

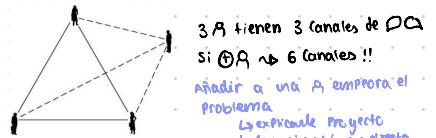


# T14: Organización de Equipos de Desarrollo de SW

## Organización de Equipo

- problema: un proyecto de 1A-facil  $\Rightarrow$  no lo hacen 3A-facil xx
- solo 1er en proyectos  $\Rightarrow$  no mucha PA & "toda la dificultad"
- código  $\rightarrow$  en horas si hay PA  
lo más es un problema de competencia técnica

## Ley de Brooks



Solución assumir q no vais a llegar  
Proteger al PPR  $\rightarrow$  no es culpa suya y si sale algo, nos beneficia a todos

## Organización Democrática

Programación sin ego  $\Rightarrow$  código

Problema El código es mío  
los errores de otros,  
quien lo toque es mi enemigo

Soluciones restituir entorno social  
reestructurar valores del programador  
animar a encontrar fallos en el código  
fallo  $\Rightarrow$  normal y aceptado  
crear identidad de grupo  
modulos  $\neq$  PPA entero  
no más de 10   
 productividad  
 investigación

¿Y si no se llega a un consenso entre todos?

¿Y si A incompetente?

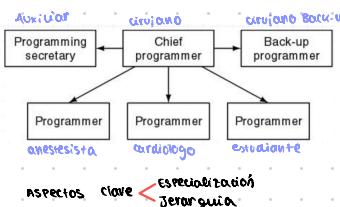
Suele haber falta de liderazgo

NO se puede imponer, se elige el PPR

## Chief Programmer Teams

centralizar la comunicación

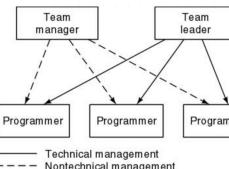
Análoga quirúrgica:



¿Y si mezclamos las dos organizaciones?  
muy inconvenientes  
chief  $\leftarrow$  manager: no reusa código  
chief  $\leftarrow$  responsable código: lo reusa

## OTRAS IDEAS

### Separar líder de manager



Y carga de gensor al programador jefe

Ventajas:

- nos hace encontrar líder a un chief programer
- Al responsable ante A Manager
- resuelve los y legal  $\rightarrow$  PPR
- líder  $\rightarrow$  también verious
- A Manager  $\rightarrow$  reuniones para enterarse

problemas:

- A Manager  $\rightarrow$  A líder
- metodología ágil

## Programador Jefe

- Buen Manager + muy cualificado
- Diseño arquitectura
- Escribe código difícil/complexo
- Maneja problemas antiguos
- Revisa trabajo otros
- Responsable de cada línea

## Programadores

- solo programa

## Programador Backup

- = skills q programador jefe  
se van turnando en los proyectos
- como mucho hace test de coja negra

## Secretario de programación

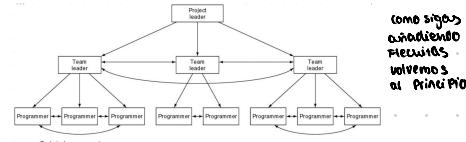
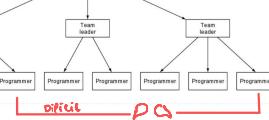
- autómatas cualificado
- documentación, compilación, rest de datos  
el software del siglo XX  
hoy todo está automatizado  
el puesto q se volviendo PPR en períodos cortos

Inconvenientes líder tiene q ser el mejor semi-dios

Back-up: no hace nada  
Secretario: solo hace el papelito

## Proyectos Grandes

Project leader  $\rightarrow$  solo gestiona, no sabe nada de los



# T14: Organización de Equipo de Desarrollo de SW

## Synchronitz-L-Stabilize teams



④ Producto → 3 o 4 BUILDs

⑤ Pequeños PPR en //

- 3-8 desarrolladores
- 3-6 testers
- PPR → tarea específica
- la desean como guion

⑥ Sincronización DIARIA

⑦ los componentes individuales A siempre trabajan juntos

⑧ Normas

↳ tener el código en la PC para el dia de la sincronización

Analogía: los niños pueden hacer lo que quieren durante el día

## Equipo de XP

### Pair Programming

Test cases

conocimiento no se pierde si un desarrollador se va

ventajas

3 desarrolladores → intercambio de conocimientos

→ "agiless programming"

experto → aprende

desarrollador → experto

NO → una solución



Depende de Productos

Depende de otros equipos.

Experiencia con ≠ estruct. equipos

poca investigación sobre desarrollo SW