#### Diseño de software y modularidad

Joaquín Hermida Giganto

Pablo Calvo Gamonal

Raúl Álvarez Álvarez



#### Contenidos

Cuándo diseñar

La complejidad

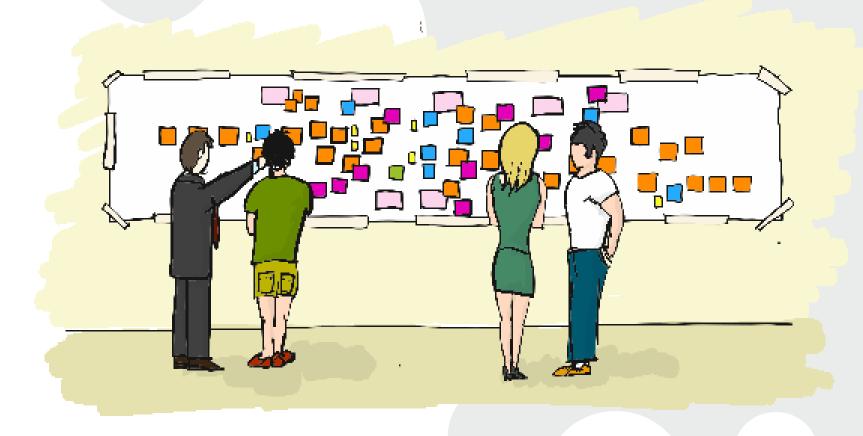
Principios de diseño

Red Flags

# Cuándo diseñar



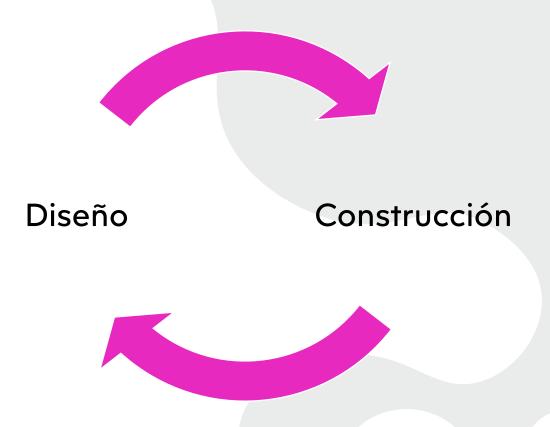
## Big Design Up Front



### No diseñar, "design by debugging"



#### Proceso iterativo





#### La complejidad

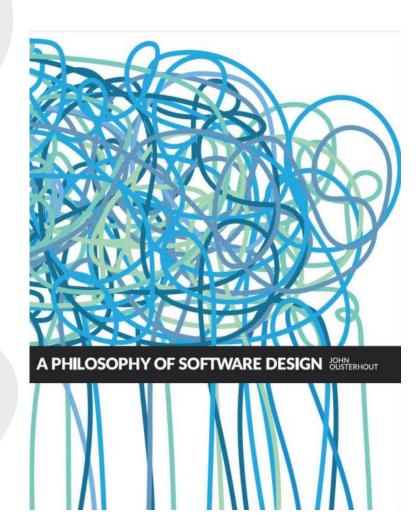


QUÉ ES



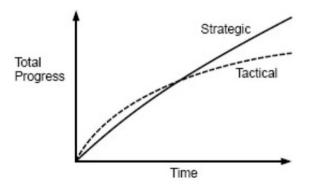
POR QUÉ ES IMPORTANTE

## Principios de diseño



#### Principios de diseño

- 1. La complejidad es incremental
- 2. Código que funcione no es suficiente
- 3. Hacer pequeños incrementos continuos para mejorar el diseño del sistema



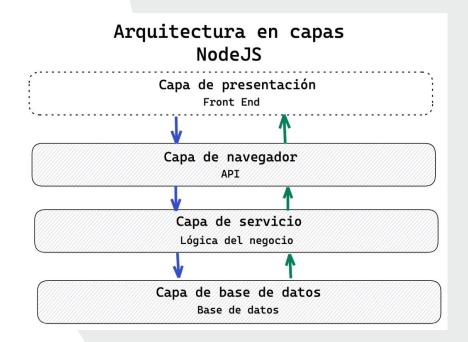
#### Principios de diseño II

- 4. Los módulos deben ser profundos
- 5. Las interfaces han de ser diseñadas para hacer que el uso más común sea lo más simple posible.
- 6. Es más importante que un módulo tenga una interfaz simple que una implementación simple.



#### Principios de diseño III

- 7. Los módulos de propósito general son más profundos.
- 8. Separar código de propósito general y el de propósito específico.
- 9. Diferentes capas deben de tener diferentes abstracciones.



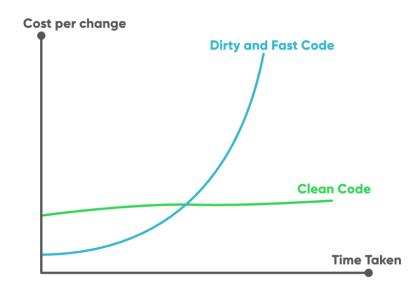
## Principios de diseño IV

"Selected" + str(modified) bpy.context.selected\_obj mint("please select exaction OPERATOR CLASSES --ontext):
oxt.active\_object is not

- 10. Arrastrar complejidad.
- 11. Definir errores y casos especiales para eliminarlos.
- 12. Diseñarlo dos veces.

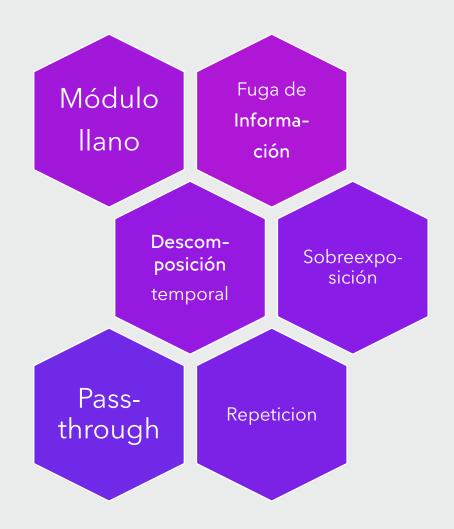
#### Principios de diseño V

- 13. Comentarios deben describir cosas que no son obvias en el código.
- 14. Software debería ser diseñado con la idea de facilidad de lectura, no de escritura.
- 15. Los incrementos del desarrollo deben de ser abstracciones, no funcionalidades.



#### Red flags en el diseño





#### Mezcla Métodos Especialconjuntos general Documen-Comentarios | tación repetidos impl. Nombres Código no difíciles / obvio impreciso

#### Red flags en el diseño





# Preguntas