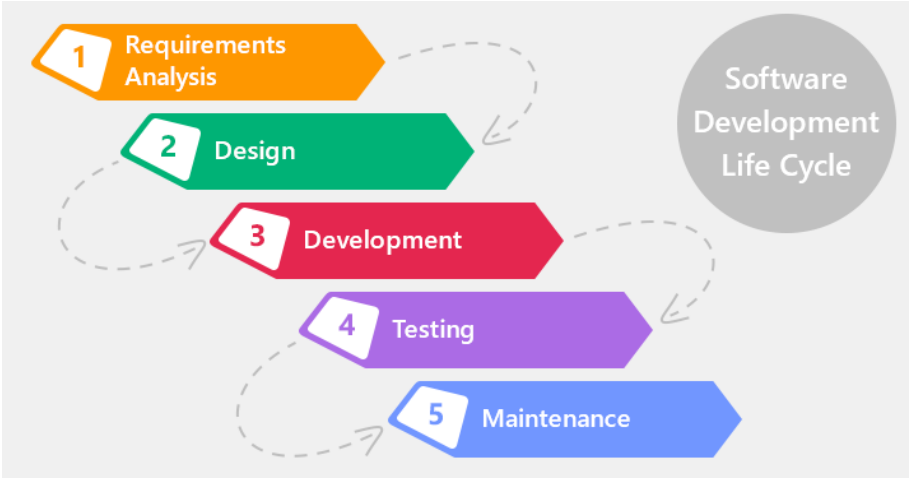
UT5 – OPTIMIZACIÓN Y DOCUMENTACIÓN

PARTE 2: DOCUMENTACIÓN

Introducción

- **¿A qué le llamamos documentación de software?**
- En realidad serán todos los documentos relacionados con el producto durante su desarrollo, uso y mantenimiento.
- Es un término muy genérico.
- Durante el ciclo de vida del software se producen infinidad de documentos:
  - Documentos de análisis y diseño.
  - Comentarios en código.
  - Manuales de usuario...



Introducción

- El objetivo de la documentación es que el software sea entendido por desarrolladores, usuarios, mantenedores, instaladores, etc.
- En el caso concreto de los desarrolladores, además de la documentación, ayuda la refactorización. Sin embargo, la refactorización no incluye información en lenguaje natural, por lo que la documentación sigue siendo igualmente necesaria.
- Además en el desarrollo web suele incluir documentación necesaria para el despliegue.
- En esta unidad vamos a tratar tres temas específicos asociados a la documentación, los comentarios, y las herramientas automáticas de documentación.

Comentarios

- Tras la refactorización del código, el uso de comentarios debería ser menos importante. El código debe ser tan claro que no necesite explicación alguna.
- Sin embargo, los comentarios van a ayudar a que el proceso de comprensión del código sea más ágil, además de permitir incluir notas para poder manejar diversas versiones, partes del código que puedan ser opcionales, etc.
- Además de notas aclaratorias y de definir partes opcionales, los comentarios sirven también para documentar funciones o métodos. Entradas, salidas esperadas, objetivo, etc. es parte de la información que se suele incluir en la documentación de funciones o métodos.
- Que un código esté bien comentado es casi tan importante como que no tenga errores.

Comentarios

- El objetivo a la hora de comentar un código es que un programador pueda usarlo (bien para modificarlo, o bien para usar una función o método) sin tener que mirar el código.
- Desde un punto de vista técnico hay dos formas de comentar. Comentar una línea o comentar un bloque de líneas. En cualquier caso, los comentarios no serán evaluados.
- Los caracteres que permiten comentar una línea dependen del lenguaje de programación utilizado. Por ejemplo, en C, C++ y Java se usa // para comentar una línea y /* para comenzar un bloque comentado y */ para cerrarlo.

Comentarios: Donde Ubicarlos

- Al principio de cada archivo:
  - Llamado “Header comment” o comentario de cabecera. Debe incluir el autor del código, la fecha y el objetivo del código.
- Antes de una función:
  - Llamado “Function header” o cabecera de función. Debe describir la función (entradas, salidas, propósito, etc.)
- En una línea:
  - Cualquier instrucción que no es obvia o cuyo código puede ser difícil de entender debe llevar un comentario inmediatamente antes o en la misma línea. Se suele llamar código autodocumentado (“Self documented code”) a aquel que usa nombres de variables, de funciones, etc. apropiadas para no tener que introducir documentación extra. El uso excesivo de “i”, “j”, “aux”, etc. hace que el código sea menos legible.

Comentarios: Prácticas a evitar

- No se deben hacer comentarios obvios.
  - int x = 5; // asignamos el valor 5 a la variable x
- Los comentarios no deben ser excesivamente largos.
  - int average = (x + y)/2; //calcula la media de x e y sumando las dos variables y dividiendo entre 2.
  - int average = (x + y)/2; //media de x e y
- No se deben mezclar idiomas en los comentarios.
  - // Creates a reader instance which takes
  - // input from standard input - keyboard
  - Scanner reader = new Scanner(System.in);
  - System.out.print("Enter a number: ");
  - // Lee un número introducido por teclado.
  - int number = reader.nextInt();
- Vamos a usar preferiblemente el inglés en los comentarios.

Comentarios: Cabeceras de archivo y función

- La cabecera de archivo debe incluir:
  - El autor, la fecha y la organización o el marco en el que se engloba el archivo (p.ej. “Universidad…” o “Proyecto…” o “Copyright…”)
  - Una descripción de lo que hace el código presente en el archivo (clases, funciones etc.). Un programador debe poder leer esta parte y entender sin mirar el código cual es la idea general del programa así como las estructuras de datos y métodos usados.
  - Una lista de modificaciones asociadas a la versión del documento (p.ej. fallos corregidos).
- Toda función/método debe incluir una cabecera de función. Dicha cabecera debe permitir al lector entender los algoritmos detrás del código sin tener que interpretarlo. Es importante que quede claro cuál es la cabecera de cada función. Para eso se debe dejar el suficiente espacio en blanco y el documento, en general, debe tener un formato adecuado.
- Veamos un ejemplo de cada una de las cabeceras.

Comentarios: Cabeceras de archivo. Ejemplo.


```
/**
 * File:    compute_blackjack_odds.C
 *
 * Author1:  H. James de St. Germain (germain@eng.utah.edu)
 * Author2:  Dav de St. Germain (dav@cs.utah.edu)
 * Date:     Spring 2007
 * Partner:  I worked alone
 * Course:   Computer Science 1000
 *
 * Summary of File:
 *
 *   This file contains code which simulates a blackjack game.
 *   Functions allow the user of the software to play against the
 *   "casino", or to simulate the odds of successfully "hitting"
 *   given any two cards.
 *
 */
```

Comentarios: Cabeceras de archivo. Ejemplo.


```
/**
 * void sort( int array[] )
 *
 * Summary of the Sort function:
 *
 *      The Sort function, rearranges the given array of
 *      integers from highest to lowest
 *
 * Parameters   : array: containing integers
 * Return Value : Nothing -- Note: Modifies the array "in place".
 *
 * Description:
 *      This function utilizes the standard bubble sort algorithm...
 *      Note, the array is modified in place.
 */

void
sort( int array[] )
{
      // code
}
```

Generadores de Documentación

- La documentación de código no suele ser muy efectiva.
- Las razones son:
  - Se considera una tarea que lleva mucho tiempo.
  - Se considera tediosa.
  - Se debe actualizar continuamente para que sea útil.
- Por estas razones surge los generadores de documentación.
- Estos programas analizan los comentarios hechos en el código y generan una serie de documentación de forma automática.
- La tarea de documentar se vuelve menos tediosa aunque sigue dependiendo del criterio de un programador.
- La actualización de código y documentación se realiza en el mismo sitio.

JavaDoc

- Propiedad de Oracle.
- Genera documentación HTML de la API a partir de los comentarios de código Java.
- Funciona con una serie de etiquetas que empiezan por @ y que establecen el significado del comentario.

```
// import statements

/**
 * @author          Firstname Lastname <address @ example.com>
 * @version         1.6                (current version number of program)
 * @since           1.2          (the version of the package this class was
first added to)
 */
public class Test {
    // class body
}
```