

# La Realidad de un Proyecto

# Proceso clásico de desarrollo de software

- El típico proceso de desarrollo de software consta de las siguientes fases:
  - 1. Conceptualización y captura de requisitos
  - 2. Análisis y Descripción funcional
  - 3. Diseño
  - 4. Codificación
  - 5. Pruebas
  - 6. Distribución

# ¿La realidad de un proyecto?



## Toma de requisitos



Como el usuario  
ha expresado sus  
necesidades



La descripción del  
comercial

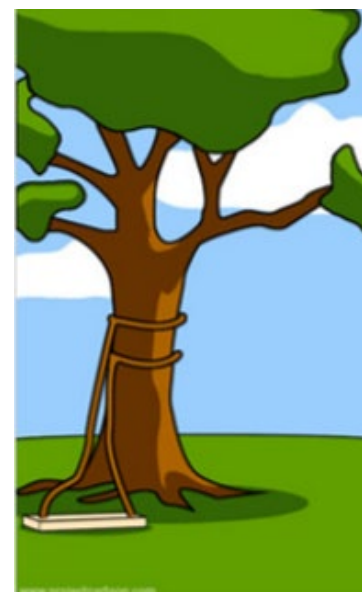
## Ejecución



Como el jefe del  
proyecto lo  
entendió



Como ha sido  
diseñado



Como ha sido  
desarrollado

## Resultado



Lo que los  
usuarios han  
recibido



Como se ha  
comportado a la  
puesta en marcha



Como se  
aportaron las  
correcciones

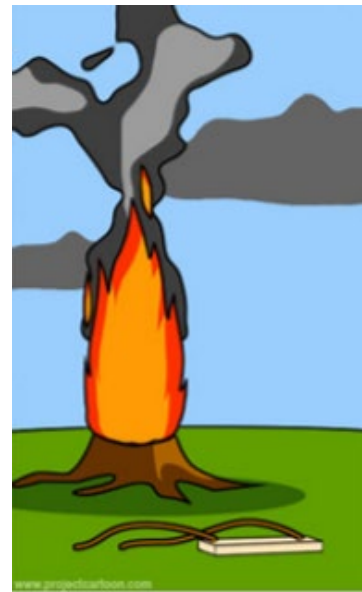
# Gestión



Cuando ha sido  
entregado

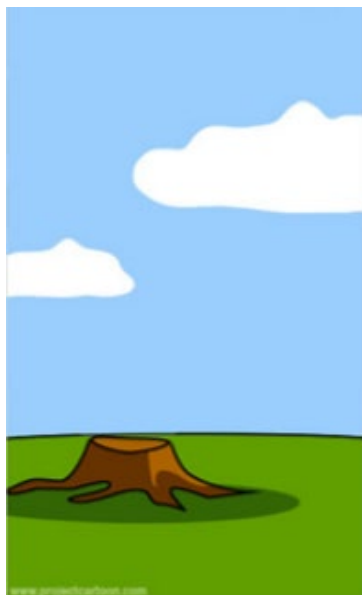


El presupuesto  
del proyecto

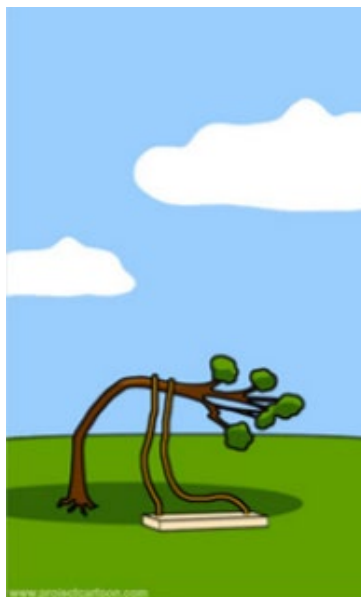


El impacto sobre  
la organización

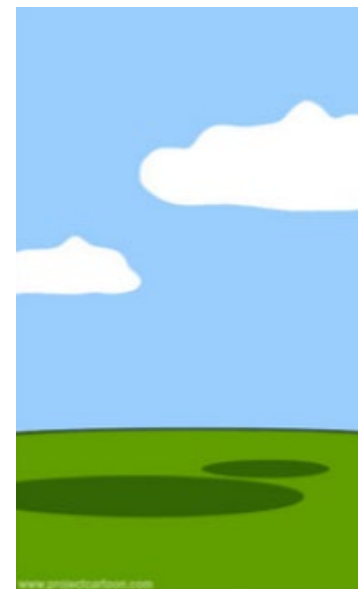
## Post-proyecto



La organización  
del soporte  
técnico



El plan de  
contingencias



La documentación  
del proyecto



Y al final...



Lo que el usuario  
realmente necesitaba

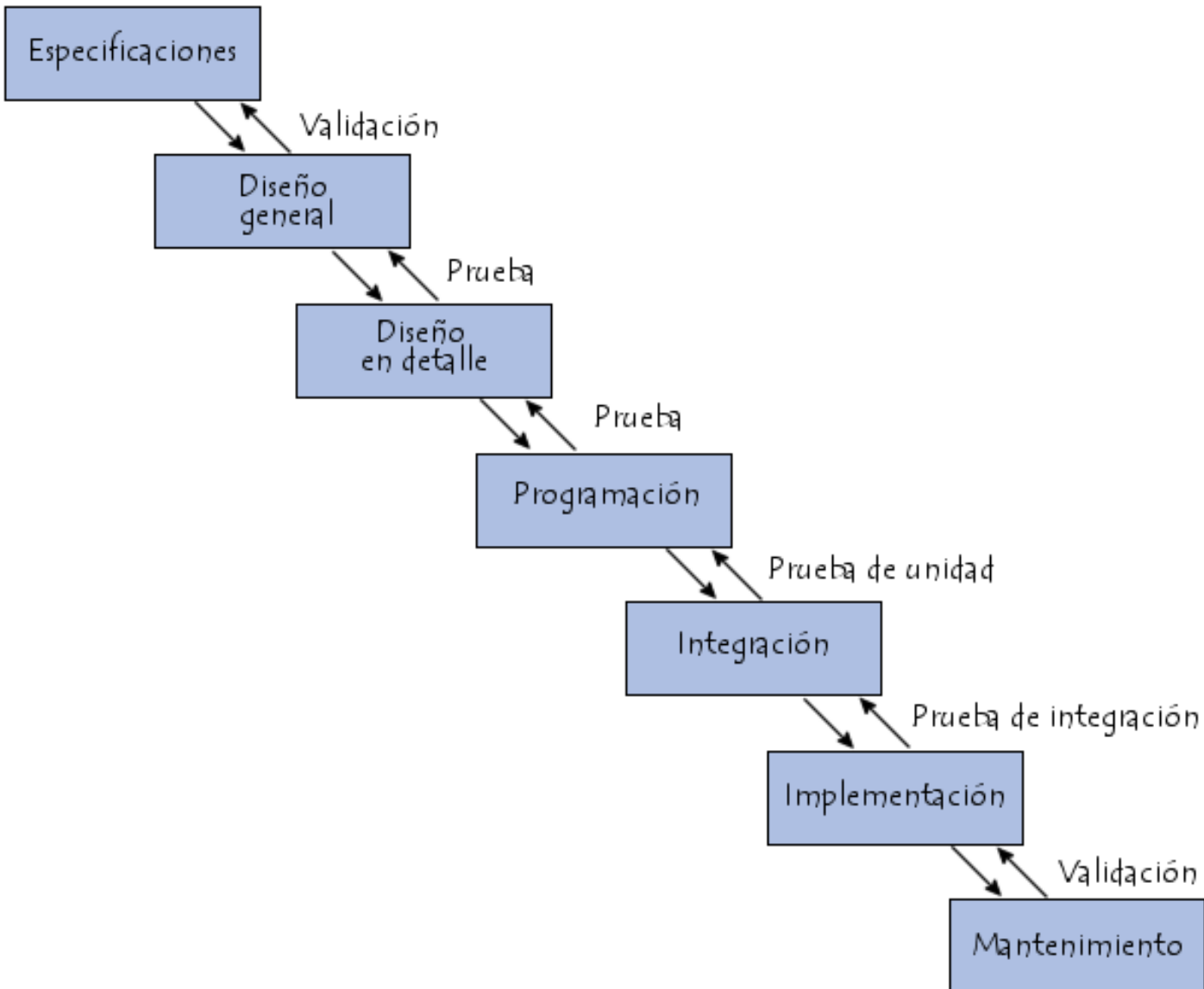


ENTRE LO QUE PIENSO,  
LO QUE QUIERO DECIR,  
LO QUE CREO DECIR.  
LO QUE DIGO,  
LO QUE QUIERES OÍR,  
LO QUE OYES  
LO QUE CREES ENTENDER,  
LO QUE QUIERES ENTENDER,  
LO QUE ENTIENDES...

EXISTEN 9 POSIBILIDADES  
DE NO ENTENDERSE.

# Modelo en cascada

- El ciclo de vida inicialmente propuesto por Royce en 1970, fue adaptado para el software a partir de ciclos de vida de otras ramas de la ingeniería.
- Es el primero de los propuestos y el más ampliamente seguido por las organizaciones (se estima que el 90% de los sistemas han sido desarrollados así).
- “Waterfall”



## Características

- Este modelo admite la posibilidad de hacer iteraciones
  - Durante el mantenimiento se puede ver la necesidad de cambiar algo en el diseño.
  - Eso significa que se harán los cambios necesarios en la codificación y se tendrán que realizar de nuevo las pruebas.
  - Es decir, si se tiene que volver a una de las etapas anteriores al mantenimiento hay que recorrer de nuevo el resto de las etapas.
- Después de cada etapa se realiza una revisión para comprobar si se puede pasar a la siguiente
- Trabaja en base a documentos, es decir, la entrada y la salida de cada fase es un tipo de documento específico.
- Idealmente, cada fase podría hacerla un equipo diferente gracias a la documentación generada entre las fases.

## Ventajas

- La planificación es sencilla
- La calidad del producto resultante es alta
- Permite trabajar con personal poco cualificado

## Inconvenientes

- Lo peor es la necesidad de tener todos los requisitos al principio. Y lo normal es que el cliente no tenga perfectamente definidas las especificaciones del sistema, o puede ser que surjan necesidades imprevistas.
- Si se han cometido errores en una fase es difícil volver atrás.
- No se tiene el producto hasta el final, esto quiere decir que:
  - Si se comete un error en la fase de análisis no lo descubrimos hasta la entrega, con el consiguiente gasto inútil de recursos.
  - El cliente no verá resultados hasta el final, con lo que puede impacientarse .
  - Es comparativamente más lento que los demás y el coste es mayor también.

## Tipos de proyectos para los que es adecuado

- Aquellos para los que se dispone de todas las especificaciones desde el principio, por ejemplo, los de reingeniería
- Se está desarrollando un tipo de producto que no es novedoso
- Proyectos simples que se entienden bien desde el principio



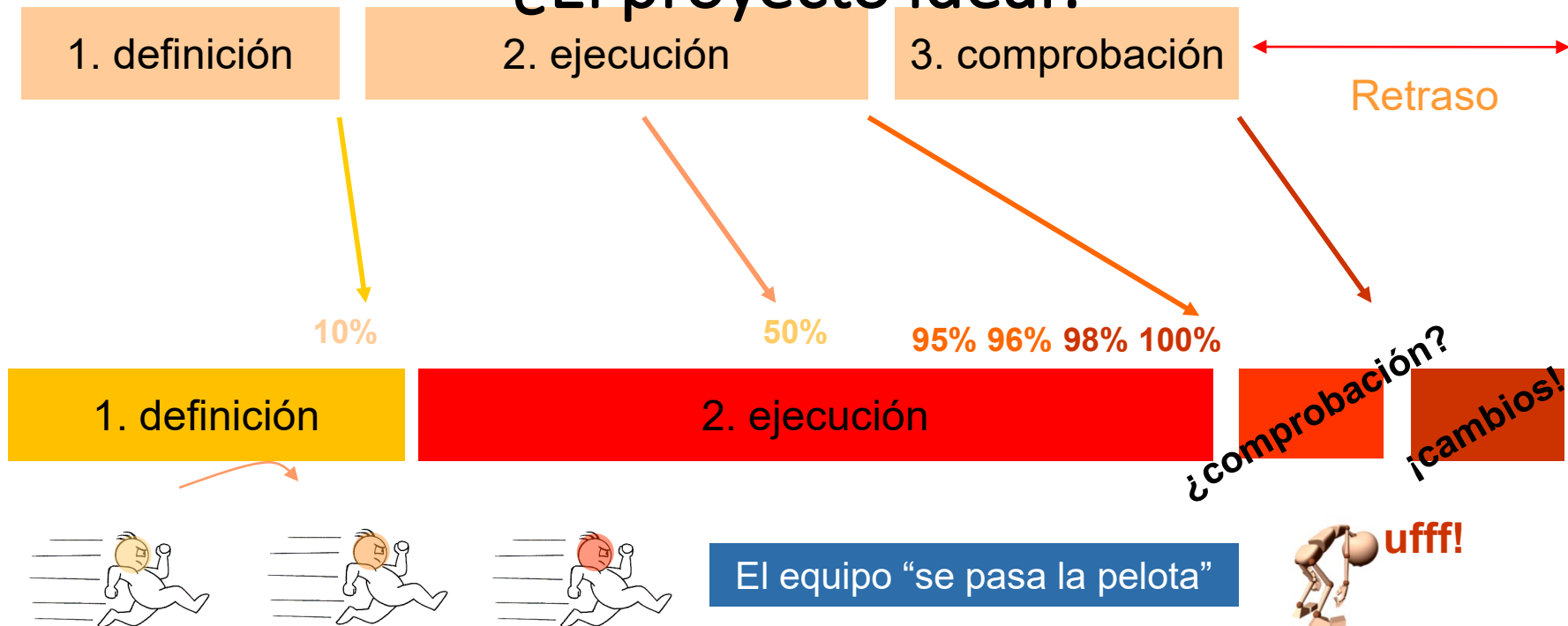
## El proyecto ideal



## ¿El proyecto ideal?



# ¿El proyecto ideal?



El equipo "se pasa la pelota"

ufff!



El proyecto **se complicó más de lo esperado**

Hay **retraso** y hay que entregar ya. Empiezan los **parches** y **no hay tiempo para pruebas** / control de calidad

El cliente tarda mucho tiempo en poder utilizar el resultado del proyecto

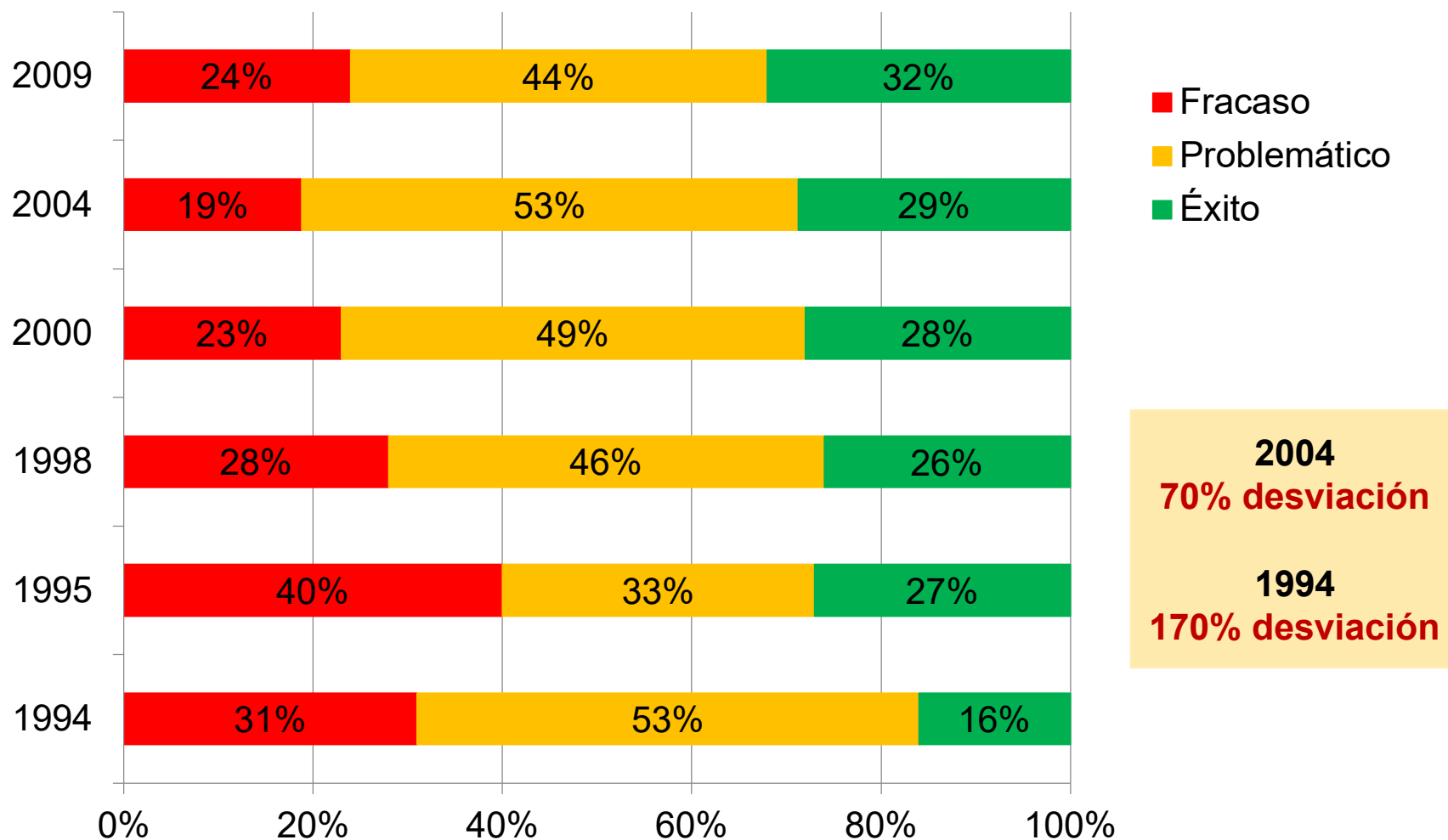
## ¿Proyecto ideal?

El equipo hace horas extraordinarias y está **poco motivado**

Apenas se dedican a cumplir órdenes. Cada uno hace sólo lo suyo

Mientras tanto, el contexto cambia y los competidores lanzan nuevos productos. **Si se cancela el proyecto** se habrá gastado el dinero a cambio de **NADA**

## Proyectos para desarrollo de sistemas de software



# ¿Por qué fracasan los proyectos de software?

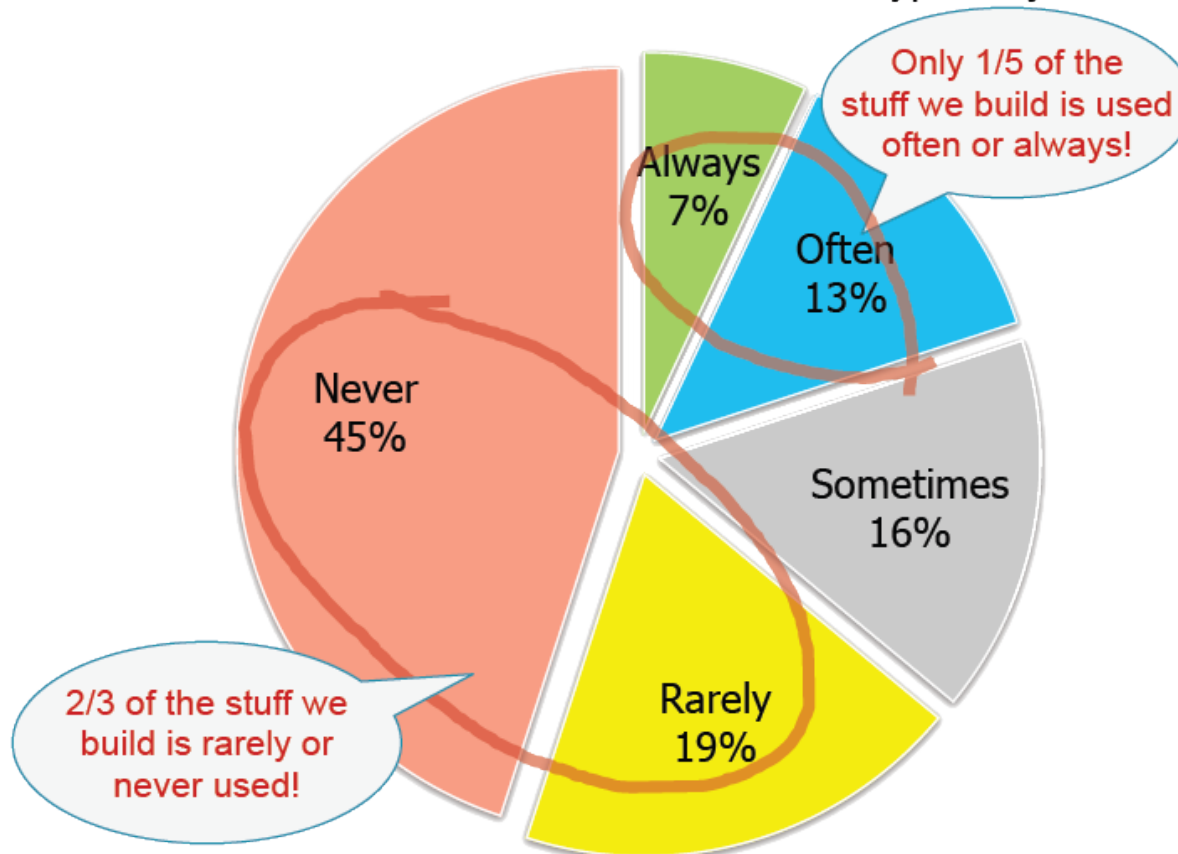
- Retrasos y desviaciones en la planificación
- Cancelación por coste o demora
- Defectos y fallos
- Requisitos mal comprendidos
- Cambios de negocio
- Falsa riqueza de características
- Cambios de personal





Dedicamos mucho esfuerzo a actividades que aportan poco valor

Features and functions used in a typical system:



Source: Standish Group Study Reported at XP2002  
by Jim Johnson, Chairman

# Triangulo de Hierro





# Fiabilidad de la planificación

- 1 característica acabada en X tiempo = Probabilidad 80%
- Varias: encadenamos probabilidades...
- Probabilidad final: BAJA

# Fracasos sonados: LIDL - SAP

**LIDL abandona la implantación del ERP de SAP tras siete años y una inversión de 500 Millones de euros.**

- El proyecto era muy ambicioso y las expectativas muy grandes. El objetivo era implementar un sistema que permitiera controlar todas las filiales y al mismo tiempo mejorar, simplificar y monitorear los procesos de compras y logística de la Compañía.
- Sin embargo, tras **siete años** de implantación, una serie de problemas persistentes en el sistema han hecho que los procesos empresariales sean más engorrosos y menos eficientes, es decir todo lo contrario de lo que era el objetivo del proyecto.
- Un fracaso que le ha costado a la cadena low-cost alemana 500 Millones de euros.

# Y un ejercicio en grupo...

# EJERCICIO

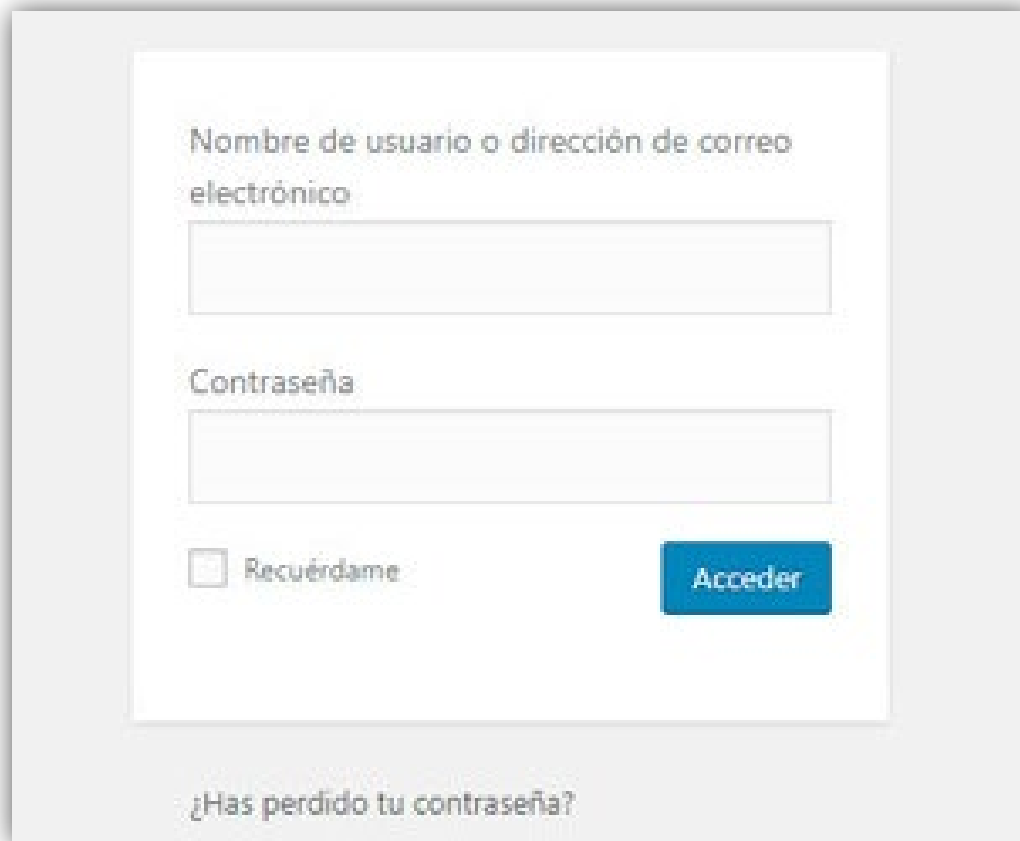
## Login – Valor = 3

Nombre de usuario o dirección de correo electrónico

Contraseña

Acceder

# Login + recordar + Olvidada Valor = ?



Nombre de usuario o dirección de correo electrónico

Contraseña

☐ Recuérdame

Acceder

¿Has perdido tu contraseña?