CI3715

Taller 3

Profesores:

- Alfonso Reinoza: jareinozacg@gmail.com

Agenda

- **Equipo**
  - **Programación por pares**:
    - **Que es?**
    - **Que aspectos afectivos influyen?**
    - **Experiencia**
    - **Testimonio de Leonardo Martínez**
- Verificación
  - TDD en PyUnit.

Programac... Es...

Un estilo de programación en la que dos programadores trabajan codo con codo en equipo , colaborando continuamente en el mismo diseño , algoritmo , código o prueba.

Programación por pares

ROTACION

Programación por pares
Preludio

experto, ta.
1. adj. Práctico, hábil, experimentado.

novato, ta.
1. adj. Nuevo o principiante en cualquier facultad o materia

promedio.
1. m. Punto en que algo se divide por mitad o casi por la mitad.

Extrovertido
1.Se aplica la a persona que tiene facilidad para manifestar sus sentimientos y para relacionarse con los demás.

Introvertido
1.Se aplica a la persona que no suele manifestar sus sentimientos y se relaciona poco con los demás.

Programación por pares
Preludio

Pareja (DRAE)

- (Del lat. **paricŭlus*, dim. de *par, paris*, igual).
- **1.** adj. Igual o semejante.
- **4.** f. Cada una de estas personas, animales o cosas considerada en relación con la otra.

Emparejar (DRAE)

- **1.** tr. Juntar dos personas, animales o cosas formando pareja. U. t. c. prnl.
- **2.** tr. Unir las personas o animales de distinto sexo formando pareja. U. m. c. prnl.
- **3.** tr. Poner algo a nivel con otra cosa.
- **9.** intr. Dicho de una persona: Ponerse al nivel de otra más avanzada en un estudio o tarea.

Principio de selección de parejas
¿Emparejarse?

Experto- Experto

Intención:"Cuando los dos expertos se ponen en sintonía, se puede oír el crepitar del rayo. Trabajar con un buen socio experto es como ganar 40 o más puntos de CI.“

Características de Éxito: Como Aretha Franklin canta en la canción "Respect", "RESPECT”.

Retos: Egos

Experto-Promedio

Intención: Hacer el trabajo de complejidad media bien hecho, mientras se mejora el nivel de habilidad de un programador.

Características de Éxito: Tiene que llegar a ser claro para el experto que el programador medio está "recibiendo", de lo contrario el experto será frustrado.

Retos: Tres situaciones

- programador promedio realmente es promedio.
- programadores promedio no interactúan bastante con el experto
- programador medio no parece “asimilar ".

Experto-Novato

Intención: Hacer bien el trabajo más fácil, mientras que se forma un programador novato.

Características de Éxito: El experto debe comprender que "En realidad no lo sabemos hasta que tenemos que enseñarlo.". El novato hace varias preguntas y, a menudo hace la pregunta más importante: "¿Por qué?"

Retos: Claramente, la vinculación a un experto con un novato requiere que el experto sea capaz de interactuar con los novatos como un maestro con sus estudiante. Paciencia, la voluntad de explicar, la capacidad de articular claramente el trabajo que se está haciendo, la compasión por el estudiante, y, sobre todo, paciencia.

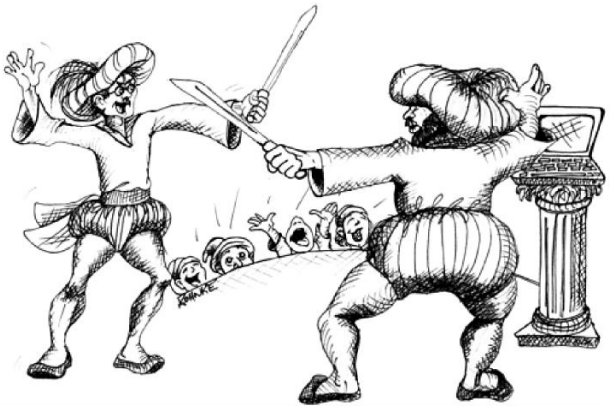
Novato-Novato

Intención: Programar el código de producción en un área relativamente no compleja del proyecto, dando una valiosa experiencia a ambos programadores.

Características de Éxito: (Sin vergüenza) Si ambos no saben algo, tienen que pedir ayuda rápidamente, obtener la respuesta, y seguir adelante. Fuerte advertencia: tiene que haber un entrenador, instructor o tutor disponible para responder preguntas y también para ayudar a guiar a la pareja.

Retos: Irse por un camino equivocado ... Sin embargo un par de novatos es una mejor opción a un novato en solitario

Extrovertido-Extrovertido



Intención: Después de largas reflexiones y discusiones constructivas, se crea una solución creativa excelente.

Características de Éxito: Todo sobre emparejar es: buscar la comunicación y los extrovertidos generalmente se encuentran entre los mejores comunicadores

Retos: Las parejas extrovertidas podrían dedicar mucho de su tiempo a hablar, discutir y argumentar. Si ellos no son productivos, el emparejamiento falla. Por lo tanto es importante que el gerente observe estos equipos extrovertidos y asegurarse de que se están produciendo buenos resultados.

Extrovertido-Introvertido

Intención: Permitir a cada socio explotar sus puntos fuertes y mejorar sus puntos débiles

Características de Éxito: La clave para que esta asociación funcione es que cada pareja reconozca su propia personalidad. Los extrovertidos deben trabajar conscientemente para no hablar todo el tiempo. Ellos pueden usar su “extravertida” para sacar información valiosa de su pareja, deben hacer preguntas a su pareja.

Retos: El emparejamiento no funcionará si ambos no dan un poco. Si cualquiera de los dos usa sus tendencias para encubrir deficiencias, entonces es muy difícil hacer que la pareja funcione.

Introvertido-Introvertido

Intención: Una silenciosa intensidad conduce a soluciones sólidas como una roca.

Características de Éxito: Los introvertidos suelen tener una intensidad tranquila y sobresalir en la difícil tarea de la programación

Retos: La dificultad de esta pareja es que los introvertidos pueden ser muy pobres comunicadores ... Muchas veces, se oponen firmemente a emparejarse.

El tema del género

Intención: El género no es un problema.

Características de Éxito: Lo que realmente importa son los mismos temas que hemos abordado

Retos: El único problema que podemos imaginar es el que se relaciona con la falta de respeto de género. Debido a que los pares funcionan tan estrechamente juntos, este tipo de chovinismo de género podría ser más notable y convertirse en un problema real.

El tema cultural

Intención: Tener parejas con diferentes orígenes culturales es maravilloso para la construcción de la confianza y la comunicación dentro del equipo. Mientras existe una comunicación, el par puede tener éxito.

Características de Éxito: La programación en parejas y la rotación mejora el trabajo en equipo y la comunicación. Tienden a ser particularmente eficaces cuando se trata de aprender acerca de la cultura del otro y la eliminación de las barreras culturales.

Retos: La intolerancia no debe ser tolerada en ninguna forma. Si el emparejamiento se plantea la cuestión cuanto antes, entonces eso es probablemente bueno. No hay lugar para la intolerancia en un entorno de colaboración.

Conductor profesional

Causa: El deseo de poder, la falta de confianza del conductor en el navegador

Características de Éxito: Un patrón de conducción profesional beneficiosa puede legítimamente ocurrir, pero sólo temporalmente. Si usted está cerca de una fecha límite crítica, es posible que no tenga tiempo para que el navegante aprenda una nueva herramienta de desarrollo..

Retos: Ayudar al conductor profesional a ceder el control y a ser un navegante

“Perdedor total"

Causa: Ego o algún tipo de problema de actitud. Muchas veces no tiene nada que ver con la pareja, la persona siente que él o ella es mejor que cualquier otra persona.

Características de Éxito: técnica de gestión, "Gestión por caminar alrededor" (MBWA) y su hermana "Gestión por caminar y escuchar." (MBWL) ... El gerente que está utilizando MBWA debe ser capaz de detectar el problema del exceso de ego después de observar el funcionamiento de la pareja. Observar las interacciones, observando el lenguaje corporal

Retos: Asesoría y enseñar a la persona a mantener su ego bajo control puede ser eficaz.

"Mi pareja es tan inteligente“

Causa: Cuando alguien simplemente tiene en cero la confianza en su propia habilidad y se siente inadecuado en el cumplimiento de la tarea incluso más básica. A menudo es incluso peor si se combina con una persona con mejores habilidades.

Características de Éxito: Es evidente que hay problemas psicológicos profundos que tendrían que superar en los casos más graves. Sin embargo, para el resto, hay varias soluciones posibles. Una de las claves es que la persona que carece de la confianza conduzca.

Retos: Darse cuenta de inmediato e informar al gerente. También es cierto que este problema puede ser más sutil: las personas que carecen seriamente la confianza a menudo han aprendido técnicas de ocultarlo,

Programación por pares

Los siete mitos de la programación en parejas

Mito 1: La carga de trabajo será doble, con dos haciendo el trabajo que uno puede hacer.

Mito 2: Nunca volveré a trabajar solo. Yo no podría soportarlo!

Mito 3: Va a funcionar bien sólo con el socio adecuado**.**

Mito 4: La programación en parejas es bueno para la formación. Pero, una vez que sabes lo que estás haciendo, es una pérdida de tiempo

Mito 5: Nunca voy a conseguir el crédito por hacer cualquier cosa. Voy a tener que compartir todo el reconocimiento con mi pareja

Mito 6: El navegador sólo encuentra errores de sintaxis. Qué aburrido es eso! Los compiladores pueden hacer mejor eso que los humanos.

Mito 7: Las únicas veces que he hecho bien algún trabajo interesante, es cuando estoy solo. Ahora con esto de PP, nunca voy a hacer nada! La programación por pares me volverá loco.

Programación por pares

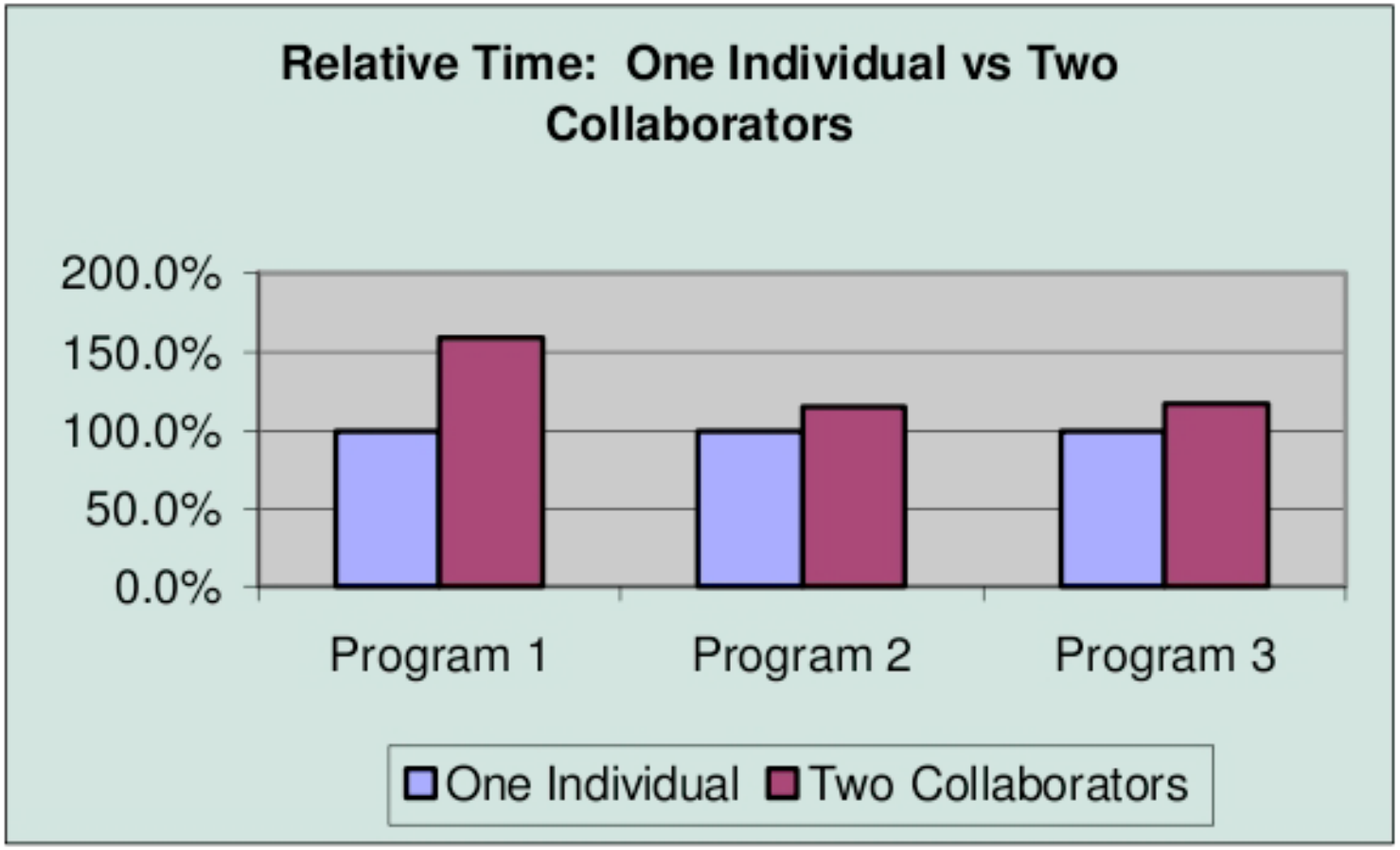
¿Cómo ayuda?

- Revisión continua.
- Menos defectos, estos se detectan a tiempo
- Mejor Calidad del Diseño
- Mejor Solución de Problemas
- **Más económico**
- La presión de pareja asegura la entrega oportuna
- Rápido método práctico para el aprendizaje
- Mejor inducción de nuevos miembros del equipo

- Ahorra esfuerzo de documentación Intra-Equipo
- Menos distracción conduce a una mayor productividad
- Mejora de la satisfacción
- Progreso sostenible: Ayuda a reducir la velocidad y pensar
- Mejor Trabajo en Equipo y Comunicación
- Gestión de proyectos: Estrategia de Mitigación del Riesgo

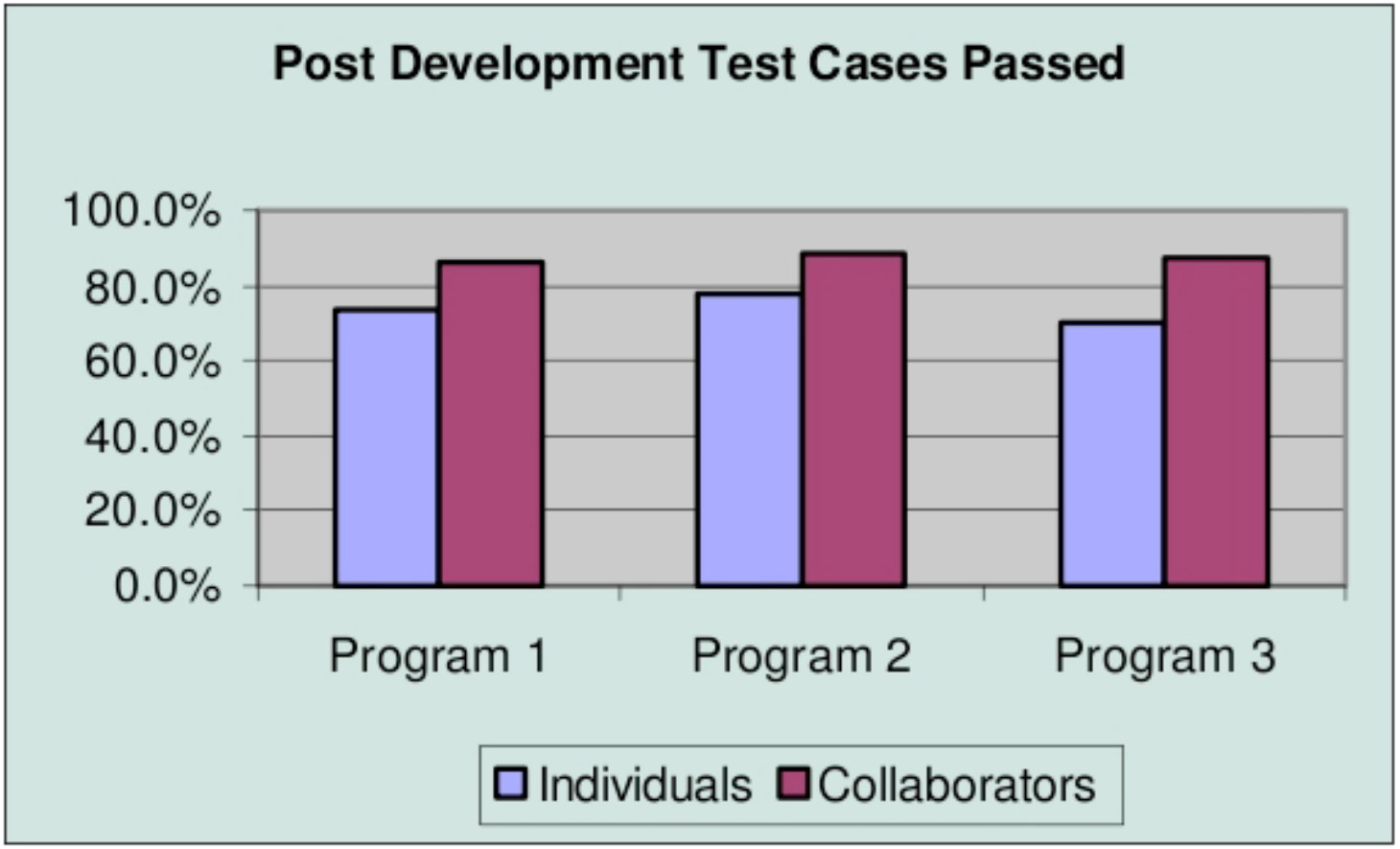
Programación por pares

Más económico



Programación por pares

Más económico



Programación por pares

El emparejamiento es delicado - Cosas a tener en cuenta

- Parejas sin rotación
- Sólo una persona conduce
- Par Distraído
- Emparejamiento selectivo más cerca de una salida
- Gerente decide los pares

Programación por pares

Puntos de resistencia - Cosas a tener en cuenta …

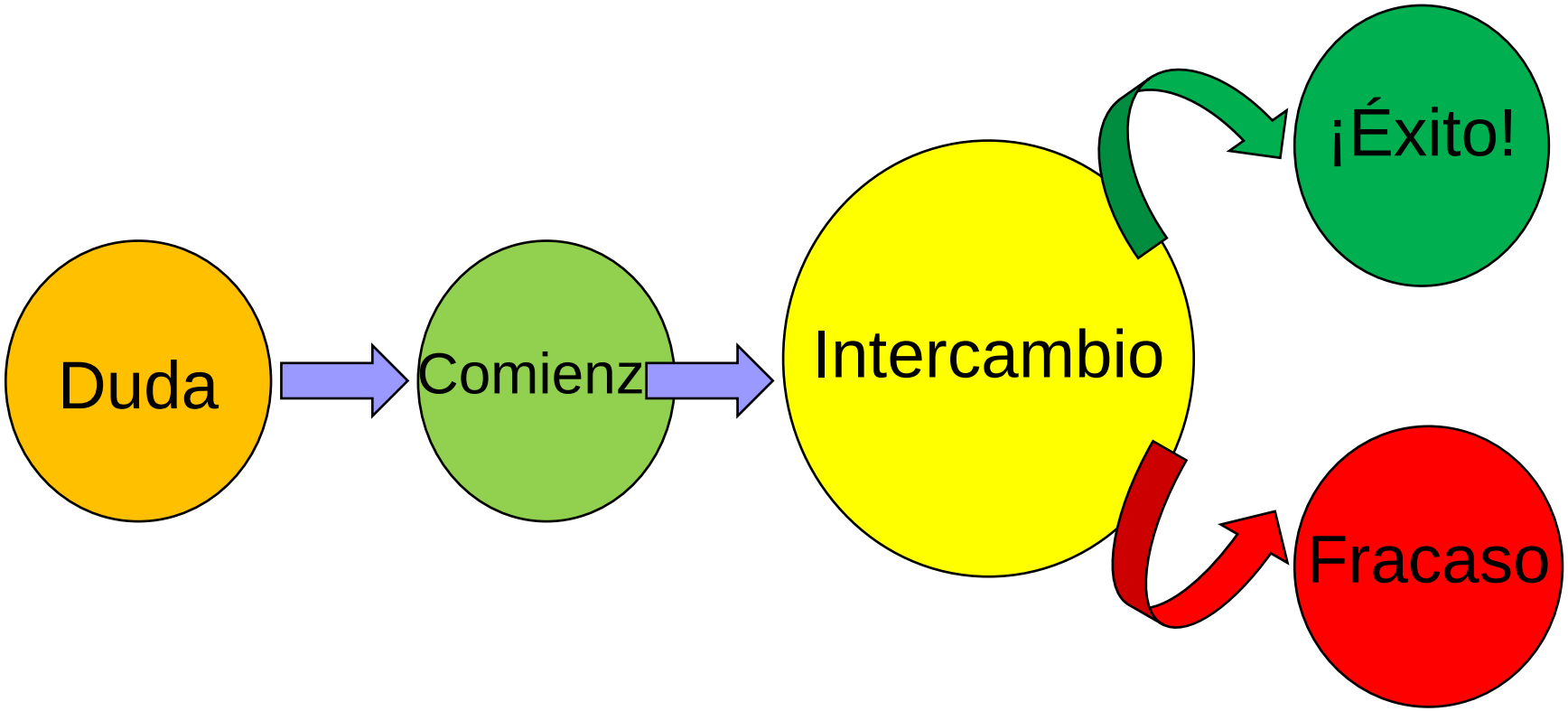
- Dificultad para convencer a las partes interesadas para que dos personas trabajen en una misma tarea
- Todo el mundo se establece en una "zona de confort" El emparejamiento podría alterarlo
- Sentido de dueño del código
- Las personas que no se preocupan
- Los desarrolladores no participan en Estimación y Planificación
- Configuración de estaciones de trabajo inconsistente
- Área pequeña de trabajo (Cubículos)

Programación por pares

Principios de selección de parejas de

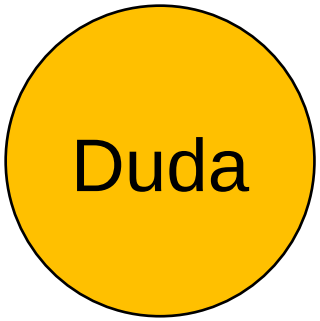
Programación por pares

Toda persona que practica la Programación por Pares la primera vez, pasa por diferentes etapas…



Programación por pares

La Experiencia



¡¿Por qué a mi?!

¿Será una buena pareja de trabajo?

¡Me gusta trabajar solo!

¡Hay que trabajar!

Compatibilidad en los métodos de trabajo

Revisar los roles:
¿Conductor?
¿Navegante?

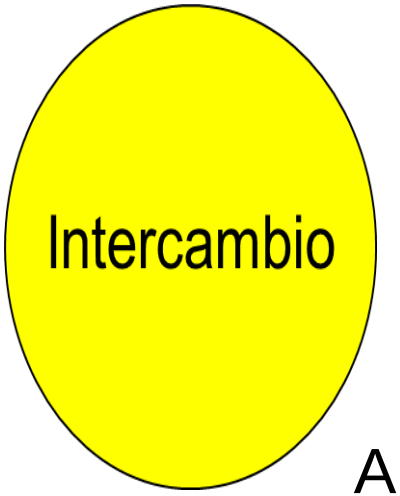
Programación por pares

La Experiencia

Diferentes puntos de vista

¿Qué tan buenas son mis ideas?

¿Qué sé y qué puedo aprender?



Conocer diferentes métodos para hacer las cosas

Conflictos...

Aprendizaje mutuo

¡Aquí ocurre lo interesante!
Dos piensan mejor que uno

Programación por pares

La Experiencia

¡Lo logramos!

¡Qué bueno trabajar contigo!

Evaluar las cosas que se aprendieron

Aquí es donde se diferencian una buena y una mala Programación por Pares

¡Jamás trabajo contigo de nuevo!

Qué pérdida de tiempo…

Trabajo mejor solo…

Programación por pares

En Conclusión

- ¡La meta es el Éxito!
- Hay que ser receptivos hacia las nuevas ideas y métodos
- Hay que estar dispuestos a aprender y enseñar
- Es importante ser tolerantes, pero también firmes. Cada quién tiene su personalidad, mas no caer en el abuso.
- Si no funcionó la primera vez… ¡No te rindas! **Siempre** se aprende. No te cierres para volver a intentarlo con alguien más.

Programación por pares

youtube.com/watch?v=q-QWdFa4awI

Normal Pair Programming Session

youtube.com/watch?v=ReuFZYtGeCc

Pair Programming Anti Patterns

Agenda

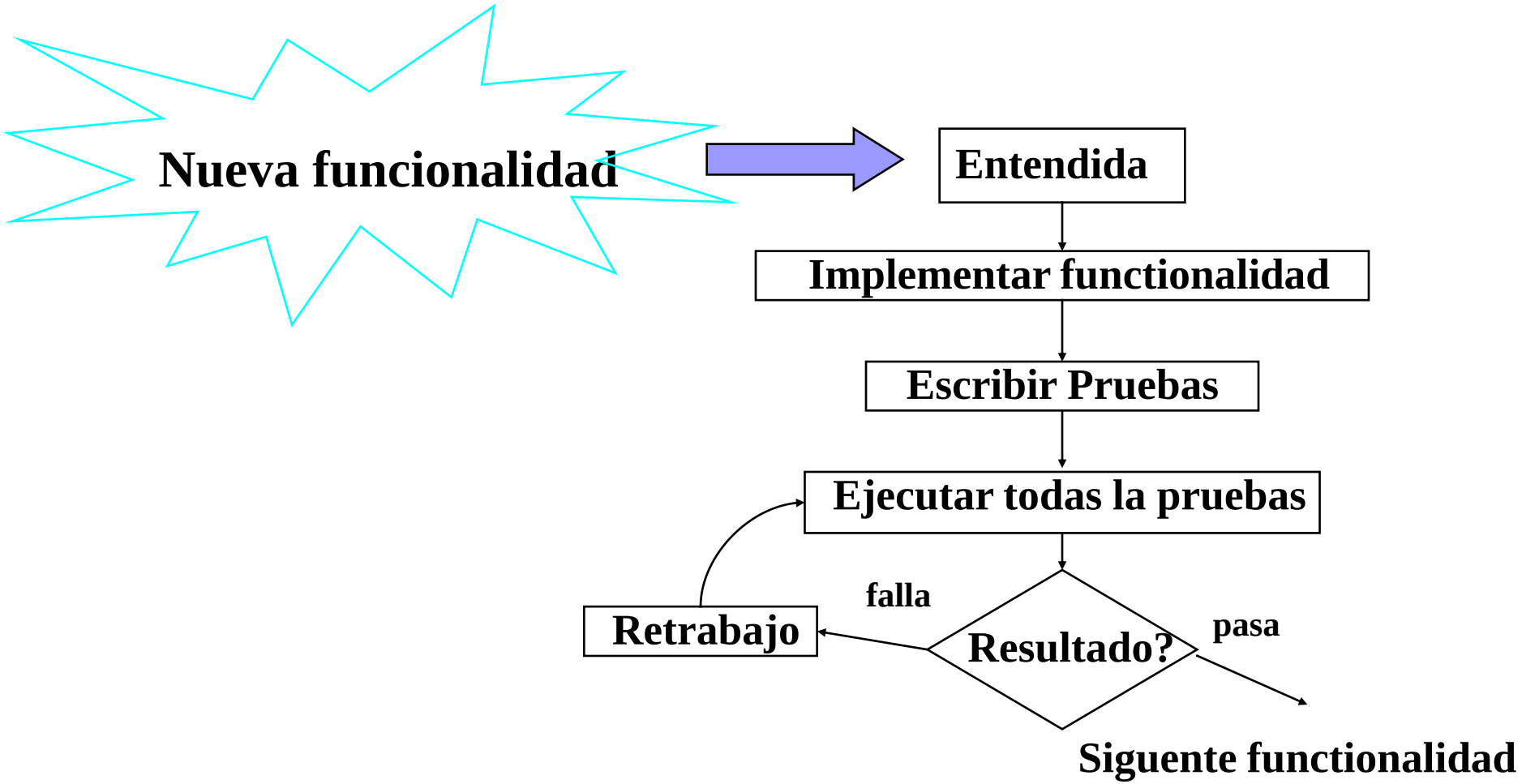
- Equipo
  - Programación por pares:
    - Que es?
    - Que aspectos afectivos influyen?
    - Experiencia
    - Testimonio de Leonardo Martínez
- **Verificación**
  - **TDD en PyUnit.**

¿Cuándo probar los programas?

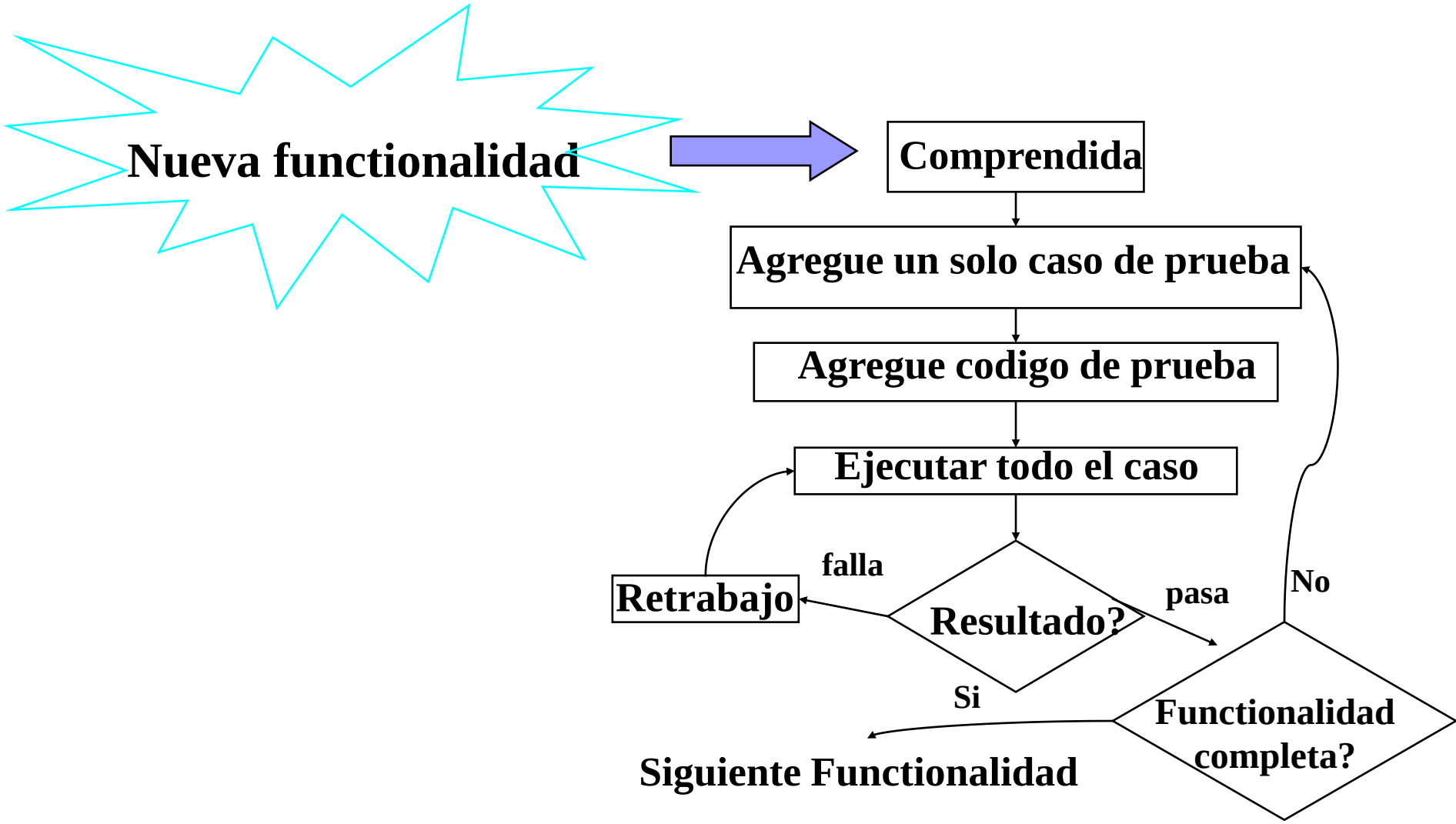
- Pruebe después
  - La forma convencional para la prueba en la que las pruebas siguen a la implementación

- Pruebe antes
  - La visión de la programación extrema en la que las pruebas se utilizan como una herramienta de desarrollo

Pruebe despues



Pruebe primero

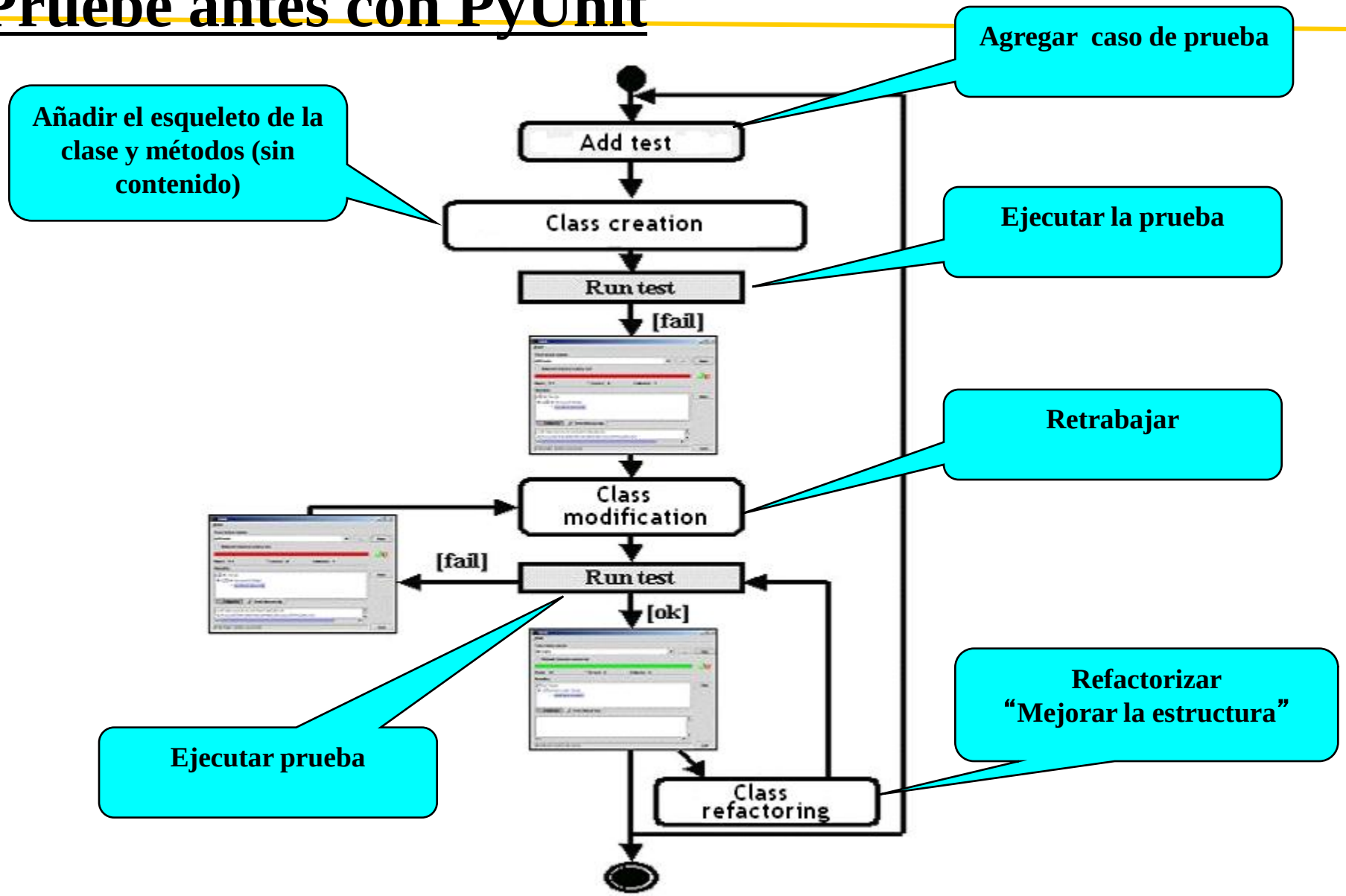


Ventajas de Probar antes

- Cada método tiene asociado un caso de prueba
  - incrementa la confianza del código ...
- Se simplifica:
  - refactorización / reestructuración
  - mantenimiento
  - la introducción de nuevas funcionalidades
- Pruebe primero ayuda a construir la documentación
  - testcases son un buen "muestreo de uso"
- La programación es más divertida ...

somos USB

Pruebe antes con PyUnit



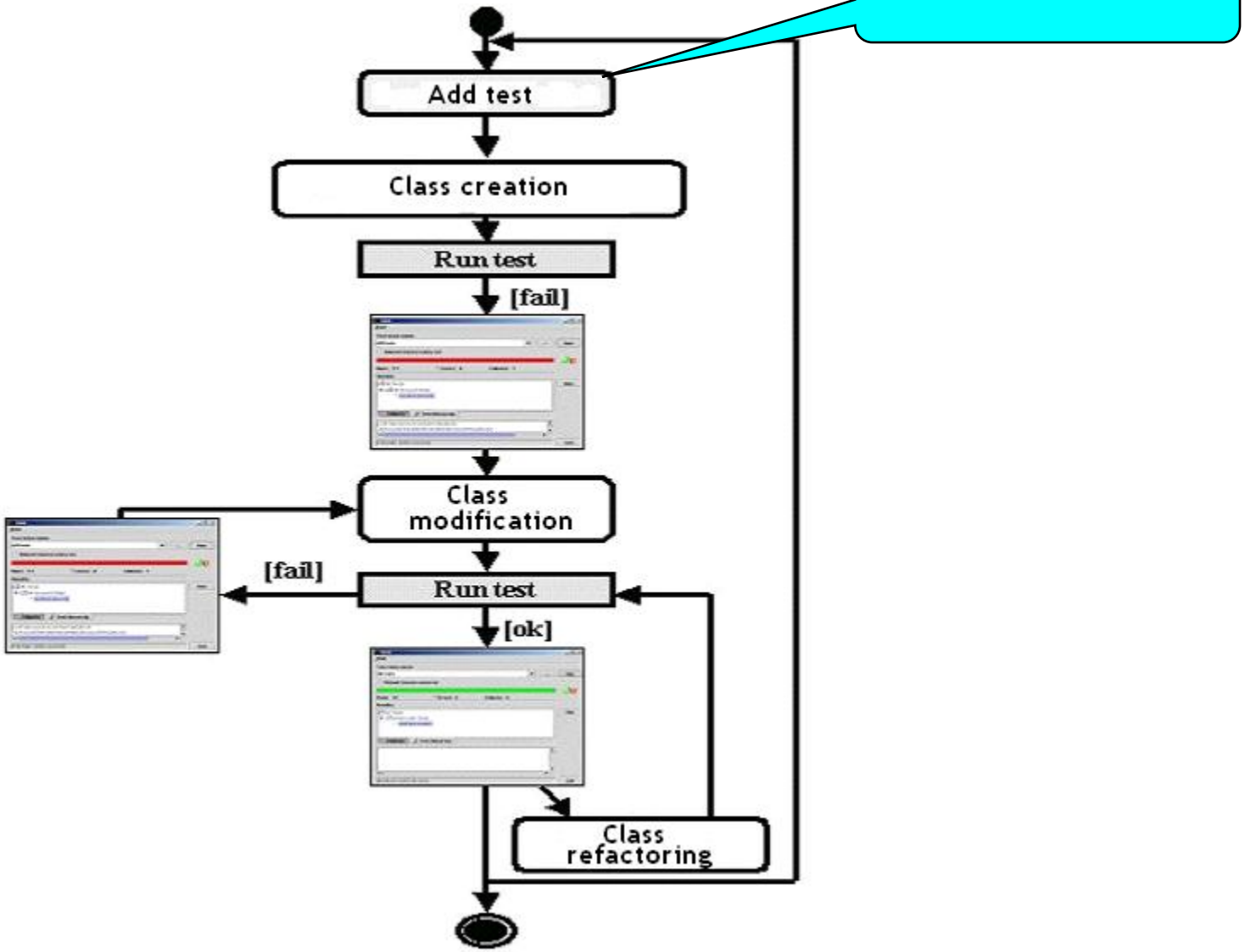
PyUnit en la practica…

- Sistema existente: una clase cuenta corriente para manejar una cuenta bancaria
  - depositar
  - retirar
- Añadir una nueva funcionalidad
  - asentar

Ejemplo:

```
cc = CurrentAccount()
cc.deposit(12)
cc.draw(-8)
cc.deposit(10)
cc.settlement()
```

expected value **14 euro!**


Agregar caso de prueba
Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring

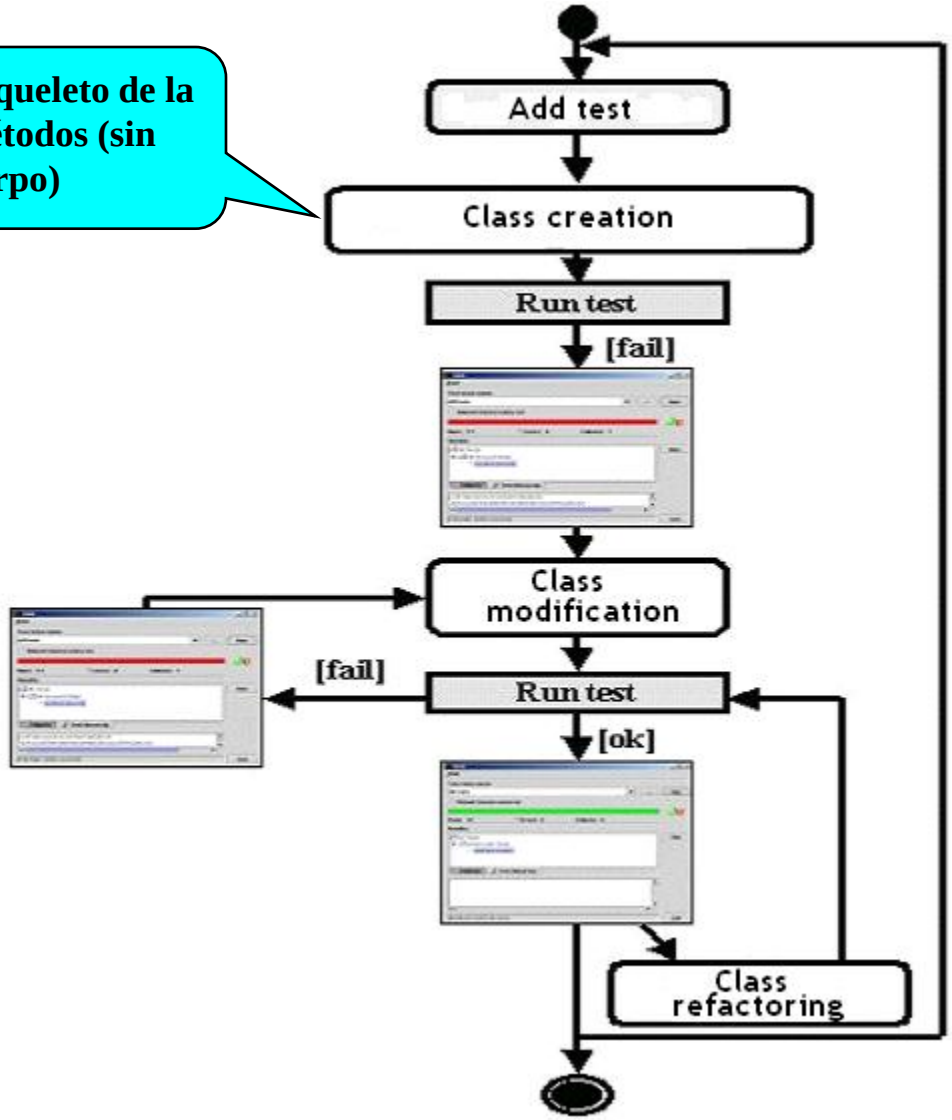
somos USB

Agregar casos de prueba para el método **settlement**

Probar antes…

```
class Test_CurrentAccount(unittest.TestCase):

 def test_settlement_none(self):
   c = CurrentAccount()
   assertEquals(0, c.settlement())

 def test_settlement(self):
    c = CurrentAccount()
    c.deposit(12)
    c.draw(-8)
    c.deposit(10)
    assertEquals(14, c.settlement())
```

Añadir el esqueleto de la clase y métodos (sin cuerpo)

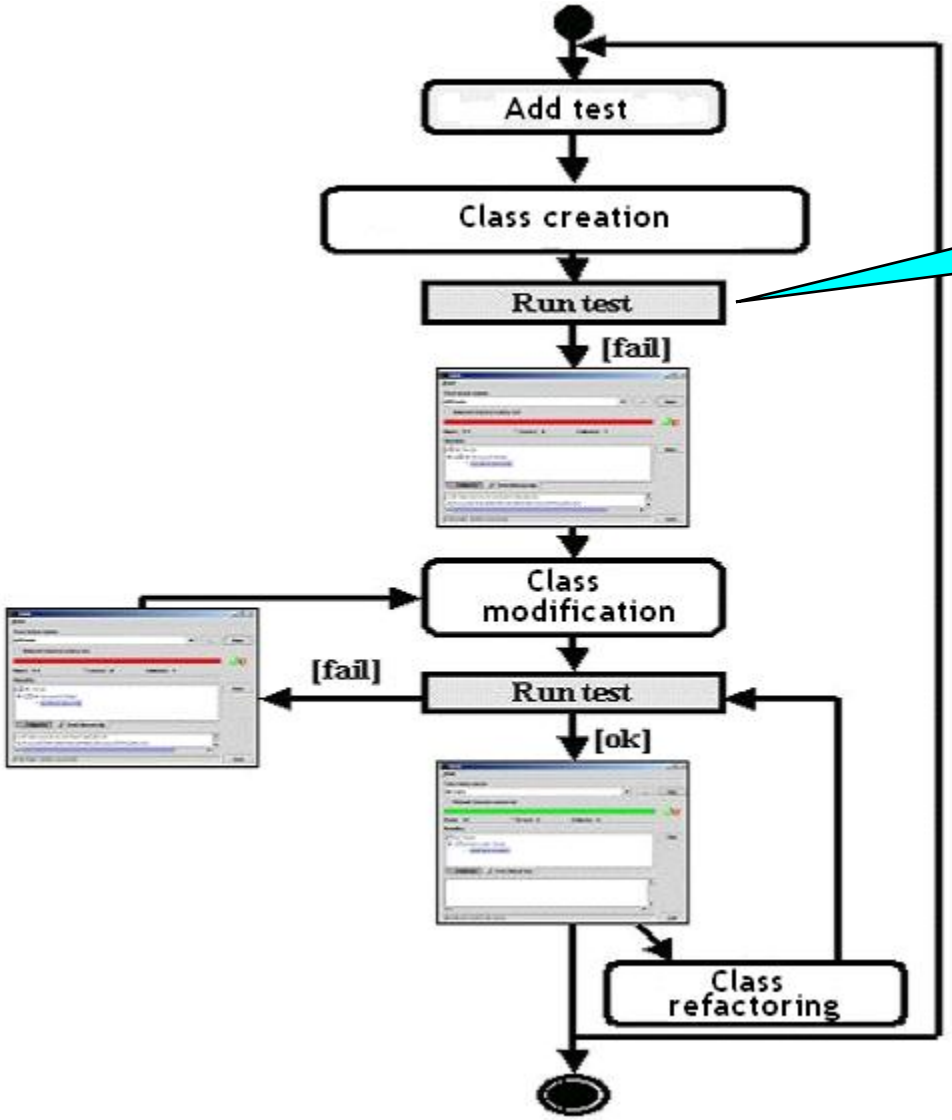


somos USB

Agregue el esqueleto del código del método

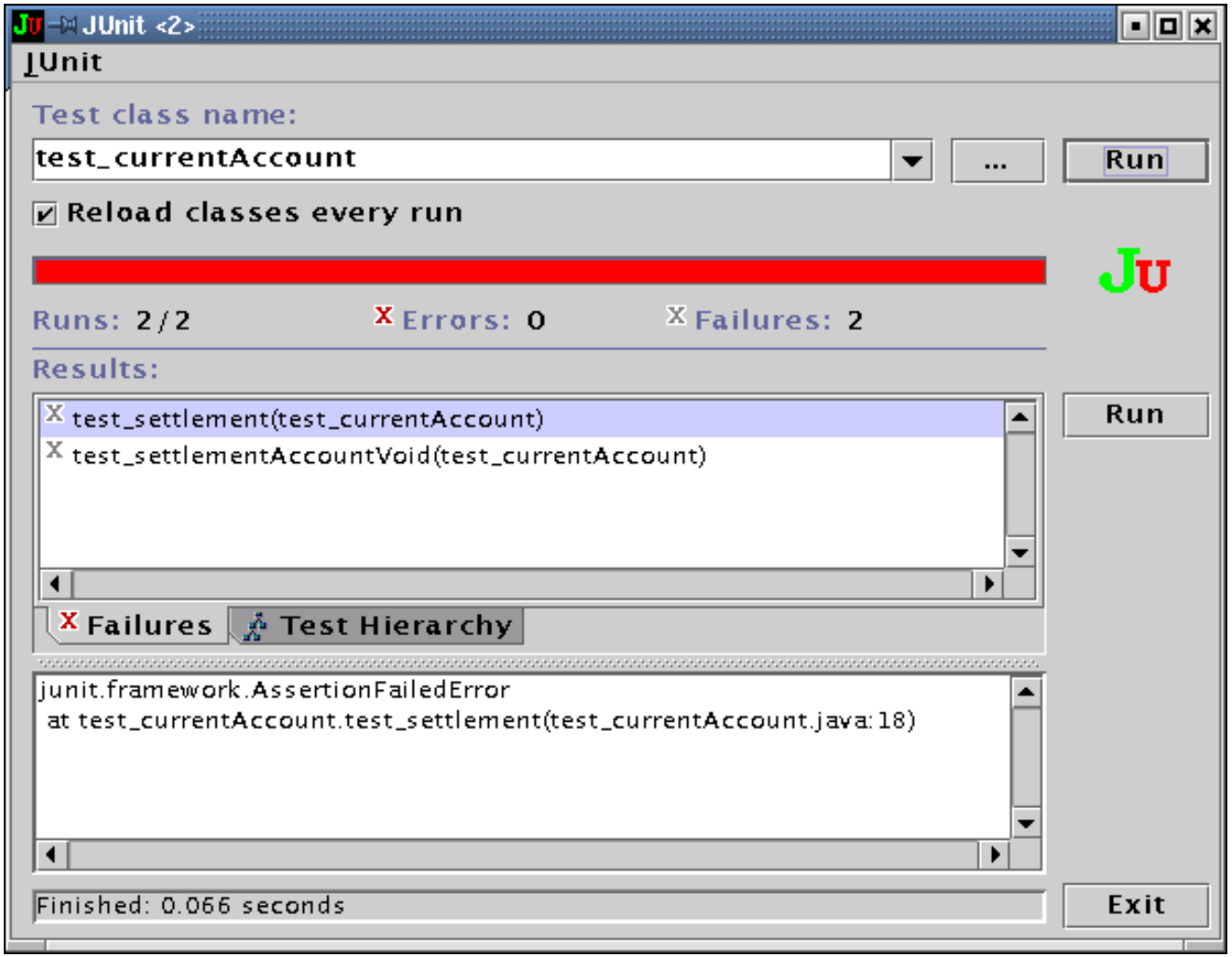
```
class CurrentAccount:
  account = []
  lastMove = 0

 def deposit(value):
   account.append(value)
   lastMove += 1

 def draw(value):
   account.append(value)
   lastMove += 1

 def settlement():
   return 0
```

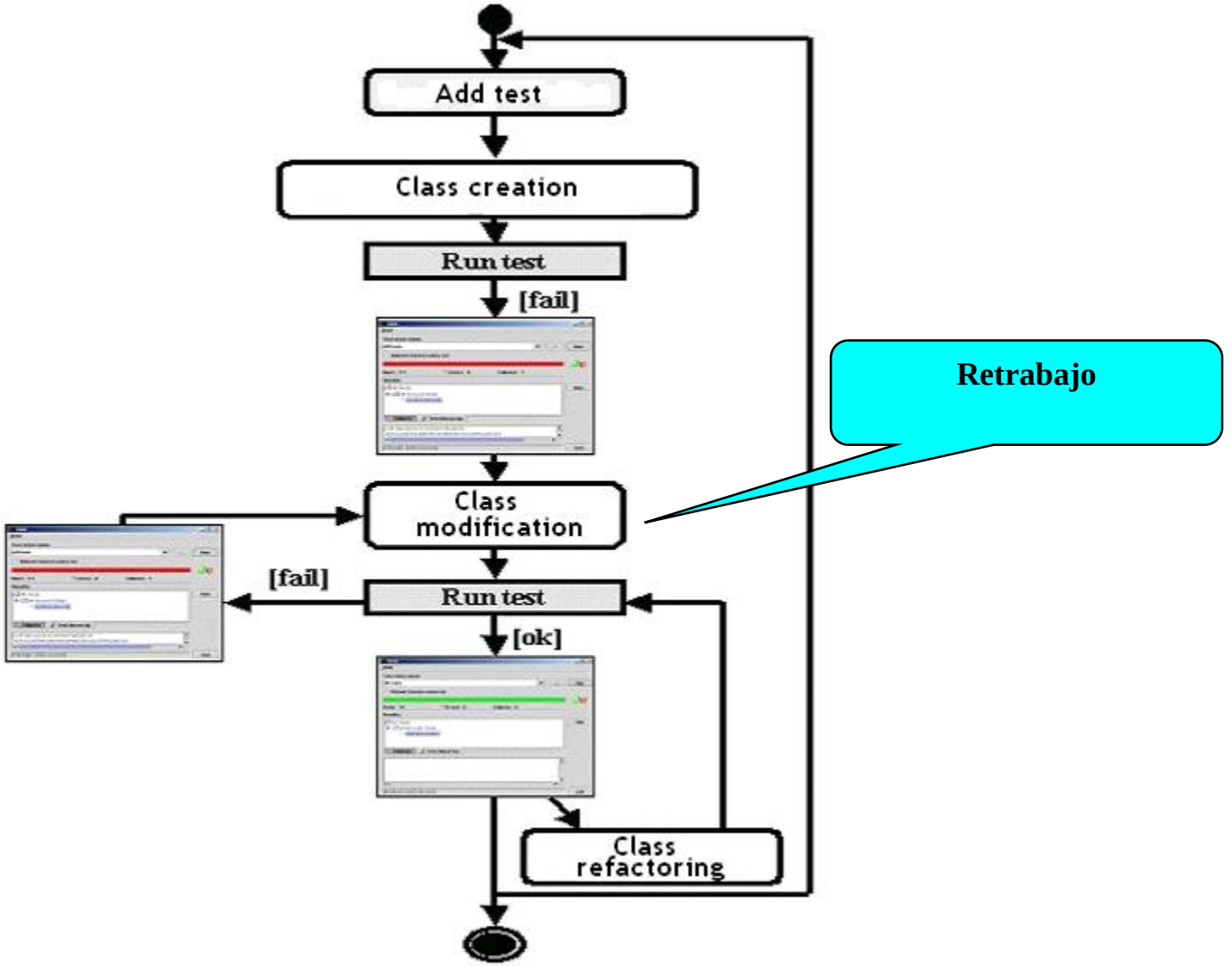
```
class Test_CurrentAccount(unittest.TestCase):

 def test_settlementVoid(self):
    c = currentAccount()
    assertEquals(0, c.settlement())

 def test_settlement(self):
    c = currentAccount()
    c.deposit(12)
    c.draw(-8)
    c.deposit(10)
    assertEquals(14, c.settlement())
```



Ejecutar prueba

Ejecutar PyUnit (*primera vez*)

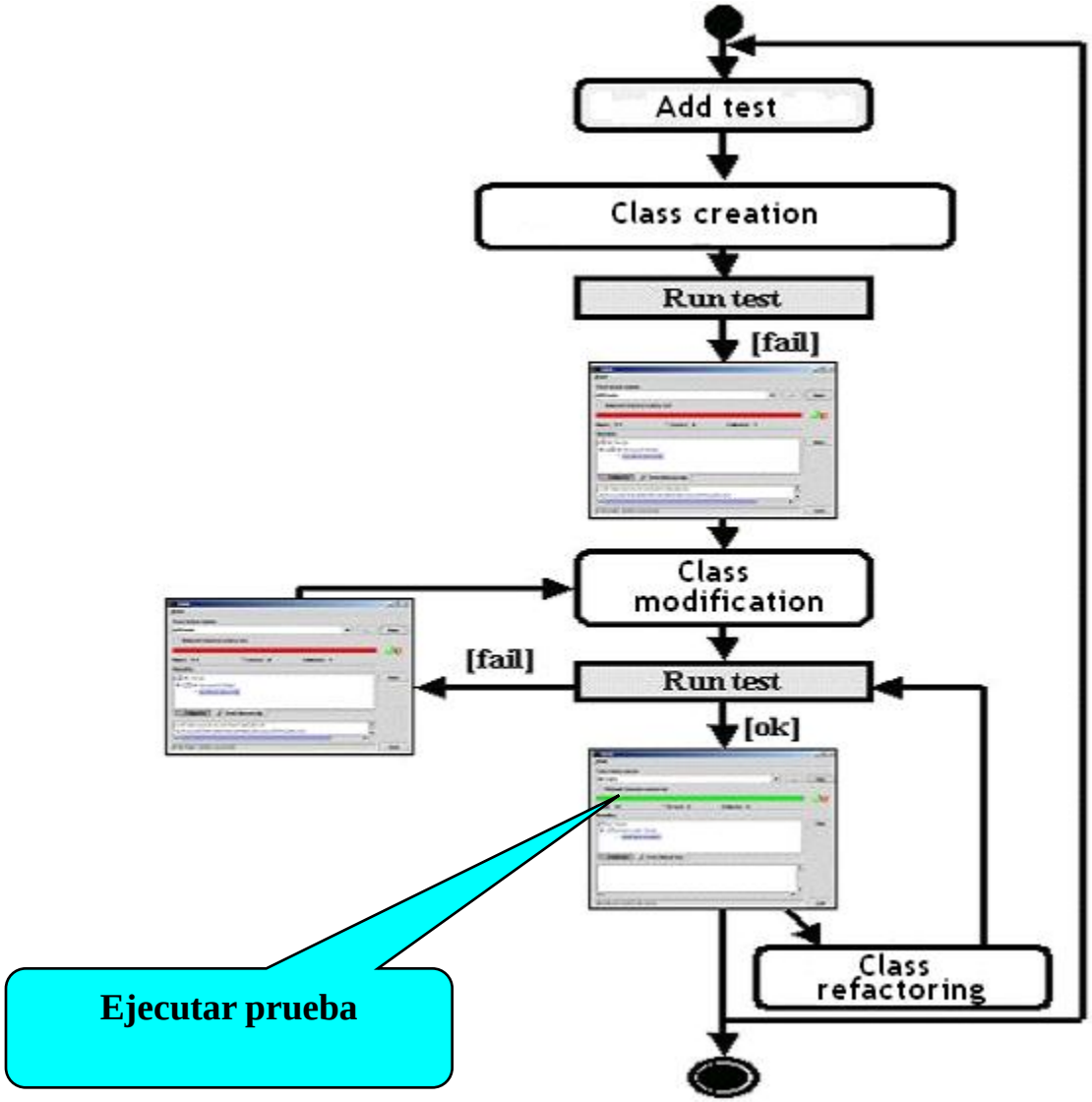



Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring
Retrabajo

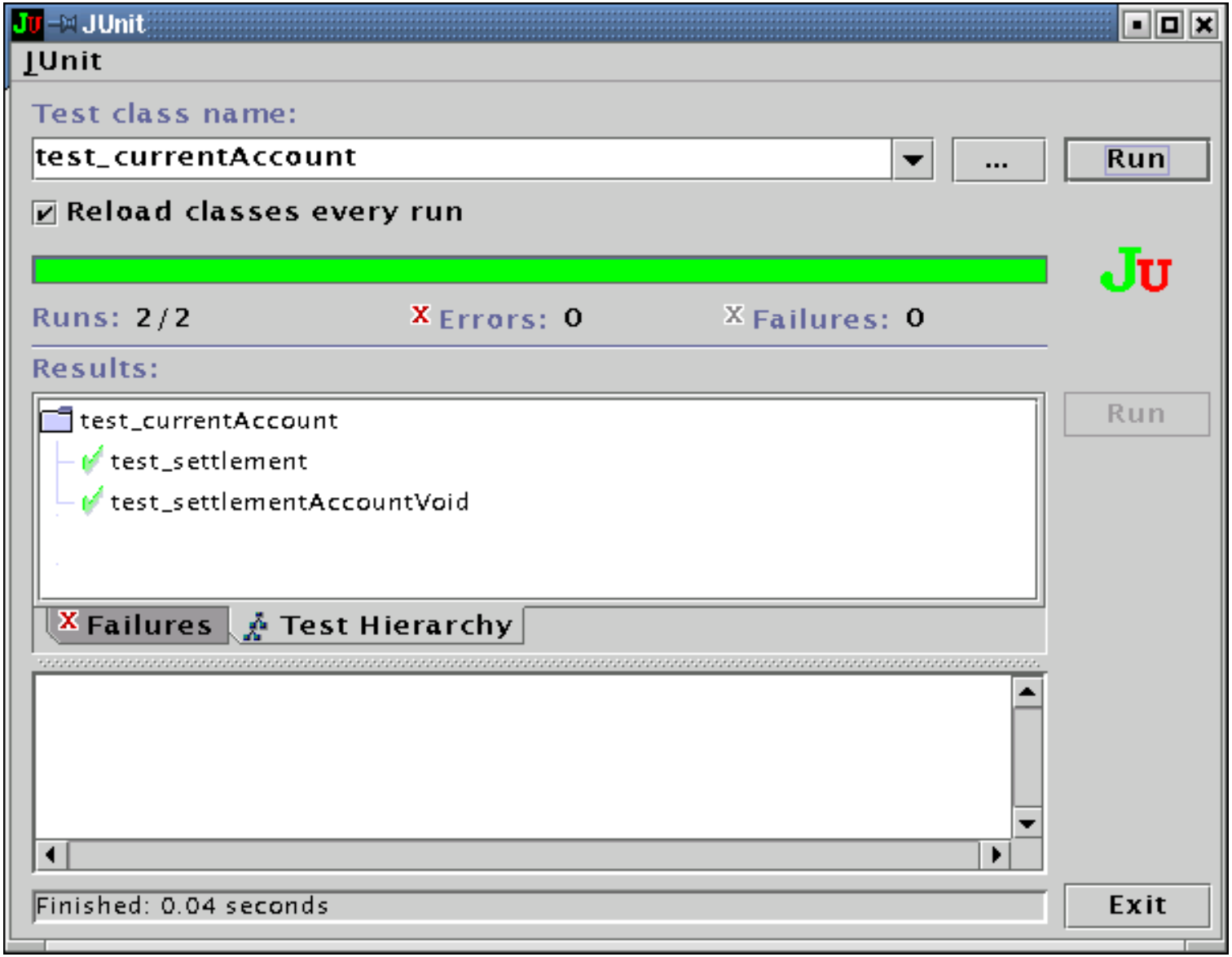
Retrabajo

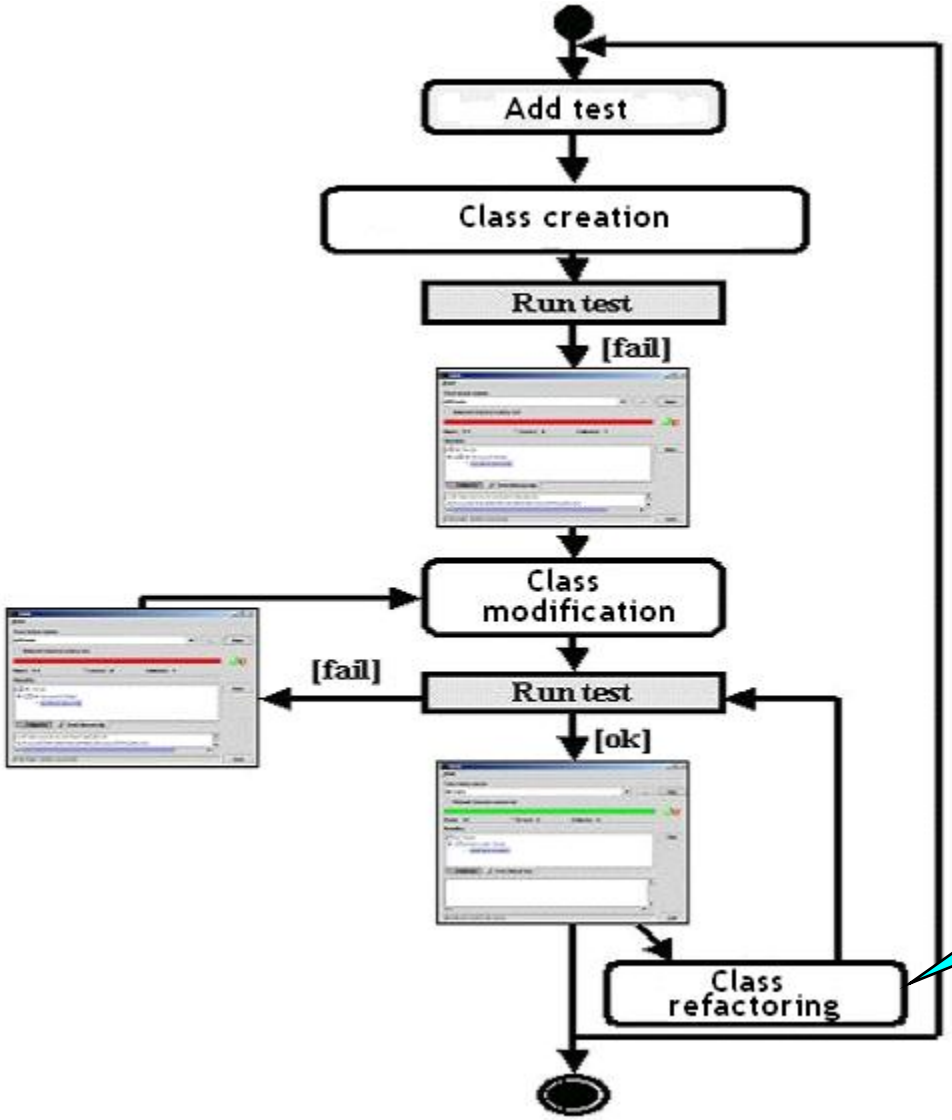
```
class CurrentAccount:
  account = []
  lastMove = 0

 def deposit(value):
   account.append(value)
   lastMove += 1

 def draw(value):
   account.append(value)
   lastMove += 1

 def settlement() {
   result = 0
   for value in account:
     result += value

   return result
```

```
class Test_CurrentAccount(unittest.TestCase):

 def test_settlementVoid(self):
    c = currentAccount()
    assertEquals(0, c.settlement())

 def test_settlement(self):
    c = currentAccount()
    c.deposit(12)
    c.draw(-8)
    c.deposit(10)
    assertEquals(14, c.settlement())
```


Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring
Ejecutar prueba

Ejecutar PyUnit (*Segunda vez*)

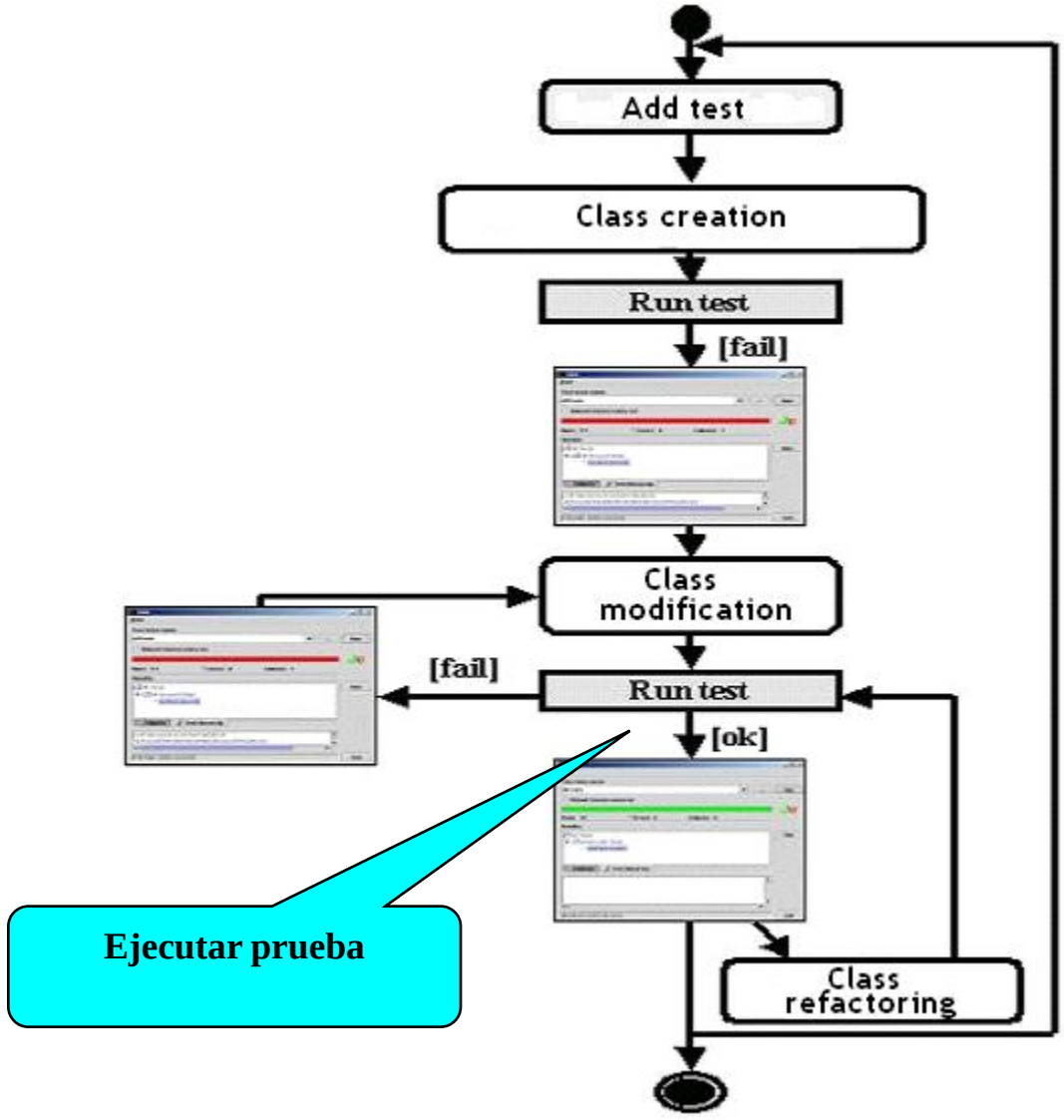



Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring
Refactorizar
"Mejorando la estructura"

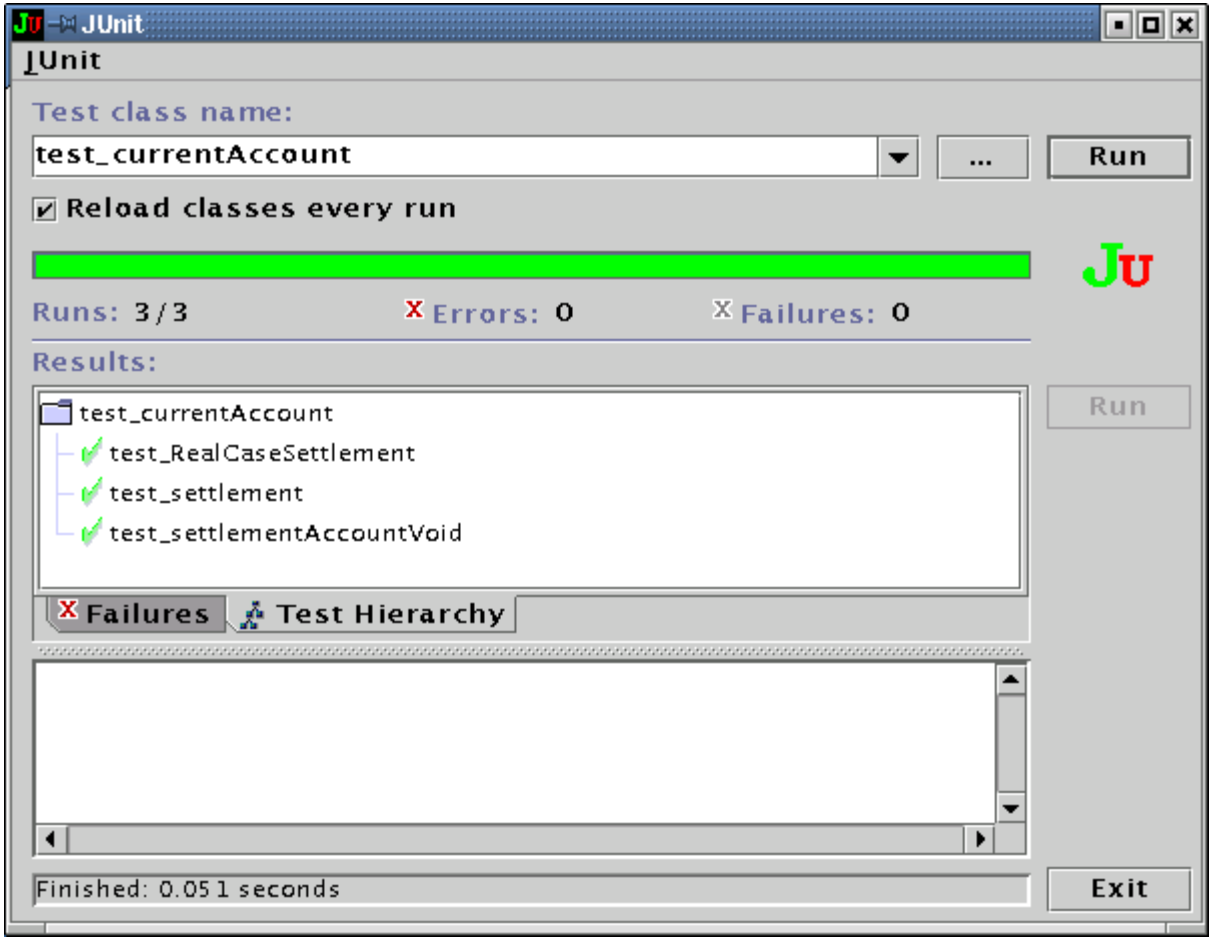
Refactorizacion

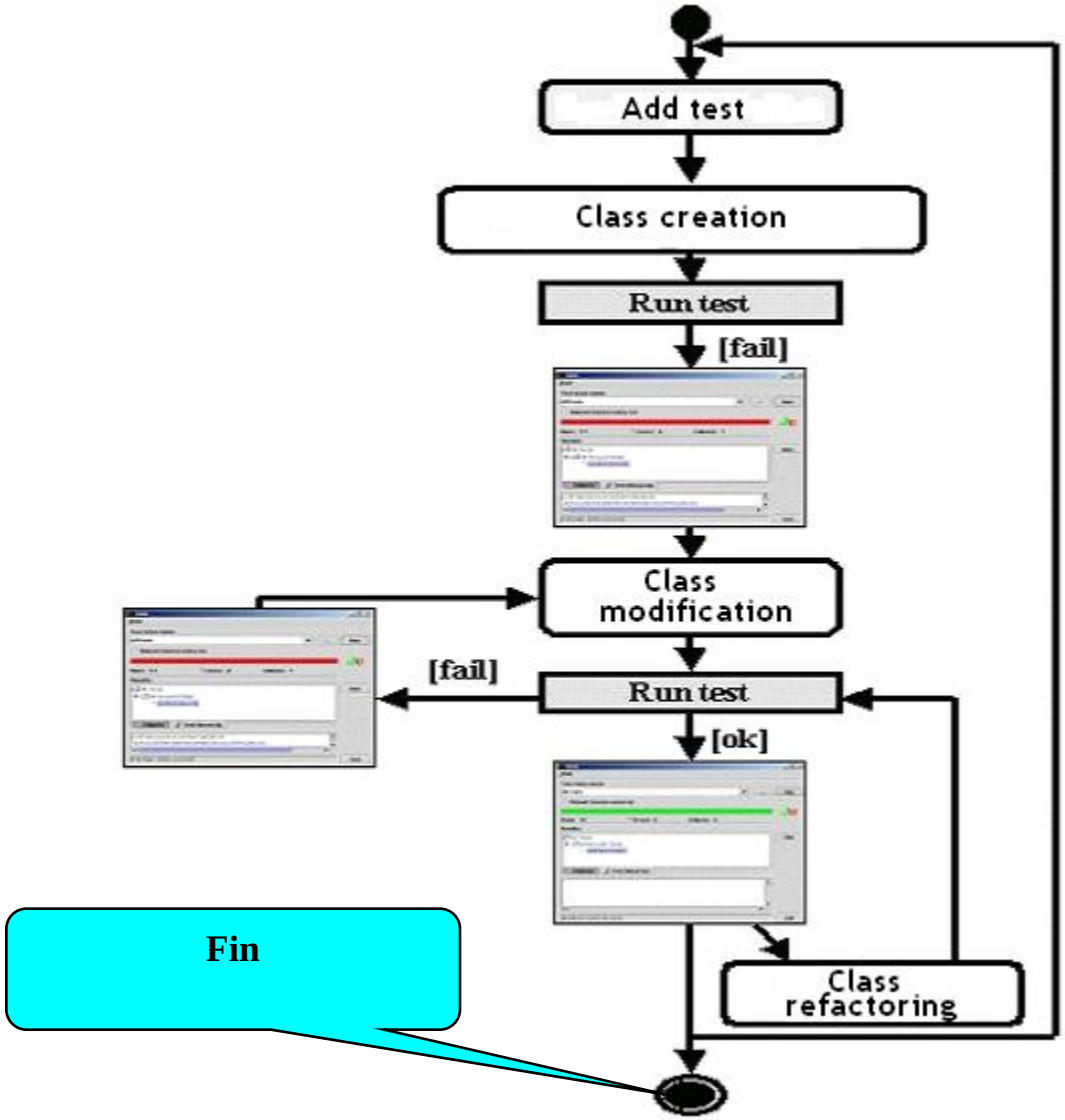
```
class CurrentAccount:
    account = []
    lastMove = 0

    def deposit(value):
        account.append(value)
        lastMove = len(account)

    def draw(value):
        account.append(value)
        lastMove = len(account)

    def settlement() {
        result = 0
        for value in account:
            result += value

        return result
```

"Cambiando la estructura dde datos: Array --> List"


Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring
Ejecutar prueba

Ejecutar PyUnit (*tercera vez*)




Add test
Class creation
Run test
[fail]
Class modification
[fail]
Run test
[ok]
Class refactoring
Fin

Referencias