



Universidad de Oviedo

ES
Español



Escuela de
Ingeniería
Informática

Rol de Arquitecto de software y *stakeholders*

2025-26



Jose Emilio Labra Gayo

Rol del arquitecto de software



Rol del arquitecto

Expectativas sobre un arquitecto

Tomar decisiones arquitectónicas

Analizar continuamente la arquitectura

Estar al día de las tendencias actuales

Asegurar cumplimiento decisiones existentes

Experiencia diversa

Conocimiento del dominio de negocio

Poseer habilidades interpersonales

Comprender y navegar en política empresarial

Leyes de arquitectura del software

Arquitecto de software es un rol, no un rango



Tomar decisiones arquitectónicas

Definir decisiones de arquitectura

Definir principios de diseño

Guiar las decisiones tecnológicas

Mantener registros de decisiones

Analizar puntos a favor y en contra



Analizar continuamente la arquitectura

Revisar continuamente tecnología y arquitectura

Ser responsable del éxito técnico del proyecto

Estar al tanto de posible deterioro estructural

Perseguir la consistencia

Organizar código en paquetes, directorios, módulos,...

Definir límites, principios, guías,...

Incluir entornos de prueba y entrega en proyectos



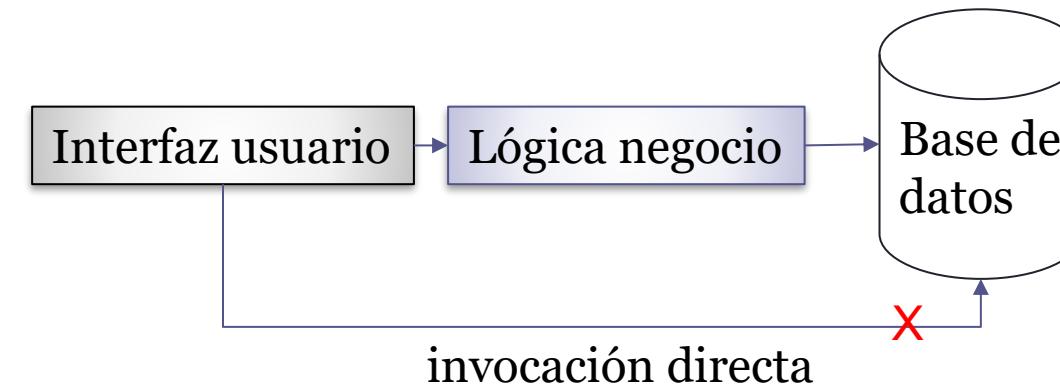
Asegurar cumplimiento de decisiones

Los arquitectos normalmente imponen restricciones

Ejemplo:

Acceso a base de datos desde interfaz de usuario

Los desarrolladores se las podrían saltar

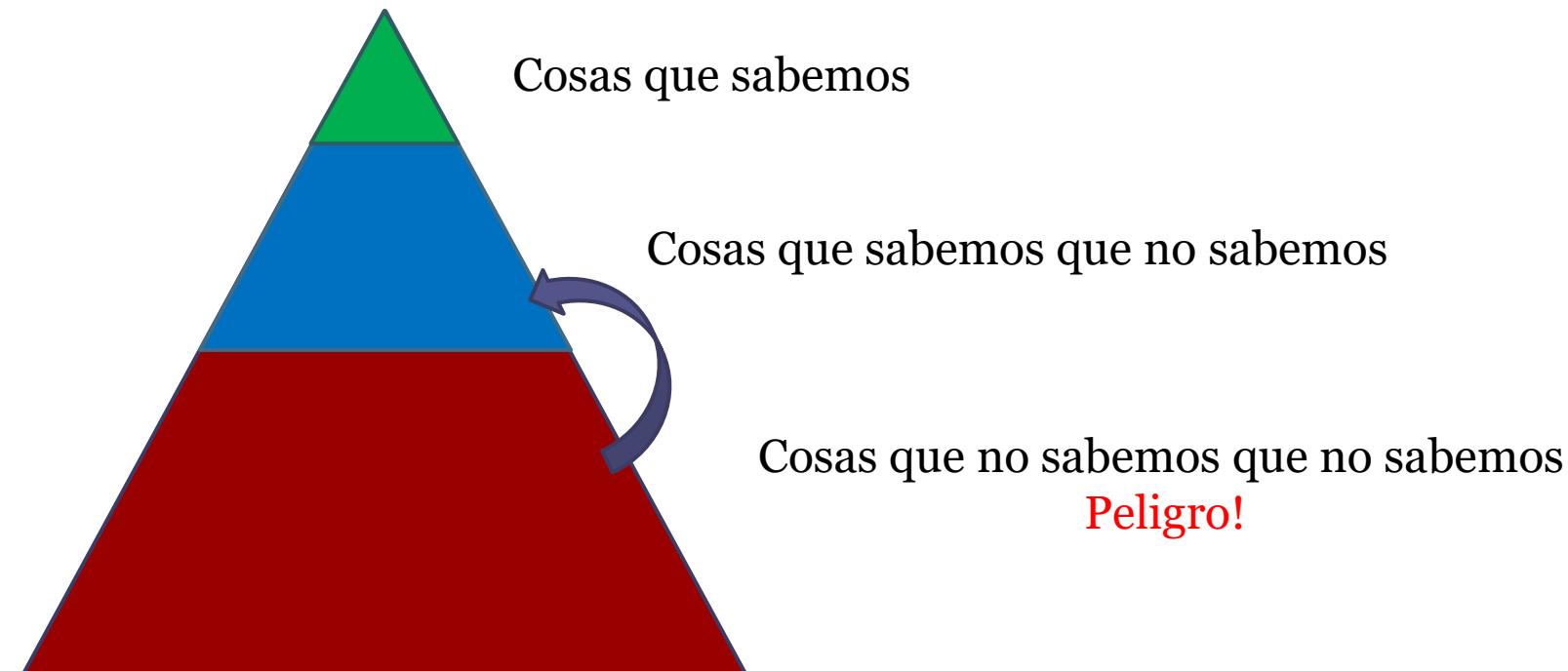


Estar al día de las tendencias

Conocer últimas tendencias tecnológicas e industriales

Las decisiones de un arquitecto pueden tener larga duración y ser muy costosas

Conocer lo que sabes y lo que sabes que no sabes



Experiencia diversa

Estar expuesto a múltiples y diversas tecnologías, marcos, plataformas, entornos, etc.

No quiere decir ser un experto en todas ellas

...pero al menos estar familiarizado con varias tecnologías

Amplitud técnica mejor que profundidad técnica



Conocimiento dominio de negocio

Se espera que el arquitecto tenga un cierto conocimiento del dominio de negocio

Comprendión del problema de negocio, los objetivos y los requisitos

Comunicarse de forma efectiva con ejecutivos y usuarios del dominio utilizando su lenguaje



Habilidades interpersonales

Arquitecto del software = líder

Habilidades de trabajo en equipo y liderazgo

Liderazgo técnico

Ser inclusivo y colaborador

Ayudar a desarrolladores a comprender la estructura general
(the big picture)

Participar en desarrollo

Formar parte de la entrega

Comprensión de bajo nivel

Codificación como parte del rol

Revisiones de código y tutorización

"no importa lo que te digan, siempre es un problema de personas", G. Weinberg



Fuente: <https://geraldmweinberg.com/>

Comprender y navegar la política

Comprender el clima político de la empresa y ser capaz de navegar la política empresarial

Decisiones arquitectónicas afectan a *stakeholders*

Dueños de producto, gestores de proyecto, personas de negocio, desarrolladores, etc.

Casi cualquier decisión tomada por un arquitecto va a ser discutida y puesta en duda

Las habilidades de negociación son necesarias

Presentar y defender la arquitectura

Ascensor del arquitecto del software

Comunicación con diferentes capas



3 preocupaciones fundamentales

Determinar soluciones de compromiso/trade-offs

Porqué hacer algo

Priorizar atributos de calidad

Cómo hacer algo

Contener la entropía

Definir estándares, convenciones, herramientas a utilizar



Trabajo en equipo

La ingeniería del software es una labor de equipo

Patrones de organización diferentes para equipos

Equipo por tecnología

Equipo por proyecto

Algunos patrones

Ocultarse y mito del genio

El factor Bus

3 pilares de interacciones sociales



Ocultarse y mito del genio

Inseguridad

Las personas tienen miedo de que otros juzguen su trabajo

Intentos de esconder el código

Mito del genio:

Tendencia a atribuir éxito de un equipo a una persona

Ejemplos: Bill Gates, Linus Torvalds, etc.

Ocultarse se considera perjudicial

El trabajo en solitario incrementa el riesgo



El factor Bus*

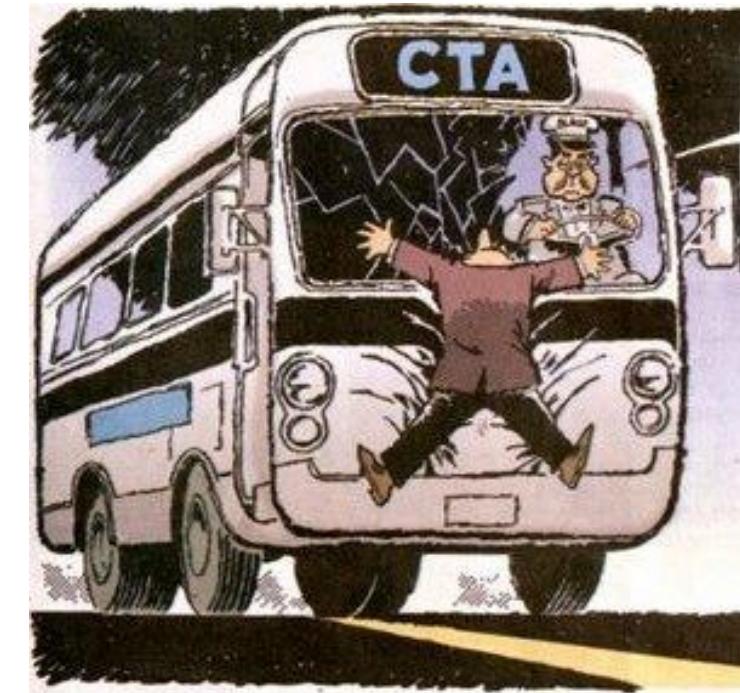
Nº de personas que tienen que ser golpeadas por un autobús para que el proyecto fracase del todo

Pueden ocurrir sucesos impredecibles

Trabajo en equipo es obligatorio para reducir riesgo

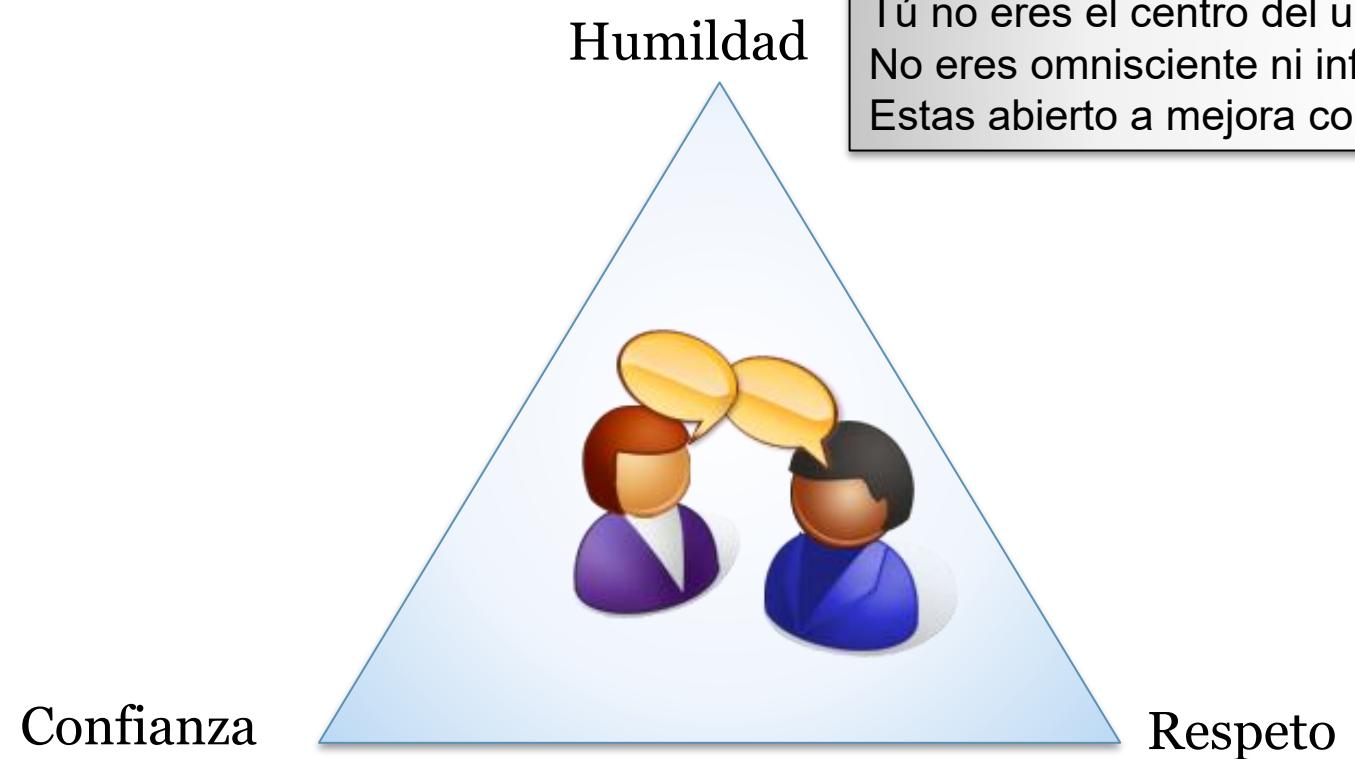
Asegurar al menos 2 personas

Buena documentación



*Término acuñado en Google (Software Engineering at Google, 2020)

3 pilares de interacción social



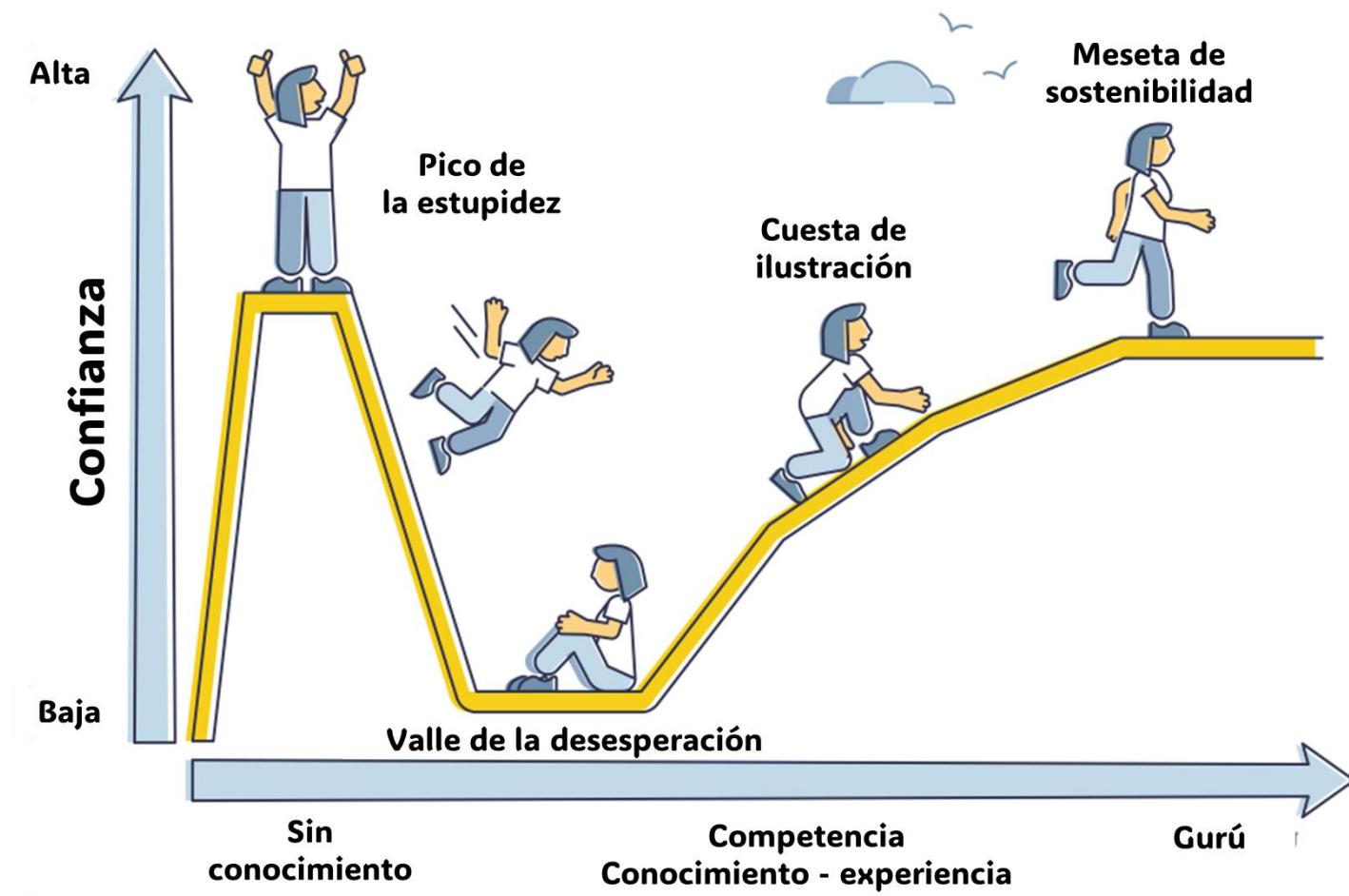
Tú no eres el centro del universo (tampoco tu código)
No eres omnisciente ni infalible
Estás abierto a mejora continua

Efecto Dunning-Kruger

Gente con poco conocimiento sobreestima su capacidad

Consecuencias:

Toma de decisiones pobre
Confianza ≠ Competencia



Personalidades del arquitecto

Arquitecto efectivo = compromiso entre
friki de control

y

arquitecto de sillón



Friki de control

Participa en todas las decisiones
Decisiones de detalle y bajo nivel
Participa en desarrollo de código (cuello de botella)



Arquitecto de sillón

Desconectado de desarrolladores
Nunca está (salta de proyecto en proyecto)
Solo participa en diagramas iniciales

Topologías de equipos

Las topologías de los equipos afectan a los sistemas

Estructuras de comunicación

Dinámica de equipos

Tamaño de los equipos

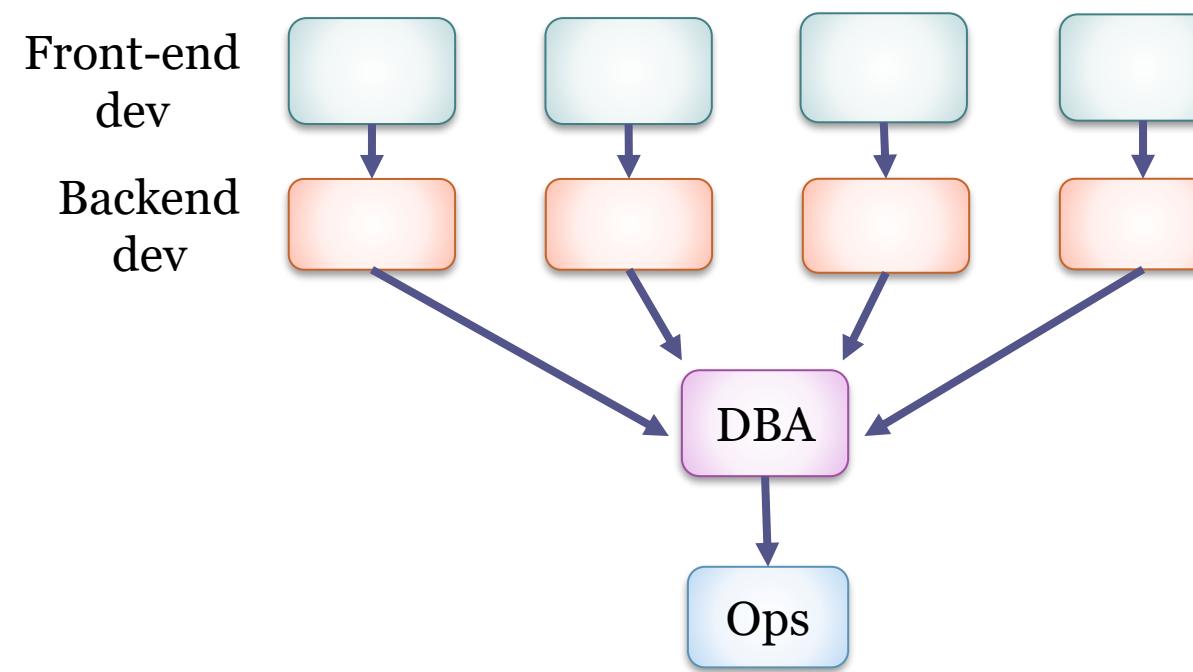
"La asignación de equipos es el primer borrador de la arquitectura", M. Nygaard

Topología tradicional de equipos

Disposición de trabajo tradicional:

Equipos existentes son asignados a cada nuevo proyecto

Ejemplo: 4 equipos: *front-end*, *back-end*, DBA y *Operations*



Ley de Conway

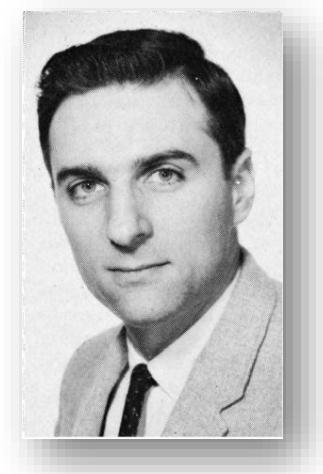
Las organizaciones que diseñan sistemas...están abocadas a producir diseños que son copias de las estructuras de estas organizaciones [M. Conway, 1967]

Corolario:

La mejor estructura de un sistema está influenciada por la estructura social de la organización

Ejemplo:

Si hay 3 equipos (diseño, programación, bases de datos), el sistema tendrá de forma natural 3 módulos

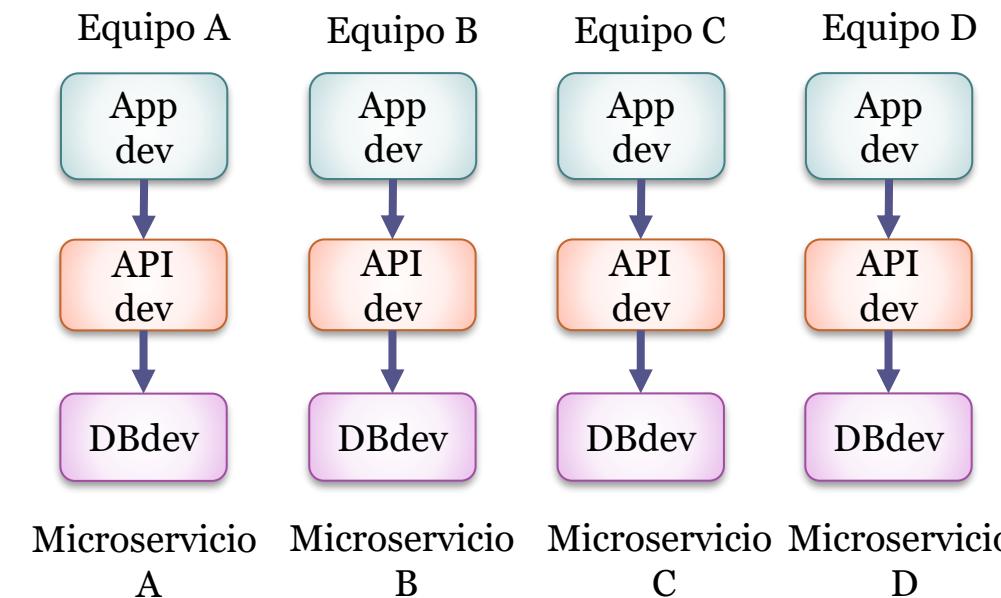


Maniobra inversa de Conway

Evolucionar equipos y estructura organizativa para promover la arquitectura deseada

Crear equipos después de la descomposición modular

Ejemplo con microservicios



Principio de Amazon: Tú lo construyes, tú lo ejecutas

Tamaño de equipo

Tamaño eficiente de un equipo influencia el éxito del proyecto

Algunos avisos a tener en cuenta:

Pérdida por proceso

Ignorancia colectiva

Difusión de responsabilidad

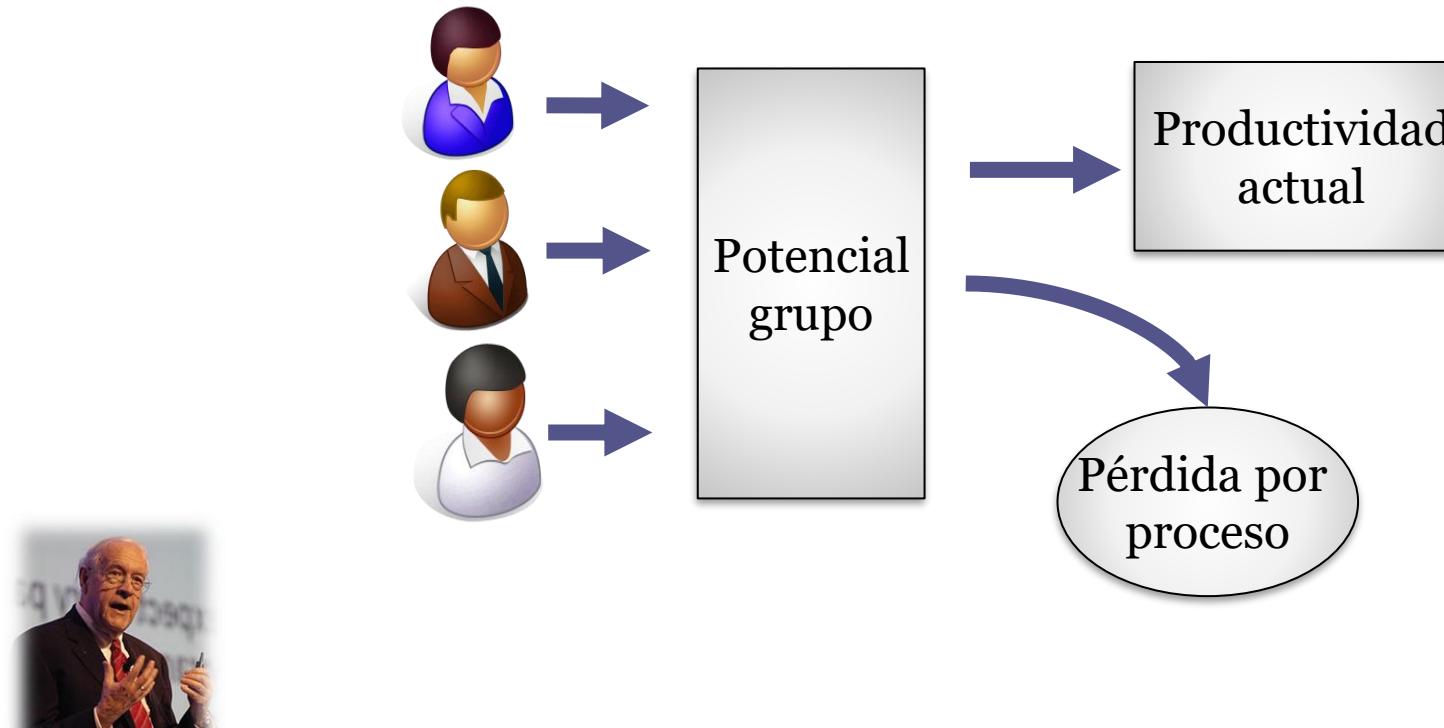


Regla de 2-pizzas: "si no puedes dar de comer a tu equipo con 2 pizzas, entonces es muy grande", J. Bezos

Pérdida por proceso

Diferencia entre potencial de grupo y productividad actual

Razones: sobrecarga comunicación, reuniones,...



Ley de Brook. Añadir más personas a un equipo que va retrasado hace que se retrase más todavía

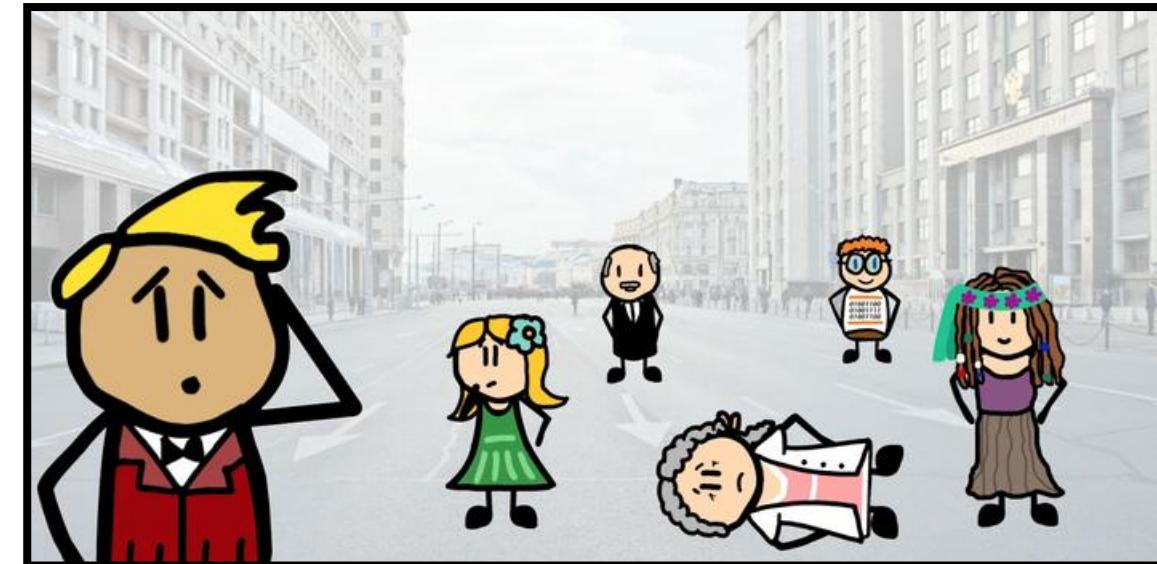
Difusión de responsabilidad

Tamaño de equipo grande impacta la comunicación

Algunas señales:

Confusión sobre quién es responsable de qué

Cosas que quedan abandonadas



Ignorancia colectiva

Cuando todo el mundo está públicamente de acuerdo en algo y privadamente lo rechazan pero creen que hay algo obvio que no entienden

Decisiones arquitectónicas no cuestionadas



Fábula del nuevo traje del Emperador

<http://fablesfairytaleandsocialjustice.weebly.com/the-emperors-new-clothes.html>

Pensamiento de grupo (*groupthink*)

El deseo de armonía de grupo lleva a decisiones irracionales

Cada miembro adapta su opinión a lo que cree que es el consenso del grupo

El grupo decide una acción que cada miembro individualmente considera desaconsejable

Algunas causas

Aislamiento del grupo

Alta cohesión

Liderazgo fuerte,

Técnicas de prevención

Incluir Abogado del diablo
o *regla del décimo hombre*



Apoyarse en listas de control

Listas de control (*checklists*) = método efectivo para asegurar que algunas tareas son realizadas o cubiertas

Tareas propensas a error o que se olvidan frecuentemente

Hacer equipos más efectivos



Efecto Hawthorne: Si la gente sabe que están siendo observados, entonces cambian su comportamiento para hacer las cosas adecuadas

Stakeholders Personas interesadas



Stakeholders



Todas las partes que participan en el desarrollo o son afectadas por el sistema

Puede ser una persona, rol u organización

Normalmente tienen preocupaciones diferentes

Algunas veces contradictorias

Es necesario

Comprender la naturaleza, fuente y prioridad de sus preocupaciones

Identificar y comprometerse con ellos

Solicitar sus necesidades y expectativas

Los *Stakeholders* manejan (explicita o implicitamente) la forma y dirección de la arquitectura para que sirva a sus necesidades

Identificando *stakeholders*

Todos los individuos, roles u organizaciones que:

Deberían conocer la arquitectura

Tienen que ser convencidos de la arquitectura

Tienen que trabajar con la arquitectura o el código

Necesitan la documentación de la arquitectura para realizar su trabajo

Tienen que tomar decisiones sobre el sistema o su desarrollo



Identificando stakeholders

Internos

Analista
Diseñador
Personas de negocios
Desarrollador
Product owner
Diseñador de UX
Jefe de proyecto

...

Externos

Cliente
Usuarios finales
Auditor
Autoridades públicas
Suministradores
Proveedores servicios externos

...

Expectativas de *stakeholders*

Las expectativas ayudan a:

Identificar necesidades específicas

Objetivo: alcanzar mayor satisfacción de la audiencia

Evitar trabajo innecesario

Evitar documentar cosas irrelevantes

Formato típico:

Role/nombre	Contacto	Expectativas

Mapa de stakeholders

Mostrar personas/roles relacionados

Incluir relaciones e interacciones

Ejemplo Sistema automatización licitaciones (*)



Declarar objetivos de negocios

Objetivos de negocios centrados en personas

Normalmente entre 3 ó 5

Persona/stakeholder

Resultado: expresa la necesidad como algo medible

Cómo cambiaría el mundo si el Sistema tiene éxito?

Contexto

Alguna aclaración sobre el objetivo

Persona/stakeholder	Resultado	Contexto
Alcalde de la ciudad	Reducir costes 30%	Evitar recortes de los presupuestos para servicios esenciales
Oficina de gestión	Revisar datos de licitaciones históricos de los últimos 30 años	Los datos históricos pueden ayudar a predecir contratos futuros

Fin