Together for Tomorrow!
Enabling People

Documentación técnica:

Fundamentos de la programación

#TecnologíaConPropósito

Índice

1. Programas, datos y algoritmos

Un programa informático consiste en una serie de instrucciones que indican al computador de forma precisa lo que debe hacer, consiguiendo que la máquina resuelva el problema que nos interesa. Para lograrlo, el programa debe estar escrito en un lenguaje que entienda el computador, y que denomina **lenguaje-máquina**. Sin embargo, esos códigos resultan muy crípticos y poco entendibles para los seres humanos. Por eso, los programadores escriben los programas en lenguaje pseudo-natural, que comprenden ellos, con lenguajes apropiados como Java, C, Python, PHP u otros muy conocidos, y luego utilizan herramientas que convierten las instrucciones en estos lenguajes al lenguaje que realmente entienden las máquinas. Estas herramientas se llaman **compiladores**.

1. Programas, datos y algoritmos

Los programas suelen aceptar unos **datos de entrada**, por ejemplo, cuando seleccionamos una fecha y un destino para comprar un billete de avión, que luego procesan mediante unos **algoritmos**. Para una agencia de viajes on-line, los algoritmos serían la búsqueda de precios más baratos, las compañías aéreas más seguras, la disponibilidad de plazas libres suficientes en las fechas escogidas. Cuando el programa resuelve el algoritmo, se da una respuesta al usuario, siguiendo con el ejemplo, la posibilidad de comprar los billetes, o bien el computador proporciona unos **datos de salida** que a su vez podrán ser procesados de nuevo por otro programa, por ejemplo, para comprar con otra compañía un seguro de viajes.

Los algoritmos son parte fundamental de los programas. Cada uno de ellos está formado por un conjunto finito de líneas de código, de instrucciones, que tendrán que estar muy bien definidas y que deberán acabar en un tiempo también finito. No tiene sentido escribir algoritmos que duren una eternidad, la idea básica es que los algoritmos son como una receta práctica para resolver un problema concreto.

2. Ingeniería del software

En los inicios de los computadores, el hardware no era muy potente y por ello el software no era demasiado complejo. El desarrollo de software era casi un arte, y su calidad final dependía en gran medida de la experiencia y habilidad del programador. Poco a poco, las cosas fueron cambiando y los sistemas informáticos se hicieron grandes y complejos. Las planificaciones de tiempo y coste demostraron que las técnicas artesanales ya no servían y en los años 90's se vivió una gran crisis del software que finalizó incorporando al desarrollo del software parte de las técnicas ingenieriles utilizadas en las fábricas. Así, se definió el término **Ingeniería del Software** como la aplicación sistemática, disciplinada y cuantificable al desarrollo, operación y mantenimiento del software.

Con este nuevo punto de vista, el desarrollo de software trasciende de la mera programación. La ingeniería del software considera un **ciclo de vida** que va desde la detección y análisis del problema que se quiere informatizar, la concepción y diseño del programa, su implementación en código que entienda la máquina, la realización de pruebas y detección de errores, su uso y su mantenimiento, como se muestra en la siguiente figura.

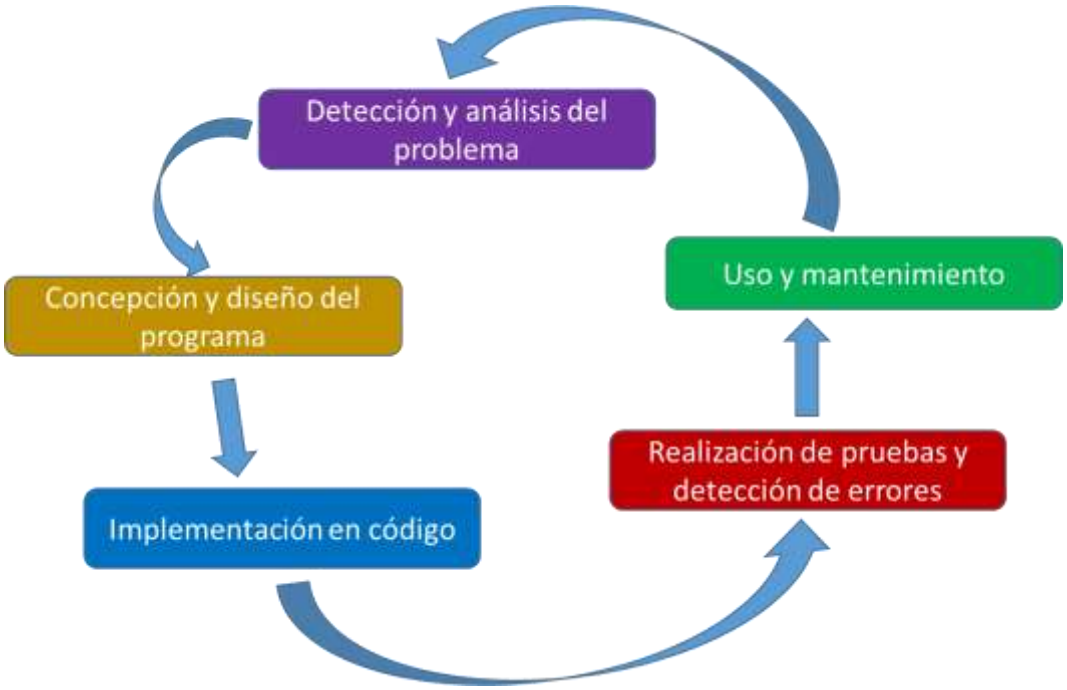

Figura 1. Fases dentro del ciclo de vida del software

2. Ingeniería del software

Un programa informático se considera **eficaz** si consigue los objetivos deseados y funciona correctamente. En cambio, un programa será **eficiente** si consigue los resultados consumiendo los menores recursos y tiempo posible. Un programa no solo es bueno por funcionar bien, sino que será mejor cuántos menos recursos utilice.

Al escribir un programa siempre se producen errores y nunca se pueden evitar del todo. Por ello, un proceso fundamental en la ingeniería del software es la **depuración** de los errores y considerar que hay que realizar operaciones de mantenimiento con una determinada frecuencia. Para facilitar el trabajo, se pueden incluir fases específicas que estén ya sistematizadas. Los errores se pueden clasificar en distintos tipos y es muy importante intentar eliminarlos en el proceso de desarrollo del software:

1. **Errores de compilación**: estos errores son los más fáciles de corregir ya que son detectados por los programas compiladores. Pueden ser errores de sintaxis en las instrucciones o uso de tipos de datos incompatibles.
2. **Errores en tiempo de ejecución**: aunque el código del programa compile bien y no dé problemas puede ocurrir que dé un error grave al ejecutarse y el programa se pare y deje de funcionar. Por ejemplo, que se produzca una división por 0 (el valor infinito no existe para las máquinas) o se escriba mal en un fichero del disco duro. Algunos de estos errores pueden ser recuperables en tiempo de ejecución y entonces se denominan **excepciones.** Para ello, los programas deben contener subrutinas específicamente diseñadas para atender cada tipo de excepción.
3. **Errores lógicos**: este caso se produce cuando no hay errores de compilación ni de ejecución, pero el programa produce un resultado incorrecto, que no es el esperado o incluso que es imposible. Por ejemplo, el precio de un billete de avión en cifras negativas (menor que cero). Estos errores son los más importantes pero los más difíciles de descubrir ya que habitualmente exigirá revisar con mucho cuidado grandes trozos de código de la aplicación contrastando resultados intermedios, que pueden suponer miles de líneas de programación.

SAMSUNG