Introducción

Hacer código **entendible** es tan importante como hacer código ejecutable.

En empresas se producen menos líneas de código por persona por mes. Porque hay mucho extra que toma tiempo y esfuerzo. Por ejemplo se agrega documentación, y testing.

La **ingeniería del software** se define como el enfoque sistemático al desarrollo, operación, mantenimiento y retiro del software.

Software (IEEE 610.12-1990):
Colección de programas, procedimientos, y la documentación y datos asociados que determinan la operación de un sistema de computación.

El Dominio del Problema

El dominio del problema es el software industrial (*industrial strength software*).

Priman características como: **usabilidad**, **confiabilidad** y **portabilidad**

Diferencias entre Software Industrial y "de Estudiantes"

industrial strength software	Student system
Para resolver problemas de clientes. Radica en la calidad.	Para propósitos demostrativos.
Se debe testear muy cuidadosamente, del 30% al 50% de los esfuerzos puede ser usado en testing.	Los *bugs* no son una preocupación mayor, menos del 5% del esfuerzo se usa en testing.
Se requiere documentación tanto para el usuario como para la organización y el proyecto.	No se necesita documentación.
Es clave una interfaz usable y accesible.	No es importante la interfaz de usuario.
El software es caro (lo más caro son los programadores*).	No existe inversión (que perder).

industrial strength software	Student system
Se divide en etapas para evaluar y revisar cada fase.	En general se programa todo de una vez y luego se prueba.
Hay requerimientos de backup y recuperación, tolerancia a fallas.	Esos requerimientos no son necesarios.
Debe ser portable.	No es necesaria la portabilidad.
Debe acatar estándares	No necesita seguir estándares porque en general solo es usado por el mismo desarrollador.

* La productividad frecuentemente se mide en términos de LOC (lines of code o KLOC) por persona, por mes. **KLOC/PM**

Tarde y No Confiable

El presupuesto y la planificación están fuera de control. Código *runaway* (desbocados).

Los errores de software son diferentes de otras ingenierías porque no envejecen (no se modifican con el tiempo). **En el software los errores son introducidos en el proceso de diseño y desarrollo**.

Mantenimiento y Rehacer trabajo

Una vez que se entrega el software se entra en la *fase de mantenimiento*.
Los costos de mantenimiento en general exceden los costos de desarrollo del sistema.

Mantenimiento correctivo	Mantenimiento adaptativo
Correcciones de errores residuales que quedaron en el sistema y hay que corregirlos a medida que se detectan.	Adaptación del software a las necesidades del **entorno cambiante**.

** Uno de los mayores problemas en el desarrollo del software es entender lo que se desea del producto. Tanto los clientes como los desarrolladores deben *visualizar* cómo debería ser el comportamiento del software una vez que este listo. A veces eso no ocurre y hay que rehacer trabajo. También puede suceder que al cambiar el entrono cambien las necesidades y por eso se debe acomodar el software a los nuevos requerimientos.

El desafío de la Ingeniería del Software

Enfoque sistemático: Para predecir el trabajo que va a hacer falta y calcular un presupuesto. Las metodologías para el desarrollo deben ser *repetibles*.

Desafíos principales

1. **Escala**: Queremos software escalable de chico a grande y de grande a chico: no exija recursos de más.
2. **Calidad** y **Productividad**
3. **Consistencia** y **Repetibilidad**
4. **Cambios**: de programadores o hardware.

Otro Desafío:
6. Un cliente no sabe describir sus necesidades antes de sufrir su falta: Por lo tanto saber y poder satisfacer esas necesidades es difícil.

Calidad

Es uno de los desafíos principales de la ingeniería del software.
Para medir la calidad, se utilizan los siguientes parámetros:

1. **Funcionalidad**: capacidad de proveer las funciones que uno quiere y necesita
2. **Confiabilidad**: que no haga más cosas de las que queremos. Que se hagan las cosas en tiempo y forma.
3. **Usabilidad**: Fácil de usar. Nunca lo usa la persona lo programó.
   Debe ser comprensible.
4. **Eficiencia**: Usar la menor cantidad de recursos y tiempo. No siempre se puede acceder al hardware potente, los usuarios no tienen los mejores procesadores.
5. **Mantenibilidad**: Es importante para adaptarse al usuario. Es clave la legibilidad.
6. **Portabilidad**: que funcionen en diferentes entornos (sin necesidad de toquetear el código o tocando poquito).

Otras características de la calidad según Amy K. Jo

- **Correctitud:** se comporta según especificaciones
- **Confiabilidad:** < 1 defecto / KLOC
- **Robustez:** recuperación de errores o entradas inesperadas.
- **Rendimiento:** eficiencia. Minimizar la cantidad de instrucciones por tarea.
- **Portabilidad:** sin que se modifique
- **Interoperabilidad:** mediante el uso de estándares. interacción con otros sistemas.
- **Seguridad:** No cualquiera pueda meterse

Propiedades del código:

- **Verificabilidad:** solo se pueden verificar porciones muy chiquitas.
- **Mantenibilidad:** adaptar, corregir y perfeccionar

- **Reutilización**
- **Diseño** y **Experiencia** del usuario
- **Eficiencia del usuario**
- **Accesibilidad**
- **Utilidad**
- **Privacidad:** balance entre la información que se provee
- **Coherencia:** reutilizar habilidades.
- **Usabilidad**
- **Sesgo:** evitar ser excluyente en la sociedad

Es necesario priorizar algunas cualidades para cada proyecto

Cambios:

Dos tipos:

- **Cambios de programadores**: (aumento de sueldo a través de cambios) El código se debe entender para modificarse.
- **Cambio del hardware**.

El enfoque de la Ingeniería del Software

Para alta C&P (relación calidad y precio) necesitamos buena **tecnología**, buenos **procesos** o **métodos** y **gente** que haga bien su trabajo.

El proceso: es lo que menos varía en el tiempo. Queremos la sistematización de procesos.

Administración del proceso: Fases del proceso de desarrollo

Es importante separar en fases para la separación de incumbencias (echar culpas). Cada grupo tiene una tarea específica. Cuando la tarea general falla, es importante saber qué parte falló.

Permite verificar la calidad de cada clase.
El proceso en fases es central en el enfoque de la Ingeniería del Software para solucionar la *crisis del software* (no saber dar un buen presupuesto y fecha de entrega).

En general los modelos de procesos consisten de:

1. **Análisis de requisitos y especificación:** qué quiere el cliente
2. **Arquitectura y Diseño:** el sistema se separa en módulos
    1. Diseño
    2. Diseño detallado
    3. Diseño de alto nivel

3. **Codificación**
4. **Testing**
5. **Entrega e instalación:** entregar, instalar y mantener

Queremos que cada una tenga una entrada y una salida definidas para testear.

El planeamiento del proyecto es central a la administración del proceso para poder determinar cuestiones como:

Se cumplen...

- En tiempo?
- en Presupuesto?
- Objetivos de calidad?

Práctico 1