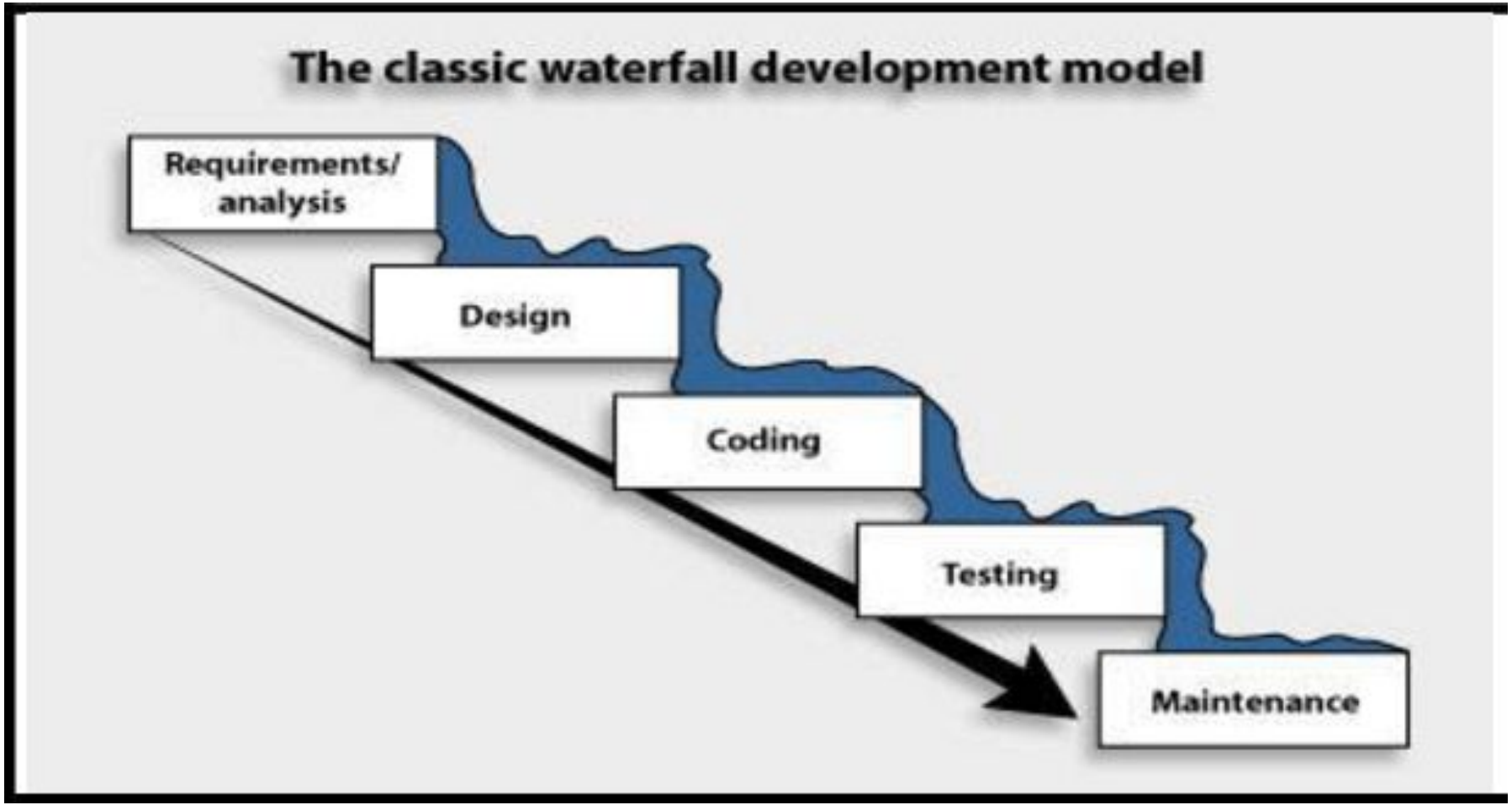
TESTING IIC3745

Ejercicio

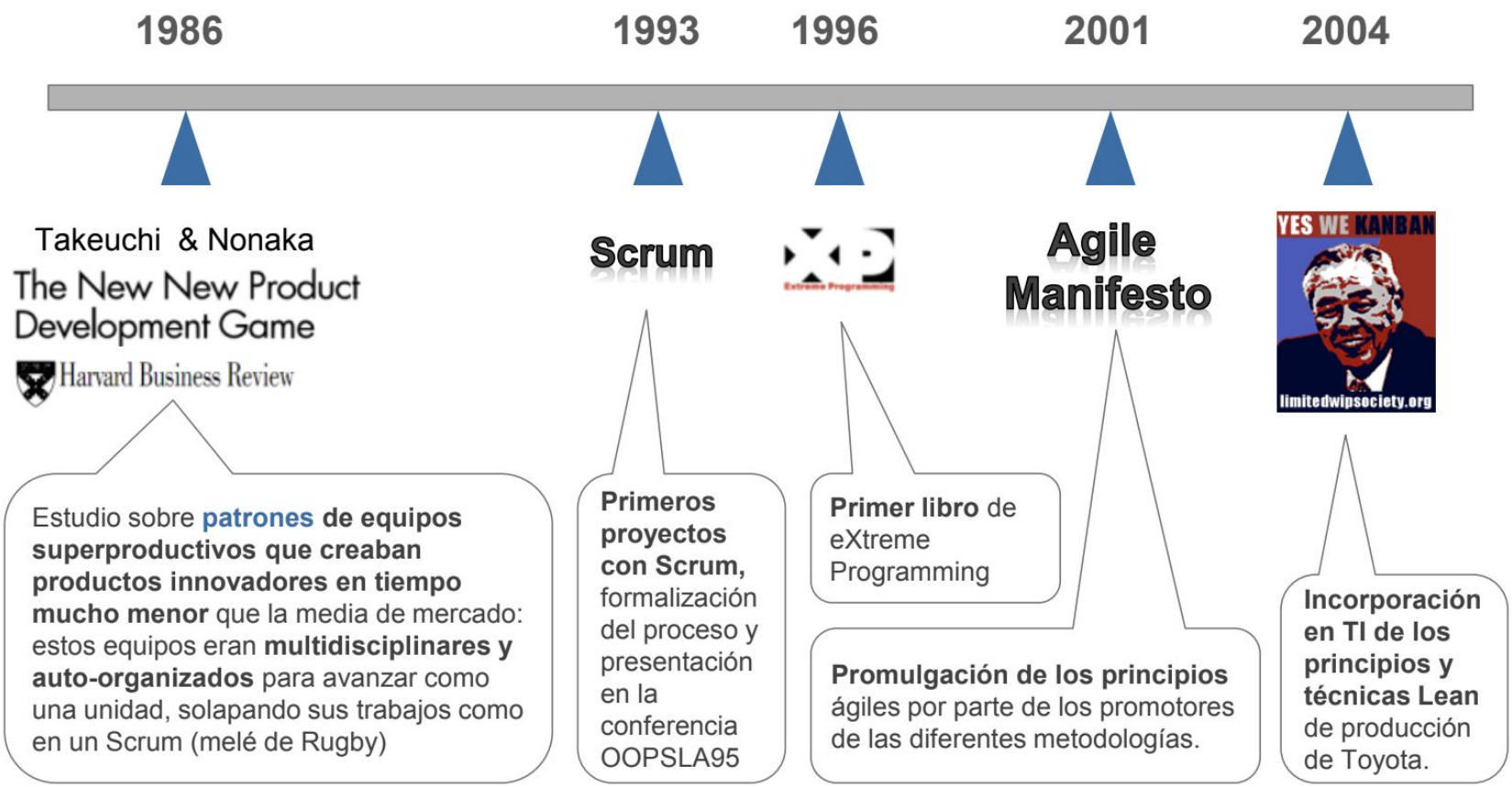
¿QUÉ expectativas tenéis?


The classic waterfall development model
Requirements/ analysis
Design
Coding
Testing
Maintenance

La calidad interna se degrada

Aplicaciones que por fuera parecen “humanas”…

… pero que por dentro son “La Cosa”.


1986
1993
1996
2001
2004
Takeuchi & Nonaka
The New New Product Development Game
Harvard Business Review
Estudio sobre patrones de equipos superproductivos que creaban productos innovadores en tiempo mucho menor que la media de mercado: estos equipos eran multidisciplinares y auto-organizados para avanzar como una unidad, solapando sus trabajos como en un Scrum (melé de Rugby)
Scrum
Primeros proyectos con Scrum, formalización del proceso y presentación en la conferencia OOPSLA95
Extreme Programming
Primer libro de eXtreme Programming
Agile Manifesto
Promulgación de los principios ágiles por parte de los promotores de las diferentes metodologías.
YES WE KANBAN
limitedwipsociety.org
Incorporación en TI de los principios y técnicas Lean de producción de Toyota.

Manifiesto ágil

2001

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 que funciona sobre *documentación exhaustiva*

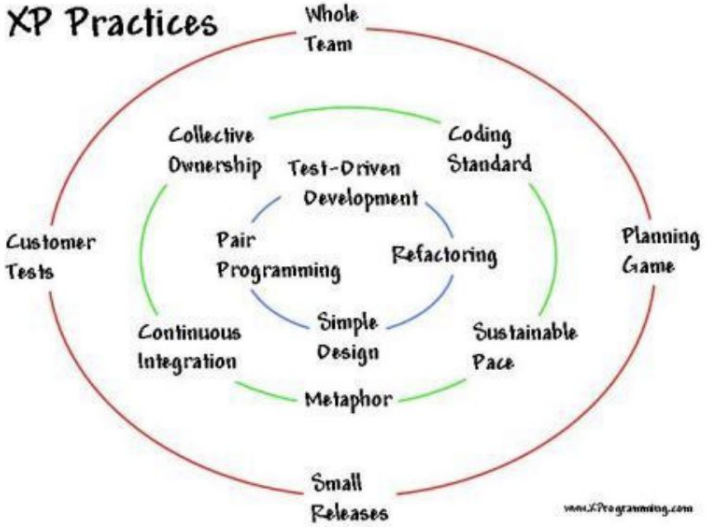
Colaboración con el cliente sobre *negociación de contratos*

Responder ante el cambio sobre *seguimiento de un plan*

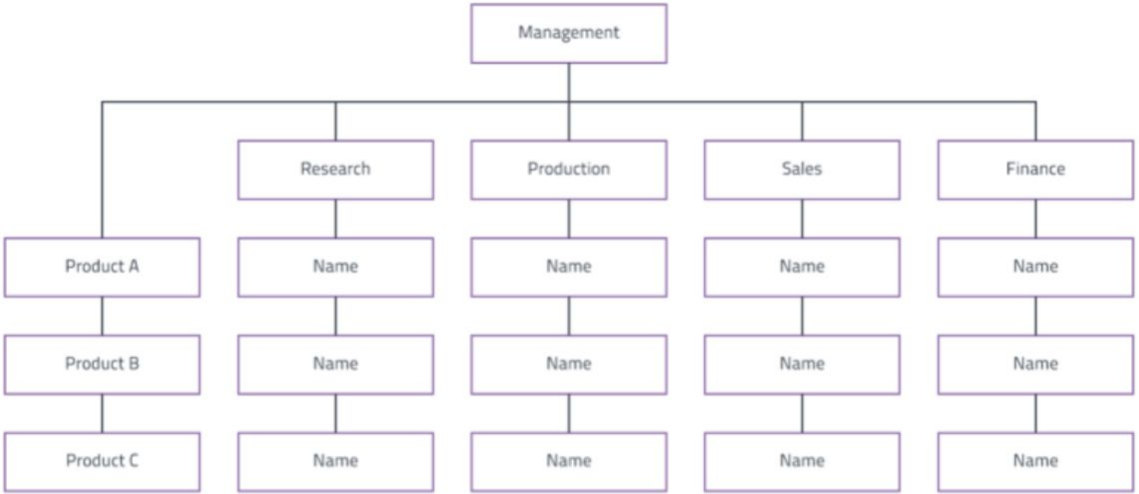
Esto es, aunque los elementos a la derecha tienen valor, nosotros valoramos por encima de ellos los que están a la izquierda.

eXtreme Programming - Prácticas

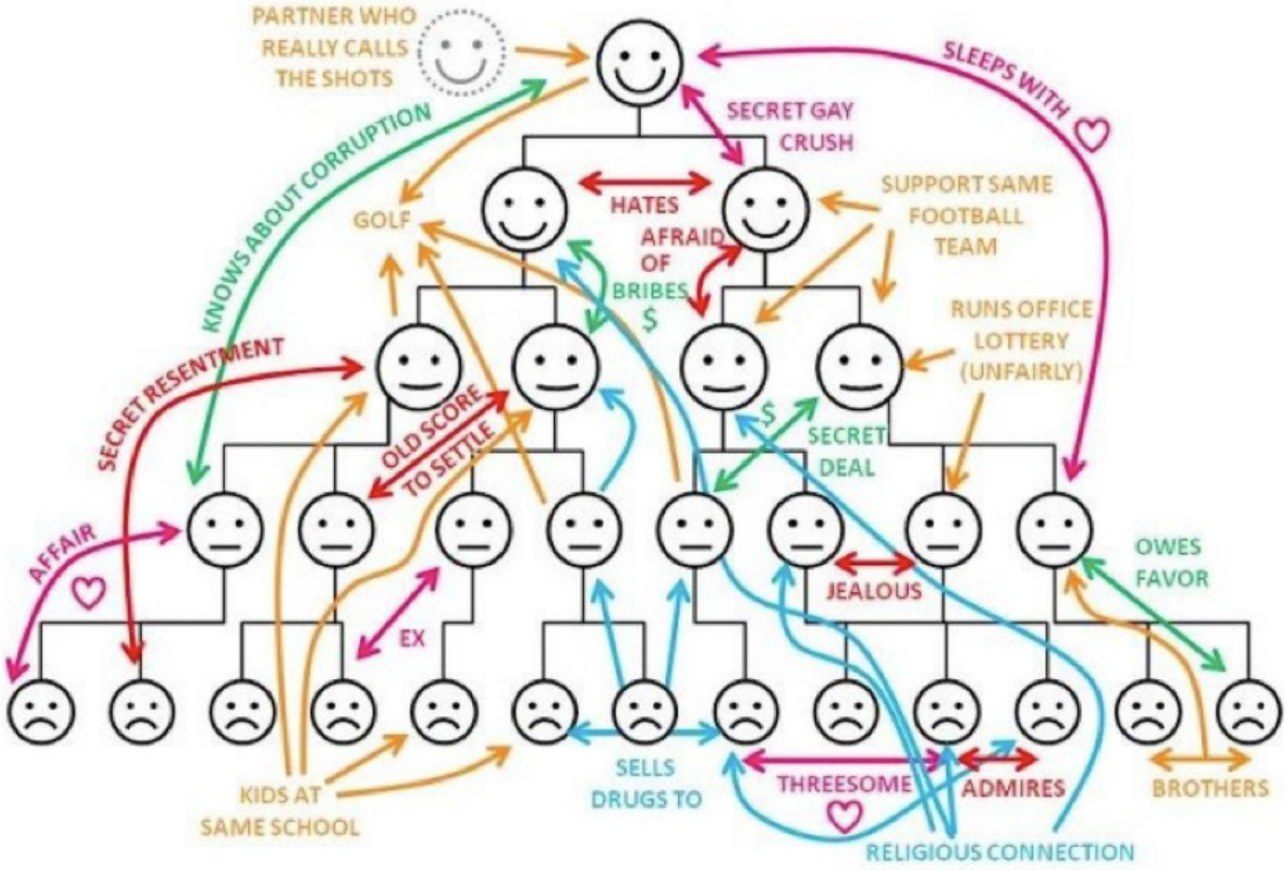
- Historias de usuario + BDD.
- Pruebas de concepto extremo a extremo.
- Estándares de codificación.
- Pruebas unitarias.
- TDD (Test Driven Development)
- Refactorización. Arquitectura emergente.
- Integración y pruebas continuas, automatizadas.
- Revisiones de código y/o programación en parejas.
- Propiedad colectiva del código.



Esto es una empresa



¿O se parece más a esto?

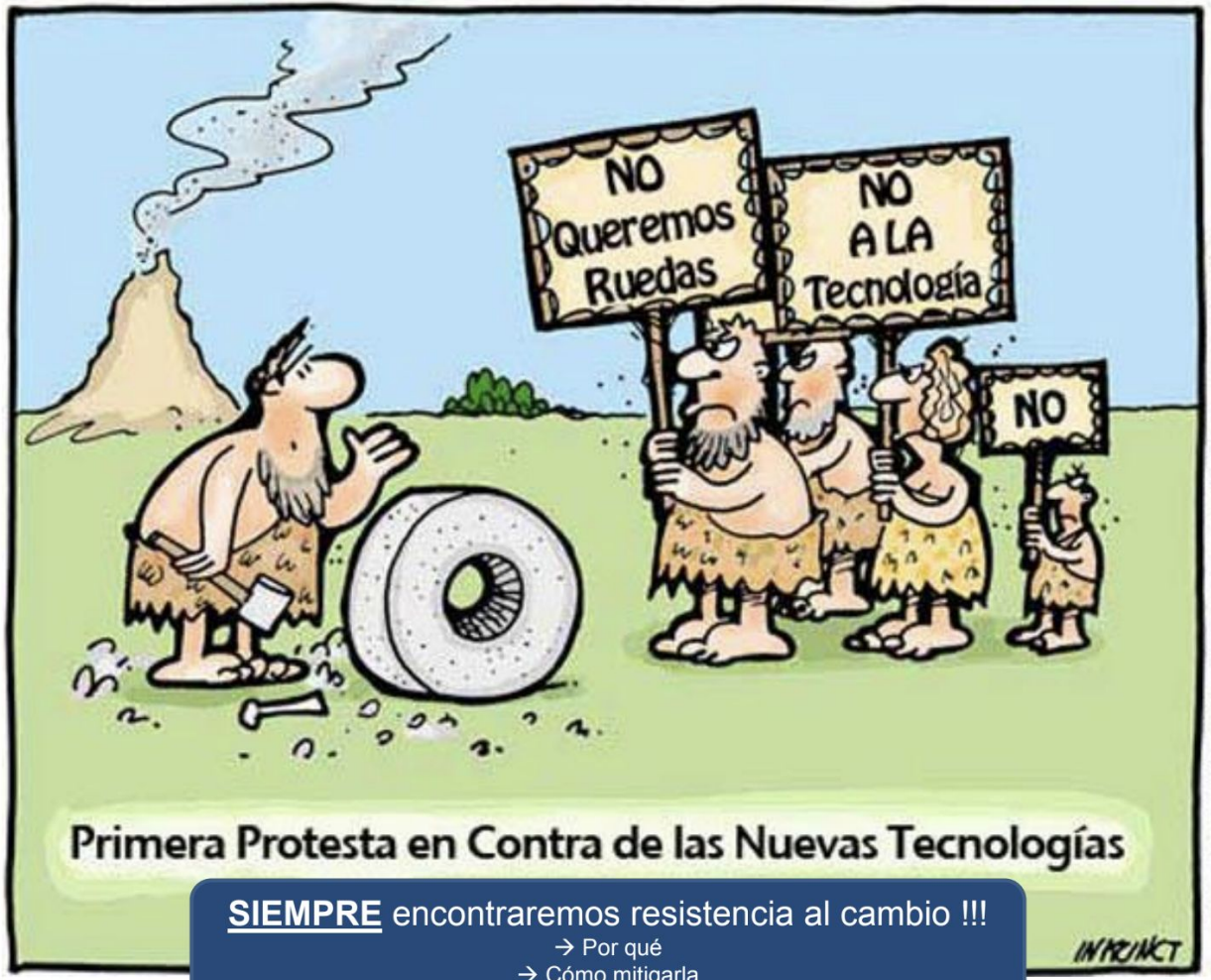


Contratar
Despedir
Evaluaciones anuales
Poner objetivos
Evaluar rendimiento
Plan formación
Plan de carrera
Aprobar vacaciones
Definir tareas
Asignar tareas al equipo
Controlar progreso tareas
Comprobar resultados
Aprobar notas de gastos
Dar bonus
Echar broncas

Hablar con otros equipos
Comunicación con dirección
Gestionar roadmap
Delegar
Controlar gastos
Habla con el cliente
Coordina dependencias
Elimina impedimentos
Gestión expectativas clientes
Gestiona desvíos
Presiona al equipo
Controla que se trabaje
Decide salarios
Cierra proyectos
Identifica y sigue riesgos

Ok, todo está mal…

¿Reconstruimos el *management*?



SIEMPRE encontraremos resistencia al cambio !!!

→ Por qué

→ Cómo mitigarla

El cambio del paradigma

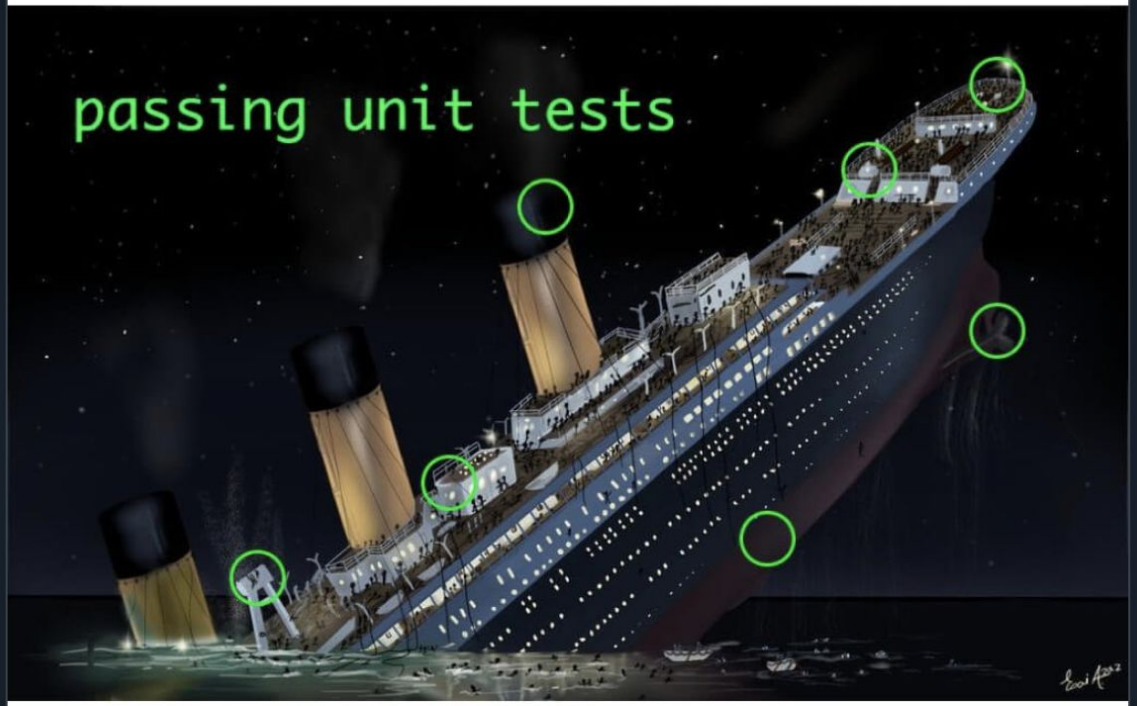
2004

El buen tester debía reportar el máximo número de defectos, aislados, reproducibles y sin investigar el **porqué**, sólo el **cómo**.

2018

El buen tester no reporta muchos defectos, trabaja con el equipo para que no se lleguen a desarrollar. Y cuando encuentra uno, su cometido es que sea la primera y la última vez que se detecte.

I Am Devloper
@iamdevloper



12 MAR 2018

Dividing frontend from backend is an antipattern

Rufus Raghunath
Software Developer and Consultant

Languages, Tools & Frameworks »

Agile Project Management »

Career Hacks »

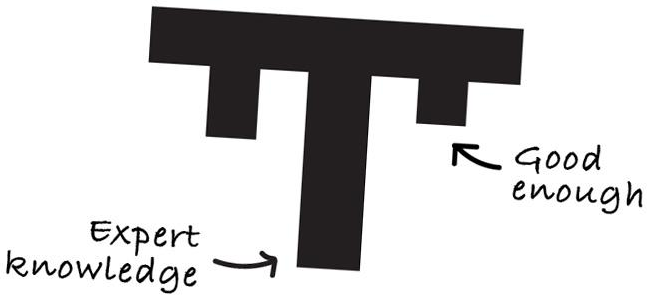
Technology »

Careers »

X

T-shaped people

DANDY PEOPLE



In cross functional teams skills are more important than roles. Team members who have a T-shaped competence profile improve the teams collaboration, delivery flow and reduce the dependency on specific individuals.

Tu haces la introducción, tu la conclusión, yo la bibliografía y mañana unimos todo eso en la universidad.

Cualquier argumento sobre calidad del código que no se resuelva en menos de 5 minutos no es un argumento sobre la calidad del código.

If we’re using **technology** X poorly and are frustrated, then moving to Y won’t help. We won’t experience frustration for a while, but only because we are distracted. All the same dysfunctions will reappear. Instead, start working better and then re-evaluate the **technology**.

Unit tests are written by programmers for programmers. Acceptance tests are specified by customers for customers. Integration tests are written by architects for architects.

Gracias,

Albert Gil Escura
@agescura
agilescura@gmail.com