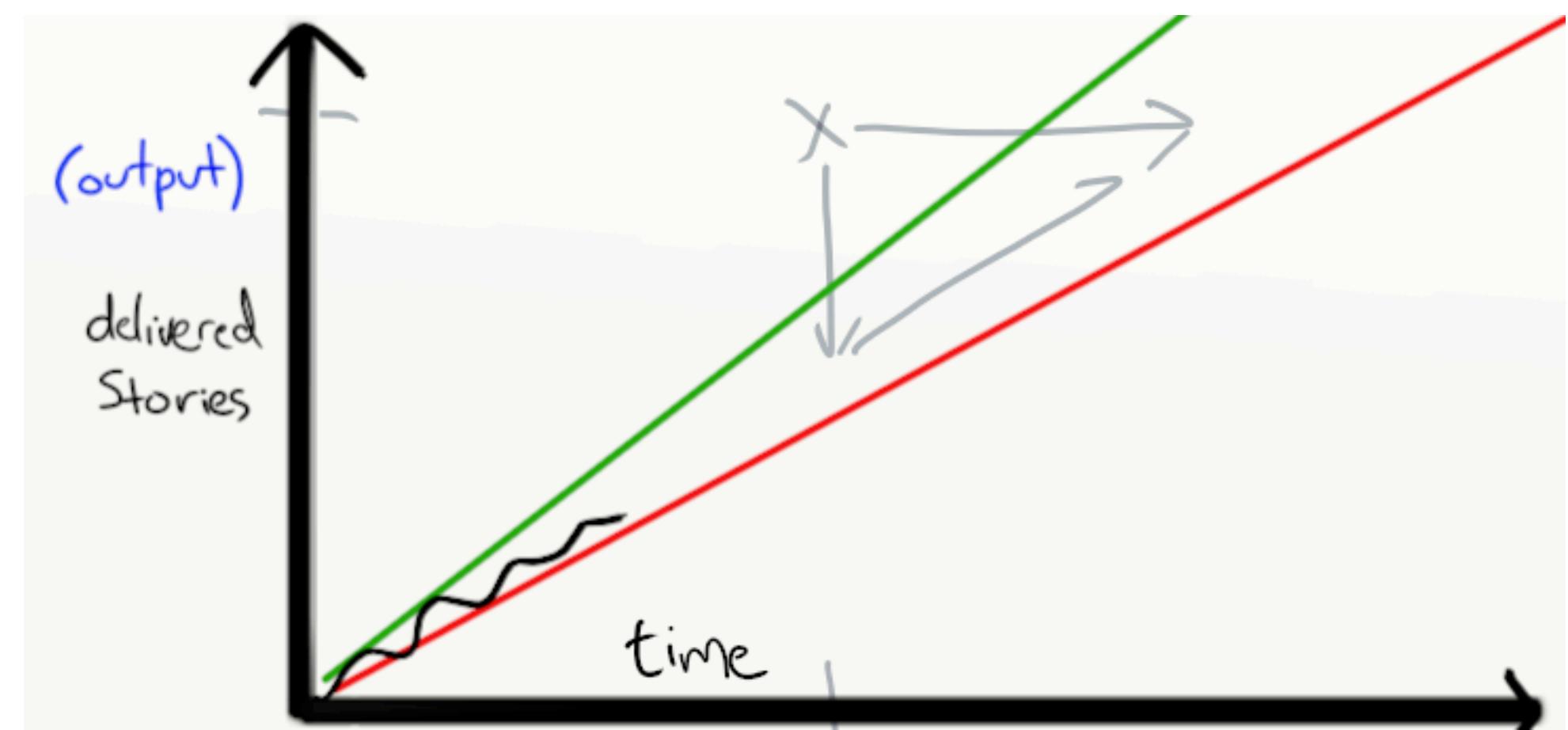
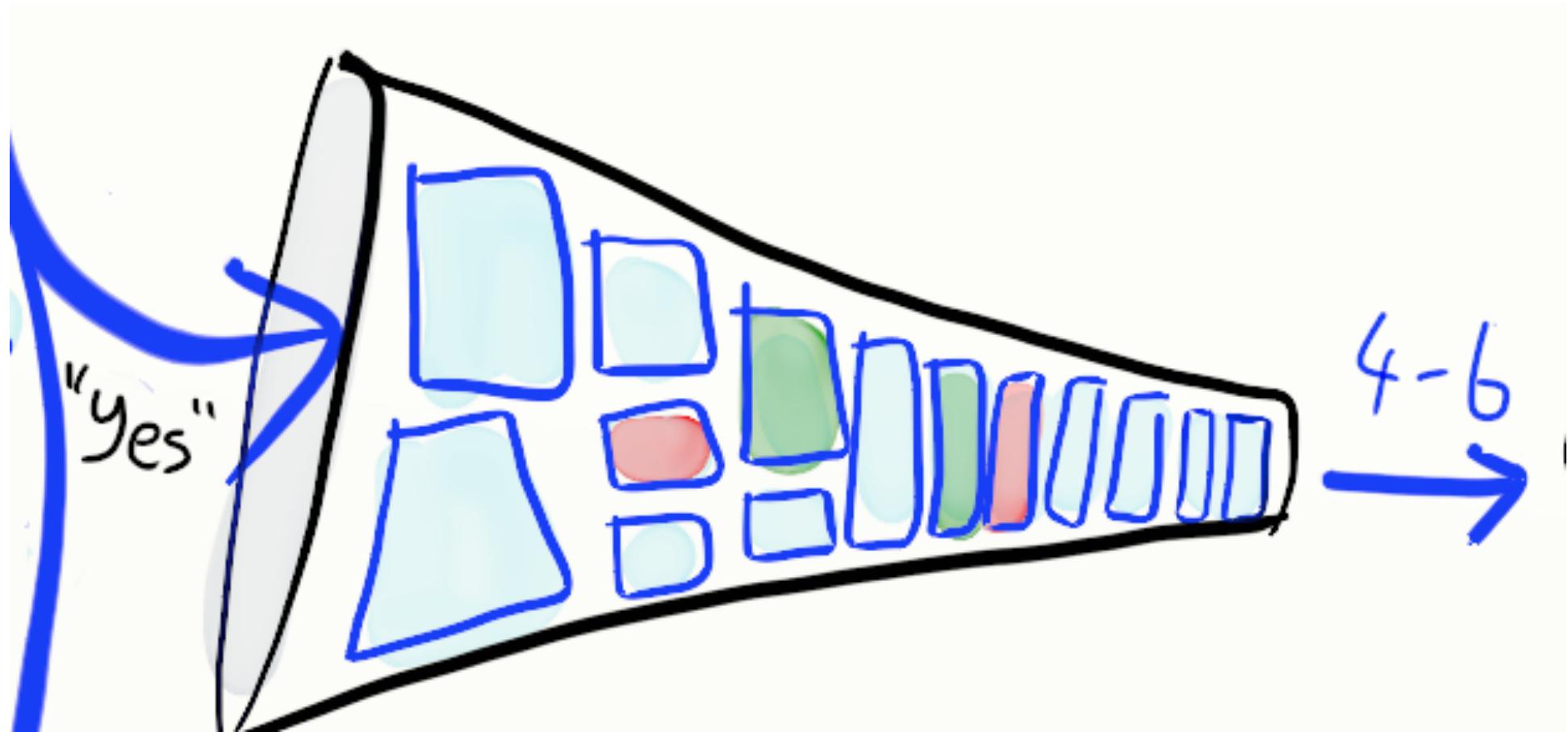
EMBRACE CHANGE

No es asumible que añadir
nuevos campos a un modelo
obligue a revisar y modificar las
consultas SQL ya desarrolladas.

¿Cómo controlar un proceso no predecible?

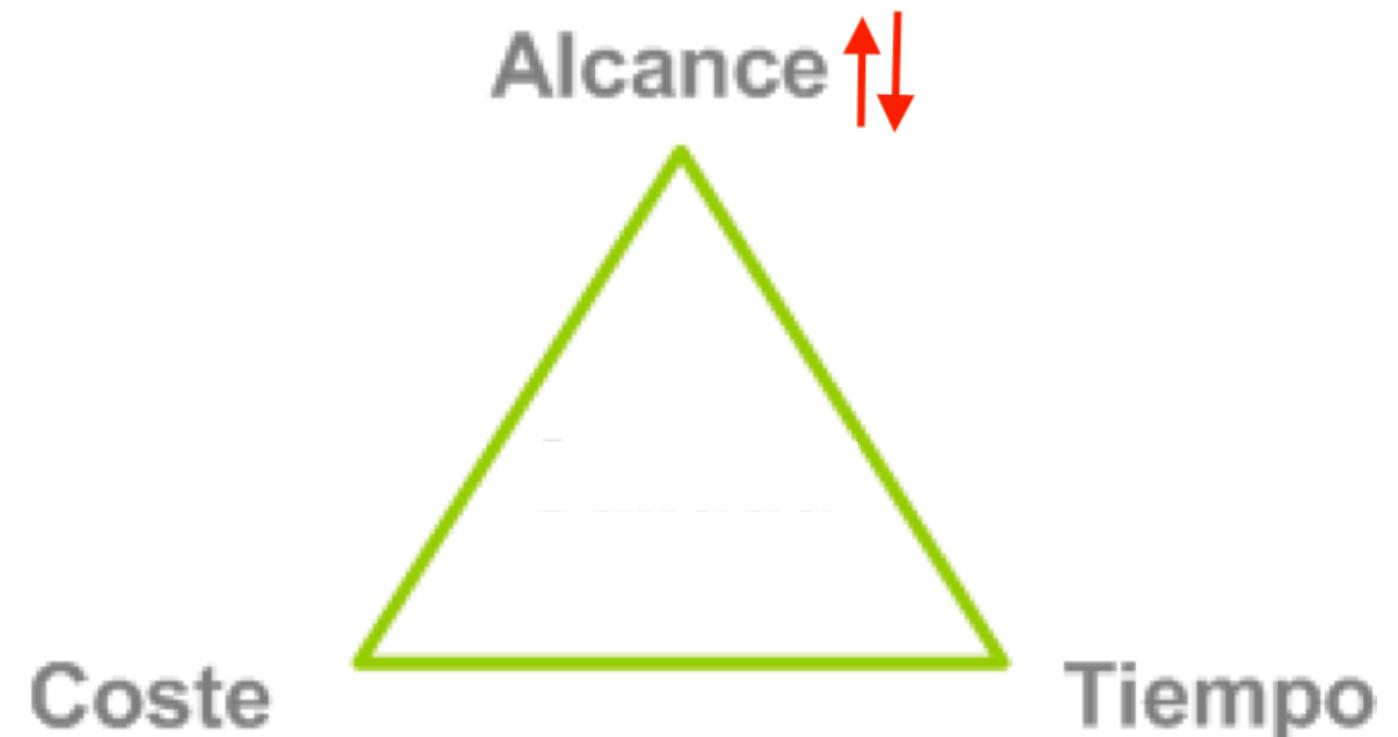


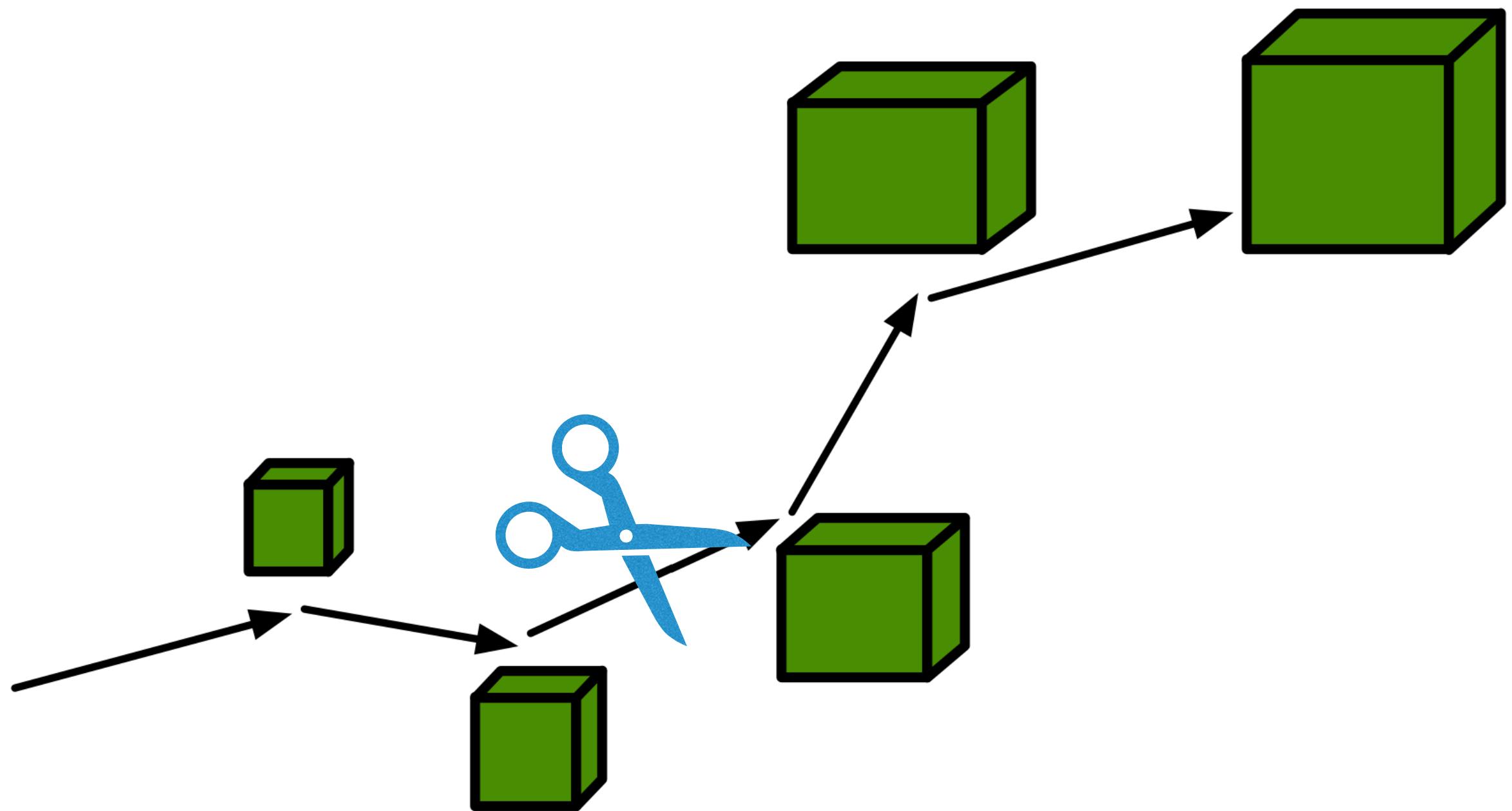




Contratos ágiles

- **Contrato tradicional:** precio fijo, el cliente paga al final por unos requisitos en un plazo. Si se incumple el plazo o los requisitos el cliente puede no pagar y la empresa de software no entrega nada.
- **Contrato ágil:** presupuesto fijo, se paga a intervalos establecidos y la empresa de software entrega periódicamente el producto. Cualquiera de los dos puede cancelar el contrato en cualquier momento.



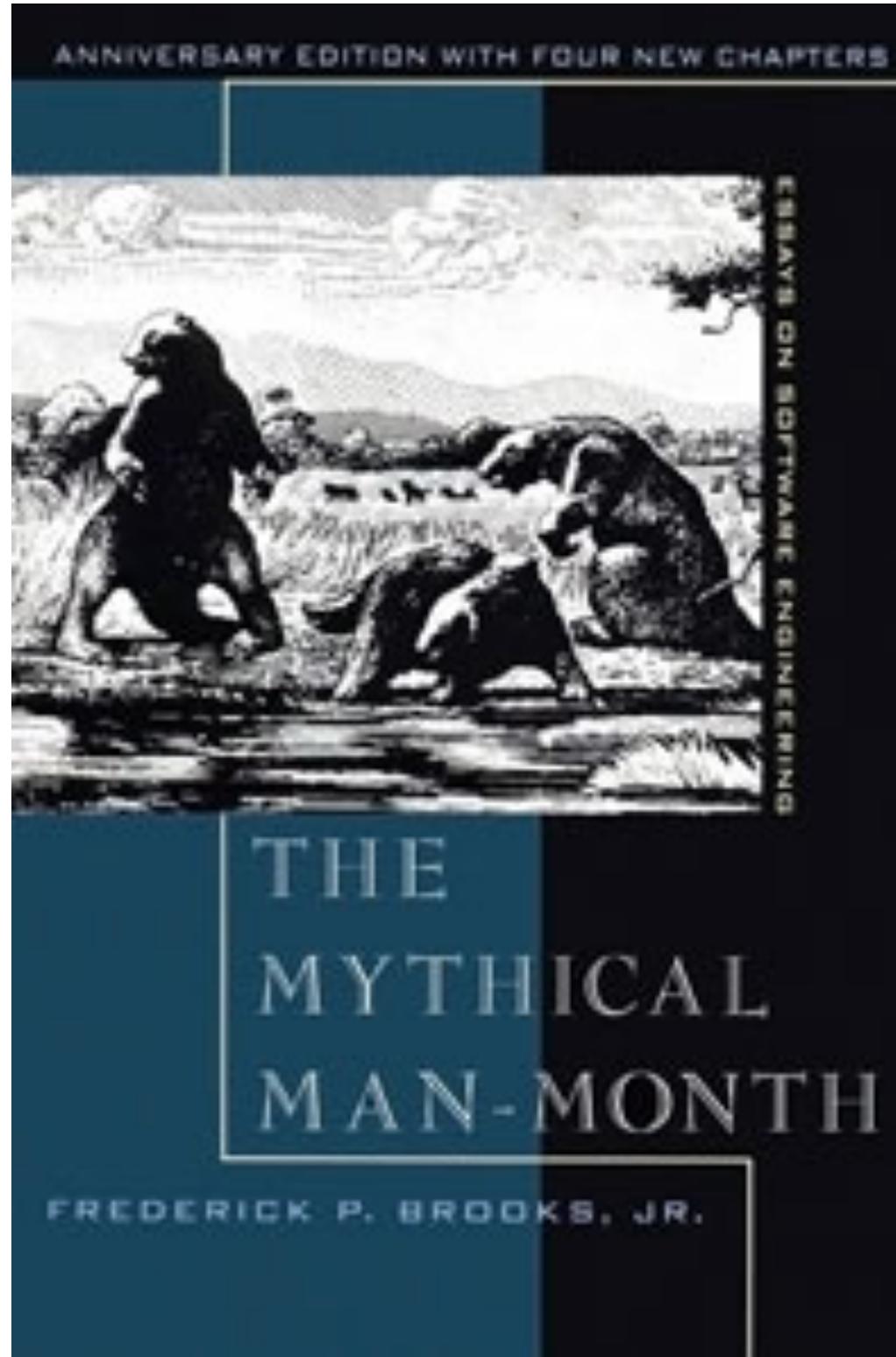




"Un cambio de última hora en los requerimientos es una ventaja competitiva."

Mary Poppendieck

4. El desarrollo de software es una actividad creativa



1975

"Adding
manpower to a
late software
project makes it
later."

Fred Brooks

No Silver Bullet

Essence and Accidents of Software Engineering

Frederick P. Brooks, Jr.

University of North Carolina at Chapel Hill

Fashioning complex conceptual constructs is the *essence*; accidental tasks arise in representing the constructs in language. Past progress has so reduced the accidental tasks that future progress now depends upon addressing the essence.

Of all the monsters that fill the nightmares of our folklore, none terrify more than werewolves, because they transform unexpectedly from the familiar into horrors. For these, one seeks bullets of silver that can magically lay them to rest.

The familiar software project, at least as seen by the nontechnical manager, has something of this character; it is usually innocent and straightforward, but is capable of becoming a monster of missed schedules, blown budgets, and flawed products. So we hear desperate cries for a silver bullet—something to make software costs drop as rapidly as computer hardware costs do.

But, as we look to the horizon of a decade hence, we see no silver bullet. There is no single development, in either technology or in management technique, that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity. In this article, I shall try to show why, by examining both the nature of the software problem and the properties of the bullets proposed.

Skepticism is not pessimism, however. Although we see no startling break-

throughs—and indeed, I believe such to be inconsistent with the nature of software—many encouraging innovations are under way. A disciplined, consistent effort to develop, propagate, and exploit these innovations should indeed yield an order-of-magnitude improvement. There is no royal road, but there is a road.

The first step toward the management of disease was replacement of demon theories and humours theories by the germ theory. That very step, the beginning of hope, in itself dashed all hopes of magical solutions. It told workers that progress would be made stepwise, at great effort, and that a persistent, unremitting care would have to be paid to a discipline of cleanliness. So it is with software engineering today.

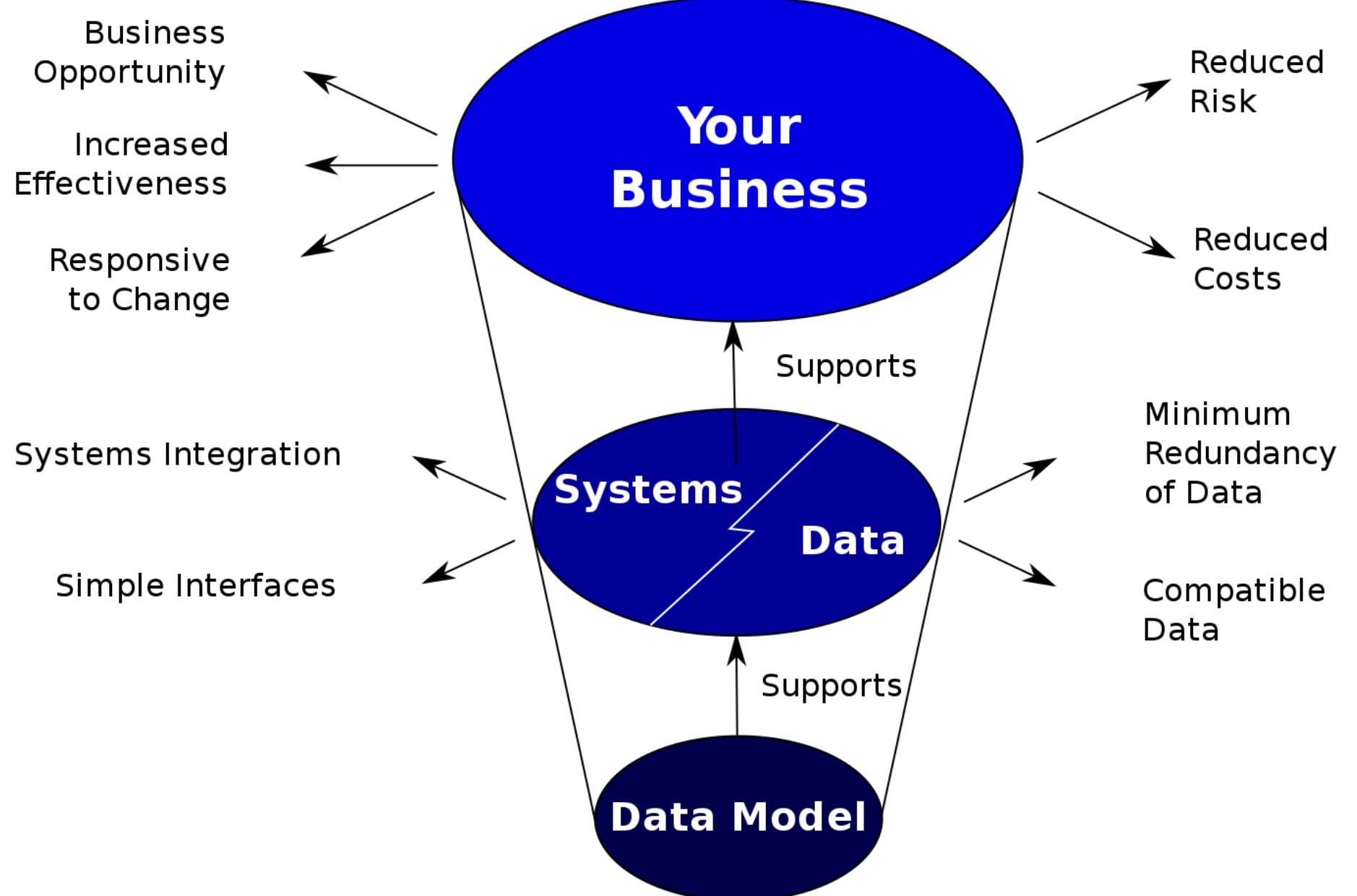
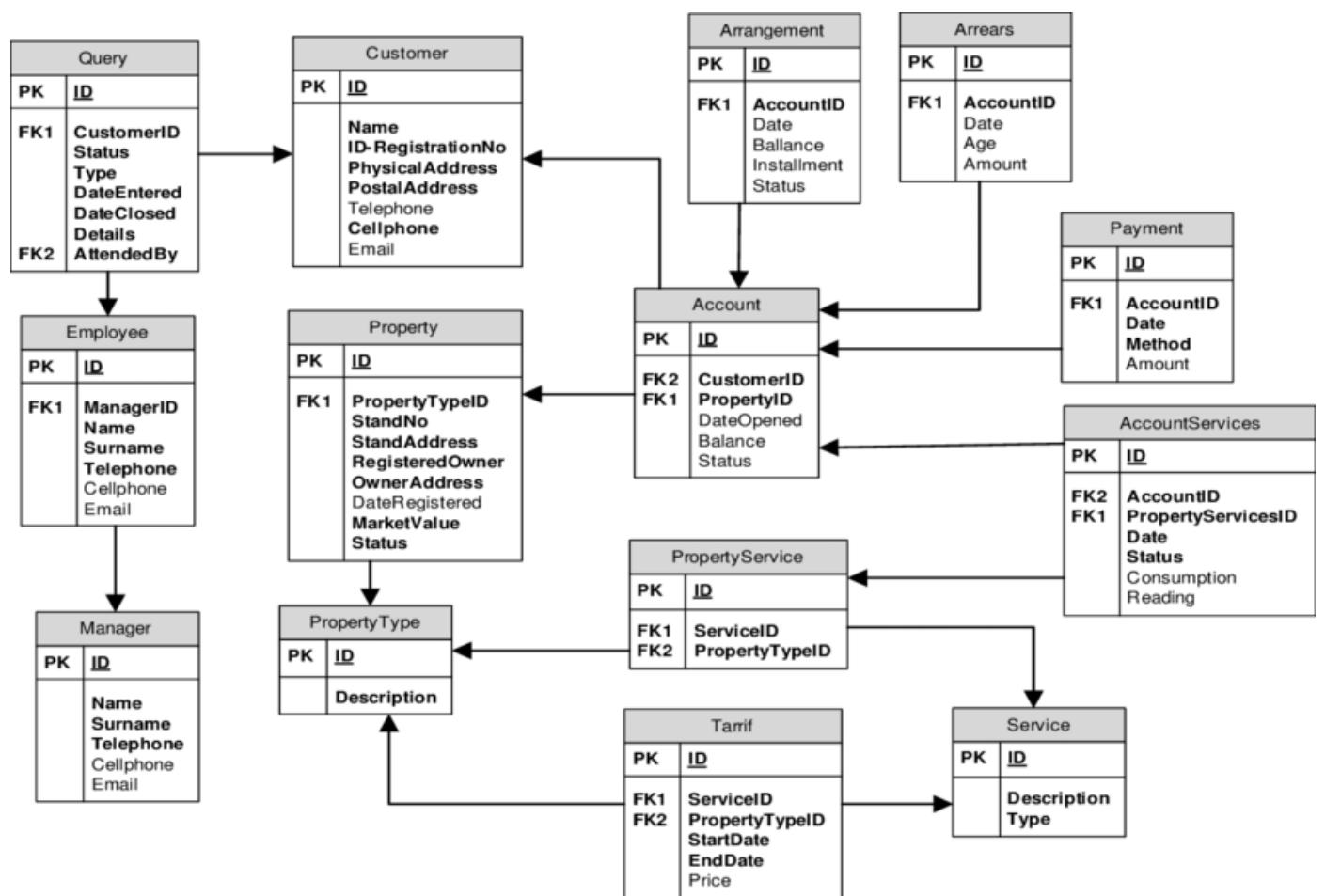
Does it have to be hard?—Essential difficulties

Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any—no inventions that will do for software productivity, reliability, and simplicity what electronics, transistors, and large-scale integration did for computer hardware.

This article was first published in *Information Processing '86*, ISBN No. 0-444-70077-3, H.-J. Kugler, ed., Elsevier Science Publishers B.V. (North-Holland) © IFIP 1986.

1986

Tareas esenciales

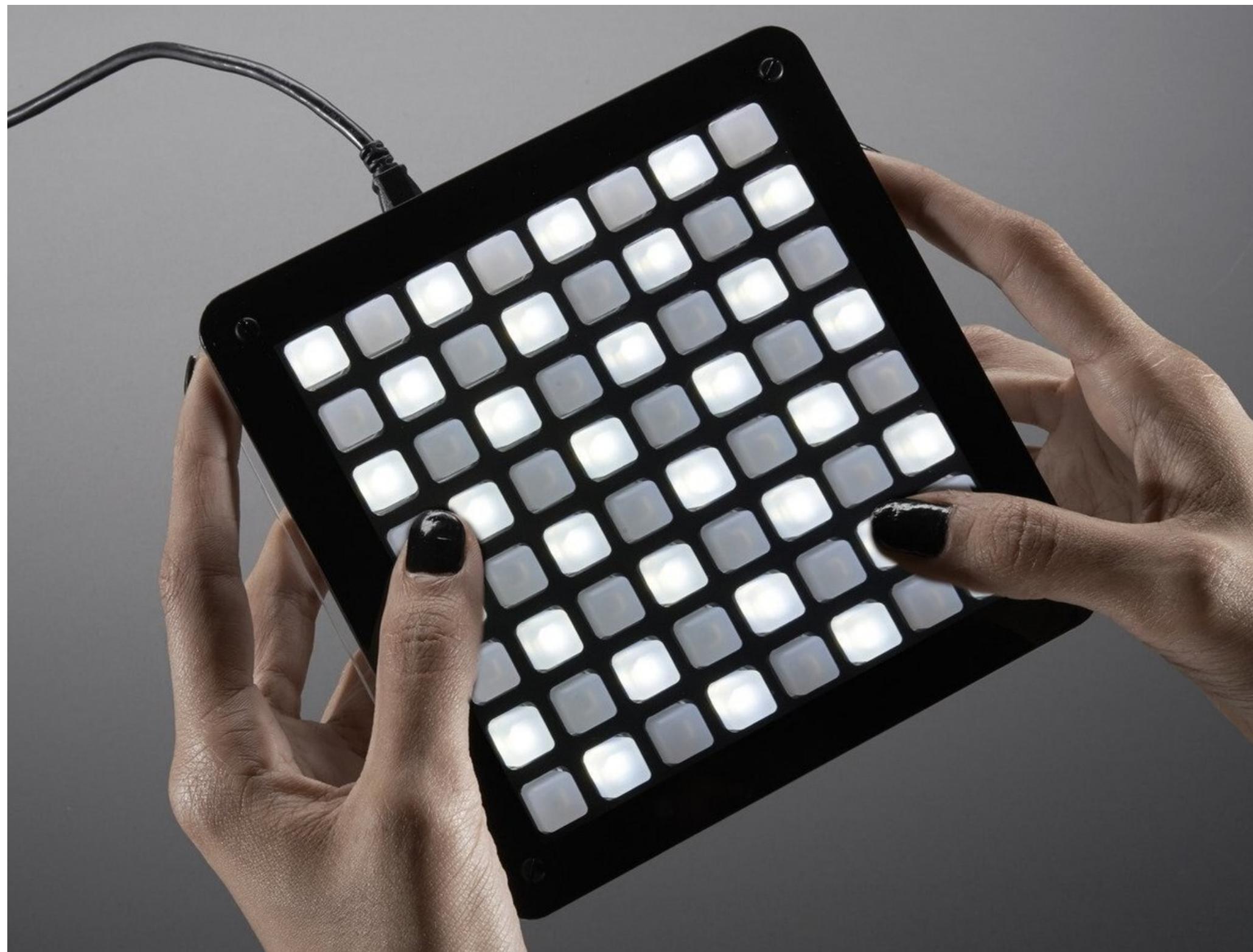


Tareas accidentales

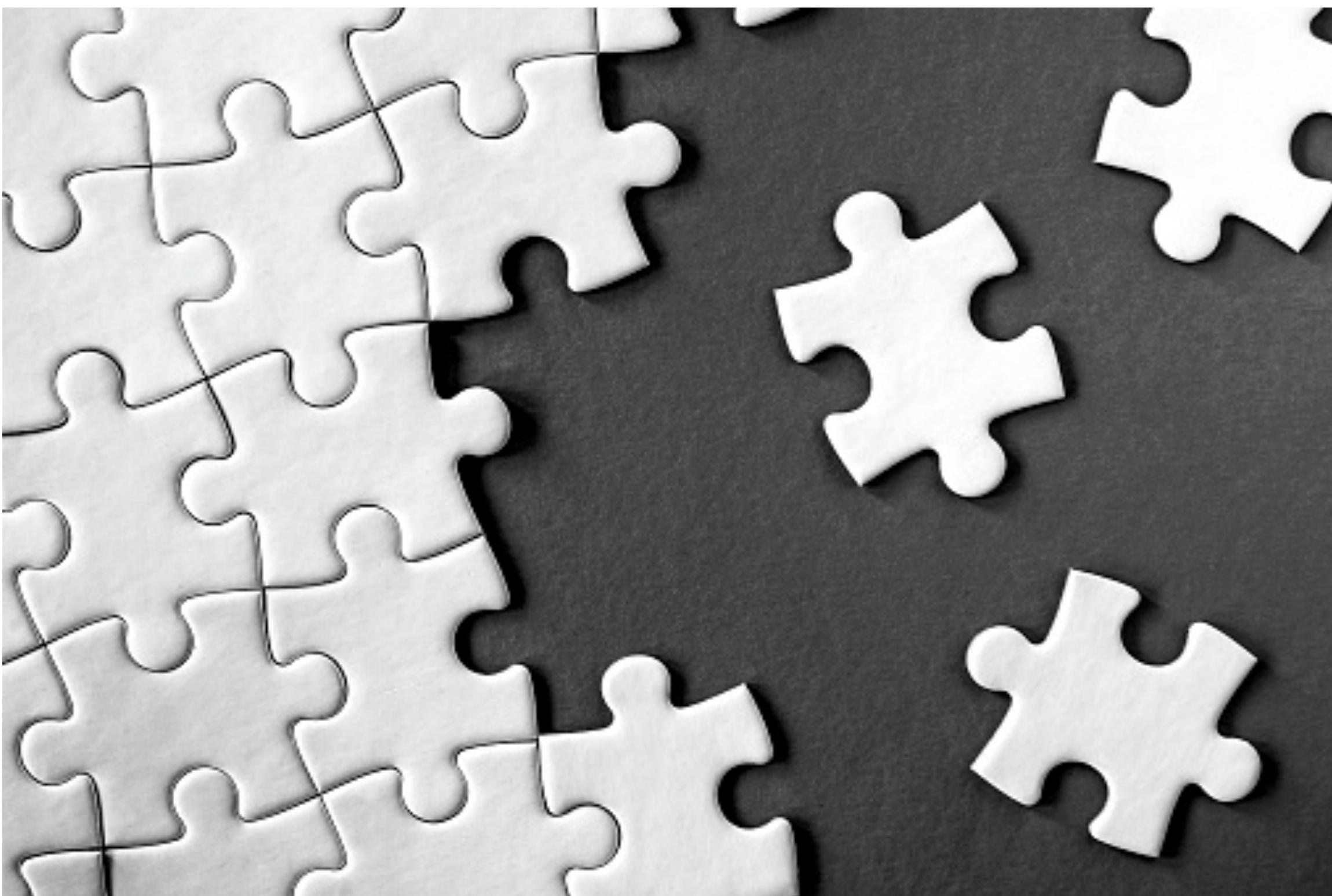


**Las tareas esenciales son
difícilmente optimizables**

Complejidad



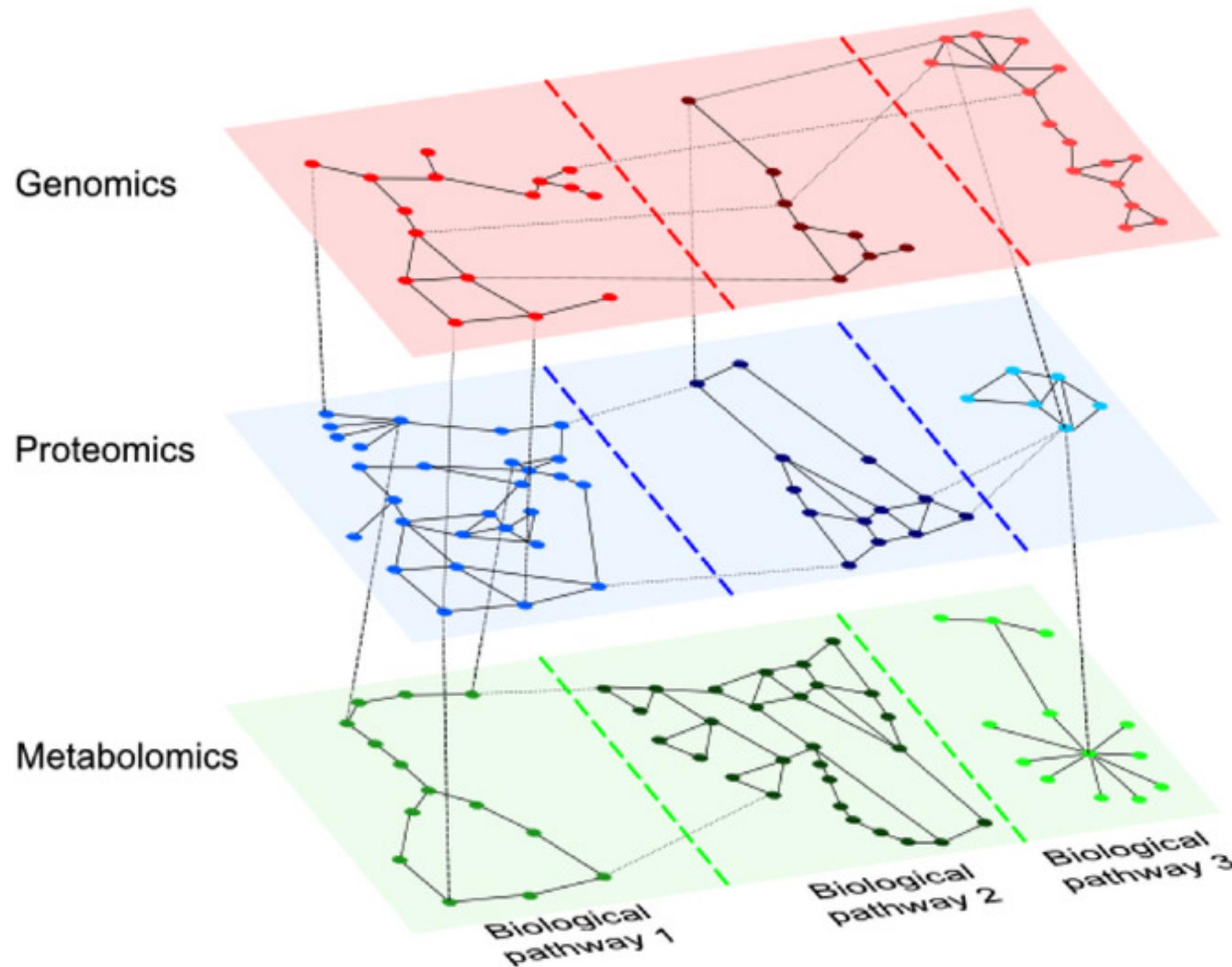
Conformidad

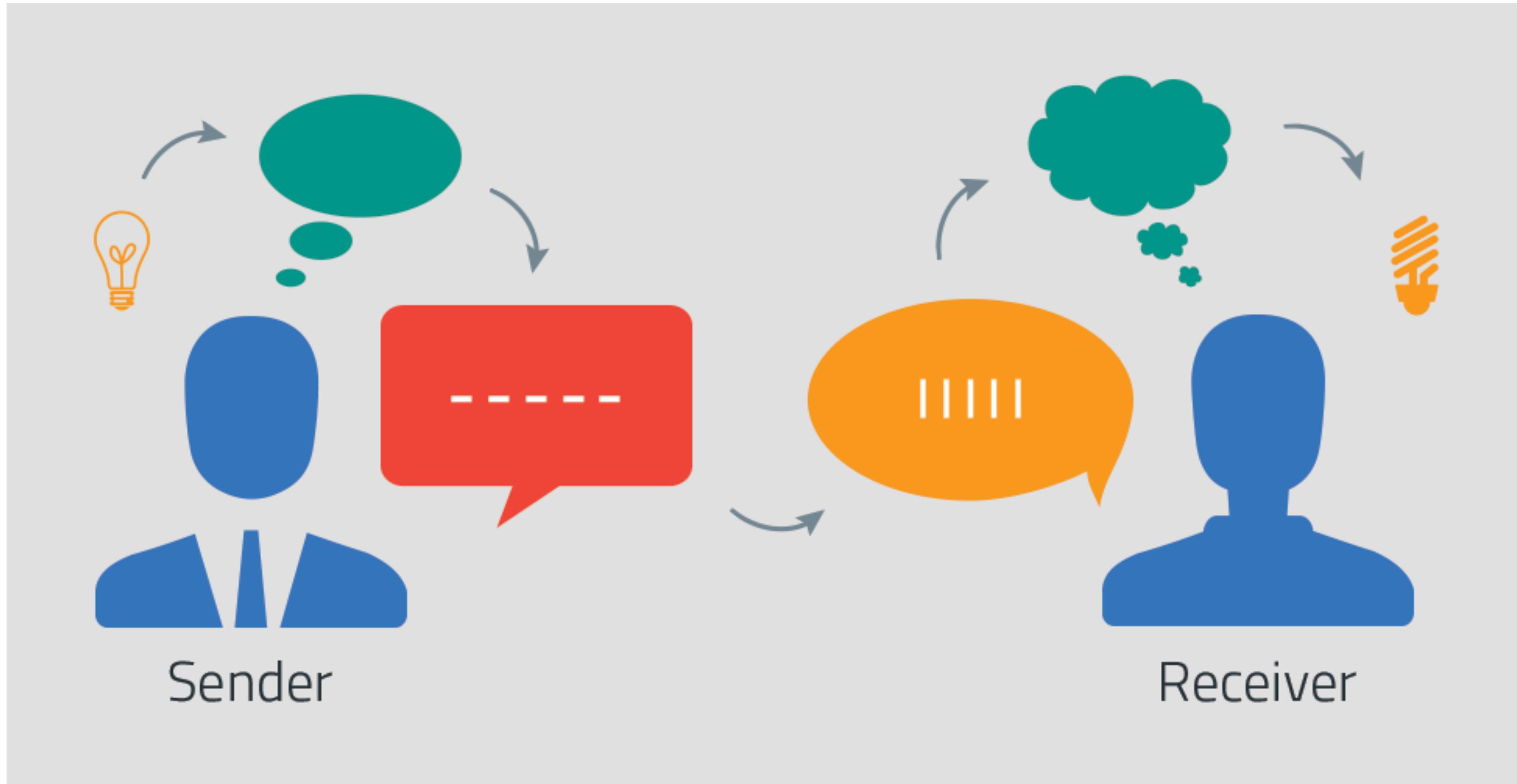


Cambiabilidad

changeability

Invisibilidad





Algunas posibles optimizaciones

Comprar siempre que sea posible

```
1 import Stripe from 'stripe';
2 const stripe = new Stripe('sk_t
3
4 await stripe.paymentIntents.cre
5
6   amount: 2000,
7   currency: 'USD',
8 });
~
```

Wildlife Expedition

SELECT TICKETS > TICKET INFORMATION > PAYMENT INFORMATION

Credit card   

Cardholder name: Jane Diaz

Card number: MM/YY:

PAY NOW



The Azuga website features a navigation bar with links for Products, Solutions, Industries, Insurance, Partners, Resources, and Company.

Fleet Tracking
GPS Fleet Tracking Platform built for safety, savings, and insights

Fleet Tracking App
Stay connected with the Fleet Mobile App

Asset Tracking
Protect your high-value equipment with real-time GPS asset tracking

eLogs (ELD)
Avoid fines with an ELD-compliant fleet solution

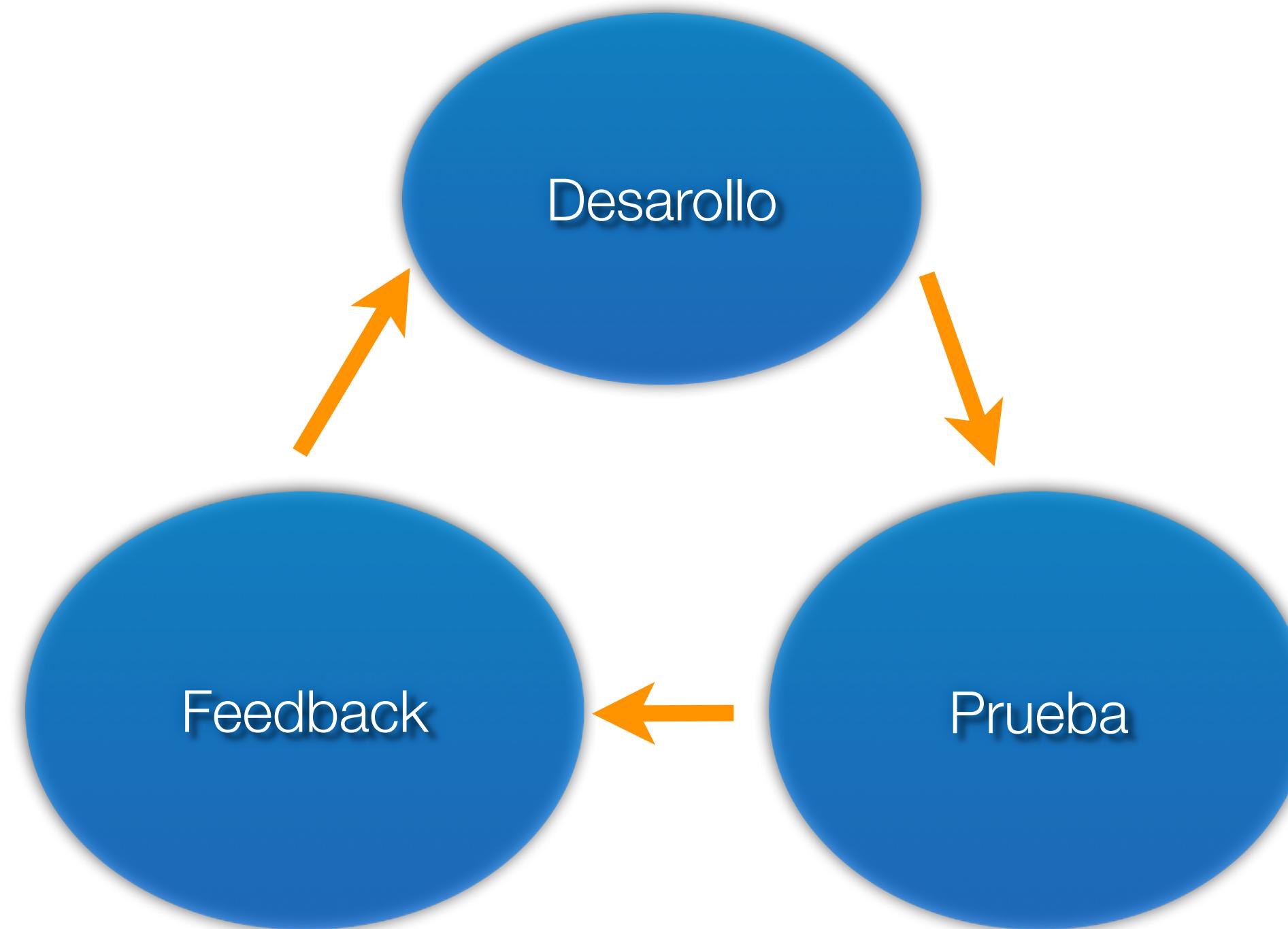
Driver Safety
Fleet safety and training programs for drivers

Fleet Bundles
Find the package that fits your business

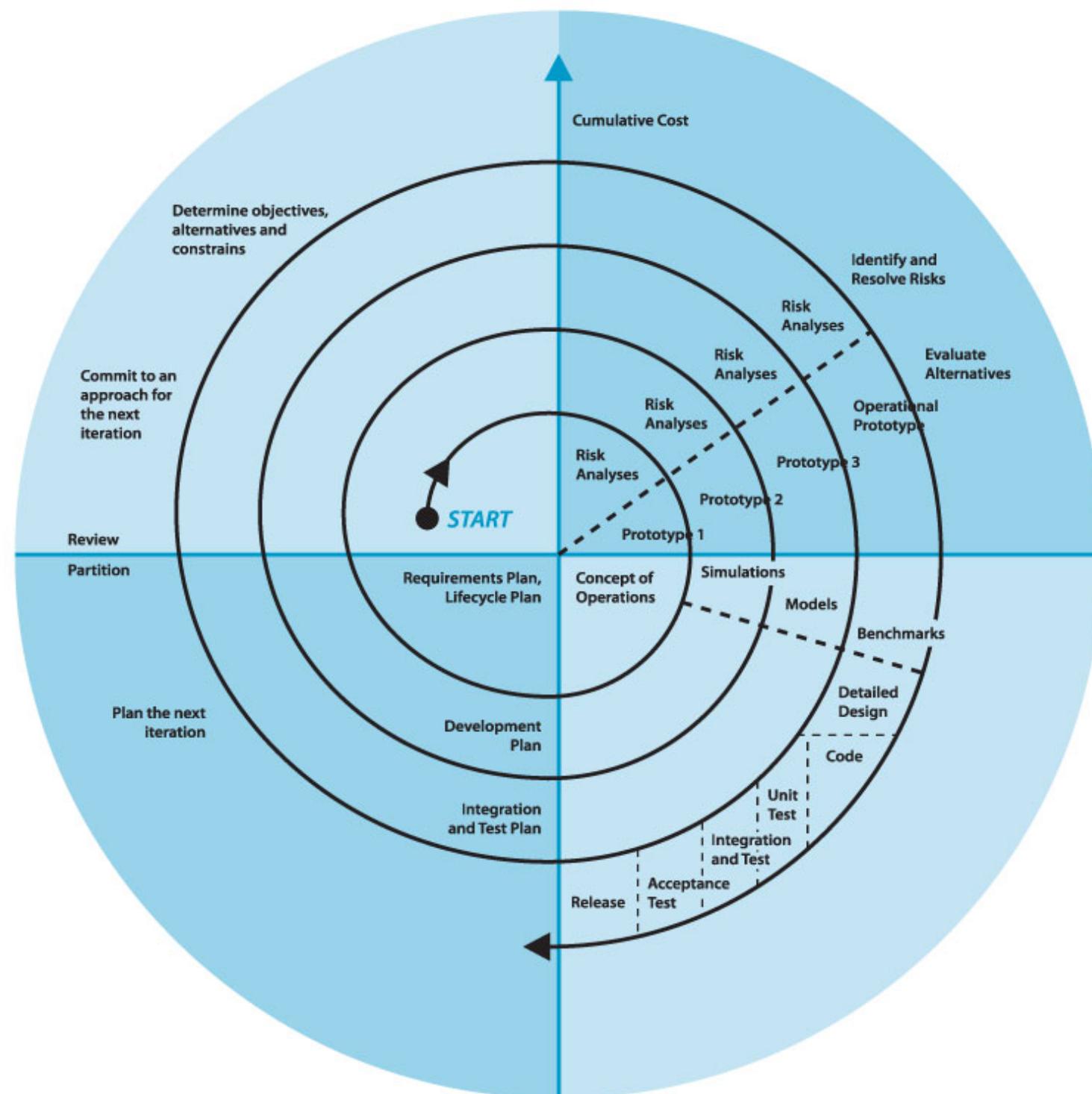
Azuga Fleet Management Software

Improve your business at every turn with Azuga Fleet management

Prototipado rápido y refinamiento de los requisitos



El software debe crecer orgánicamente, no ser construido





Search or jump to...

/ Pull requests Issues Codespaces Marketplace Explore

domingogallardo / apuntes-mads

Code

Issues

Pull requests

Actions

Projects

Wiki

Security

Insights

master ▾

apuntes-mads / apuntes / desarrollo-software / desarrollo-software.md



domingogallardo Corrección

1 contributor

858 lines (707 sloc) | 43.6 KB

Desarrollo de software

Veremos en este apartado las características propias del software y de la forma de desarrollarlo que comparar con ingenierías tradicionales como la construcción.

Software

El software es un invento muy reciente de la humanidad. Fue a mitad del siglo XX cuando se empezaron a utilizar los primeros computadores electrónicos programables en organismos oficiales y grandes empresas. Y los primeros lenguajes de programación fueron creados para manejarlos.