Desarrollar software

1) Qué es el desarrollo
2) El universo de nuestro software
3) Aplicación vs. Script
4) Aplicación web / no web
5) Tips / enunciados

1. Qué es el desarrollo de software

a. Es un proceso iterativo
b. No es solamente programar
c. Es la traducción de reglas de negocio a instrucciones lógicas precisas
d. Es el definir el universo que representará las instrucciones lógicas

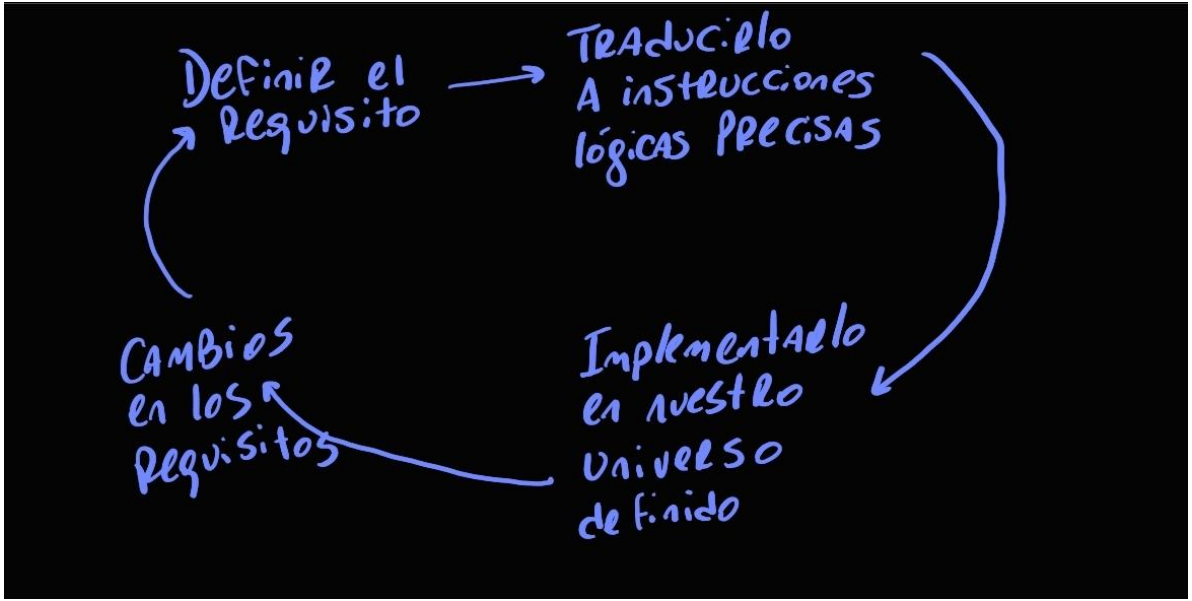
1.a. Desarrollar software es un proceso iterativo

Software etimológicamente significa “producto blando” (soft + ware). Un objeto maleable, no estricto, modificable y adaptable.

Permite la modificación de un sistema de uso específico.

La modificación de este sistema es causado por los cambios de requerimientos

La implementación de estos cambios debe ser considerada como una tarea más en el proceso de implementar nuevos requisitos o requisitos modificados en cuanto a que lo que el software representa lo necesite.



Qué genera los cambios en los requisitos

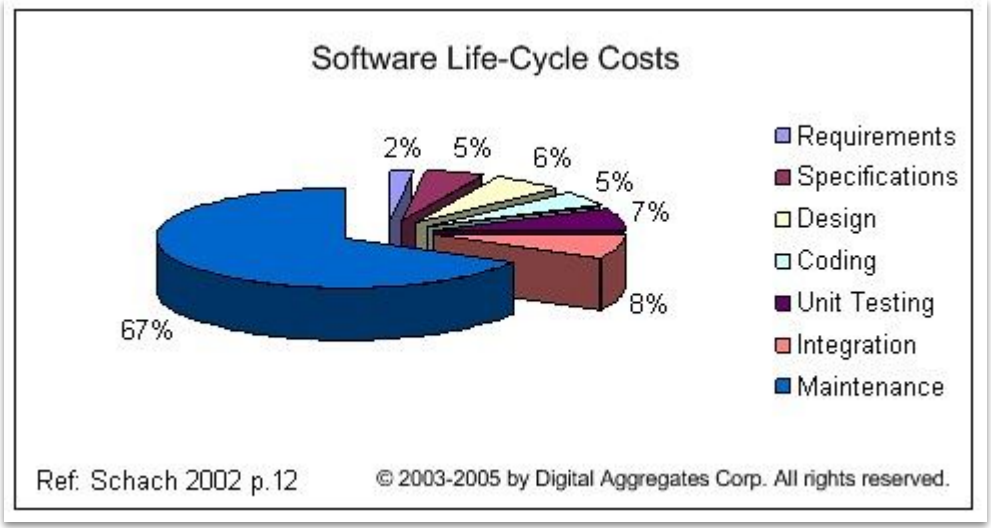
Los requisitos son propensos al cambio a causa de:

- Cambios en las reglas de negocio (reglas, enunciados que existirían incluso sin software. Por ejemplo, la inclusión de una nueva moneda en un sistema de transacciones)
- Cambios en el universo de nuestro software (una vulnerabilidad de seguridad en una librería, la necesidad de soportar más usuarios que los pensados originalmente)
- Mantenimiento del mismo software

1.b. Desarrollar software no es solo programar

Programar es la acción de traducir los requisitos que dan sentido a nuestro software a instrucciones lógicas, lo que conforma una pequeña parte del desarrollo. Entre estas tareas tenemos:

- Definición de requisitos
- Diseño de software
- Diseño de la infraestructura (universo)
- Programar
- Integrar lo programado al sistema existente
- Testear el software
- Mantenimiento de software



1.c. La traducción de reglas de negocio a instrucciones lógicas precisas

Desafortunadamente, aún la tecnología no permite que le cobremos a un usuario 680$ por un Sixpack Imperial e imprimamos el comprobante con estas 3 líneas de código:

```
1 const usuario = registrar_usuario();
2 const cobro = cobrar_usuario(usuario, 680, 'Sixpack Imperial');
3 imprimir_cobro($cobro);
```

Para llegar a eso, primero deberíamos definir qué es “registrar_usuario”, qué es “cobrar_usuario”, qué es “imprimir_cobro”.

Y para ello, es necesario desmenuzar lo más posible el requisito en instrucciones lógicas precisas.

registrar_usuario es: capturar los datos del usuario, agruparlos, guardarlos en una base de datos.

De la misma manera, debemos seguir desmenuzando: capturar los datos del usuario es: tener un campo de texto llamado ‘nombre’, un botón llamado ‘registrar’, una acción asociada a ese botón, etc, etc, etc.

Ejemplo más completo en pseudocódigo (código ficticio)

```
const usuario = document.getElementById('campo_nombre').value;
const cobro = cobrar_usuario(usuario, 680, 'Sixpack imperial');
imprimir_cobro(cobro);

function registrar_usuario(nombre_de_usuario){
  const usuario = new Usuario();
  usuario.nombre = nombre_de_usuario;
  return usuario;
}

function cobrar_usuario(usuario, precio, producto){
  const transaccion = new Transaccion();
  transaccion.to(usuario).setPrice(precio).setDescription(producto);
  transaccion.emit();
  return transaccion;
}

function imprimir_cobro(cobro){
  const impresora = System.Printers.HP238;
  const texto = `Se cobró ${cobro.precio} a ${cobro.usuario} por ${cobro.producto}`;
  impresora.imprimir(texto);
}
```

1.d. Representación de instrucciones lógicas en un universo

En el ejemplo anterior, nuestras instrucciones lógicas ‘registrar_usuario’, ‘cobrar_usuario’ e ‘imprimir_cobro’ estaban escritas en Javascript.

Con Javascript obtenemos el nombre de usuario desde un campo en un formulario, imprimimos un documento mediante el objeto accesor de ‘impresora’, importado desde la librería System.Printers y emitíamos una transacción con el objeto accesor ‘Transacción’ de la librería ‘transactions’.

Todo esto es parte de nuestro universo, el encargado de representar nuestras instrucciones lógicas.

2. El universo de nuestro software

Correctamente llamado “infraestructura de software”, es la selección de lenguajes, librerías, metodologías, protocolos, configuraciones, programas, etc. El software que va a ejecutar nuestro software.

El diseño de la infraestructura es un proceso arquitectural.

¿Cómo definir un universo?

La definición del universo (de ahora en más infraestructura) es una tarea de investigación de campo, jugarsela y elegir.

- Buscar en google una implementación de un caso de uso similar al que se está trabajando (‘cómo crear un conversor de MP3 a FLAC’, ‘cómo crear una aplicación de ecommerce).
  - Lo interesante de estos resultados no es, cómo hacerlo, si no, qué tecnologías y metodologías son las que más aparecen.
  - También puede buscarse ‘qué tecnología es la mejor para...’
- Tener en cuenta los conocimientos propios
  - Es mil veces mejor trabajar con una tecnología conocida que con una ‘más nueva’
- Leer la documentación oficial de la librería (siempre) para identificar posibles problemas o ventajas
- Elegir

Script versus aplicación: las dos son software, los dos son programas

Las aplicaciones están orientadas al software que va a ser utilizado por usuarios, personas humanas. Es necesario un motor visual (conocido como front-end, ui o gui -user interface o graphical user interface-) que interactúe con la funcionalidad computacional (backend, 'lo que hace las cosas', 'lo que calcula', 'lo que inserta en la base de datos').

Contrariamente, los scripts son programas destinados a ser utilizados por la computadora, sin un motor visual que represente sus funcionalidades. Es común desarrollar scripts para tareas específicas, con un campo de datos específico.

Aplicación

- Programa destinado al usuario humano
- Motor gráfico que represente las funcionalidades
- Validaciones estrictas de datos
- Pensadas para una variedad de usos más amplia
- La invocación a la aplicación no es controlada (no sabemos qué va a hacer el usuario)

Script

- Programa destinado a la realización de una tarea específica
- Sin una validación de datos tan estricta
- La invocación al script es controlada (sabemos qué vamos a hacer nosotros)

Desarrollo web/no web

“La web” es un modelo de infraestructura, donde desde un comienzo está definido el protocolo de intercambio de datos (HTTP, Hyper Text Transfer Protocol), el motor gráfico (el navegador) y una limitación en la selección de lenguajes y librerías.
Una aplicación ‘no web’, podría estar de igual manera estar escrita en un lenguaje de programación que usualmente se use para la programación web, podría también usar el protocolo HTTP, pero no usar el navegador para mostrarse.

¿Qué hace que una aplicación sea web? La utilización del protocolo HTTP, ser renderizada en el navegador web y ser accesible desde el internet. Pero la aplicación puede ser de cualquier manera y cumplir cualquier funcionalidad. También una aplicación no web puede ser convertida a web, o ejecutada en un ambiente web, y viceversa.

En sintesis, no hay mucha diferencia entre una aplicación web y no web, personalmente, no utilizaría tal concepto para diferenciar aplicaciones.

La diferenciación más apropiada en mi opinión, pasa por la funcionalidad de la aplicación, si es una aplicación orientada a usuarios, a desarrolladores, o a la misma computadora.

Tips y enunciados

- Todo es posible
- Y probablemente alguien lo haya hecho
- No por eso no lo vas a hacer
- Googlear es esencial, no es sinónimo de no saber, es sinónimo de querer saber más.
- Desarrollar software también es diseñarlo. Diagramar sirve un montón.
- No es lo mismo leer que ver un video tutorial
- No hay mejor información que la documentación oficial de la tecnología que usemos
- Siempre leé los errores