INTRODUCCIÓN A LA INGENIERIA DE SW

Primera Unidad
Concepto de Ingeniería de SW

¿CUÁLES SON LAS CARACTERÍSTICAS DEL SOFTWARE?

CARACTERÍSTICAS DEL SOFTWARE

a) Intangible,

b) No se desgasta pero queda obsoleto en poco tiempo,

c) Mantenible, para perdurar en el tiempo deben adaptarse a los cambios operacionales y del entorno.

d) Confiable y eficiente.

Calidad del diseño.
Costes más importantes en la ingeniería
Gestión especial de los proyectos

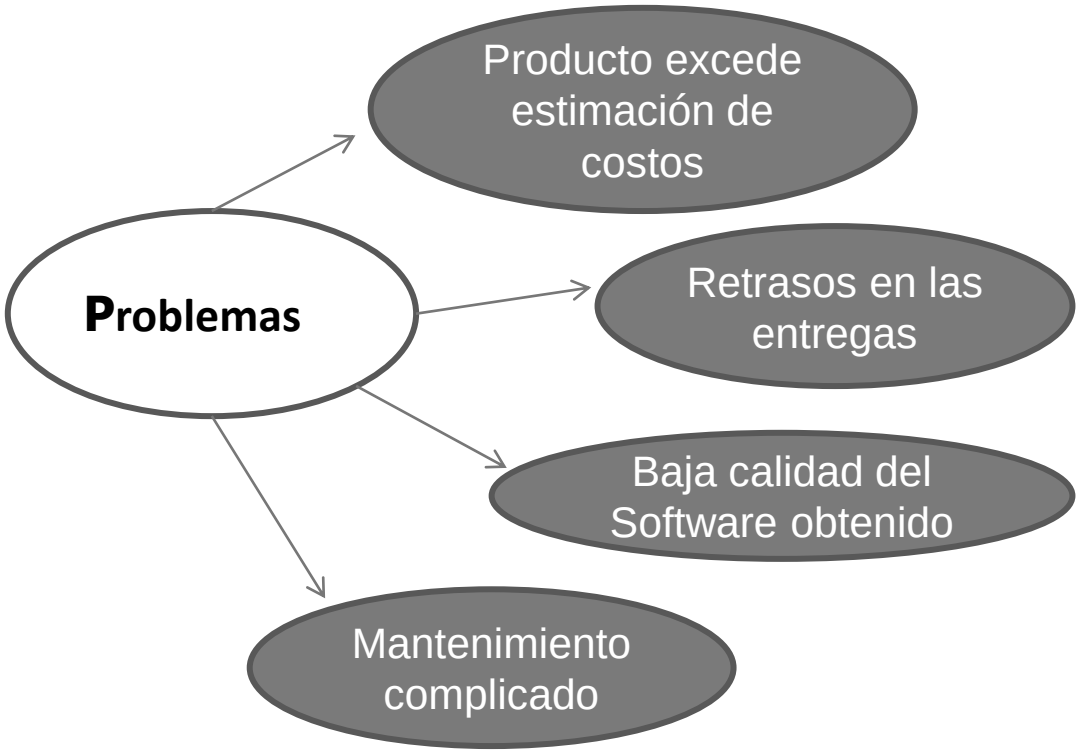
CATACTERÍSTICAS DEL SOFTWARE

- La importancia relativa de las características depende del tipo de producto y del ambiente en el que será utilizado.
- En algunos casos, algunos atributos pueden dominar.
- En sistemas de seguridad críticos de tiempo real, los atributos clave pueden ser la confiabilidad y la eficiencia.
- Los costos tienden a crecer exponencialmente si se requieren altos niveles de alguna característica.

INTRODUCCIÓN

LA CRISIS DEL SOFTWARE

Término acuñado en los años 70 para explicar los problemas del desarrollo de software asociado a su creciente demanda :



INTRODUCCIÓN

LA CRISIS DEL SOFTWARE

El informe Chaos del 2009 muestra que los proyectos considerados 100% exitosos suben de un 16% (año 1985) a un 32%, mientras que los cancelados han bajado de un 31% a un 24%.

¿ Cuando se considera que un proyecto de desarrollo de sw es exitoso?.

COSTOS DEL SOFTWARE

- Causas de la crisis del software
  - Naturaleza lógica del software
  - Mala gestión de los proyectos (ausencia de datos, deficiente comunicación, ...)
  - Ausencia de entrenamiento formal en nuevas técnicas (programadores vs. ingenieros de software)
  - Resistencia al cambio
  - Mitos del software:

MITOS DE GESTIÓN

- Uso de estándares
- Uso de herramientas
- Mala planificación: aumento de programadores

MITOS DE LOS DESARROLLADORES

- Programa funcionando = fin del trabajo
- Calidad = el programa se ejecuta sin errores
- Entrega al cliente: programa funcionando

MITOS DEL CLIENTE

- Requisitos establecidos como una declaración general de objetivos
- Flexibilidad del software ante los cambios

INGENIERÍA DE SOFTWARE

¿QUÉ ES LA INGENIERÍA DE SOFTWARE?

El IEEE define la ingeniería del software como "La aplicación de un enfoque sistemático, disciplinado y cuantificable en el desarrollo, la operación y el mantenimiento del software".

La Ingeniería de Software concierne teorías, métodos y herramientas para el desarrollo profesional del software (sommerville 2005).

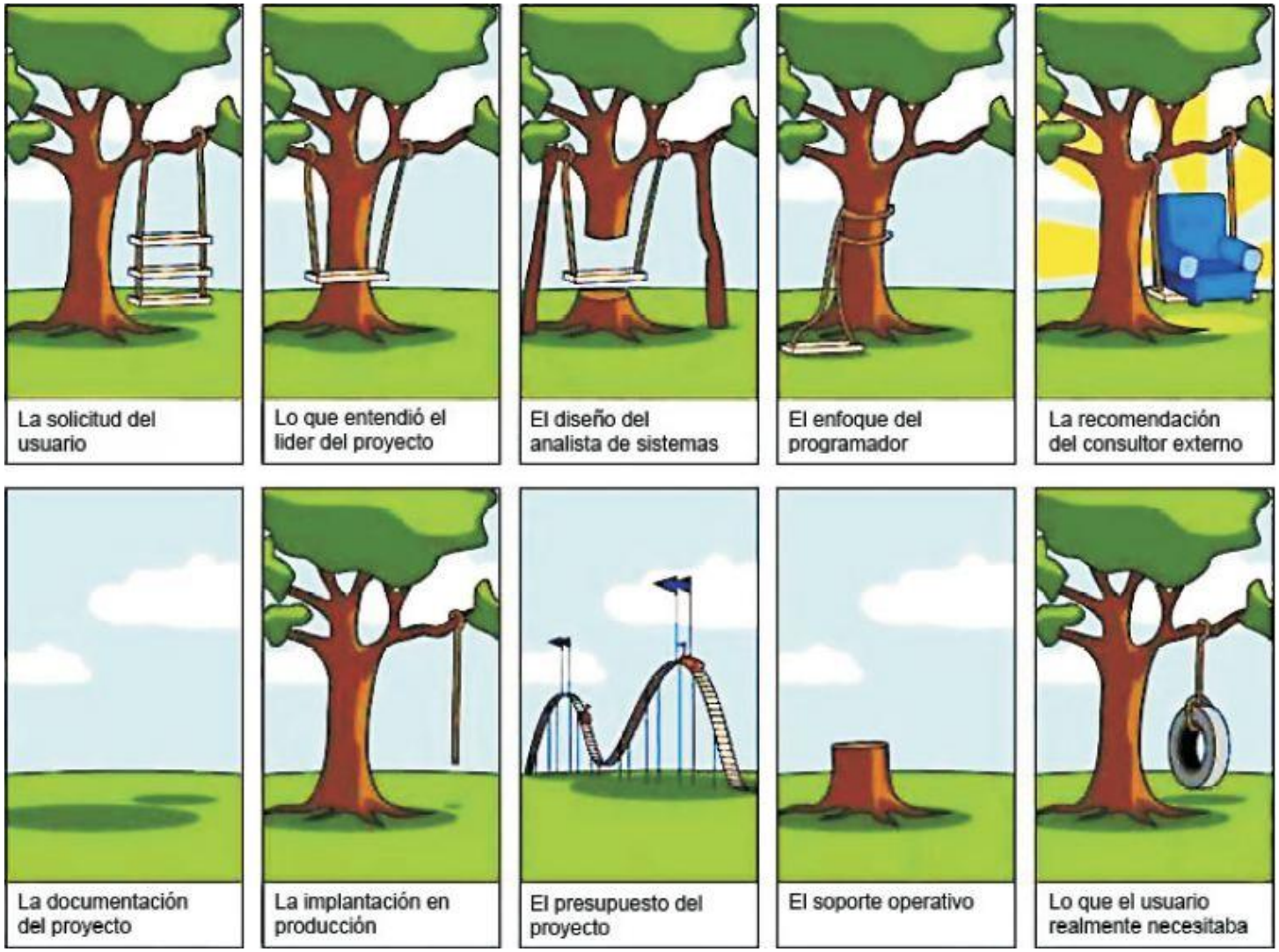
COSTOS DEL SOFTWARE

A menudo dominan los costos de un sistema computacional. Los costos de software en un computador son a menudo mayores que el costo del hardware.

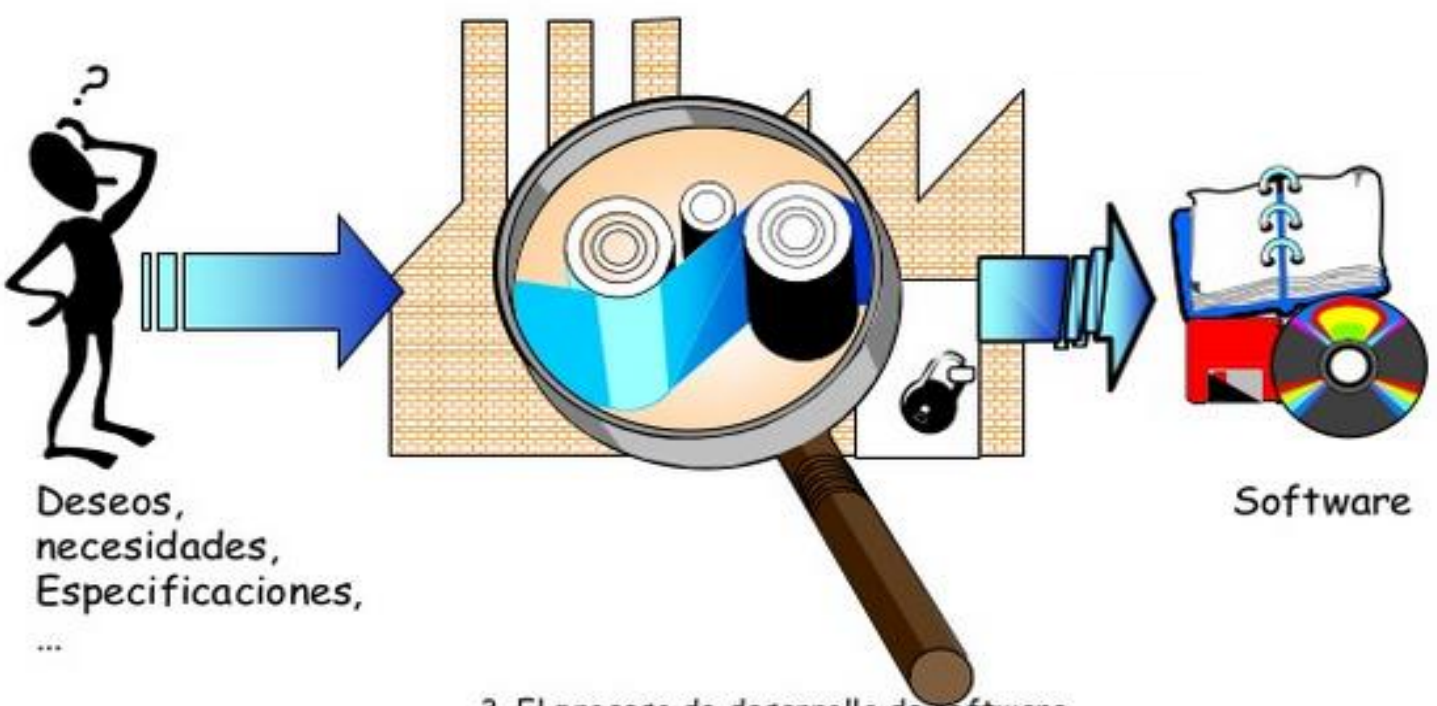
Cuesta más el mantenimiento del software que el desarrollo del mismo.

A la Ingeniería de Software le compete el desarrollo de software rentable.

PROBLEMAS DE LA INGENIERÍA DE REQUISITOS



¿QUÉ ENTENDEMOS POR PROCESO DE SOFTWARE?



3. El proceso de desarrollo de software

¿QUÉ ENTENDEMOS POR PROCESO DE SOFTWARE?

Sommerville:
"Un proceso de desarrollo de software es un conjunto de actividades y resultados asociados que conducen a la creación de un producto de software"

Pressman:
"Marco de trabajo de las tareas que se requieren para construir software de alta calidad"

¿QUÉ ENTENDEMOS POR PROCESO DE SOFTWARE?

Un proceso de desarrollo de software describe normalmente:

- Qué **tareas** y en qué orden se deben llevar a cabo.

- Qué **roles** deben tener las diferentes personas que participan en el desarrollo, cuál es la responsabilidad de cada rol y qué tareas deben llevar a cabo.

- Qué **artefactos** elemento que se produce durante el desarrollo (documento, fragmento de software,...) deben usarse como punto de partida para cada tarea y qué se debe generar como resultado.

STAKEHOLDERS

- Los ***stakeholders*** son aquellas personas o entidades que tienen algún impacto o interés en este sistema.

- Algunos participantes pueden ser:

  - cliente: encarga y paga el sistema
  - desarrolladores: construyen el sistema (analistas, diseñadores, programadores,...)
  - Gerente o director del proyecto: planifica y calcula el presupuesto, coordina a los desarrolladores y cliente
  - Usuarios finales: los que van a utilizar el sistema

- **Roles**
  - conjunto de responsabilidades en el proyecto o en el sistema
  - asociado con un conjunto de tareas y se asigna a un participante
  - un mismo participante puede cumplir varios roles

ACTIVIDADES, TAREAS Y RECURSOS

- actividad (o fase): conjunto de tareas que se realiza con un propósito específico (obtención de requisitos, entrega, administración,...) que pueden componerse de otras actividades

.

- tarea: unidad elemental de trabajo que puede ser administrada; consumen recursos, dan como resultado productos de trabajo y dependen de productos de trabajo producidos por otras tareas

- recursos: bienes que se utilizan para realizar el trabajo:
  - tiempo, equipamiento y recursos humanos
  - al planificar, el gerente divide el trabajo en tareas y les asigna recursos

ACTIVIDADES, TAREAS Y RECURSOS

- actividad (o fase): conjunto de tareas que se realiza con un propósito específico (obtención de requisitos, entrega, administración,...) que pueden componerse de otras actividades

.

- tarea: unidad elemental de trabajo que puede ser administrada; consumen recursos, dan como resultado productos de trabajo y dependen de productos de trabajo producidos por otras tareas

- recursos: bienes que se utilizan para realizar el trabajo:
  - tiempo, equipamiento y recursos humanos
  - al planificar, el gerente divide el trabajo en tareas y les asigna recursos

OTROS CONCEPTOS DE LA INGENIERÍA DEL SOFTWARE

objetivos, requerimientos y restricciones

- objetivos:
  - principios de alto nivel que se utilizan para guiar el proyecto
  - definen los atributos realmente importantes del sistema (seguridad, fiabilidad,...)
  - a veces hay conflicto entre objetivos (por ejemplo, seguridad y bajo coste) que aumentan la complejidad del proyecto

OTROS CONCEPTOS DE LA INGENIERÍA DEL SOFTWARE

objetivos, requerimientos y restricciones

- requerimientos
  - características que debe tener el sistema
  - requerimiento funcional: área de funcionalidad que debe soportar el sistema (por ejemplo, *proporcionar billetes de tren*)
  - requerimiento no funcional: restricción que se establece sobre el funcionamiento del sistema (por ejemplo, *proporcionar billetes de tren en menos de un segundo*)

PROCESO DE SW

Las actividades estructurales de un proceso de desarrollo son:

- comunicación: Abarca investigación de requisitos y actividades relacionadas.
- Planeación: Plan de trabajo de la Ing de Sw.
- Modelado: Creación de modelos de análisis y diseño.
- Construcción: Generación de código/pruebas.
- Despliegue: El software se entrega para evaluación.

Las actividades varían dependiendo del tipo de proyecto, del equipo de desarrollo, de l naturaleza misma del problema y de la cultura de la organización.

PROCESO DE SW

Actividades de Comunicación:

- Antes de empezar el desarrollo, un factor crítico es la comunicación con el cliente. .
- Se busca entender los objetivos de los participantes respecto del proyecto, y reunir los requerimientos que ayuden a definir las características y funciones del software.

PROCESO DE SW

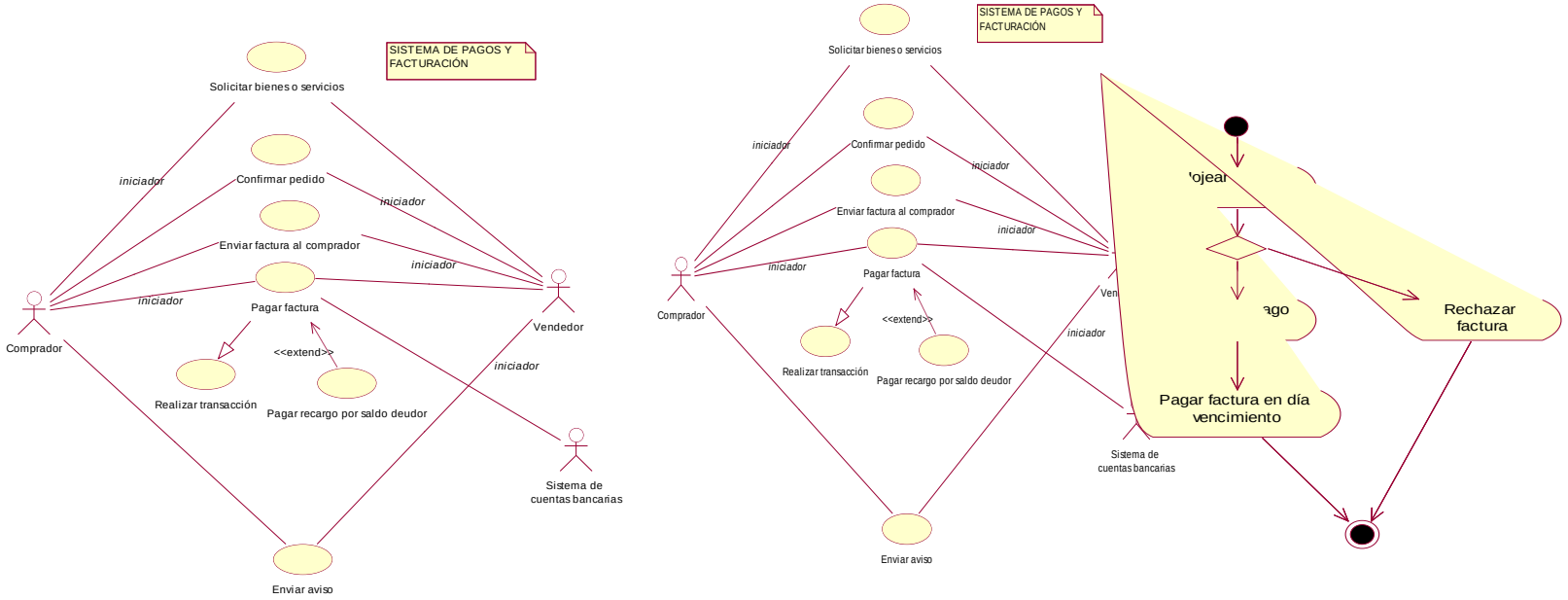
Actividades de Planeación:

- Define el trabajo a realizar al describir las tareas técnicas por realizar, los riesgos probables, los recursos que se requieren, los productos del trabajo que se obtendrán y una programación de las actividades.

PROCESO DE SW

Actividades de Modelado:

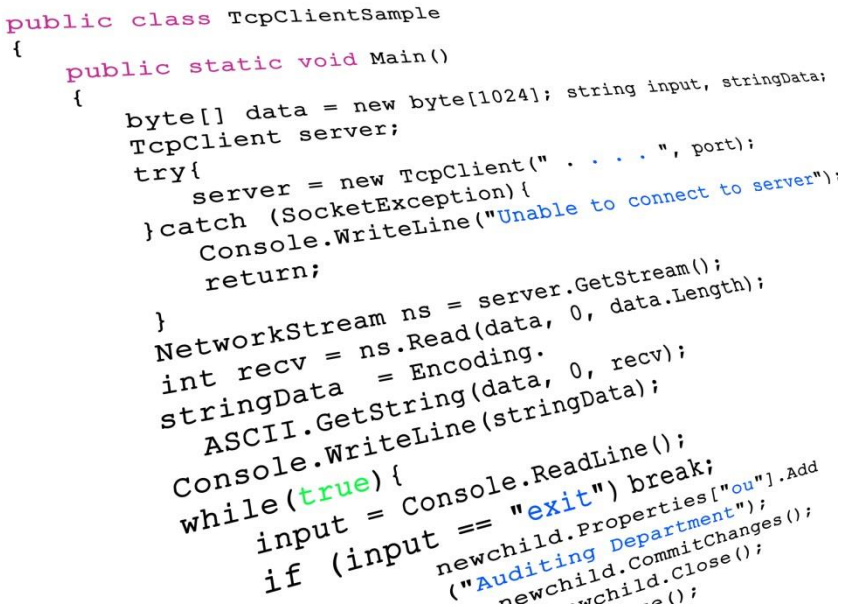
- Crear modelos a fin de entender mejor los requerimientos del software y el diseño que los satisfará.
- Comprender el ambiente de funcionamiento del·sistema, comprender los distintos sistemas.
- Usar técnicas y herramientas para construir los modelos



PROCESO DE SW

Actividades de Construcción:

- Se combina la generación de código con las pruebas para descubrir errores en él.

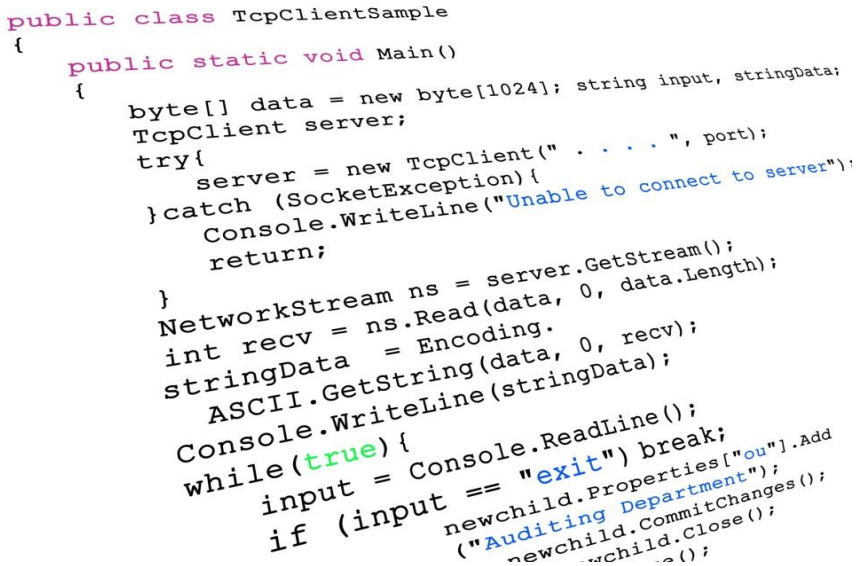


```
public class TcpClientSample
{
    public static void Main()
    {
        byte[] data = new byte[1024]; string input, stringData;
        TcpClient server;
        try{
            server = new TcpClient(" . . . . ", port);
        }catch (SocketException){
            Console.WriteLine("Unable to connect to server");
            return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData  = Encoding.
            ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while(true){
            input = Console.ReadLine();
            if (input == "exit") break;
                    newchild.Properties["ou"].Add
                    ("Auditing Department");
                    newchild.CommitChanges();
                    newchild.Close();
```

PROCESO DE SW

Actividades de Despliegue:

El software se entrega al consumidor que lo evalúa y que le da retroalimentación.



```
public class TcpClientSample
{
    public static void Main()
    {
        byte[] data = new byte[1024]; string input, stringData;
        TcpClient server;
        try{
            server = new TcpClient(" . . . . ", port);
        }catch (SocketException){
            Console.WriteLine("Unable to connect to server");
            return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData  = Encoding.
            ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while(true){
            input = Console.ReadLine();
            if (input == "exit") break;
                newchild.Properties["ou"].Add
                ("Auditing Department");
                newchild.CommitChanges();
                newchild.Close();
```

Diseño: definición

En sentido general, **diseñar** es una forma de **resolución de problemas**.

• Por ello, al diseñar se utilizan nociones como : Objetivos, Restricciones, Alternativas, Representaciones, Soluciones.

> Hay dos formas de realizar un diseño software:
> Hacer que sea tan simple que sea obvio que no tiene deficiencias,
> o hacer que sea tan complicado que no tenga deficiencias obvias.
> El primer método es más difícil.
>
> **C.A.R. Hoare** (creador del quicksort, lenguaje CSP)

DISEÑO OO

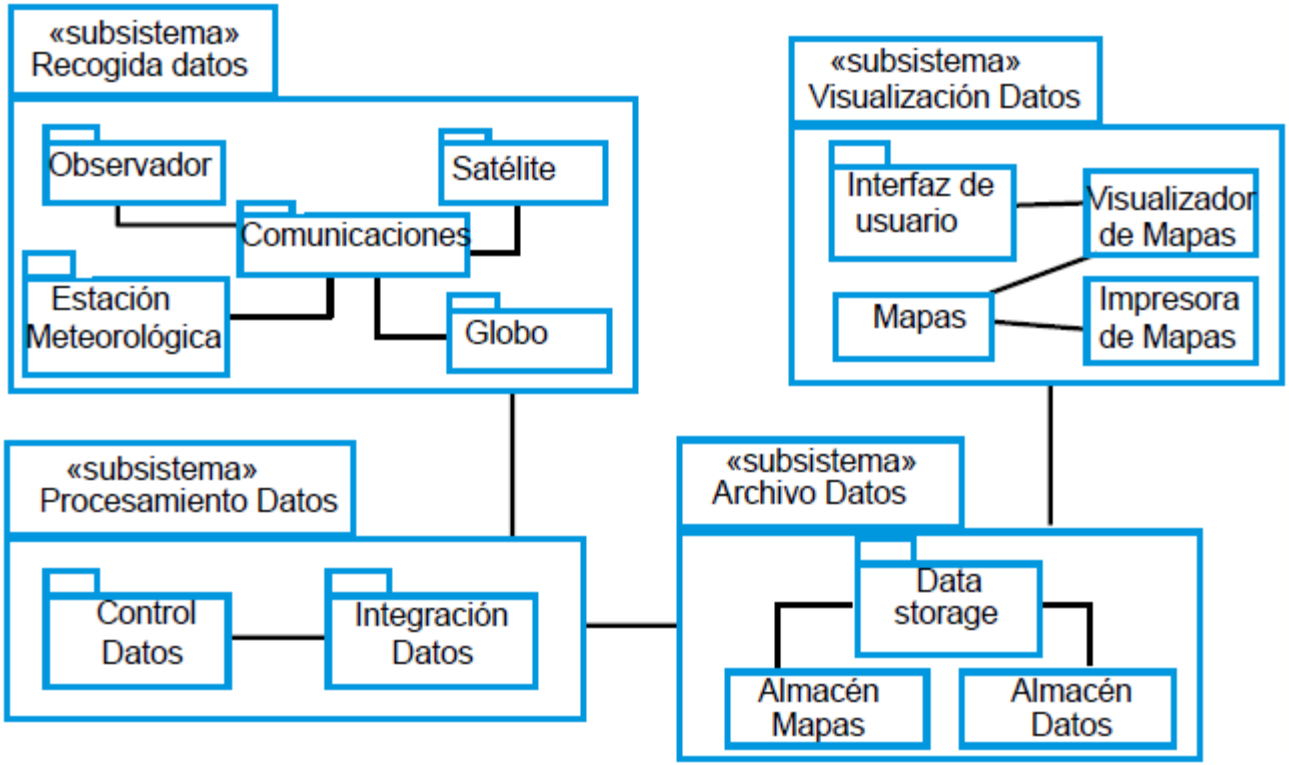
Ejemplo

Se necesita un **sistema de mapas climáticos** para generar mapas del tiempo de una forma regular usando los datos recopilados de estaciones meteorológicas remotas automáticas y otras fuentes como observadores, globos y satélites. Las estaciones meteorológicas transmiten sus datos a la computadora del sistema cuando reciben una petición al respecto desde dicha máquina.

el computador del sistema valida los datos recopilados y los integra con los datos de otras fuentes diferentes. Los datos integrados se archivan y, usando dichos datos y una base de datos de mapas digitalizados, se elabora un juego de mapas del tiempo locales. Los mapas pueden imprimirse para su distribución en una impresora de mapas especializada o pueden mostrarse en varios formatos diferentes.

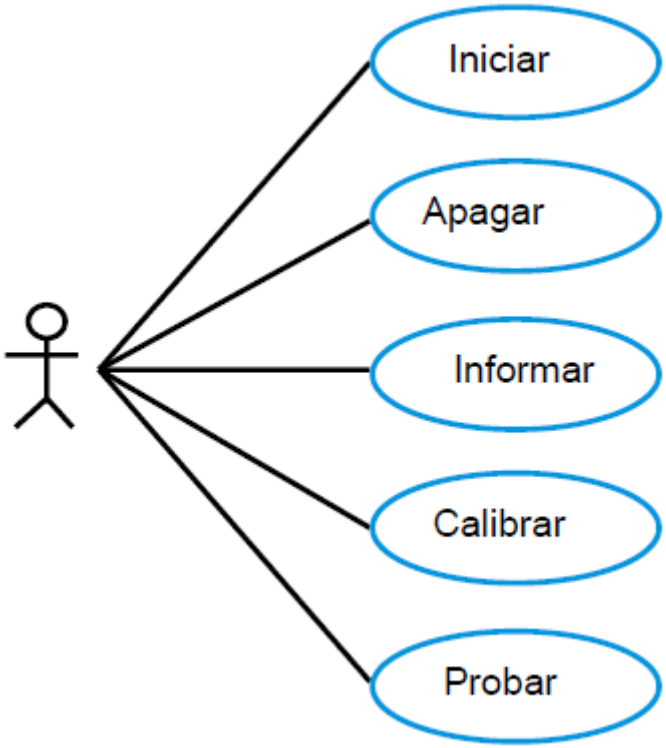
DISEÑO OO

Subsistemas de Ejemplo



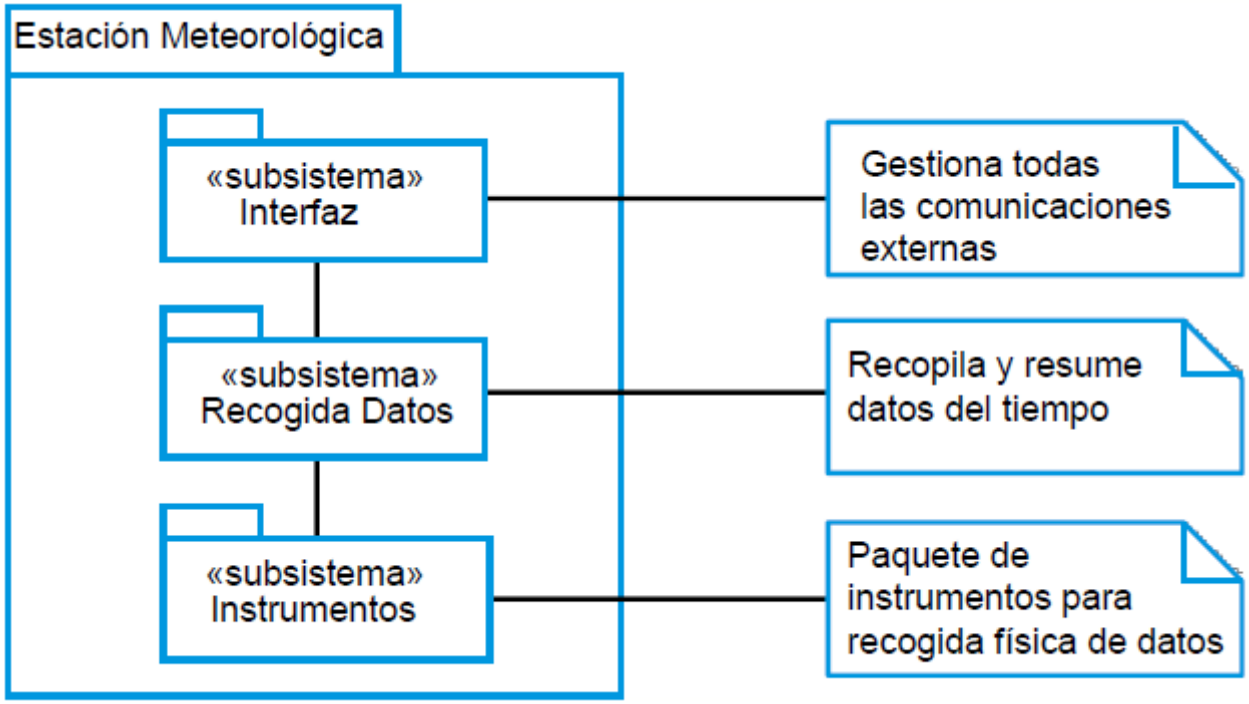
DISEÑO OO

Modelo de Casos de Uso del Ejemplo



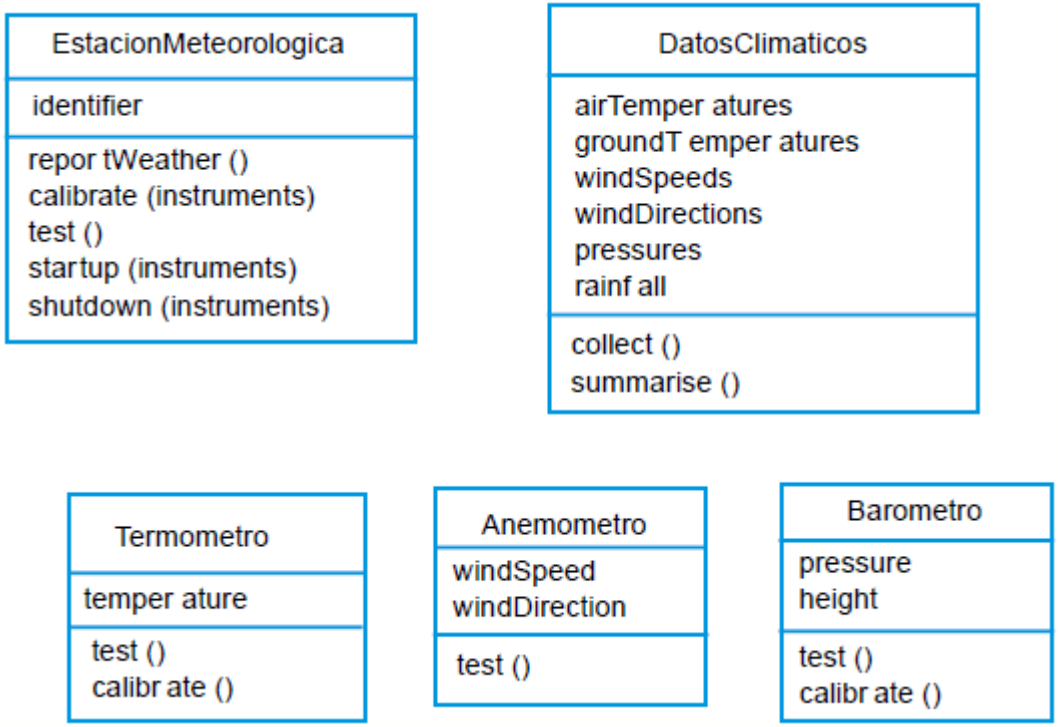
DISEÑO OO

Arquitectura en Capas



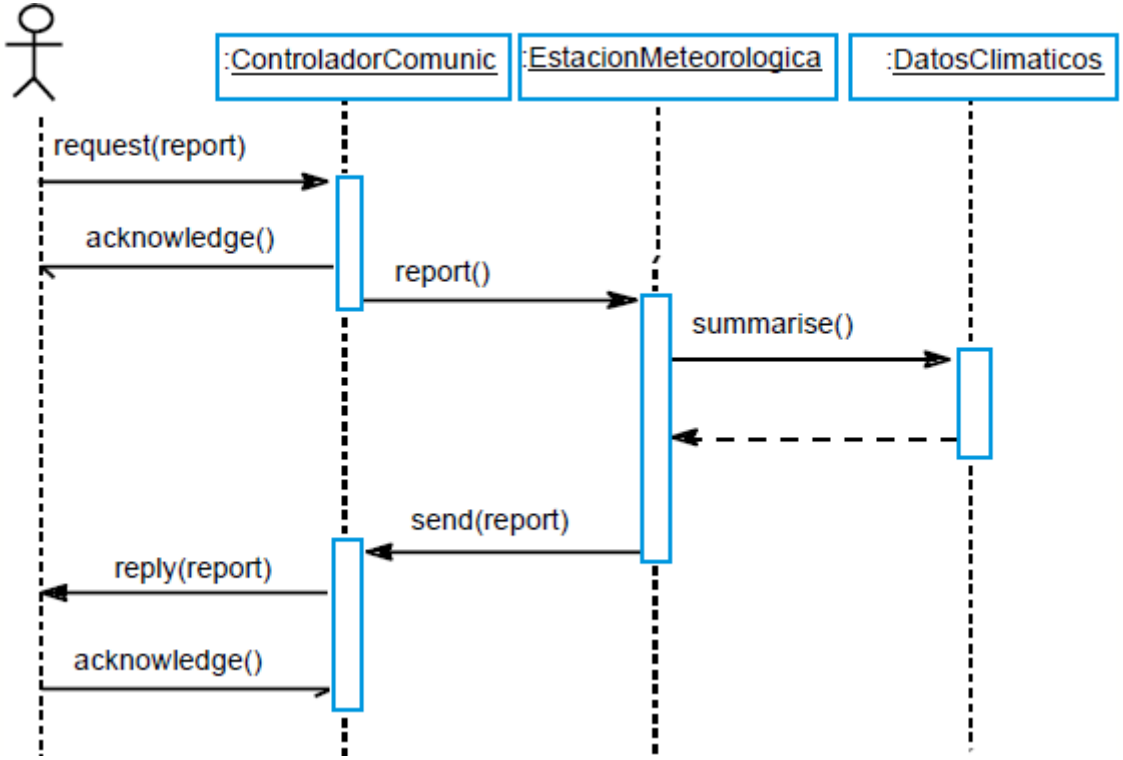
DISEÑO OO

Identificación de Objetos (clases)



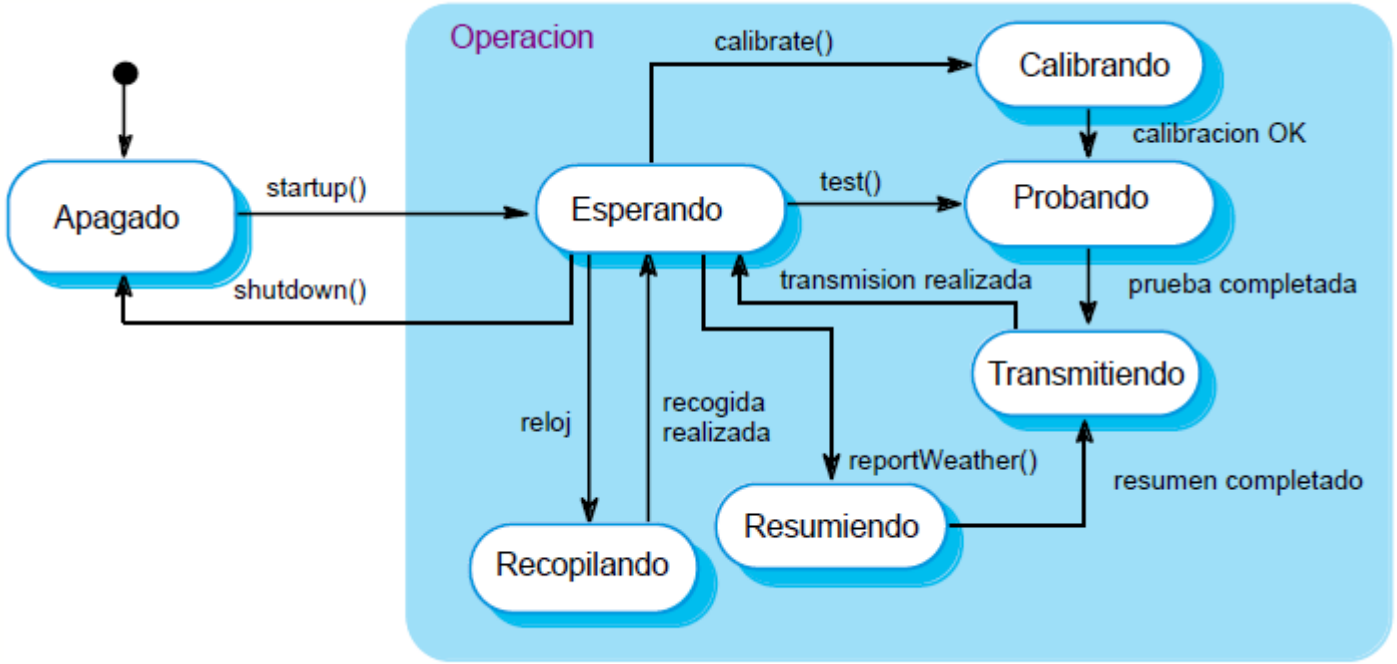
DISEÑO OO

Modelo de comportamiento: diagrama de secuencia caso de uso 'informar'



DISEÑO OO

Modelo de comportamiento: diagrama de estado estación metereológica



EVALUANDO

ANALISIS DEL PROBLEMA

- ¿Quiénes tienen que ver con la solución del problema? Es decir, ¿quiénes son los participantes?
- ¿Cuáles son las incógnitas? ¿Cuáles datos, funciones y características se requieren para resolver el problema en forma apropiada?
- ¿Puede fraccionarse el problema? ¿Es posible representarlo con problemas más pequeños que sean más fáciles de entender?
- ¿Es posible representar gráficamente el problema? ¿Puede crearse un modelo de análisis?

EVALUANDO

PLANEACIÓN

- ¿Ha visto antes problemas similares? ¿Hay patrones reconocibles en una solución potencial?
- ¿Hay algún software existente que implemente los datos, funciones y características que se requieren?
- ¿Ha resuelto un problema similar? Si es así, ¿son reutilizables los elementos de la solución?
- ¿Pueden definirse problemas más pequeños? Si así fuera, ¿hay soluciones evidentes para éstos?
- ¿Es posible crear un modelo del diseño?