**Preguntas**

**¿Quiénes de ustedes no conocen nada de TDD?**

**¿Quiénes lo aplican en su trabajo diario?**

--------------------------------------------------------------------------

DISCLAIMER: I am *not* saying "don't do any up-front design at all". This is an exercise to stretch your refactoring and feedback-driven design muscles.

* Let TDD drive your design
* Doing TDD will help you archive other solutions for your problem instead of the predefined one in your head



Continuous Testing.- Delegar toda la responsabilidad de probar a QA, Yo voy aprobar las partes de mi código que no funcionan luego de haberlo terminado, yo voy a probar las partes de mi código a medida que las voy escribiendo utilizando el debugger. Escribiré pruebas para todos lo que escriba. Escribiré las pruebas antes de escribir mi código. == Test First Programming (hago todo como lo venía haciendo antes, sigo haciendo mi diseño upfront, con la diferencia de que hago los tests antes). Es muy útil ya que permite tener tu código bajo control a través de una suite de test de regresión que los puedes ejecutar en cualquier momento.

De TFP a TDD es gradualmente hacer menos Design up front, cada vez hacer tu primer borrador de tu diseño más pequeño y más pequeño hasta que solo las cosas vítales (arquitectónicas) tengan que diseñarse de manera anticipada.

TDD as “testing” es el primer paso para entender tdd como una actividad de“diseño”. Primero logras tener tus defectos bajo control y tener la confianza de que tu sistema está libre de defectos, es decir tu restricción ya no es tu calidad sino el tiempo de feedback en el cuál verificas tu diseño.