# Introducción a la ingeniería de software

Verónica E. Arriola-Rios

Facultad de Ciencias, UNAM

4 de octubre de 2020

# Crisis del software

# La crisis del software

- El término *crisis del sofware* se originó en la primer conferencia de *Ingeniería de Software de la OTAN* en 1968 en Alemania, con la cual nacía este campo.
- Edsger Dijkstra lo describió en 1972 del modo siguiente:

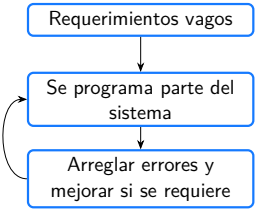
  *"La principal causa de la crisis del software es que ¡las computadoras se han vuelto varios órdenes de magnitud más poderosas! En concreto: mientras no había máquinas, la programación no era un problema; cuando tuvimos unas pocas computadoras modestas, la programación se convirtió en un ligero problema; ahora que tenemos computadoras gigantes, la programación se ha convertido en un problema igualmente gigante."*[1]

Es decir, "el incremento en la capacidad de cómputo rebasó la habilidad de los programadores para utilizar esas capacidades efectivamente."[2]

[1] The humble programmer by Dijkstra

[2] Software crisis in Wikipedia

# Problema: el método *"programa y arregla"* University of Cape Town 2011




- Si nos recuerda nuestra forma de programar... sólo sirve para programas pequeños.
- No hay manera de estimar presupuestos.
- No es posible evaluar riesgos errores en el diseño: es posible llegar a tener un producto **casi terminado** con problemas técnicos irresolubles y tener que volver a empezar.

Ejemplos: Wikipedia: List of failed and overbudget custom software projects

En estos años varios proyectos fallaron al no poder escalar sus técnicas para sistemas pequeños a sistemas más grandes y complejos con los problemas siguientes:

- Sistemas que nunca pudieron ser completados,
- fechas de entrega rebazadas,
- presupuestos excedidos,
- sistemas que no lograron cumplir con todos los requerimientos,
- sistemas ineficientes
- sistemas que funcionaban, pero eran muy difíciles de usar,
- sistemas muy difíciles de modificar para ajustarse cambios en necesidades y prácticas organización que los usaba,
- pérdida de confianza de los usuarios, quienes experimentaban problemas al usar el software. University of Cape Town 2011

# Nuevos paradigmas de programación

# Temas

# Programación procedimental

- Introducen el uso de *procedimientos* o *subrutinas*, de donde derivan las actuales **funciones**, que operan sobre *variables* y *estructuras de datos*.
- Permiten la creación de *bibliotecas* o *librerías* de procedimientos desarrollados por diferentes programadores o incluso equipos.

Los primeros lenguajes procedimentales fueron:

- Fortran, ALGOL, COBOL, PL/I y BASIC entre 1957–1964
- Pascal y C entre 1970 y 1972.

https://en.wikipedia.org/wiki/Procedural_programming

# Cualidades

Las cualidades que promueven son:

**Modularidad** Las entradas se reciben como *argumentos* y las salidas se entregan como *valores de regreso*.

**Alcance** Evita que un procedimiento tenga acceso a los datos (variables) de otros procedimientos.

# Temas

# Programación orientada a objetos

- Se basa en el concepto de *objeto*.
- Cada objeto contiene su propios datos como *atributos* y *métodos* para comunicarse con el objeto y manipular esos datos.
- El paradigma[3] de programación orientada a objetos (POO) es un modelo de organización de programas y una colección de patrones y prácticas para diseñar programas; no se centra en el desarrollo de algoritmos Eliëns 1994.

[3] Para una discusión más detallada de lo que significa la palabra *paradigma* a secas, consultar Porto y Merino Publicado: 2008. Actualizado: 2012.

# Cualidades

Las cualidades que promueven son:

**Encapsulamiento** Los datos de cada objeto sólo pueden ser manipulados por el objeto mismo a través de sus métodos.

**Herencia** Permite reutilizar código mendiante los mecanismos de *generalización* y *especialización* en una jerarquía de tipos de objetos, donde objetos de tipos más específicos comparten el código de acciones comunes a todos ellos en clases *ancestras*.

# Pioneros

- En 1962, Kristen Nygaard y Ole-Johan Dahl, en el centro de computación de Noruega, crean el lenguaje **Simula**.
- En 1970, Alan Kay, Dan Ingalls y Adele Goldberg, lanzan la primera versión de **Smalltalk**.

Kay escribe al respecto:

> *Imaginé a los objetos como células biológicas y/o computadoras individuales en una red, siendo capaces solamente de comunicarse a través de mensajes.*

- Entre 1974 y 1975 Barbara Liskov y sus estudiantes del MIT desarrollan CLU, aunque no incluía herencia.

# Ciclo del software

# Etapas del ciclo del software


- *Especificación* de los requisitos del cliente.
- *Análisis* del problema a resolver y sus posibles soluciones. Plan de trabajo para completar la solución, tomando en cuenta posibles eventualidades.
- *Diseño* o modelado de la solución a nivel abstracto.
- *Implementación* en una plataforma con lenguaje(s) de programación seleccionados.
- *Pruebas* de la correctez y funcionalidad de la solución implementada. Vendría luego la entrega o *despliegue*.
- *Mantenimiento*: corrección de errores detectados, mejoras del sistema, extensión de sus funcionalidades, adaptación a nuevos requerimientos del cliente.

# Bibliografía

# Bibliografía I


- Eliëns, Anton (1994). *Object-Oriented Software Development*. Addison Wesley.
- Porto, Julián Pérez y María Merino (Publicado: 2008. Actualizado: 2012.). *Definición de Paradigma*. URL: http://definicion.de/paradigma/.
- University of Cape Town (1 de ene. de 2011). *The software crisis in Software Engineering*. Ed. por University of Cape Town. MSc-IT Study Material. The Department of Computer Science, University of Cape Town. URL: https://www.cs.uct.ac.za/mit_notes/software/htmls/ch02s02.html (visitado 02-10-2020).

# Licencia

Creative Commons
Atribución-No Comercial-Compartir Igual