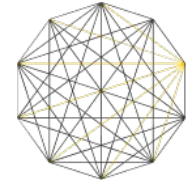


V&V

Desarrollo software, una aproximación



540
quinientoscuarenta



¿Por qué estamos aquí?



Software economics




Proteger una inversión





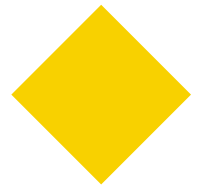
Todos somos negocio





Somebody has to pay for all this. Software development that doesn't acknowledge economics risks the hollow victory of a 'technical success'. ***Make sure what you are doing has business value, meets business goals, and serves business needs.*** For example, solving the highest priority business need first maximizes the value of the project. Two aspects of economics that affect software development are the ***time value of money*** and the ***option value of systems and teams***. The time value of money says that a dollar today is worth more than a dollar tomorrow. ***Software development is more valuable when it earns money sooner and spends money later.*** Incremental design explicitly defers design investment until the last responsible moment in an effort to spend money later. All the practices are intended to enhance the option value of both the software and the team while keeping in mind the time value of money by not investing in speculative flexibility.

Kent Beck



Óptica economics

- Una mentalidad: debemos ser conscientes de que todas nuestras acciones tienen un impacto económico en el producto que realizamos
- Herramientas: ¿cómo afectan las prácticas técnicas al coste, valor, riesgo y deuda de un producto?
- Un marco para la toma de decisiones
- Un lenguaje común



Valor





Coste





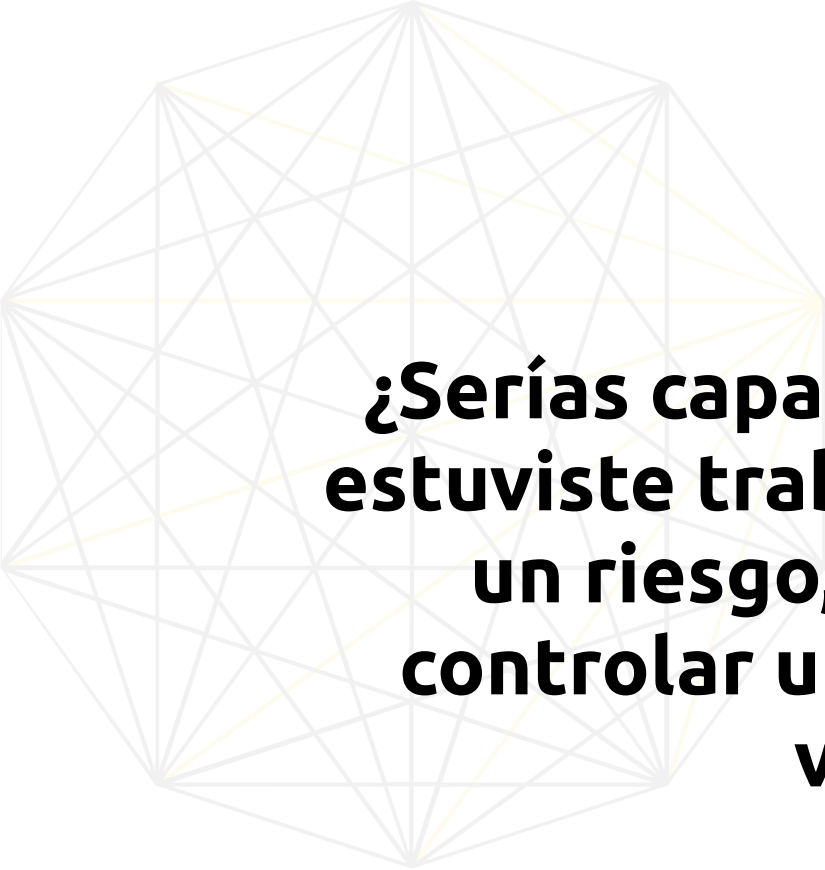
Deuda





Riesgo





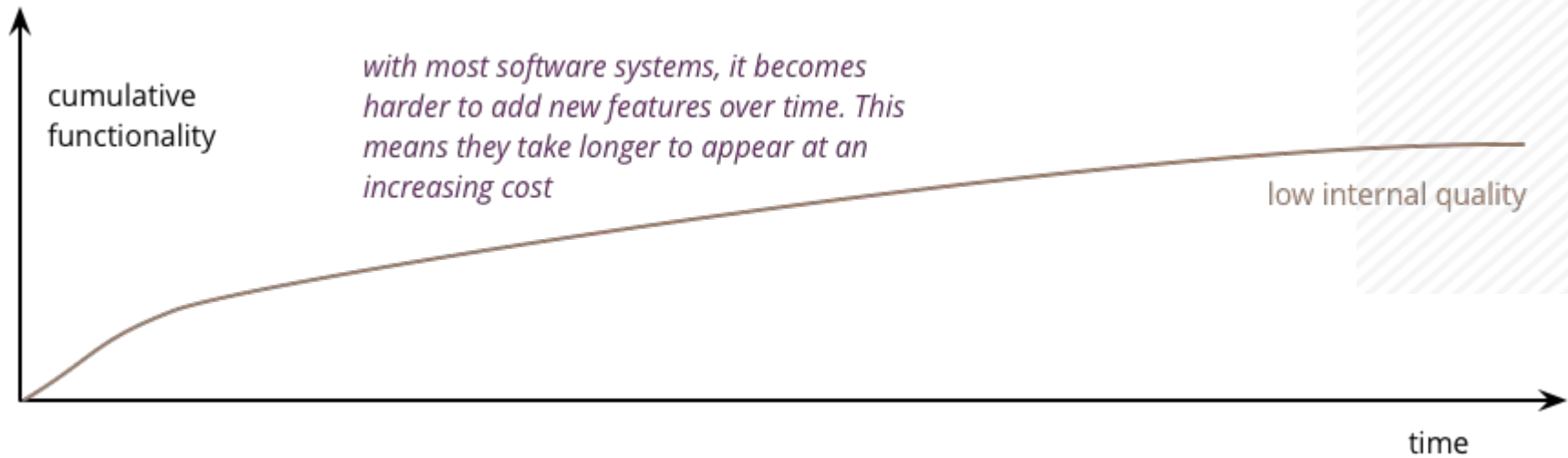
**¿Serías capaz de decir si ayer
estuviste trabajando en reducir
un riesgo, pagar deuda,
controlar un coste o añadir
valor?**



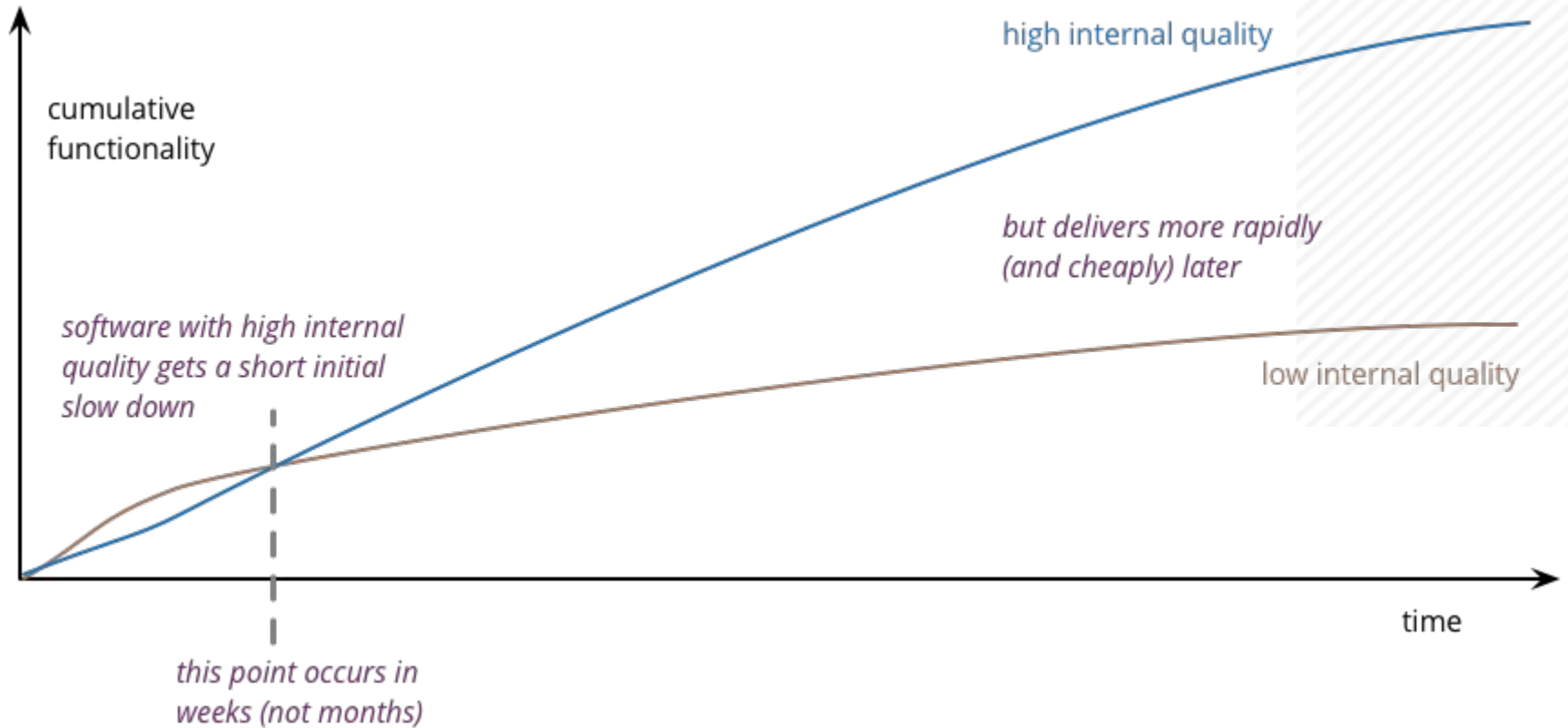
Desarrollo software de calidad

**Pero, ¿qué es
desarrollo de
software de
calidad?**





Martin Fowler - Is High Quality Software Worth the Cost?

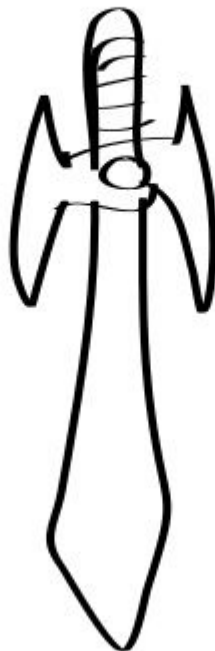


Martin Fowler - Is High Quality Software Worth the Cost?

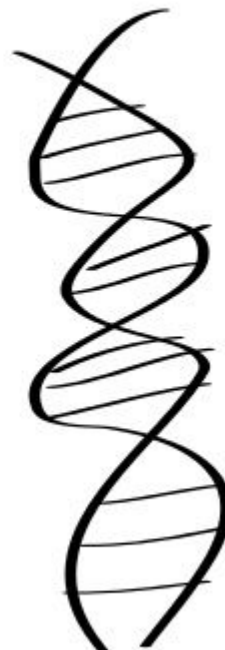
Fragile



Robust



Anti-Fragile



Anti-frágil





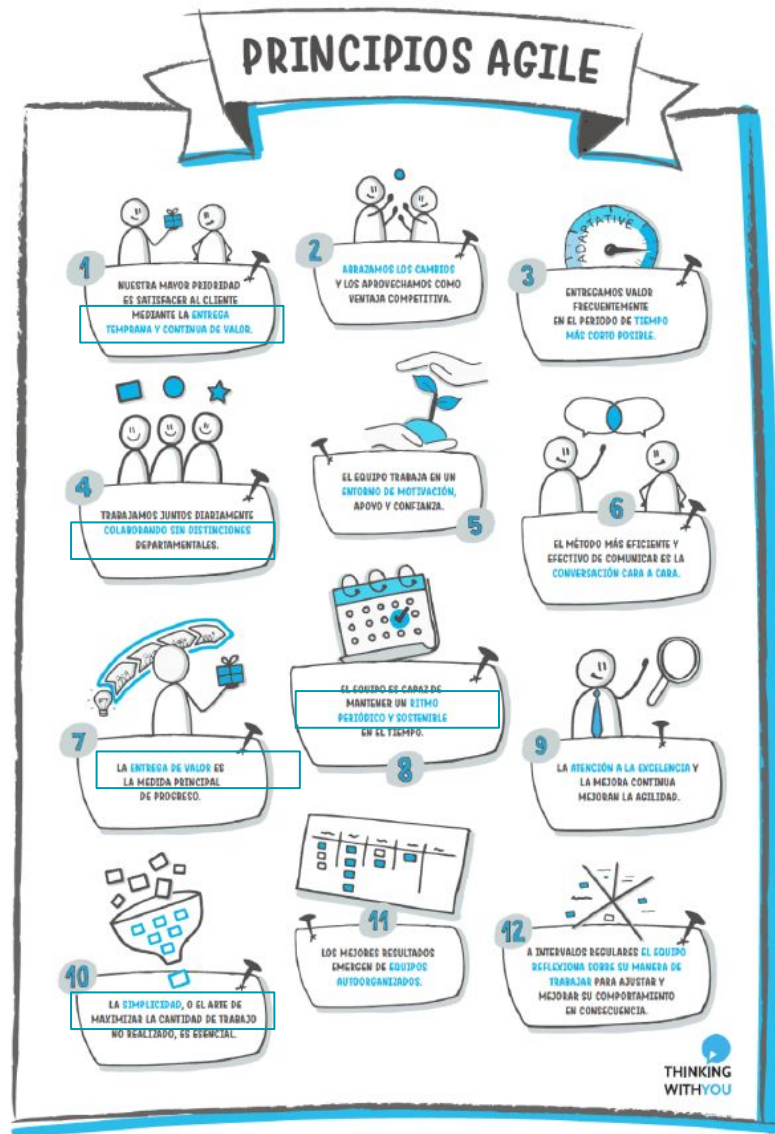
**Propiedad de los sistemas que aumentan
su capacidad para sobrevivir como
resultado de factores estresantes: golpes,
etc.**



Software economics - Luis Artola

Agile, XP, software craftsmanship





Metodologías ágiles

- Scrum
 - Sprints
 - Dailies
 - Demos y Retrospectivas
 - Panel scrum
- Kanban
 - Panel kanban
 - Priorización clara
 - WIP
 - Visualización clara

Metodología de desarrollo que abarca desde la cultura de las relaciones entre las personas hasta técnicas de programación.

¿Qué es XP?

- Centrado en construir software mantenible.
- Permite ahorrar costes.
- Aplica en proyectos largos.

Riesgo

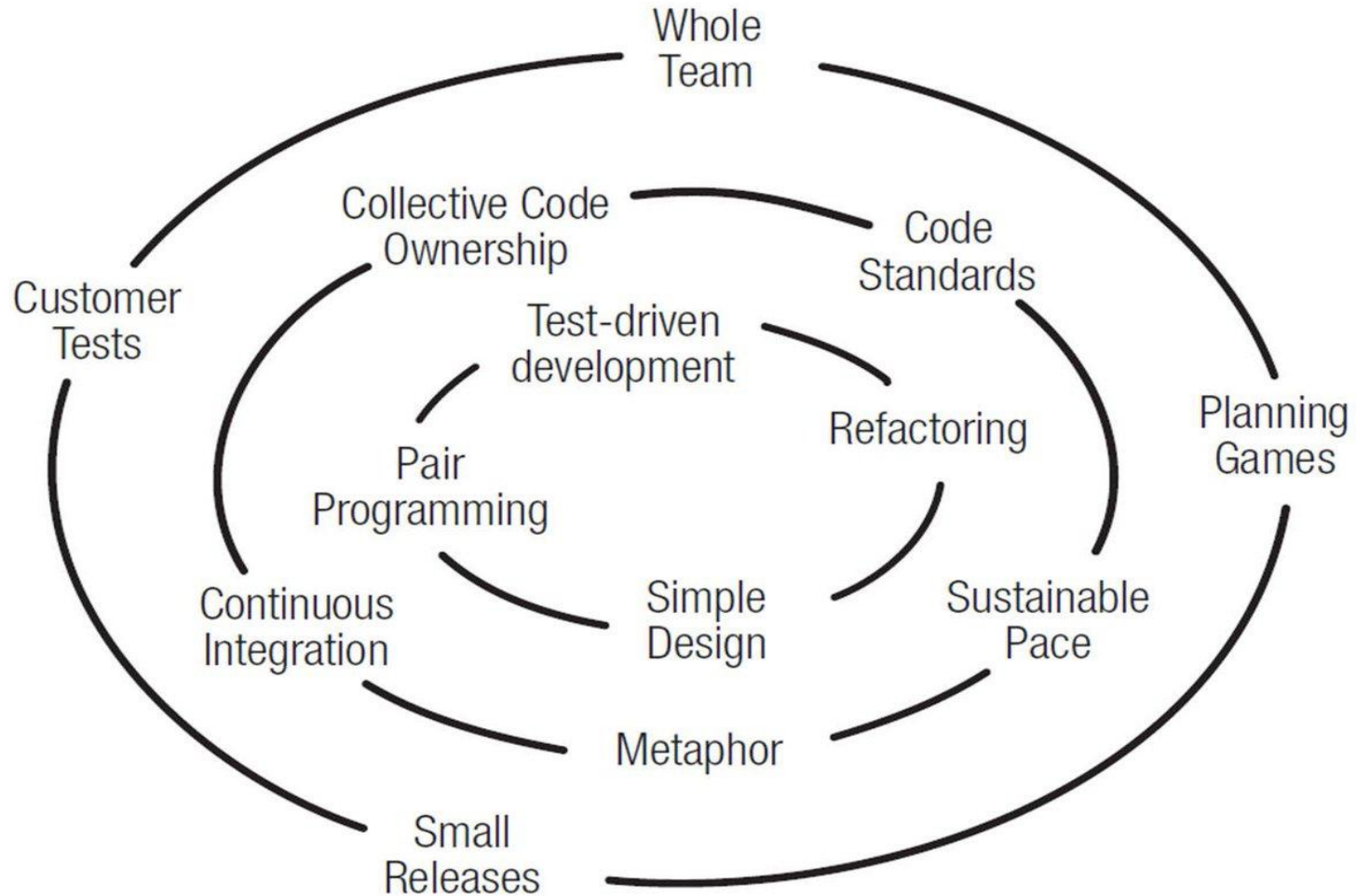
“El problema básico del desarrollo software es el riesgo.”

Kent Beck

¿Qué no es XP?

Agile y XP no es ir más rápido.

Core practices



Core practices

Test Driven Development

Classic TDD
Test doubles
Outside in TDD / London
School of TDD

Simple Design

4 elements of simple design
SOLID principles
Design patterns
Domain Driven Design

Refactoring

IDE productivity
Code smells
Refactoring smells
Refactoring legacy code

Pair programming

Driver-navigator
Ping-pong / Chess clock
Pomodoro
Pair rotation

Manifiesto Software Craftsmanship / manifiesto ágil

Software craftsmanship

- No solo software que funciona, sino también software bien hecho.
- No solo respondiendo al cambio, sino también agregando valor de manera constante.
- No solo individuos e interacciones, sino también una comunidad de profesionales.
- No solo colaboración con el cliente, sino también asociaciones productivas.

Agile

- Individuos e interacciones por encima de procesos y herramientas
- Software funcionando por encima de documentación exhaustiva
- Colaboración con el cliente por encima de negociación contractual
- Respuesta ante el cambio por encima de seguir un plan

XP: introducción

¿Para qué XP y SC?

- ~~Calidad~~
- ~~Código limpio~~
- ~~Profesionalidad~~
- ~~Hacer lo correcto~~



- Economics
- Felicidad

Nuestra forma de entender el desarrollo de software

Principios





Desarrollar software es una actividad de grupo





Desarrollo por consenso





Hacemos las cosas bien: la calidad no es negociable





**Real options: post-ponemos decisiones
hasta el último momento responsable**





Todos somos negocio





Pragmatismo





Relaciones de largo plazo





Inspección, adaptación y mejora continua





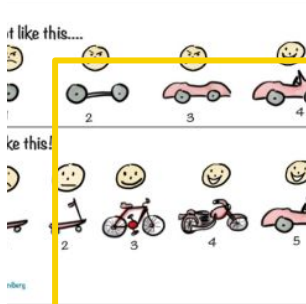
Fusión UX/UI y devs



Principios

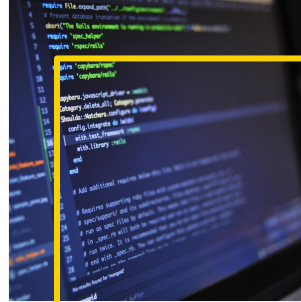
- Hacer software es una actividad de grupo.
- Desarrollo por consenso.
- Hacemos las cosas bien: la calidad no es negociable.
- Real options: post-ponemos decisiones hasta el último momento responsable.
- Todos somos negocio.
- Pragmatismo.
- Relaciones de largo plazo.
- Inspección, adaptación y mejora continua.
- Fusión UX/UI y devs.

Principios fundamentales



Be Agile my friend

+



**eXtreme
Programming**

=



Felicidad



Somos una empresa de desarrollo de software, especialmente de aplicaciones web y móvil.

Queremos **hacer felices a todas las personas que intervienen en el desarrollo de software.**

Esto lo hacemos desarrollando un **software bien diseñado, de calidad, mantenible** y que esté orientado a ayudar a nuestros clientes a **alcanzar continuamente sus objetivos de negocio.**





<https://540deg.com/>

@540deg