Ingeniería del Software I

2º

Tema 5.2
Diseño detallado e implementación

Universidad Francisco de Vitoria
UFV Madrid

Grado en Ingeniería Informática
Escuela Politécnica Superior

Diseño e implementación

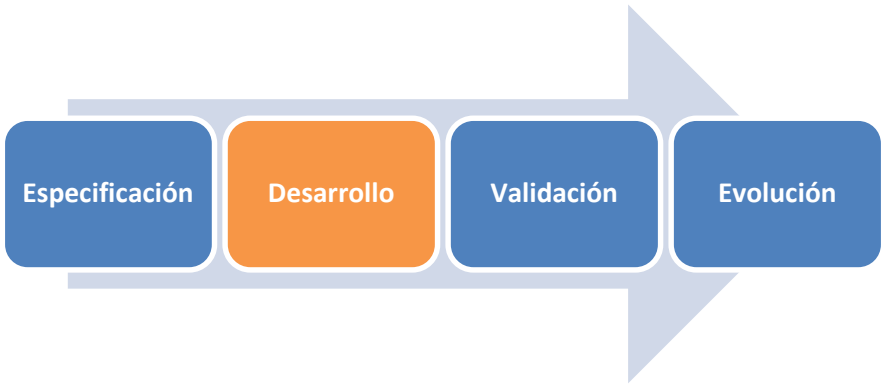
Universidad Francisco de Vitoria UFV Madrid

Objetivos

- Actividades más importantes dentro de un proceso
- Modelos diferentes que pueden usarse para diseño orientado a objetos
- Idea de patrones de diseño
- Introducir conflictos clave que se debe considerar al implementar

Índice

1. Diseño orientado a objetos con UML
2. Patrones de diseño
3. Conflictos de implementación
4. Desarrollo de código abierto



Diseño e implementación

Universidad Francisco de Vitoria UFV Madrid

- **Diseño e implementación entrelazadas**. Las actividades de diseño e implementación de software se encuentran invariablemente entrelazadas.
- El diseño trata sobre cómo resolver un problema, de modo que siempre existe un proceso de diseño
- **Siempre hay un proceso de diseño**. no tiene sentido en absoluto si implementa su sistema al configurar un paquete comercial.
- **¿Compro o desarrollo?** Una de las decisiones de implementación más importantes, que se toman en una etapa inicial de un proyecto de software, consiste en determinar si debe comprar o diseñar el software de aplicación.

- *Libro: El Mítico Hombre-Mes (añadir recursos humanos a un proyecto retrasado lo hace demorarse aún más)*

Diseño orientado a objetos con UML

Universidad Francisco de Vitoria UFV Madrid

Objetos = datos + métodos

- Los objetos incluyen datos y operaciones para manipular dichos datos. En consecuencia, pueden entenderse y modificarse como entidades independientes. Cambiar la implementación de un objeto o agregar servicios no afectará a otros objetos del sistema.
- Puesto que los objetos se asocian con cosas, con frecuencia hay un mapeo claro entre entidades del **mundo real** (como componentes de hardware) y sus **objetos controladores** en el sistema. Esto mejora la **comprensibilidad** y, así la **mantenibilidad** del diseño.

Diseño orientado a objetos con UML

Universidad Francisco de Vitoria UFV Madrid

- Para desarrollar un diseño de sistema desde el concepto hasta el diseño detallado orientado a objetos, hay muchas **cuestiones por hacer**:
  1. **Comprender y definir el contexto y las interacciones externas con el sistema.**
  2. **Diseñar la arquitectura del sistema.**
  3. **Identificar los objetos principales en el sistema.**
  4. **Desarrollar modelos de diseño.**
  5. **Especificar interfaces**

Contexto e interacciones del sistema

- La primera etapa en cualquier proceso de diseño de software es desarrollar la comprensión de las relaciones entre el software que se diseñará y su ambiente externo.
- La comprensión del contexto permite también determinar las fronteras del sistema.

Patrones de diseño

- El patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede reutilizarse en diferentes configuraciones. El **patrón no es una especificación detallada**. Más bien, puede considerarla como una descripción de experiencia acumuladas, una solución bien probada a un problema común.

Los problemas del diseño

- Para utilizar los patrones en su diseño, es necesario **reconocer que cualquier problema de diseño** que se enfrente puede tener un patrón asociado que se puede aplicar.
- **Recomendar a varios objetos que el estado de un objeto se ha cambiado** (patrón Observador).
- **Poner en orden las interfaces a un número de objetos relacionados** que a menudo se han desarrollado de **forma incremental** (patrón de la fachada).
- **Proporcionar una forma estándar de acceder a los elementos de una colección**, con independencia de la forma en que la colección es implementada (patrón Iterator).
- **Permitir la posibilidad de ampliar la funcionalidad de una clase** existente **en tiempo de ejecución** (patrón Decorador).

Cuestiones de aplicación

Universidad Francisco de Vitoria UFV Madrid

- El **enfoque aquí no está en la programación**, aunque esto es obviamente importante, sino en otras cuestiones de implementación que a menudo no están cubiertos en los textos de programación:
  - **Reutilización**. El software más moderno se construye mediante la reutilización de componentes o sistemas ya existentes. Cuando se está desarrollando software, usted debe hacer el mayor uso posible del código existente.
  - **Gestión de la configuración**. Durante el proceso de desarrollo, se tiene que llevar la cuenta de las muchas versiones diferentes de cada componente de software en un sistema de gestión de la configuración.
  - **Desarrollo con objetivo al anfitrión**. El software de producción no suele ejecutar en el mismo computador que el entorno de desarrollo de software. Más bien, se desarrolla en una computadora (el sistema anfitrión) y lo ejecuta en un equipo independiente (el sistema de destino).

Reutilización

Universidad Francisco de Vitoria UFV Madrid

- 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

Universidad Francisco de Vitoria UFV Madrid

- **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.

Costes de Reutilización

Universidad Francisco de Vitoria UFV Madrid

- **Los costes del tiempo** invertido en busca de software para la reutilización y la evaluación de si es o no ajustable a sus necesidades.
- En casos específicos, los **costes de la compra** de software reutilizable. Para grandes sistemas off-the-shelf, estos costos pueden ser muy altos.
- **Los costes de adaptación y configuración** de los componentes o sistemas de software reutilizables para reflejar los requisitos del sistema que se está desarrollando.
- **Los costes 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.

Universidad Francisco de Vitoria UFV Madrid

- 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.
- Más detalle en libro de referencia (Sommerville)

Actividades de gestión de configuración

Universidad Francisco de Vitoria UFV Madrid

- **Gestión de versiones**, en el que se prestó apoyo 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.
- La **integración de sistemas**, donde se proporciona apoyo para ayudar 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**, donde se brindó apoyo para permitir a los usuarios reportar errores y otros problemas, y para permitir que todos los desarrolladores vean quién está trabajando en estos problemas y cuando son arreglados.

Desarrollo de objetivo al anfitrión

Universidad Francisco de Vitoria UFV Madrid

- La mayoría del software está **desarrollado en un equipo** (el anfitrión), 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

Universidad Francisco de Vitoria UFV Madrid

- Un **compilador integrado y sistema de edición** dirigido a la sintaxis que le permite crear, editar y compilar código.
- Un sistema de **depuración** de idioma.
- 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)

Universidad Francisco de Vitoria UFV Madrid

- 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.

Factores de implementación de componentes / sistema

Universidad Francisco de Vitoria UFV Madrid

- **Si un componente está diseñado para una arquitectura de hardware específico**, o si recurre **a otro sistema de software**, es obvio que se debe implementar en una plataforma que proporciona el **soporte de hardware y software necesarios**.
- **Los sistemas de alta disponibilidad** pueden requerir componentes para ser desplegado en **más de una plataforma**. Esto significa que, en el caso de fallo de la plataforma, una implementación alternativa del componente está disponible.
- **Si hay un alto nivel de tráfico de comunicaciones entre los componentes**, por lo general **tiene sentido desplegarlos en la misma plataforma o en plataformas que están físicamente cerca uno del otro**. Esto reduce el retardo entre el tiempo que un mensaje es enviado por uno de los componentes y recibida por otro.

Desarrollo Open Source

Universidad Francisco de Vitoria UFV Madrid

- **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 propietaria sino siempre debe estar disponible para que los usuarios examinen y modifiquen a su antojo.
- El software de código abierto extendió esta idea a través de Internet para reclutar a una población mucho más grande de desarrolladores voluntarios. Muchos de ellos son también usuarios del código.

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 y el sistema de gestión de base de datos MySQL.

Cuestiones de código abierto

Universidad Francisco de Vitoria UFV Madrid

- 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

Universidad Francisco de Vitoria UFV Madrid

- 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

Universidad Francisco de Vitoria UFV Madrid

- Un **principio fundamental** de desarrollo de código abierto es que el código fuente debe estar **disponible gratuitamente**, esto no quiere decir que cualquiera puede hacer lo que quieran con ese código.
- Legalmente, el desarrollador del código (ya sea una empresa o un individuo) todavía posee el código. Ellos pueden imponer restricciones sobre cómo se utiliza mediante la inclusión de condiciones jurídicamente vinculantes en una licencia de software de código abierto.
- Algunos desarrolladores de software libre creen que si un componente de código abierto se utiliza para desarrollar un nuevo sistema, luego ese sistema debe también ser de código abierto.
- **Otros están dispuestos a permitir que su código sea usado sin esta restricción**. Los sistemas desarrollados pueden ser propietarios y vendidos como sistemas de código cerrado.

Gestión de licencias

Universidad Francisco de Vitoria UFV Madrid

- Establezca un sistema de mantenimiento de la información acerca de los componentes de código abierto que se descargan y utilizan.
- Sea consciente de los diferentes tipos de licencias y entienda cómo se licencia un componente antes de su uso.
- Esté al tanto de las vías de evolución para los componentes.
- Eduque a la gente sobre el código abierto.
- Disponga de sistemas de auditoría en orden.
- Participe en la comunidad de código abierto.

Código abierto

Universidad Francisco de Vitoria UFV Madrid

Desarrollo de código abierto

- El desarrollo de código abierto es un enfoque al desarrollo de software en que se publica el código de un sistema de software y se invita a voluntarios a participar en el proceso de desarrollo.
- La Free Software Foundation (http://www.fsf.org), que aboga porque el código fuente no debe ser propietario sino, más bien, tiene que estar siempre disponible para que los usuarios lo examinen y modifiquen como deseen.
- El software de código abierto extendió esta idea al utilizar Internet para reclutar a una población mucho mayor de desarrolladores voluntarios. La mayoría de ellos también son usuarios del código.
- Sin embargo, en la práctica, **los sistemas exitosos de código abierto aún se apoyan en un grupo central de desarrolladores** que controlan los cambios al software.
- Ejemplos: Linux, Java, Apache (servidor Web), mySQL

Licencia de código abierto

Universidad Francisco de Vitoria UFV Madrid

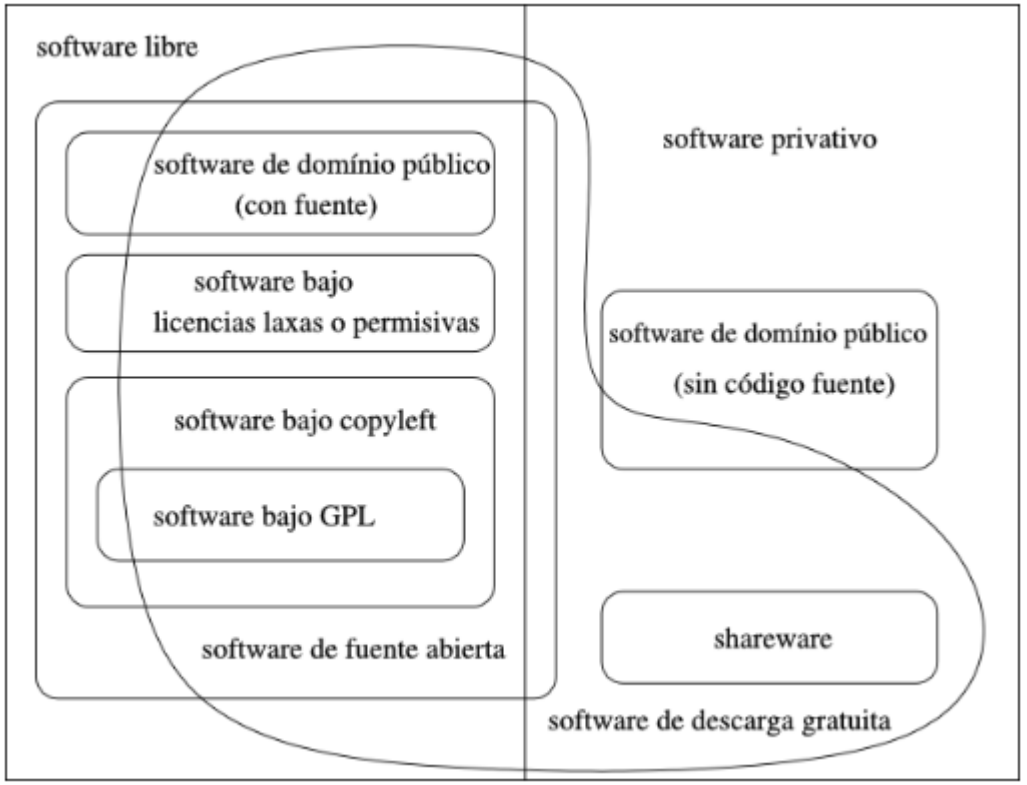
- La mayoría de las licencias de código abierto se derivan de uno **de tres modelos generales**:

1. **La licencia pública general GNU - GPL** se conoce como licencia “recíproca”; de manera simple, significa que si usted usa software de código abierto que esté permitido bajo la licencia GPL, entonces debe hacer que dicho software sea de código abierto.
2. **La licencia pública menos general GNU - LGPL** es una variante de la licencia anterior, en la que usted puede escribir componentes que se vinculen con el código abierto, sin tener que publicar el código de dichos componentes. Sin embargo, si cambia el componente permitido, entonces debe publicar éste como código abierto.
3. **La licencia Berkeley Standard Distribution - BSD** es una licencia no recíproca, lo cual significa que usted no está obligado a volver a publicar algún cambio o modificación al código abierto. Puede incluir el código en sistemas propietarios que se vendan. Si usa componentes de código abierto, debe reconocer al creador original del código.

Licencia de código abierto

Universidad Francisco de Vitoria UFV Madrid

Este diagrama, diseñado originalmente por Chao-Kuei



Software libre

Software libre es aquel que se suministra con autorización para que cualquiera pueda usarlo, copiarlo y/o distribuirlo, ya sea con o sin modificaciones, gratuitamente o mediante pago. En particular, esto significa que el código fuente debe estar disponible. «Si no es fuente, no es software»

Software de código abierto («*Open Source*»)

Algunas personas utilizan la expresión software de «código abierto» para referirse más o menos a la misma categoría a la que pertenece el software libre. Sin embargo, no son exactamente el mismo tipo de software: ellos aceptan algunas licencias que nosotros consideramos demasiado restrictivas, y hay licencias de software libre que ellos no han aceptado. De todos modos, las diferencias entre lo que abarcan ambas categorías son pocas: casi todo el software libre es de código abierto, y casi todo el software de código abierto es libre. Nosotros preferimos la expresión «software libre» porque se refiere a libertad, cosa que no sucede con la expresión «código abierto».

Licencia de código abierto

Universidad Francisco de Vitoria UFV Madrid

Software de dominio público

El software de dominio público es aquel que no tiene derechos de autor. Si el código fuente es de dominio público, se trata de un caso especial de software libre sin copyleft, lo que significa que algunas copias o versiones modificadas pueden no ser libres en absoluto.

Software con copyleft

El software con copyleft es software libre cuyos términos de distribución garantizan que todas las copias de todas las versiones tengan aproximadamente los mismos términos de distribución. Esto significa, por ejemplo, que las licencias copyleft generalmente no permiten que terceros le agreguen requisitos adicionales al software (aunque puede estar permitido agregar un conjunto limitado de requisitos que se consideran seguros) y exigen que el código fuente esté disponible. Esto tutela el programa y sus versiones modificadas contra algunas de las formas más comunes de convertirlo en software privativo.

Software libre sin copyleft

Los programas publicados sin copyleft vienen con permiso de redistribución y modificación, como así también con el permiso de agregarle restricciones.

Si un programa es libre pero no tiene copyleft, es posible que algunas copias o modificaciones no sean libres en absoluto. Una empresa de software puede compilar el programa, con o sin modificaciones, y distribuir el archivo ejecutable como software privativo.

Software con licencia permisiva, laxa

Entre las licencias permisivas, laxas, se incluyen la licencia X11 y ambas licencias BSD. Estas licencias permiten utilizar el código de cualquier manera, inclusive la distribución de binarios privativos con o sin modificaciones del código

Licencia de código abierto

Universidad Francisco de Vitoria UFV Madrid

Software con licencia GPL

La Licencia Pública General de GNU (General Public License - GNU GPL) consiste en un conjunto específico de cláusulas de distribución para publicar programas con copyleft. El Proyecto GNU la usa para la mayoría de los programas que distribuye.

Equiparar el software libre con software cubierto por la licencia GPL es por lo tanto un error.

El sistema operativo GNU

El sistema operativo GNU es un sistema completamente libre de tipo Unix, que el Proyecto GNU comenzó a desarrollar en 1984.

Un sistema operativo de tipo Unix está constituido por muchos programas. El sistema GNU incluye todos los paquetes oficiales de GNU. También incluye muchos otros paquetes, como el sistema X Window y TeX, que non son software de GNU.

Software GNU

El software GNU se publica con el auspicio del Proyecto GNU. Si un programa es software GNU, también decimos que es un programa o un paquete de GNU, y el manual o el archivo README de un paquete de GNU así lo debe indicar. Además, en el Directorio de Software Libre están identificados todos los paquetes de GNU.

Software GNU bajo copyright de la FSF

Los desarrolladores de paquetes de GNU pueden transferir los derechos de autor a la FSF, o pueden quedárselos. La elección es suya.

Licencia de código abierto

Universidad Francisco de Vitoria UFV Madrid

Software que no es libre

El software «que no es libre» [*nonfree software*] es cualquier software que no es libre. Está prohibido su uso, redistribución o modificación, o requiere que se solicite permiso, o tiene tantas restricciones que de hecho no se puede hacer libremente.

Software Privativo

El software privativo es otro nombre para designar el software que no es libre. En el pasado habíamos subdividido el software que no es libre en «software semilibre», que podía ser modificado y redistribuido sin fines comerciales, y «software privativo», que no podía ser modificado ni redistribuido. Pero hemos abandonado esta distinción y ahora utilizamos el término «software privativo» como sinónimo de software que no es libre.

Conclusión

Universidad Francisco de Vitoria UFV Madrid

- En el desarrollo de software, siempre se debe considerar la posibilidad de la reutilización de software existente, ya sea como componentes, servicios o sistemas completos.
- La gestión de la configuración es el proceso de gestión de cambios en un sistema de software en constante evolución. Es esencial cuando un equipo de personas están cooperando para desarrollar software.
- El desarrollo de código abierto implica hacer el código fuente de un sistema públicamente disponible. Esto significa que muchas personas pueden proponer cambios y mejoras en el software.