



7 al 18 de Marzo@ Df UBA  
CABA, Argentina

# Introducción al desarrollo de Software

Cecilia Jarne

[cecilia.jarne@unq.edu.ar](mailto:cecilia.jarne@unq.edu.ar)

## Ideas básicas para empezar

- 1) Herramientas y procesos de organización, desarrollo y mantenimiento de soft.
- 2) Estrategias y **buenas prácticas**.
- 3) Desarrollo de soft **colaborativo**.
- 4) Favorecer el uso de soft **reutilizable**.
- 5) Trabajo en grupo para **poner en práctica** estas ideas.

# Desarrollo de Software

## 1) Procesos de organización y desarrollo

-¿Para quién estoy programando?

Para mí, Para nosotros, Para terceros...

(set de requerimientos que necesito, acordar con el par o hacer un buen esquema de lo que necesita quien encarga la tarea.)

-¿Que tareas se necesitan, cómo las vamos a implementar?

(usar si es necesario un papel para un primer esquema.)

-Traducir los requerimientos en sub-tareas

-Usar algún método de project management

(para ser estrictos con los tiempos de desarrollo, implementación y optimización.)

-No inventar la pólvora!

(Invertir algún tiempo en investigar si actualmente existe una manera de implementar las cosas documentada)

## 2) Estrategias y buenas prácticas.

- Invertir algo de tiempo en pensar que es necesario y posibilidades de implementación.  
(Observar ejemplos antes de decidir la mejor estrategia.)
- Comunicarse fluidamente con el que pide el software o con quien desarrolla junto con nosotros.
- Documentar el proceso.  
(Utilizar las herramientas existentes)

## 2) Estrategias y buenas prácticas.

-Invertir algo de tiempo en pensar y leer.  
(Observar ejemplos antes de decidir la mejor estrategia)

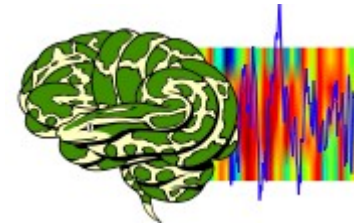
ALGUNAS IDEAS DE DONDE MIRAR:

<http://stackoverflow.com/>

<http://numerical.recipes/>

<https://projecteuler.net/>

<http://nipy.org/nitime/index.html>



(este es un ejemplo un poco más específico de sitios que existen para distintos temas)

# Desarrollo de Software

## 3) Desarrollo de soft colaborativo.

- Escribir software mas modular y reusable.  
=> Escribir frameworks y librerias.

- Software modificable.

Donde las componentes puedan ser combinadas sin tener que recompilar (si se puede)

=> Combinar codigo script y compilado.

- Intentar que las componentes puedan ser (re-)testeadas y (re-)validadas.

-BUENA DOCUMENTACION SOBRE EL SOFT QUE ESCRIBIMOS!!!!!!

# Desarrollo de Software

## 3) Desarrollo de soft colaborativo.

Algunos ejemplos de soft colaborativo

<http://nipy.org/nitime/index.html>

<https://root.cern.ch/>

<http://auger.le.infn.it/dpa.html>

<http://audiotools.sourceforge.net/>

**Collaboration**



# Desarrollo de Software

## 4) Favorecer el uso de soft reusable.



- Intentar escribir módulos para combinar o usar script code dependiendo de las características y tamaños del problema a resolver.
- El software científico tiene características propias que hacen que uno se incline por estas herramientas.



## ¿Qué hace distinto al soft científico?

- Los requerimientos no están del todo definidos a veces.
- Limitaciones en el cálculo de punto flotante pueden perjudicarnos.
- Algunas aplicaciones pueden ser usadas una vez.
- No todos los científicos saben programar o hacerlo del modo mas eficiente.
- A veces las implementaciones deben ser hechas por gente inexperta.

## ¿Por qué usar lenguaje de script?

- Portabilidad.
- No hay necesidad de re-compilear.
- Disponibilidad de librerías en la propia plataforma.
- Flexibilidad.
- Adaptabilidad.
- Posibilidad de adaptar múltiples extensiones de archivos.
- Conveniencia.
- Los script lenguajes tienen gran facilidad para pre- y post-proceso de datos.
- Las partes que lleven tiempos grandes se pueden hacer en lenguaje compilado.

# Desarrollo de Software

## ¿Por qué usar lenguaje modular y librerías?

- Muchas tareas distintas requieren enfoques de cálculo o proceso análogos.
- Algunas tareas difieren solo en el subset de datos.
- Cálculos de operaciones comunes.  
(Fast Fourier transforms, basic linear algebra, etc.)
- Los datos pueden colocarse en archivos estructurados soportados por herramientas de análisis y visualización comunes.
- Gran potencial de re-uso de código!!!
- Los módulos independientes pueden ser validados más fácilmente.

# Desarrollo de Software

## ¿Por qué usar lenguaje modular y librerías?

**Provee niveles de abstracción.**

-> No es necesario saber como funciona TODO siempre.

-> Oportunidad para hacer más clara la optimización.

**Acceso organizado a los datos.**

-> datos+funciones para modificarlos.

-> control de acceso de solo lectura o lectura-escritura.

# Desarrollo de Software

## ¿Por qué usar documentación embebida?

Tres tipos de documentación se necesitan.

- **Información para los desarrollares que quieren agregar código.**

- > Documentación API (e.g. via doxygen)

- > Comentarios en el código que explican elecciones.

- **Información para los usuarios.**

- > Manual de referencia para los usuarios.

- **Información para los usuarios que quieren aprender a usar una herramienta específica.**

- > Tutoriales, HOWTO?

Arrancar a escribir la documentación desde el comienzo.

## Source Code Management

- Un lugar para colocar los archivos fuente y una herramienta de comunicación entre los desarrolladores.
- Un modelo de código distribuido entre los desarrolladores.
- Trabajar con varias ramas y mergeear luego.
- Hacer los cambios de a poco y no combinar cambios que no se relacionan en un mismo commit.
- Tener una documentación consistente de los cambios y las reglas para realizarlos.

## Conclusiones

- Intentar quitar malos hábitos de programar solo.
- Adoptar herramientas tecnológicas útiles que pueden hacer más exitosa la tarea del desarrollo colaborativo.
- O al menos poder hacer crosscheck de nuestro trabajo con otros!