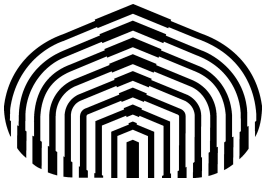

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la información
CI-2691- Laboratorio de algoritmos I

Laboratorio 7

El objetivo de este laboratorio es dar una introducción de los principios de desarrollo ágil, práctica de programación en parejas, ritmo sostenido y desarrollo de casos de prueba.

Principios de los procesos de desarrollo ágil.

El desarrollo ágil es muy utilizado actualmente para crear sistemas de software. En éste participan pequeños equipos que trabajan en colaboración organizada y disciplinada. La idea es desarrollar software en lapsos cortos. Se basa en ciertos principios, entre los cuales destacan:

- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenido, es decir, el equipo debe mantener un ritmo constante de forma indefinida.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.

El desarrollo ágil promueve ciertas prácticas que son esenciales para obtener software de calidad. El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Algunas de ellas que podemos aplicar en un primer curso de programación son:

- Diseño simple: Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
- Refactorización (Refactoring): Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.
- Programación en parejas: Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores, entre otras).

- Propiedad colectiva del código: Cualquier miembro del equipo puede cambiar cualquier parte del código en cualquier momento.
- Estándares de programación: Se enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

Principios de buena programación.

Para adiestrarse en las prácticas de la programación ágil es imprescindible seguir los principios que rigen una buena programación. A continuación se listan aquellos que pueden ser aprovechados en un primer curso de programación.

- Evitar la repetición de código o documentación y en su lugar reutilizar.
- Principio de abstracción. “Cada pieza de funcionalidad significativa en un programa debe ser implementada en un sólo lugar del código fuente”
- La mejor solución a un problema es la más simple. Ejemplo:

```
# EJEMPLO DE UNA SOLUCIÓN POCO SIMPLE

i=0
j=100
while ( i < j ):
    # Este ciclo tiene 50 iteraciones.
    i = i+1
    j = j-1


# EJEMPLO DE UNA SOLUCIÓN SIMPLE

i = 0
while ( i < 50 ):
    # Este ciclo también tiene 50 iteraciones.
    i = i + 1
```

- Hacer pruebas unitarias, es decir probar partes pequeñas del código, como subprogramas, de forma individual.
- Usar identificadores descriptivos. Los nombres de las funciones, variables, etc. deben declarar claramente lo que hacen. Ejemplos:

```
# EJEMPLO DE NOMBRES POCO DESCRIPTIVOS
dia0 = int( input( "Dia de nacimiento: " ) )
mes0 = int( input( "Mes de nacimiento: " ) )
año = int( input( "Año de nacimiento: " ) )
```

```
dia1 = int( input( "Dia actual: " ) )
mes1 = int( input( "Mes actual: " ) )
año1 = int( input( "Año actual: " ) )


# EJEMPLO DE NOMBRES DESCRIPTIVOS
diaDeNacimiento = int( input( "Dia de nacimiento: " ) )
mesDeNacimiento = int( input( "Mes de nacimiento: " ) )
añoDeNacimiento = int( input( "Año de nacimiento: " ) )
diaActual = int( input( "Dia actual: " ) )
mesActual = int( input( "Mes actual: " ) )
añoActual = int( input( "Año actual: " ) )
```

- Principio del menor asombro: El nombre de un subprograma debería corresponder con lo que hace.
- Principio de responsabilidad única. Un componente de código (subprograma) debe ejecutar una única y bien definida tarea.
- Minimizar el acoplamiento. Cada componente (bloque de código, subprograma, etc.) debe minimizar las dependencias de otros componentes.
- Maximizar cohesión. Evitar implementar en un componente dos funcionalidades que no están relacionadas, cumpliendo tareas que no tienen relación.
- La reutilización de código es buena.
- No usar más parámetros de entrada de los que se necesitan. Asegurarse de que los parámetros recibidos son los que se esperan.

Programación en Pareja.

La Programación por Parejas es considerada como uno de los pilares de una metodología ágil. Es una técnica relativamente sencilla de aplicar en la cual dos programadores trabajan codo con codo, en un único computador, para desarrollar el código de un programa. En este contexto, la persona que teclea recibe el nombre de conductor, mientras que la persona que revisa el código recibe el nombre de observador o navegante. Estos dos roles, conductor y observador/navegante, en ningún momento, deben ser considerados como roles estáticos.

Los miembros de la pareja deben intercambiar sus roles de manera frecuente durante toda la actividad de desarrollo. Si no se cumple este sencillo requisito, no se podrá hablar de una verdadera Programación por Parejas. El objetivo de la Programación por Parejas es mejorar la calidad del código. Las buenas prácticas en la programación por parejas son:

- Todo se comparte: Los dos miembros de la pareja, sin importar el rol que estén desempeñando en un determinado momento, son los autores del código. Ambos tendrán que aprender a compartir sus éxitos y sus fracasos, expresándose mediante

frases como "cometimos un error en esta parte del código" o, mucho mejor, "hemos pasado todas las pruebas sin errores".

- Observador activo: No debe caerse en el error de confundir el rol de observador con disponer de un periodo de descanso, realizar una llamada o contestar a un mensaje de texto recibido en el teléfono. Por el contrario el observador activo ocupa su tiempo realizando un proceso continuo de análisis, diseño y verificación del código desarrollado por el conductor, tratando en todo momento de que la pareja sea capaz de alcanzar la excelencia del código.
- Deslizar el teclado, no cambiar las sillas: Esta regla nos indica que el espacio de trabajo debe estar dispuesto de modo que no sea necesario intercambiar el lugar frente al teclado (cambiar las sillas) para poder intercambiar los roles. Con el simple movimiento de deslizar el teclado para que éste cambie de manos la pareja debe ser capaz de intercambiar los roles de conductor a observador/navegante y viceversa.
- La importancia del descanso: Deben realizarse descansos de manera periódica, que ayudarán a la pareja a retomar el desarrollo con la energía necesaria, por lo que es imprescindible que,durante los mismos, los dos desarrolladores mantengan sus manos y su mente alejados de la tarea que estén desarrollando en pareja. Algunas tareas que se pueden realizar durante estos periodos de descanso son: revisar el correo electrónico personal, realizar una llamada de teléfono pendiente, contestar a los mensajes de texto recibidos en el móvil, navegar por la red, comer,incluso, estudiar otra materia.

Programación Sostenida.

Es importante llevar un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es el de planificar el trabajo para mantener un ritmo constante y razonable, sin sobrecargar al equipo. Cuando un proyecto se retrasa, trabajar tiempo extra puede ser más perjudicial que beneficioso. El trabajo extra desmotiva inmediatamente al grupo e impacta en la calidad del producto.

Programación dirigida por las pruebas (*Test-driven programming*)

En las metodologías tradicionales, la fase de pruebas, incluyendo la definición de los tests, es usualmente realizada sobre el final del proyecto, o sobre el final del desarrollo de cada módulo. Las metodologías ágiles proponen un modelo inverso, en el que, lo primero que se escribe son las pruebas que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Estas pruebas, llamadas pruebas unitarias, corresponden a probar cada componente de software (o subprograma) de manera individual. Para ello, hay que intentar que los casos de prueba elegidos cubran la posibilidad de encontrar el mayor número de errores posibles, ya que los errores permanecerán ocultos hasta que se ejecute la porción de código que los provoca y para que dicha porción se ejecute es necesario que el programador realice las acciones necesarias para provocarlo.

Algunas recomendaciones para hacer pruebas: verificar el tipo de datos de entrada, las precondiciones, los resultados esperados (correctos o incorrectos). En resumen se está creando los "escenarios de pruebas", es decir, las diferentes condiciones en las que el programa deberá trabajar.

Ejercicios:

1. Para cada uno de los ejercicios del laboratorio 6, escriba en un documento en donde coloque las pruebas realizadas. Para ello llene una tabla con las siguientes columnas (se muestra un ejemplo):

Nombre del Programa	**Escenario de Prueba**	**Resultado Esperado**
Lab06Ejercicio1.py	Secuencia vacía	Mensaje: "No se admite secuencia vacía. El programa terminará…"

   En los escenarios de pruebas debe haber al menos: 2 entradas correctas y 4 entradas incorrectas. Guarde el documento con "PruebasLab6.pdf" y súbalo en el aula virtual.

2. Utilice los principios de desarrollo ágil, programación en pareja y ritmo sostenido durante esta semana para realizar la entrega 1 del proyecto.

Referencias

[1] José H. Canós, Patricio Letelier y Mary Carmen Penadés, "Metodologías Ágiles en el Desarrollo de Software", DSIC Universidad Politécnica de Valencia. 2003. Disponible en la web. http://www.willydev.net/descargas/prev/TodoAgil.Pdf

[2] Scrum Manager®, Programación por Parejas, Disponible en la web: http://www.scrummanager.net/bok

[3] Christopher Diggins, "The Principles of Good Programming", July 24, 2011. Disponible en la web: http://www.artima.com/weblogs/viewpost.jsp?thread=331531