DISEÑO E IMPLEMENTACIÓN

Diseño e implementación

•El diseño de software y la implementación es la etapa en el proceso de ingeniería de software en el que se desarrolla un sistema de software ejecutable.

•Las actividades de diseño e implementación de software son invariablemente entrelazadas.

•El diseño de software es una actividad creativa en la que se identifica los componentes de software y sus relaciones, sobre la base de los requisitos del cliente.

•La implementación es el proceso de realización del diseño como un programa.

Construir o comprar

•En una amplia gama de dominios, ahora es posible comprar sistemas off-the-shelf (COTS) que pueden ser adaptados y orientados a los deseos de los usuarios.

•Por ejemplo, si desea implementar un sistema de registros médicos, es posible comprar un paquete que ya se utiliza en los hospitales. Puede ser más barato y más rápido para utilizar este enfoque en lugar de desarrollar un sistema en un lenguaje de programación convencional.

•Cuando se desarrolla una aplicación de esta forma, el proceso de diseño se preocupa con el uso de las funciones de configuración de ese sistema para entregar los requisitos del sistema.

Un proceso de diseño orientado a objetos

•Procesos de diseño orientados a objetos suponen el desarrollo de una serie de diferentes modelos de sistemas.

•Ellos requieren un gran esfuerzo para el desarrollo y el mantenimiento de estos modelos y, para sistemas pequeños, esto puede no ser rentable.

•Sin embargo, para los grandes sistemas desarrollados por diferentes modelos de diseño de grupos son un mecanismo de comunicación importante.

Etapas del proceso

•Hay una variedad de diferentes procesos de diseño orientados a objetos que dependen de la organización que utilice el proceso.

•Las actividades comunes en estos procesos incluyen:

- •Definir el contexto y los modos de uso del sistema;
- •Diseñar la arquitectura del sistema;
- •Identificar los principales objetos del sistema;
- •Desarrollar modelos de diseño;
- •Especificar las interfaces de objetos.

.

Contexto del sistema y las interacciones

•La comprensión de las relaciones entre el software que se está diseñando y su entorno externo es esencial para decidir la manera de proporcionar la funcionalidad requerida del sistema y cómo estructurar el sistema para comunicarse con su entorno.

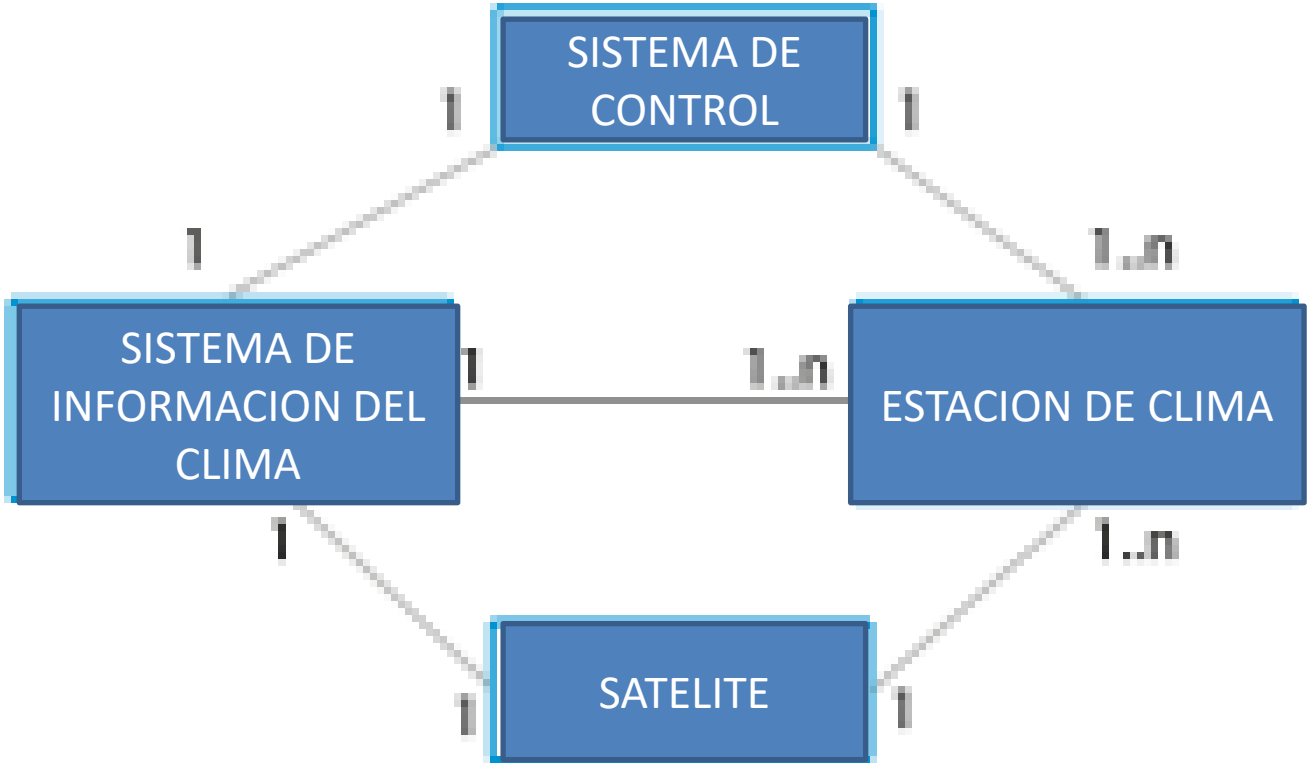
•La comprensión del contexto también permite establecer los límites del sistema. El ajuste de los límites del sistema ayuda a decidir qué características se implementan en el sistema que está siendo diseñado y qué características se encuentran en otros sistemas asociados.

Los modelos de contexto e interacción

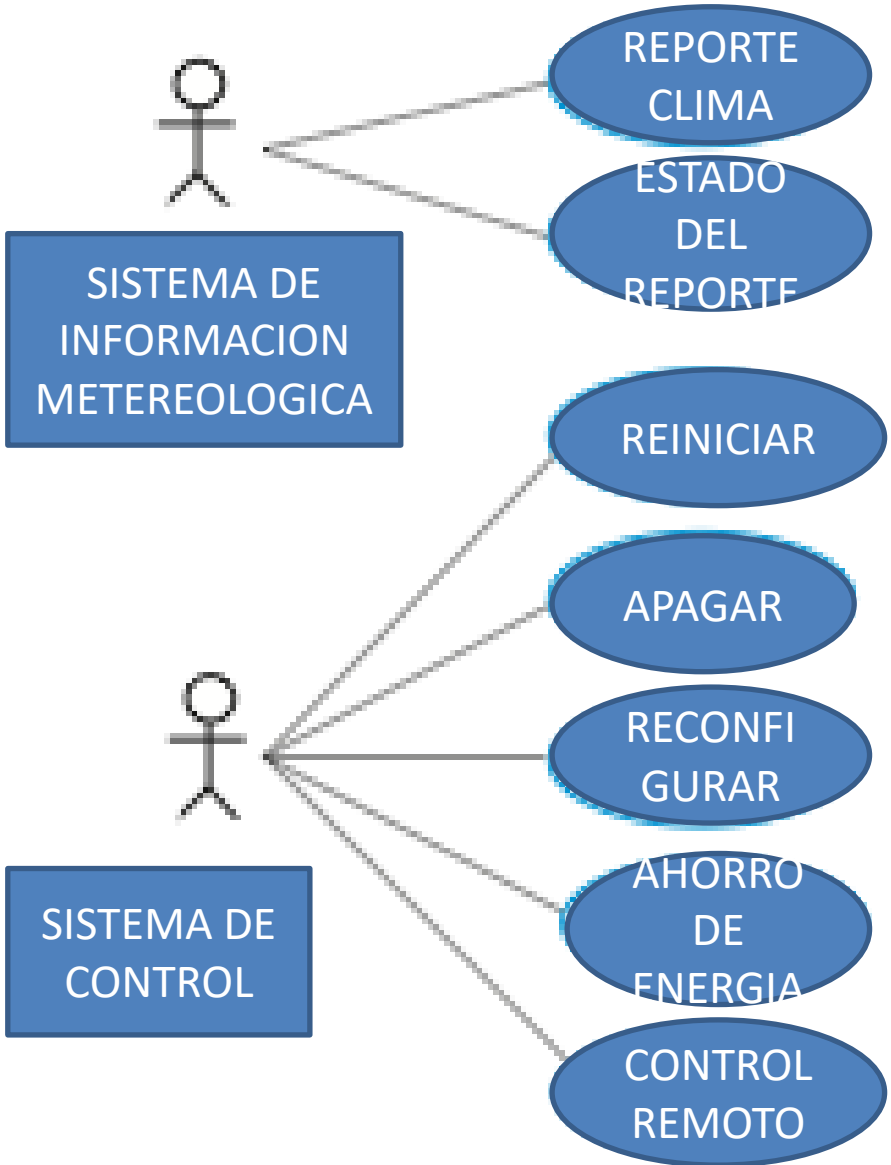
•Un modelo de contexto del sistema es un modelo estructural que demuestra los otros sistemas en el entorno del sistema en desarrollo.

•Un modelo de interacción es un modelo dinámico que muestra cómo el sistema interactúa con su entorno mientras es utilizado.

Contexto del sistema para la estación meteorológica



Casos de uso de la Estación meteorológica



Descripción de caso de uso-Informe meteorológico

Caso de uso	Reportar clima
Actores	sistema de información meteorológica, Estación meteorológica
Descripción	La estación meteorológica envía un resumen de los datos meteorológicos que se ha recogido de los instrumentos en el período de recolección para el sistema de información sobre el clima. Los datos enviados son el máximo, el mínimo, y una temperatura media de tierra y aire; máximo, mínimo y promedio de las presiones de aire; el máximo, el mínimo, y la velocidad media del viento; la precipitación total; y la dirección del viento como un muestreo a intervalos de cinco minutos.
Estimulo	El sistema de información del clima establece un enlace de comunicación por satélite con la estación meteorológica y solicita la transmisión de datos.
Respuesta	El resumen de datos se envía al sistema de información sobre el clima.
Comentarios	Las estaciones meteorológicas por lo general se les pide reportar una vez por hora, pero esta frecuencia puede variar de una estación a otra y pueden ser modificados en el futuro.

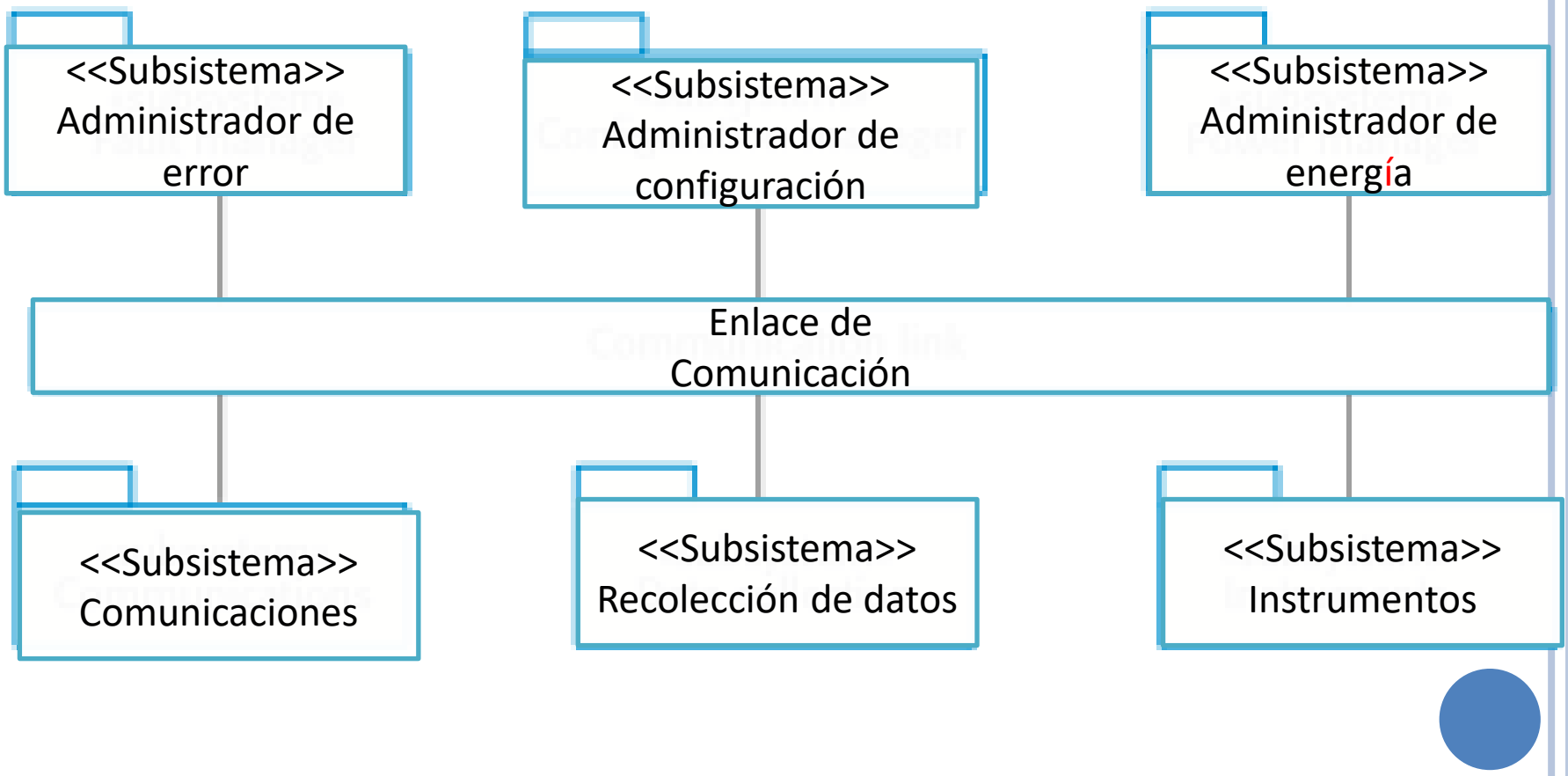
Diseño arquitectónico

•Una vez que se han entendido las interacciones entre el sistema y su entorno, se utiliza esta información para el diseño de la arquitectura del sistema.

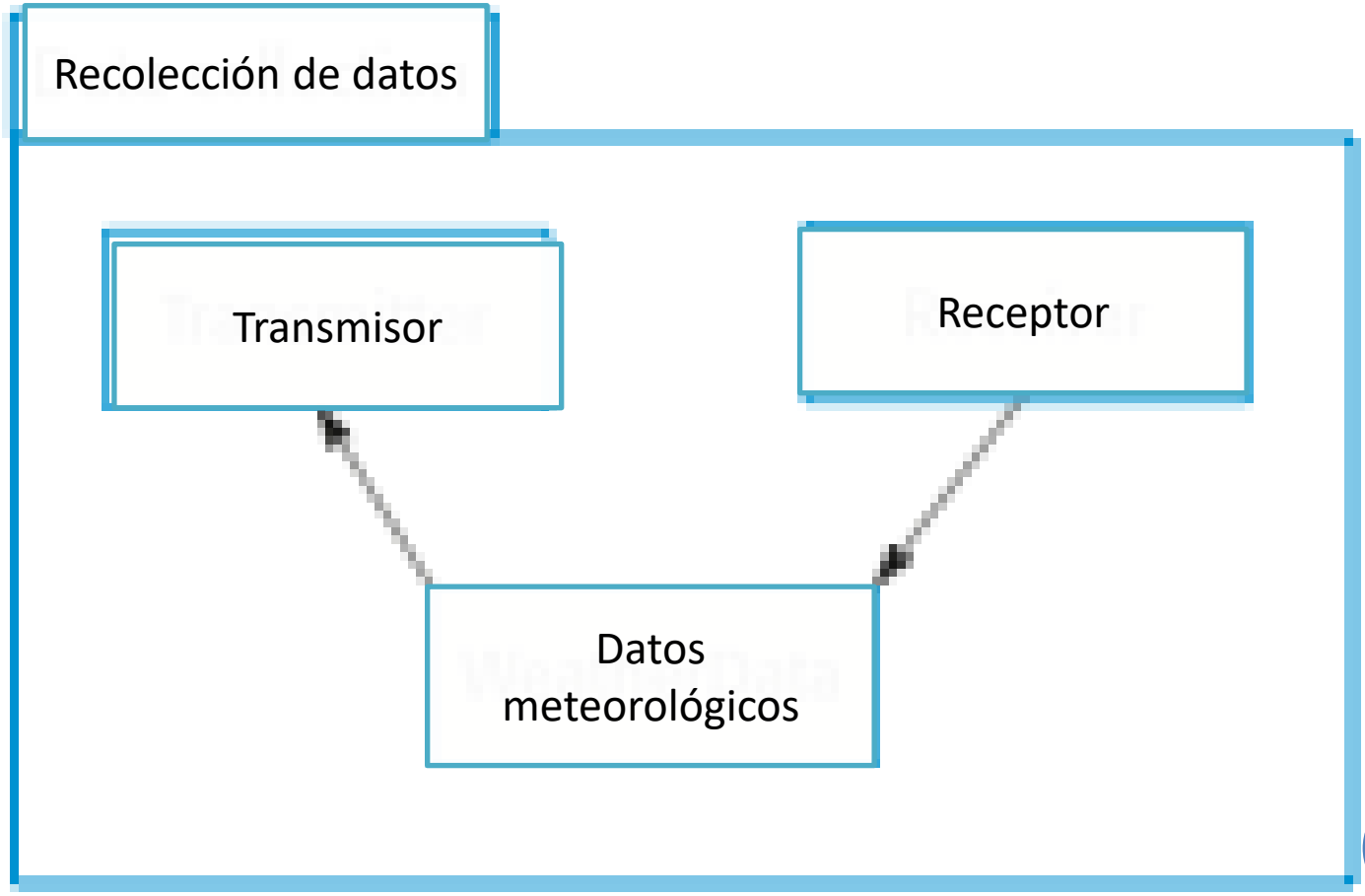
•Se identifican los principales componentes que conforman el sistema y sus interacciones, y luego puede organizar los componentes utilizando un patrón arquitectónico, como un modelo en capas o cliente-servidor.

•La estación meteorológica se compone de subsistemas independientes que se comunican mediante la difusión de mensajes sobre una infraestructura común.

Arquitectura de alto nivel de la estación meteorológica



Arquitectura del sistema de recolección de datos



Identificación de las clases Objeto

•Identificar las clases de objetos se vuelve a menudo una parte difícil del diseño orientado a objetos.

•No existe una "fórmula mágica" para la identificación de objetos. Se basa en la habilidad, la experiencia y el conocimiento del dominio de los diseñadores de sistemas

•La Identificación de objetos es un proceso iterativo. Es poco probable hacerlo bien la primera vez.

Aproximaciones para la identificación

•Utilizar un enfoque gramatical basado en una descripción en lenguaje natural del sistema

•Basar la identificación en las cosas tangibles en el dominio de aplicación.

•Utilizar un enfoque conductual e identificar objetos en función de su participación y su comportamiento.

•Utilizar un análisis basado en escenarios. Se identifican los objetos, atributos y métodos en cada escenario.

Descripción de la estación meteorológica

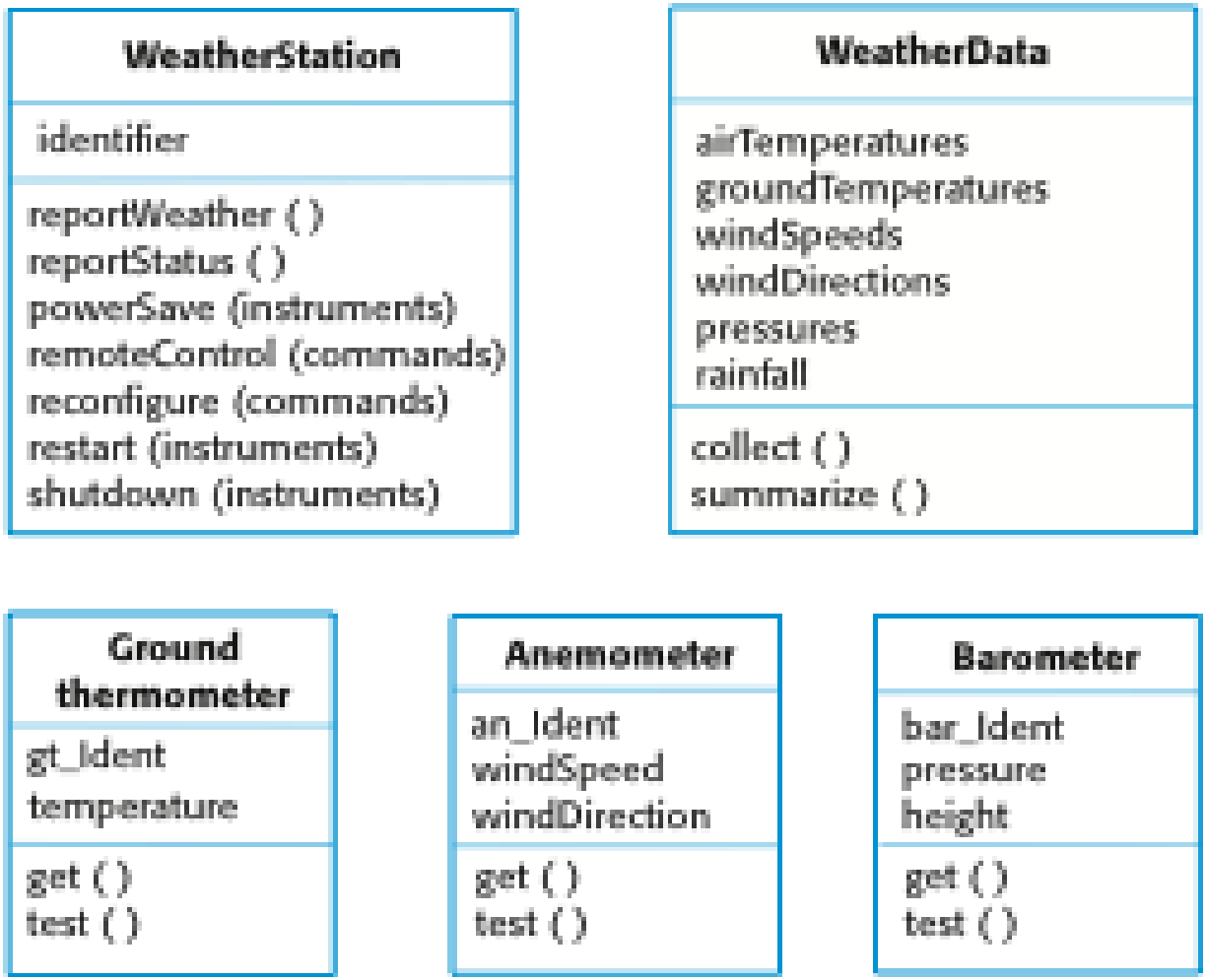
•Una estación meteorológica es un paquete de instrumentos controlados por software que recoge los datos, lleva a cabo algún tipo de procesamiento de datos y transmite estos datos para su posterior procesamiento. Los instrumentos incluyen el aire y los termómetros de tierra, un anemómetro, una veleta, un barómetro y un pluviómetro. Los datos se recogen periódicamente.

•Cuando se emite un comando para transmitir los datos del clima se resumen los datos recopilados. Los datos resumidos se transmiten control central

Clases de objeto de la estación meteorológica

- La identificación de clase de objetos en el sistema de estación meteorológica puede estar basada en el hardware y los datos tangibles en el sistema:
  - Termómetro de tierra, anemómetro, barómetro, (objetos de dominio de aplicación que son objetos 'hardware' relacionados con los instrumentos en el sistema)
  - Estación meteorológica
  - La interfaz básica de la estación meteorológica a su entorno. Por lo tanto, refleja las interacciones identificadas en el modelo de casos de uso.
    - Los datos del clima
    - Los datos resumidos de los instrumentos.

Clases de objeto de la Estación meteorológica



Modelos de diseño

- Los modelos de diseño muestran los objetos y clases de objetos y las relaciones entre estas entidades.

- Los modelos estáticos describen la estructura estática del sistema en términos de clases de objetos y relaciones.

- Los modelos dinámicos describen las interacciones dinámicas entre objetos.

Ejemplos de modelo de diseño

•Modelos de subsistemas que muestran agrupaciones lógicas de objetos en subsistemas coherentes.

•Modelos de secuencias que muestran la secuencia de interacciones de objetos.

•Modelos de estado de máquina que muestran como objetos individuales cambian su estado en respuesta a eventos.

Modelos del subsistema

•Muestra cómo el diseño se organiza en grupos de objetos relacionados lógicamente.

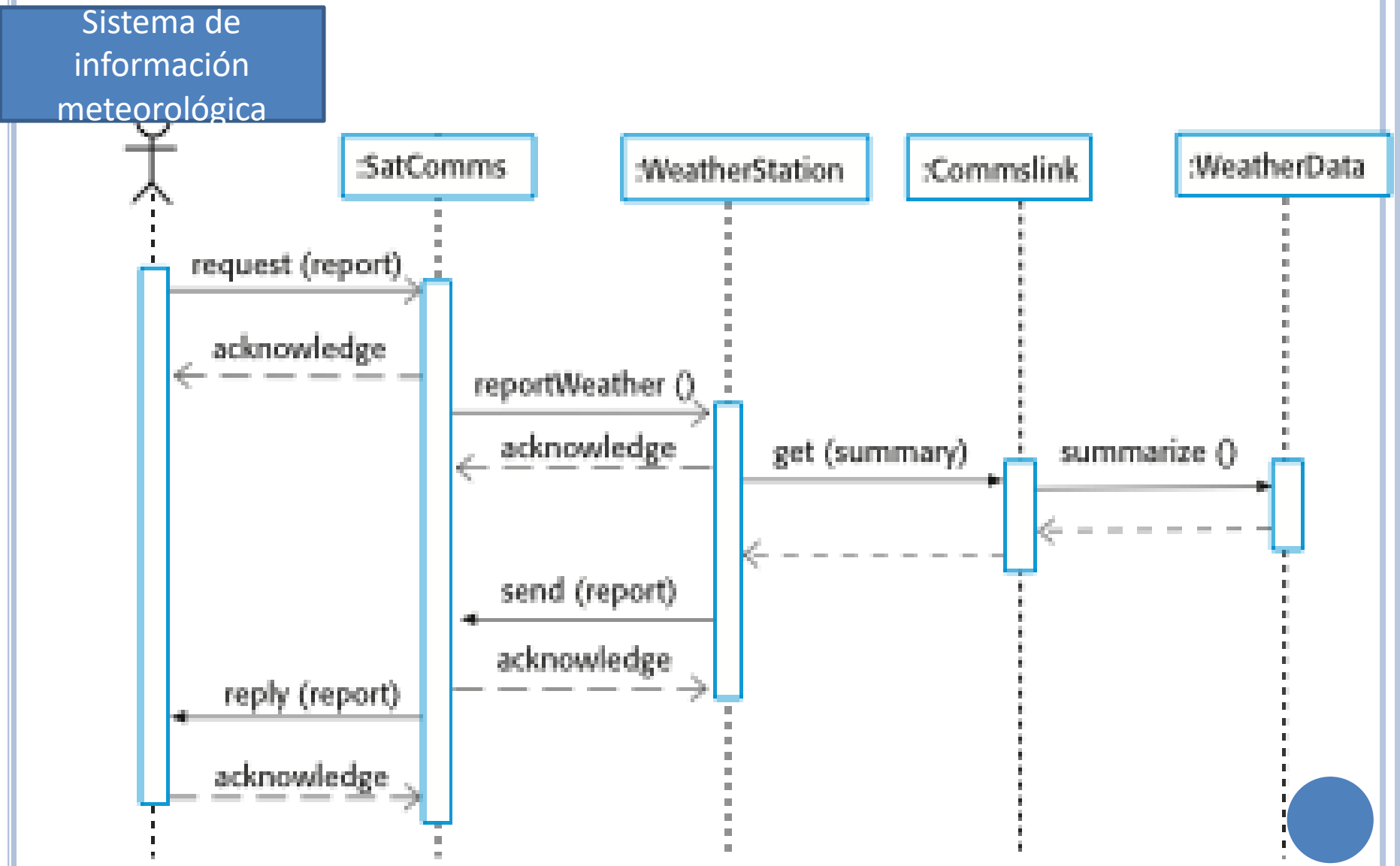
•En el UML, estos se muestran por el uso de paquetes - una construcción de encapsulación. Este es un modelo lógico. La organización actual de los objetos en el sistema puede ser diferente.

Modelos de secuencia

•Modelos de secuencia muestran la secuencia de interacción dados entre los objetos

•Los objetos se disponen horizontalmente en la parte superior;
•El tiempo se representa verticalmente, de modo que se leen los modelos de arriba hacia abajo;
•Las interacciones se representan mediante flechas etiquetadas, Diferentes estilos de flecha representan diferentes tipos de interacción;
•Un rectángulo delgado en una línea de vida de un objeto representa el momento en que el objeto esta activo

Diagrama de secuencia recopilación de datos



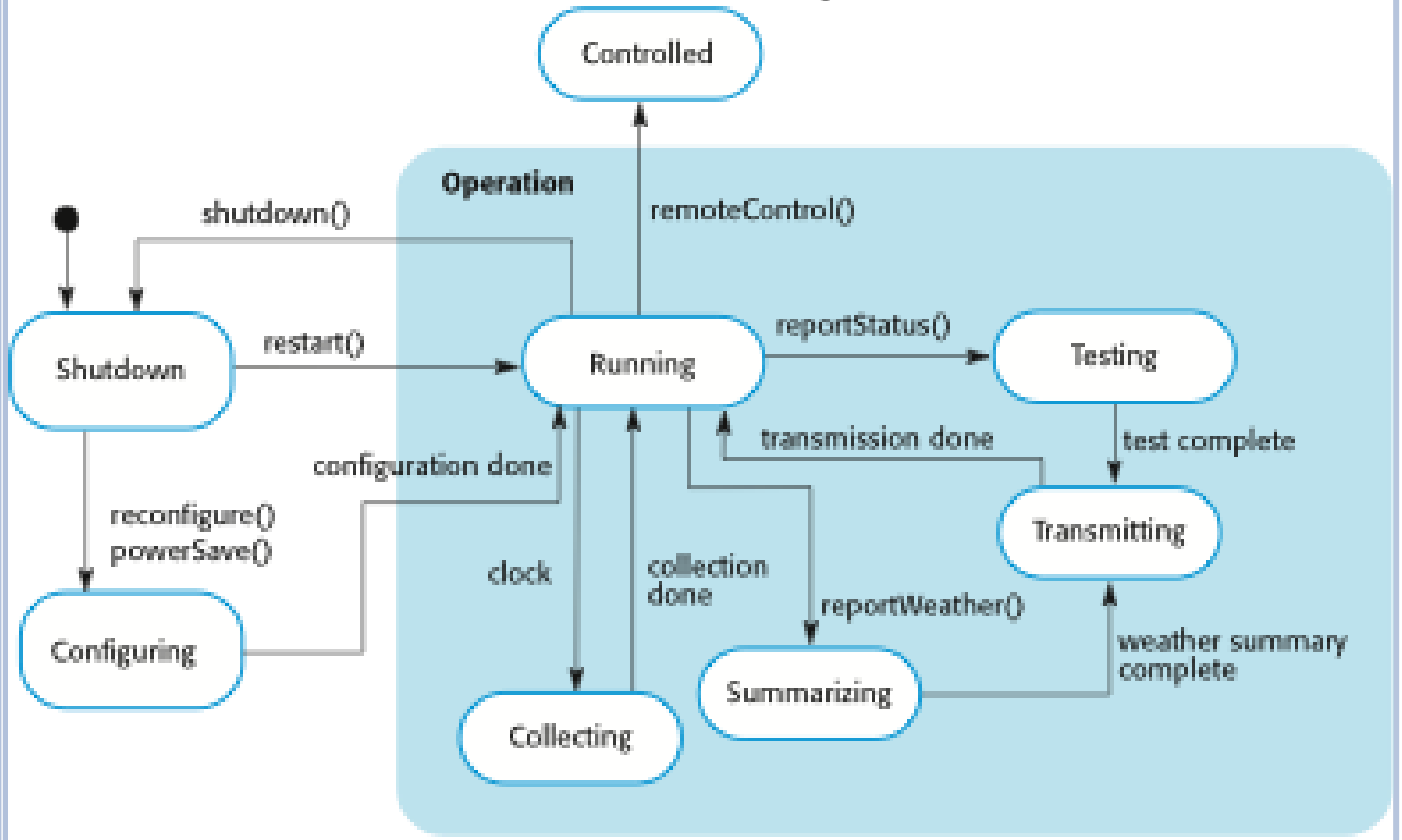
Diagramas de estado

•Los diagramas de estado se utilizan para mostrar cómo los objetos responden a diferentes solicitudes de servicio y las transiciones de estado provocados por estas peticiones.

•Los diagramas de estado son modelos útiles de alto nivel de un sistema o de un comportamiento en tiempo de ejecución de un objeto.

•Normalmente, no es necesario un diagrama de estado para todos los objetos en el sistema. Muchos de los objetos de un sistema son relativamente simples y un modelo de estados añade detalles innecesarios al diseño.

Diagrama de estados de la estación meteorológica



Especificación de interfaces

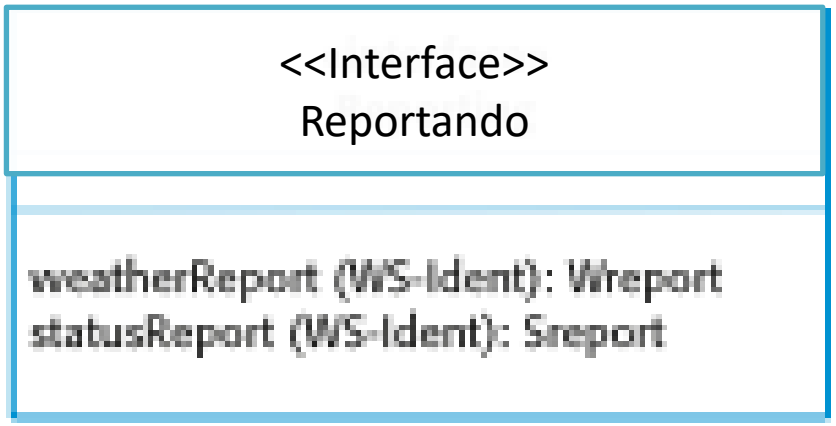
•Las interfaces de objetos tienen que ser especificados para que los objetos y otros componentes pueden ser diseñados en paralelo.

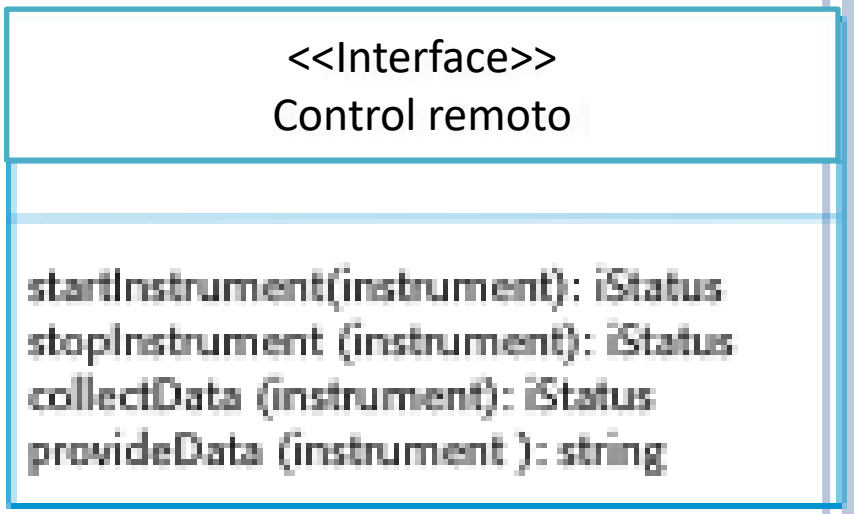
•Los diseñadores deberían evitar el diseño de la representación de la interfaz, sino que deben ocultar esto en el objeto mismo.

•Los objetos pueden tener varias interfaces que son puntos de vista sobre los métodos establecidos.

•El UML utiliza diagramas de clases para la especificación de interfaz pero Java también puede ser utilizado.

Interfaces de la estación meteorológica





Patrones de diseño

•Un patrón de diseño es una forma de reutilizar el conocimiento abstracto sobre un problema y su solución.

•Un patrón es una descripción del problema y la esencia de su solución.

•Debe ser lo suficientemente abstracto para ser reutilizados en diferentes entornos.

•Descripciones de patrón suelen hacer uso de las características orientadas a objetos como la herencia y el polimorfismo.

Elementos del patrón

- Nombre
- Un identificador de patrón significativo.
- Descripción del problema.
- Descripción de la solución.
- No es un diseño concreto, pero es una plantilla para una solución de diseño que pueden ser instanciado de diferentes maneras.
- Consecuencias
- Los resultados y las ventajas y desventajas de aplicar el patrón.

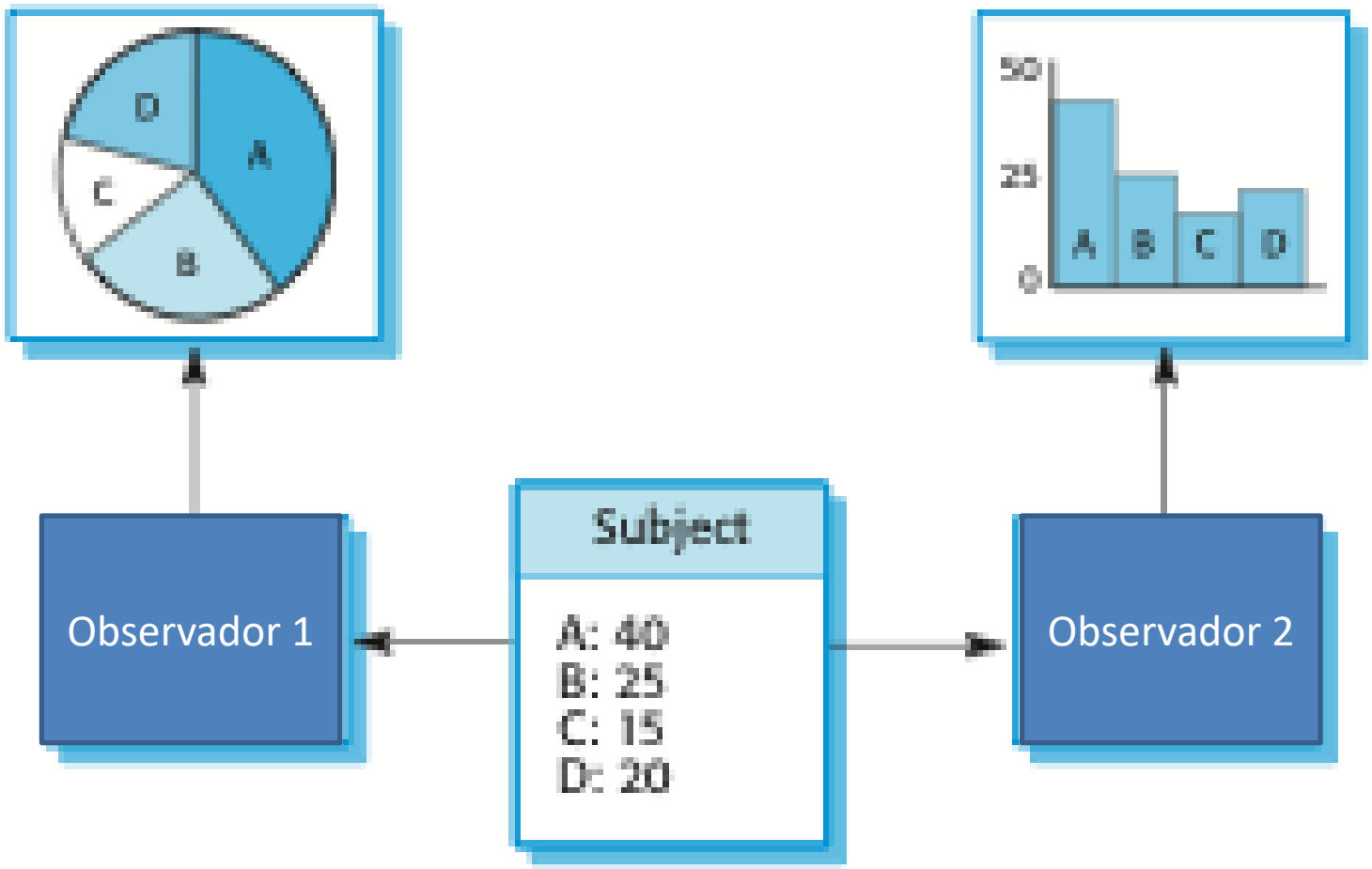
El patrón observador

•Nombre :Observador.
•Descripción:
Separa la vista del estado del objeto, del objeto en sí.
•Descripción del problema :
•Se utiliza cuando se necesitan varias vistas de estado.
Descripción de la solución
•Genera distintas vistas
Consecuencias :
•Las optimizaciones para mejorar el rendimiento de la vistas no son prácticos.

El patrón Observador

Descripción	Separa la vista del estado de un objeto del objeto en sí mismo y permite que se proporcionen vistas alternativas. Cuando el estado del objeto cambia, todas las vistas son automáticamente notificadas y actualizadas para reflejar el cambio.
Descripción del problema	En muchas situaciones, usted tiene que proporcionar varias vistas de información de estado, como por ejemplo una pantalla gráfica y una pantalla tabular. No todos ellos pueden ser conocidos cuando se especifica la información. Todas las presentaciones alternativas deben apoyar la interacción y, cuando se cambia el estado, todas las pantallas deben ser actualizados. Este patrón se puede utilizar en todas las situaciones en que se requiere más de un formato de visualización de la información de estado y donde no es necesario que el objeto, que mantiene la información de estado, sepa acerca de los formatos de visualización específicos utilizados.

Múltiples vistas utilizando el patrón Observador



Reutilización

•Desde la década de 1960 hasta la década de 1990, la mayoría del nuevo software ha sido desarrollado desde cero, escribiendo todo el código en un lenguaje de programación de alto nivel.
•La única reutilización o software significativo fue la reutilización de funciones y objetos en las bibliotecas de lenguajes de programación.
•Los costes y la presión del horario significan que este enfoque se hizo cada vez más inviable, especialmente para los sistemas comerciales y basados en Internet.
•Un enfoque del desarrollo basado en torno a la reutilización de software existente surgió y ahora se utiliza generalmente para el software empresarial y científico.

Niveles de reutilización

- El nivel de abstracción
  - En este nivel, no se emplea el software directamente, pero se utiliza el conocimiento de las abstracciones exitosas en el diseño de su software.
- El nivel del objeto
  - En este nivel, vuelve a utilizar directamente los objetos de una biblioteca en lugar de escribir el código manualmente.
- El nivel de los componentes
  - Los componentes son colecciones de objetos y clases de objetos que se reutilizan en los sistemas de aplicación.
- El nivel de sistema
  - En este nivel, se vuelve a utilizar los sistemas de aplicación enteros.

Costos de Reutilización

Considerar:

- Los costos del tiempo invertido en busca de software para la reutilización y la evaluación de si es o no ajustable a sus necesidades.
- Los costos de la compra de software reutilizable.
- Los costos de adaptación y configuración de los componentes o sistemas de software reutilizables para reflejar los requisitos del sistema que se está desarrollando.
- Los costos de la integración de los elementos de software reutilizables entre sí (si está utilizando software de fuentes diferentes) y con el nuevo código que se ha desarrollado.

Gestión de la configuración

•La gestión de la configuración es el nombre que recibe el proceso general de la gestión de un sistema de software cambiante.

•El objetivo de la gestión de la configuración es apoyar el proceso de integración de sistemas para que todos los desarrolladores puedan acceder al código y a la documentación del proyecto de una manera controlada, averiguar qué cambios se han hecho, y compilar y enlazar componentes para crear un sistema.

Actividades de gestión de configuración

•**Gestión de versiones**, para realizar un seguimiento de las diferentes versiones de los componentes de software. Los sistemas de gestión de la versión incluyen facilidades para coordinar el desarrollo de varios programadores.

•**Integración de sistemas**, proporciona apoyo a los desarrolladores a definir qué versiones de los componentes se utilizan para crear cada versión de un sistema. Esta descripción se utiliza entonces para construir un sistema de forma automática mediante la compilación y la vinculación a los componentes necesarios.

•**Seguimiento de problemas**, brinda apoyo para que los usuarios reporten errores y problemas. Los desarrolladores ven quién está trabajando en estos problemas y cuando están resueltos.

Desarrollo huesped - objetivo

•La mayoría del software está desarrollado en un equipo pero se ejecuta en una máquina diferente (el objetivo).

•De manera más general, podemos hablar de una plataforma de desarrollo y una plataforma de ejecución.

•Una plataforma es algo más que hardware. Incluye el sistema operativo instalado, más otro software de soporte, tal como un sistema de base de datos de gestión o, para las plataformas de desarrollo, un entorno de desarrollo interactivo.

•La plataforma de desarrollo por lo general tiene el software instalado diferente de la plataforma de ejecución; estas plataformas pueden tener diferentes arquitecturas.

Herramientas de plataforma de desarrollo

•Un compilador integrado y sistema de edición dirigido a la sintaxis que le permite crear, editar y compilar código.

•Un sistema de depuración.

•Herramientas de edición de gráficos, tales como herramientas para editar los modelos UML.

•Herramientas de prueba, tales como Junit que puede ejecutar automáticamente una serie de pruebas en una nueva versión de un programa.

•Herramientas de apoyo de proyectos que ayudan a organizar el código para diferentes proyectos de desarrollo.

Entornos de desarrollo integrados (IDEs)

- Las herramientas de desarrollo de software a menudo se agrupan para crear un entorno de desarrollo integrado (IDE).
- Un IDE es un conjunto de herramientas de software que es compatible con diferentes aspectos del desarrollo de software, dentro de algún marco e interfaz de usuario comunes.
- Las IDEs son creados para apoyar el desarrollo de un lenguaje específico de programación como Java. El IDE del lenguaje puede ser especialmente desarrollado, o puede ser una creación de instancias de un IDE de uso general, con herramientas específicas del lenguaje de apoyo.

Desarrollo de código abierto

•Desarrollo de código abierto es un enfoque para el desarrollo de software en el que se publica el código fuente de un sistema de software y se invita a los voluntarios a participar en el proceso de desarrollo

•Sus raíces están en la Free Software Foundation (www.fsf.org), que aboga que el código fuente no debe ser propietario sino siempre debe estar disponible para que los usuarios examinen y modifiquen.

•El software de código abierto extendió esta idea a través de Internet para reclutar a una población mucho más grande de desarrolladores voluntarios.

Los sistemas de código abierto

•El producto de código abierto más conocido es, por supuesto, el sistema operativo Linux que es ampliamente utilizado como un sistema de servidor y, cada vez más, como un entorno de escritorio.

•Otros productos de código abierto importantes son Java, el servidor web Apache, el sistema de gestión de base de datos MySQL, el sistema de elearning Moodle, etc.

Cuestiones de código abierto

- El producto que se está desarrollando, debería hacer uso de componentes de código abierto?
- Se debería utilizar un enfoque de código abierto para el desarrollo del software?

Negocio de código abierto

•Cada vez más empresas están utilizando productos de un enfoque de código abierto para el desarrollo.

•Su modelo de negocio no depende de la venta de un producto de software, sino en la venta de soporte para ese producto.

•Ellos creen que la participación de la comunidad de código abierto permitirá que el software se desarrolle de forma más barata, más rápida y creará una comunidad de usuarios para el software.

Concesión de licencias de código abierto

- Un principio fundamental de desarrollo de código abierto es que el código fuente debe estar disponible gratuitamente.
- Legalmente, el desarrollador del código (ya sea una empresa o un individuo) todavía posee el código y puede imponer restricciones sobre cómo se utiliza mediante una licencia de software de código abierto.
- Algunos desarrolladores creen que si un componente de código abierto se utiliza para desarrollar un nuevo sistema, ese sistema debe también ser de código abierto.
- Otros permiten que su código sea usado sin esta restricción y los sistemas desarrollados pueden ser como sistemas de código cerrado.