

# Mutation Testing

Universidad Autónoma de Coahuila

Facultad de Sistemas

Calidad y Pruebas de Software

Carlos Nassif Trejo García



---

Dick Lipton

---

1970s

---

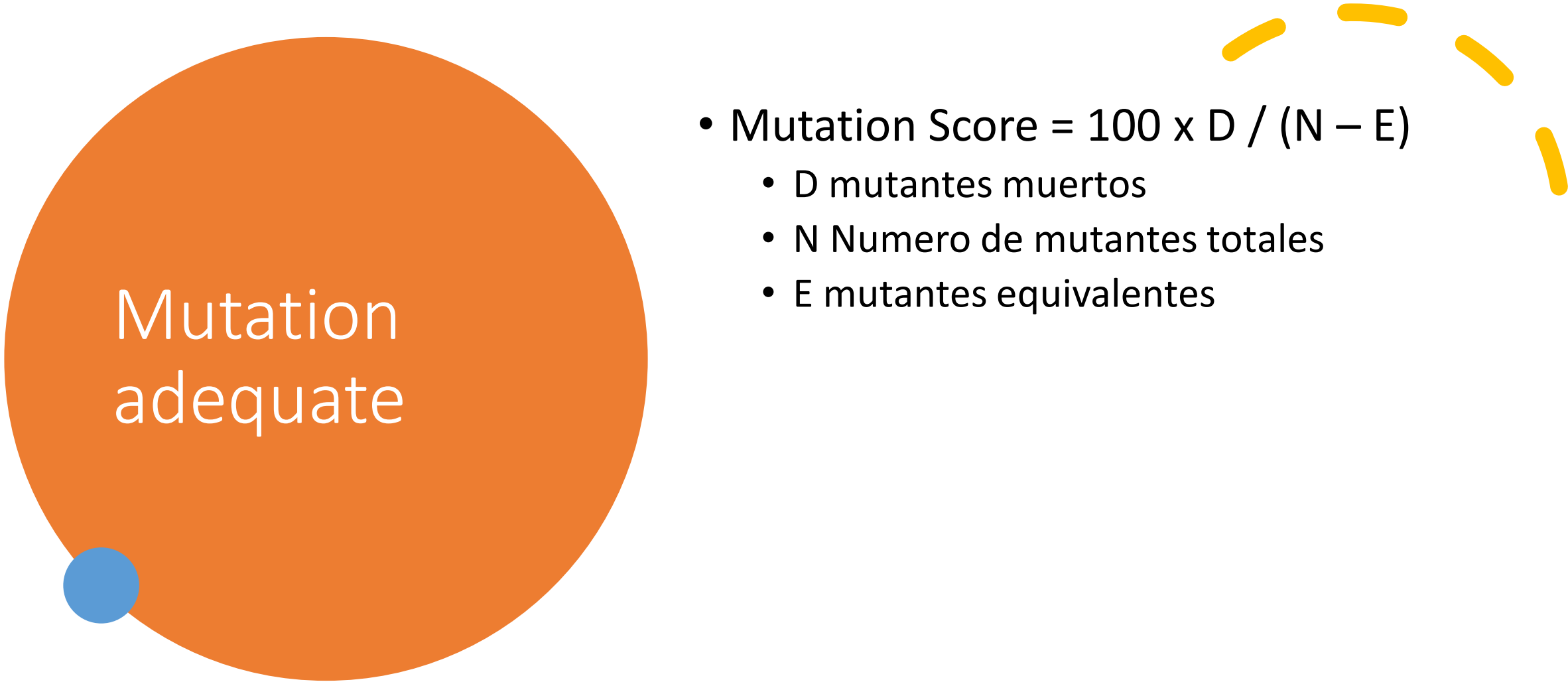
Medir la calidad de  
los casos de prueba

Mutantes

Vivo

- Equivalente
- Matable

Muerto



# Mutation adequate

- Mutation Score =  $100 \times D / (N - E)$ 
  - D mutantes muertos
  - N Numero de mutantes totales
  - E mutantes equivalentes

# Mutation Analysis

Un banco de pruebas es determinado para distinguir entre el programa original y sus mutantes

Nuevos casos de prueba son creados para matar a los “matables”

El proceso se repite hasta tener una mutation score deseable



# Pasos

1. Programa P, casos de prueba T correctos
2. Correr cada T al programa P y asegurarse que pasen
3. Crear mutantes  $\{P_i\}$
4. Ejecutar cada T a cada mutante  $P_i$ 
  1. No pasa = Muerto
  2. Pasa = Equivalente o matable
5. Calcular el “mutation score”
  1. Mutation score bajo = Crear nuevos casos de prueba e ir al paso 2
  2. Score alto = Terminar

# Suposiciones

- Hipotesis del programador competente: El programador no hace programas al azar
- Efecto de acoplamiento: Si un software tiene **fallas**, usualmente habrá un par de mutantes que solo podrán ser asesinados por casos de prueba que también detecte la **falla**

