DEPURACIÓN E INTEGRACIÓN

INTEGRACIÓN

¿QUE ES LA INTEGRACIÓN?

Es aquella actividad de software en la cual se combinan dos componentes de software separados en un solo sistema.

BENEFICIOS DE UNA INTEGRACIÓN CUIDADOSA

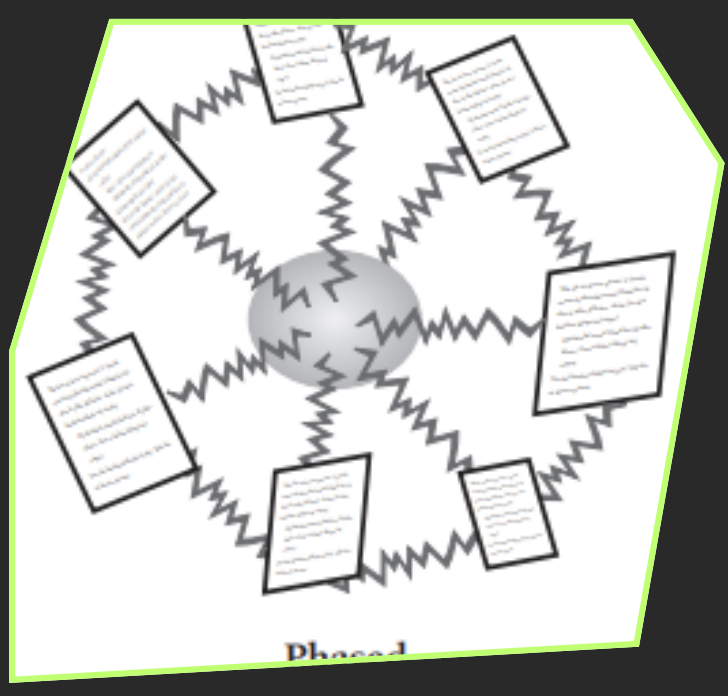
- Diagnostico efectivo de defectos
- Menos defectos
- Menor tiempo en el primer trabajo del producto
- Incrementa la modal
- Mayor precisión en los estados de reporte
- Mejora la calidad del códgio
- Menor documentación

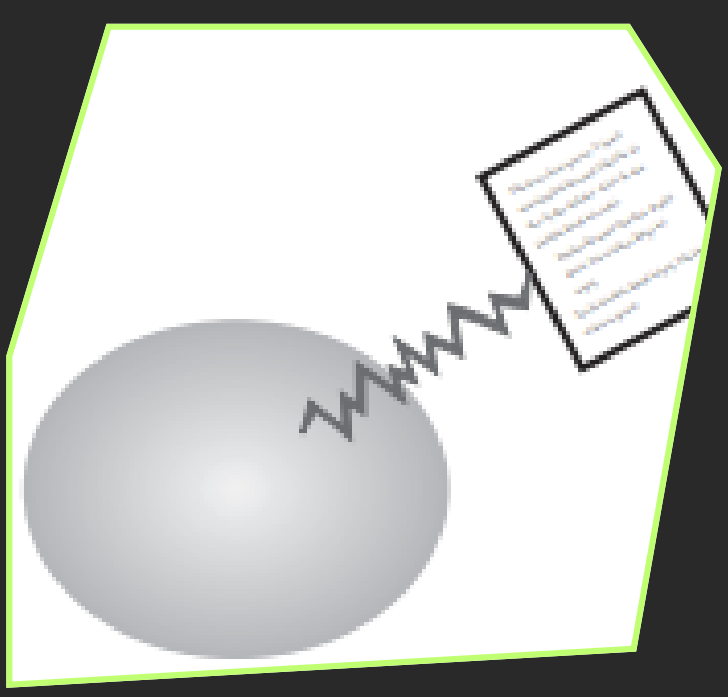
INTEGRACIÓN EN FASES (BIG BANG)

La integración en fases consiste en seguir pasos o fases bien definicad:

1. Disenar, codificar y depurar
2. Combinar las calses en un sistema
3. Testear y depurar el sistema.

Este tipo de integración te forza a no solo lidiar con problemas causados entre interacción de clases, también con problemas que son dificiles de diagnosticar ya que interactúan entre sí.





INTEGRACIÓN INCREMENTAL

La integración incremental, se codifica y testea un programa en partes pequeñas para finalmente integrarlas en una sola. Sus pasos son:

1. Desarrollar una pequeña parte funcional del sistema.
2. Diseñar, codificar, testear y depurar una clase.
3. Integrar la nueva clase con el esqueleto del sistema.

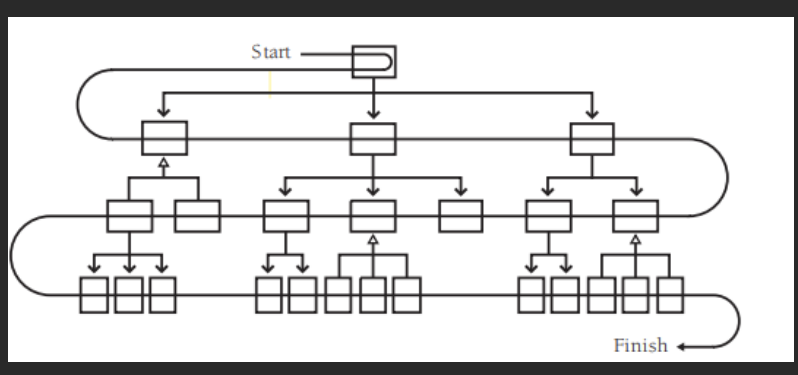
Algunas de sus ventajas son:

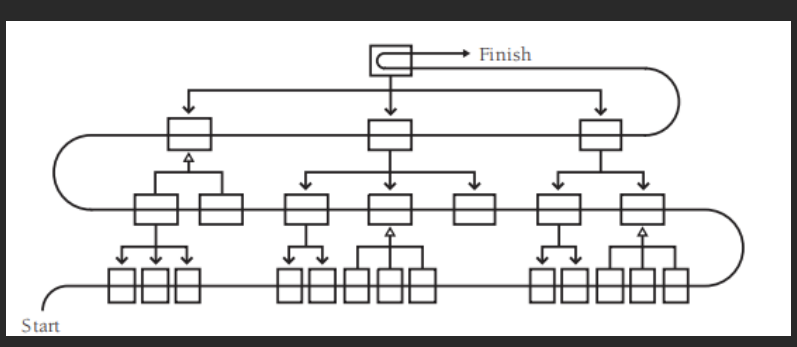
- Fácil localizacion de errores
- Se puede obtener un mejor monitoreo de progreso.
- Diseñar un sistema en un tiempo de desarrollo corto

ESTRATEGIAS DE INTEGRACIÓN

- **Top-Down Integration**

En esta integración las clases superiores son desarrolladas e integradas primero, suelen ser las ventanas principales, control de aplicacion u objetos que contienen un main. Las interfaces entre clases deben ser cuidadosamente especificadas.



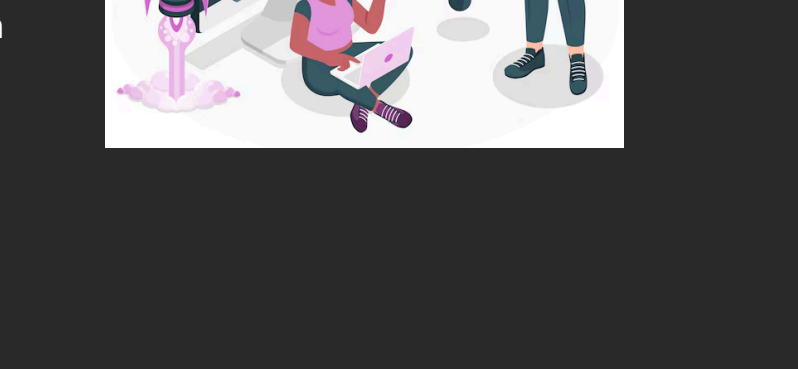


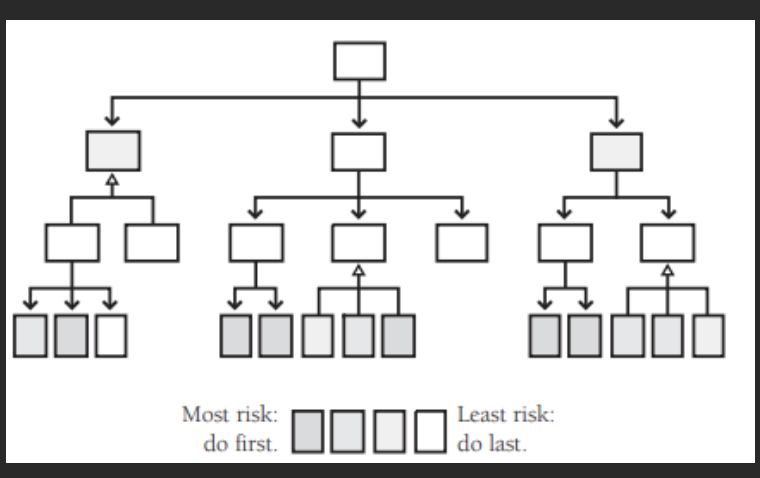
- **Bottom-Up Integration**

Las clases se desarrollan e integran desde el inferior de la jerarquía, provee un limitado asignamiento incrementar, este tipo de integración se puede iniciar al inicio de un proyecto.
Además requiere un diseño completo del sistema antes de iniciar la integración.

- **Sandwich integration**

En este tipo de integración primero se integran las clases de objeto de negocio con mayor nivel en la parte superior de la jerarquía, de ahí se integran las clases de dispositivo que utilizadas por las clases bajas. Las clases de en medio se suelen desarrollar al final.





- **Risk-Oriented integration**

También conocido como “Hard part first integration”, similar a la integración de sandwich. Tiene a integrar las clases altas y las clases bajas primero, guardando las de nivel medio para el final. En este se decide el nivel de riesgo asociada con cada clase, decidiendo cual será la parte más desafiante de implementar, siendo la primera en realizarse.

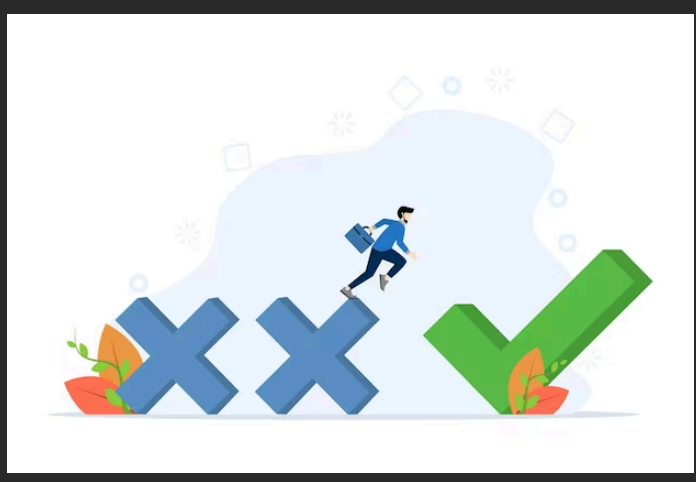
DEPURACIÓN

¿QUE ES LA DEPURACIÓN?

Es el proceso de identificar la raiz causante de un error con el fin de corregirla. La depuración no tiene por qué ser la parte difícil.

VENTAJAS DE LA DEPURACIÓN

- Incrementa la calidad del software
- Aprender acercar del programa que se está desarrollando
- Aprender de nuestros errores
- Aprender sobre la calidad de nuestro código a raíz del punto de vista de alguien que debe leerlo
- Aprender como resolvemos los problemas
- Aprender a resolver defectos

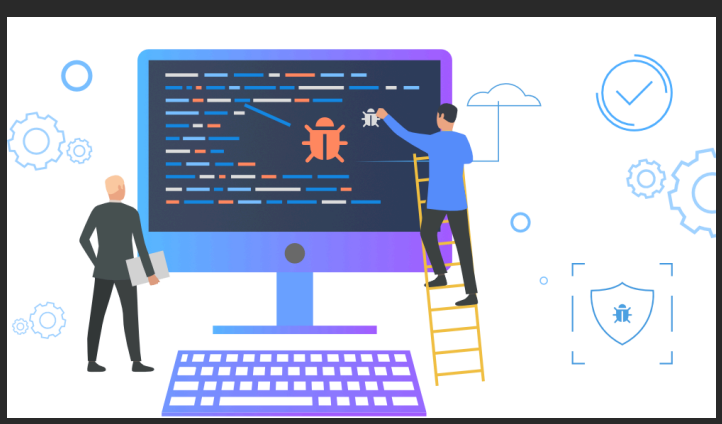


MALAS PRÁCTICAS AL DEPURAR

- Encontrar el defecto a través de adivinanza.
- Perder tiempo tratando de entender el problema completo.
- Arreglar el error de la forma mas obvia

MÉTODOS PARA DEPURAR

- **Manera científica**
  - **Recopilar datos de experimentos repetibles**
  - **Formular hipótesis que acontesca con información relevante**
  - **Diseñar un experimento para aprobar o rechazar la hipotesis**
  - **Aprobar o desaprobar la hipotesis**
  - **Repetir las veces que sea necesario**



RECOMENDACIONES

DETECCIÓN DE DEFECTOS

- Utilizar toda la información disponible para formular hipotesis acerca del origen de un defecto.
- Refinar los casos de prueba que producen el error.
- Usar herramientas que se tengan disponibles.
- Ejercitar el codigo en unidades de prueba.
- Reproducir el error de maneras diferentes, encontrando casos similares donde se producen.
- Checar el codigo creado recientemente.
- Expandir la zona sospechosa del código.
- Checar defectos comúnes.
- Tomar un descanso del problema.

CORRECCIÓN DE DEFECTOS

- Entender el problema antes de comenzar a resolverlo.
- Entender el programa y no solamente el problema.
- Confirmar el diagnostico de defectos.
- Relajarse
- Mantener una copia original del codigo fuente.
- Resolver correctamente el problema.
- Cambiar el codigo solo por una buena razón.
- Hacer un cambio a la vez.
- Checar defectos que sean similares al ocurrido.
- Añadir pruebas de unidad que expongan el defecto.
- Checar la solución que se le de al defecto.