UD 2 – Evaluación de los entornos de desarrollo integrados

CONCEPTOS

Para poder elaborar una aplicación es necesario utilizar una serie de herramientas que nos permitan escribirla, depurarla, traducirla y ejecutarla. Se conoce como entorno de desarrollo integrado (IDE, Integrated Development Environment) y su función es proporcionar un marco de trabajo para el lenguaje de programación utilizado.
Su selección y una utilización óptima será una decisión muy importante en el procedimiento de creación de software ya que el programador deberá trabajar con ella durante la mayor parte de tiempo que dedique a la creación de nuevas aplicaciones.
Debemos seleccionarlo teniendo en cuenta el lenguaje de programación y el desarrollo de una aplicación determinada, y además conocer la mayoría de las funcionalidades y saber aprovechar todas las facilidades que ofrezca el entorno.

FUNCIONES DE UN ENTORNO DE DESARROLLO

Es un programa compuesto por una serie de herramientas que utilizan los programadores para desarrollar código. Este programa puede estar pensado para utilización con un único lenguaje de programación o bien puede dar cabida a varios.

Las herramientas que normalmente componen un entorno de desarrollo integrado son un sistema de ayuda para la construcción de:

- Constructor de interfaces gráficas de usuario (GUI)
- Editor de texto
- Compilador/intérprete
- Depurador

Opcionalmente, pueden presentar

- sistema de acceso a bases de datos y gestión de archivos
- sistema de control de versiones
- sistema de pruebas
- sistema de refactorización
- generador automático de documentación.

1. Construcción de interfaces gráficas de usuario

Conjunto de funcionalidades que permiten incorporar, editar y eliminar componentes gráficos de forma sencilla en la aplicación que se está desarrollando. Estos componentes facilitarán la interacción del usuario con el ordenador.

2. Editor de texto

Programa que permite crear y modificar archivos digitales compuestos únicamente por texto sin formato, conocidos comúnmente como archivos de texto o texto plano.
El editor de texto es la herramienta más utilizada porque ofrece la posibilidad de crear y modificar los contenidos desarrollados, el código de programación que hará funcionar adecuadamente la
aplicación.
Siempre que el IDE reconozca el lenguaje de programación (o disponga del componente adecuado) el editor ofrecerá:

-Resaltado de sintaxis (syntax highlighting): las palabras clave serán reconocidas con colores, lo que facilitará mucho el trabajo del programador.

-Realización de código (code completion): se reconocerá el código que se está escribiendo y, por ejemplo en un objeto o clase, ofrecerá sus atributos, propiedades o métodos para que el programador seleccione cuál quiere referenciar.

-Corrector de errores: normalmente desde un punto de vista de sintaxis.

-Regiones plegables: ocultación de ciertas partes del código, con el fin de facilitar la visualización de aquellos fragmentos más relevantes.

3.Compilador:

En función del lenguaje de programación utilizado, el IDE podrá ofrecer la funcionalidad de compilarlo. Un compilador traduce un código de programación en un lenguaje máquina capaz de ser interpretado por los procesadores y de ser ejecutado.
A la hora de compilar un código de programación, los entornos integrados de desarrollo dispondrán de diferentes fases de análisis del código, como son la fase semántica y la fase lexicográfica. La compilación mostrará los errores encontrados o generará código ejecutable, en el caso de encontrar ninguno.

4.Intérprete

El intérprete traduce el código de alto nivel a código de bytes, pero, a diferencia del compilador, lo hace en tiempo de ejecución. No guarda el resultado.

5.Depurador

El depurador es un programa que permite probar y depurar el código fuente de un programa, facilitando la detección de errores. Algunas de las funcionalidades típicas de los depuradores son:

- Permitir la ejecución línea a línea del código validando los valores que van adquiriendo las variables.
- Pausar el programa en una determinada línea de código, haciendo uso de uno o varios puntos de ruptura (breakpoints).
- Algunos depuradores ofrecen la posibilidad de poder modificar el contenido de alguna variable mientras se está ejecutando.

6.Acceso a bases de datos y gestión de archivos

Algunos lenguajes ofrecen la posibilidad de integrar código de acceso a bases de datos (o código de sentencias estructuradas). Para facilitar esta funcionalidad es muy recomendable disponer de la posibilidad de acceder directamente a la base de datos desde el mismo entorno de desarrollo y no haber de utilizar otro. Lo mismo sucederá con la gestión de archivos.

7.Control de versiones

El control de versiones permite volver a una versión anterior que sea estable o que cumpla unas determinadas características que los cambios hechos no cumplen. Es importante tener un histórico del trabajo realizado o de las versiones que se han dado por buenas o se han modificado.

8.Refactorización

La refactorización (refactoring) es una técnica de la ingeniería de software dirigida a reestructurar un código fuente, alterando su estructura interna sin cambiar su comportamiento externo.

9.Documentación y ayuda

La documentación y ayuda provee acceso a documentación, manuales y ayuda contextual sobre las instrucciones y la sintaxis de los lenguajes soportados.

EJEMPLOS DE ENTORNO DE DESARROLLO

Hoy día los entornos de desarrollo proporcionan un marco de trabajo para la mayoría de los lenguajes de programación existentes en el mercado (por ejemplo C, C ++, C #, Java o Python, entre otros).
Además, es posible que un mismo entorno de desarrollo permita utilizar varios lenguajes de programación, como es el caso de Eclipse o Visual Studio.

TIPOS DE SOFTWARE

Cuando iniciamos el proceso de instalación de un programa, uno de los primeros pasos es siempre la aceptación de una licencia de uso, es decir, un contrato con una serie de términos y condiciones que se deben cumplir. Se nos presenta un texto legal y para poder continuar tenemos que aceptar dicha licencia. Si la licencia no se acepta el programa no puede instalarse.
Existen muchos tipos de licencias de software, pero podemos dividirlas en dos grandes grupos: propietarias y libres.

UD 2 – Evaluación de los entornos de desarrollo integrados

1.Software Libre

-Proporciona la libertad de:
- Ejecutar el programa, para cualquier propósito;
- Estudiar el funcionamiento del programa, y adaptarlo a sus necesidades;
- Redistribuir copias;
- Mejorar el programa, y poner sus mejoras a disposición del público, para beneficio de toda la comunidad.

-Muy común utilizar licencia GNU GPL.
-Puede ser gratuito o no, pero siempre otorga estas libertades. De lo contrario no es libre. (p.ej. LibreOffice el libre y es gratuito)
Una de las mayores ventajas es que programadores de todo el mundo pueden contribuir (en muchos casos de forma desinteresada) a mejorar o depurar los programas, lo que logra un producto muy robusto.

El concepto de software libre surgió en la década de los ochenta y su principal ponente fue el proyecto GNU, cuyo objetivo era desarrollar un sistema operativo, completamente libre. Actualmente, la fundación de Software libre (FSF) es el organismo más activo en la defensa y promoción de esta idea.

2. Propietario o PRIVATIVO

En el software propietario, privativo o no libre, con o sin código fuente disponible, no se permite alguna (o ninguna) de las libertades del software libre. Los programas distribuidos con esta licencia generalmente solo incluyen los archivos necesarios para ejecutarlo. El software privativo puede ser gratuito o de pago (p.ej. Microsoft Word es propietario y de pago).

3. Codigo abierto

A partir de la década de los 90 algunos desarrolladores empezaron a utilizar el término de código abierto que en la práctica implica lo mismo que el software libre. Se enfoca más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.
Existen muchas licencias de software libre, pero la más conocida es la licencia GPL (General Public Licence), creada en el marco del proyecto GNU. Es una licencia que te permite modificar cualquier software libre que llegue a tus manos y compartirlo con otros usuarios ya sea pagando o gratis pero siempre permitiendo que la nueva versión pueda ser modificada después.
Hasta el momento solo hemos hablado de derechos de uso, pero no hemos abordado el aspecto económico; es muy habitual cometer el error de confundir estos dos aspectos.

EL PRECIO DEL SOFTWARE: SOFTWARE COMERCIAL, SHAREWARE Y FREEWARE

El tipo de licencia escogido a la hora de ofrecer un programa es independiente de que se haya desarrollado o no como negocio (software comercial). Así, un desarrollador de software libre puede poner a la venta su trabajo, mientras que podrían regalarnos un programa propietario por ejemplo al comprar una revista.

Se llama freeware a los programas que podemos descargar y utilizar gratuitamente. Shareware indica que el programa tiene un costo pero se nos ofrece gratuitamente una versión limitada (en tiempo de uso o en funcionalidades): podemos así probar el programa y decidir después si queremos adquirir la versión completa.

1.Shareware
- Evaluación gratuita del producto.
- Límite de tiempo de evaluación.
- Restricciones en las capacidades finales.
- Requiere un pago para uso completo.
- También existe “shareware de precio cero” (pero esta modalidad es poco común)
- WinRAR, PC File, Paint Shop Pro, PC-Tools,Virus Scan...

2.Freeware
- Se distribuye sin coste.
- No confundir con código libre
- Mantiene el Copyright (derechos de autor)
- NO se puede modificar.
- NO se puede utilizar libremente (modificar ni vender).
- Ofrece funcionalidad completa de manera gratuita.
- Suele incluir licencia comercial

UD 2 – Evaluación de los entornos de desarrollo integrados

CRITERIOS PARA LA ELECCIÓN DE UN IDE

Ya sabemos qué es un entorno de desarrollo integrado y qué particularidades y funcionalidades ofrece, ahora necesitamos saber cuál elegir.
Para poder elegir correctamente un IDE, necesitamos saber las características que buscamos en él, si satisface nuestras necesidades y si cumple con los requisitos técnicos del sistema.
Entre los criterios nos encontramos:

- Sistema operativo
- Lenguaje de programación y frameworks
- Herramientas y disponibilidad

1. Sistema operativo

Sin lugar a dudas uno de los criterios más restrictivos a la hora de seleccionar nuestro IDE es saber en qué sistema operativo vamos a trabajar, y más importante aún para que sistema operativo vamos a desarrollar nuestro software.
Si desarrollamos aplicaciones para Linux, inusual desarrollar la aplicación en Windows. Esto se debe a que el IDE tiene un compilador integrado. Si recordamos el proceso de obtención de código ejecutable de la unidad didáctica anterior, veremos que el compilador se encarga de traducir el código fuente en código objeto, que es el que ejecutará el sistema. Este problema es fácilmente salvable compilando nuestro código fuente en un compilador de otro sistema operativo (siempre y cuando exista).

2.Lenguaje de programación y framework

Como vimos anteriormente un IDE puede soportar uno o varios lenguajes de programación, por lo que saber en qué lenguaje vamos a codificar nuestro software y qué lenguajes nos ofrecen los distintos IDEs es información a tener en cuenta.
Este criterio va de la mano con el sistema operativo, ya que si quisiéramos desarrollar en Visual Basic bajo Linux no sería Visual Estudio nuestra opción, sino que tendríamos que utilizar otro IDE como Gambas.
Lo mismo ocurre con las plataformas de trabajo, también llamadas framework. Si vamos a desarrollar con Visual Basic con la plataforma .NET bajo Linux, tendríamos que usar Mono Develop en lugar de Visual Studio.

Herramientas y disponibilidad

Las diferentes herramientas de las que dispone un IDE son el último criterio de selección, seguramente nos encontremos con varios IDE que cumplen los requisitos del lenguaje y sistema operativo, pero no todos tiene las mismas funciones, por lo que saber cuales son las
funcionalidades es un elemento a tener en cuenta en la toma de una decisión.
En ocasiones pueden ser restrictivos ya no solo por tus propias preferencias sino por el modo de trabajo si nuestros compañeros desarrolladores están usando Java con un sistema de control de versiones Subversion (SVN) podríamos usar indistintamente Netbeans, Eclipse, etc. a que este sistema de control de versiones está disponible en esos IDE.

Aún nos queda tratar el asunto de disponibilidad, el cual, una vez comprobados todos los criterios de selección, puede desbaratar nuestra toma de decisiones. El aspecto más restrictivo de la disponibilidad reside en el precio de la aplicación, dependiendo de nuestro presupuesto
podemos acceder a diferentes IDEs.
Aunque nuestro presupuesto sea escaso, existen soluciones muy económicas incluso gratuitas.

Conceptos a tener en cuenta al instalar un IDE

JVM (Java Virtual Machine , máquina virtual de Java) se encarga de interpretar el código de bytes y generar el código máquina del ordenador (o dispositivo) en el que se ejecuta la aplicación. Esto significa que necesitamos una JVM diferente para cada entorno. Se incluye dentro del JRE
JRE (Java Runtime Environment) es un conjunto de utilidades de Java que incluye la JVM, librerías y el conjunto de software necesario para ejecutar las aplicaciones Java.
JDK (Java Development Kit , kit de desarrollo de Java) es el conjunto de herramientas para desarrolladores; contiene, entre otras cosas, el JRE y el conjunto de herramientas para compilar código, empaquetarlo, generar documentación, etc.

En la siguiente figura se puede observar de forma esquemática el proceso de conversión del código Java, desde la creación de su código fuente hasta la obtención del código máquina.

UD 2 – Evaluación de los entornos de desarrollo integrados

HERRAMIENTAS CASE Computer Aided Software Engineering (Ingeniería de software asistida por ordenador)

Por tanto se refiere al desarrollo y mantenimiento de proyectos de software con la ayuda de varias herramientas automatizadas.
Las herramientas CASE son similares y se inspiraron en parte en las herramientas de diseño asistido por computadora (CAD) utilizadas para diseñar productos de hardware.
Son programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante las etapas del Ciclo de Vida de desarrollo de software.
Son herramientas destinadas a aumentar la productividad en el desarrollo de software reduciendo costes en términos de tiempo y de dinero

1.Ejemplos

- Herramientas de modelado, para reflejar el análisis del sistema. Los modelos habituales son Entidad-Relación y UML
- Gestión de proyectos, para planificar los proyectos, asignar tareas, los tiempos, etc.
- Herramientas de verificación y validación, usadas para analizar la corrección del código y su rendimiento.
- Generadores automáticos de código a partir de otras especificaciones (por ejemplo, gráficas)
- Generadores automáticos de documentación técnica y de usuario

2. Definiciones

- CASE: Ayuda por Computadora a la Ingeniería de Software.
- TECNOLOGIA CASE: instrumentos y técnicas de software para automatizar la disciplina de ingeniería incluyendo metodologías y herramientas automatizadas.
- HERRAMIENTA CASE: Una herramienta del software que automatiza (por lo menos en parte) una parte del ciclo de desarrollo de software.
- SISTEMA CASE: Un conjunto de herramientas CASE integradas que comparten una interfaz del usuario común y corren en un ambiente computacional común.
- KIT de HERRAMIENTAS CASE: Un conjunto de herramientas CASE integradas que se han diseñado para trabajar juntas y automatizar (o proveer ayuda automatizada al ciclo de desarrollo de software, incluyendo el análisis, diseño, codificación y prueba).

3.Clasificación

Existen herramientas que cubren gran parte del ciclo de vida del software. Existen herramientas que sólo cubren alguna/s fase/s dentro de la etapa de desarrollo. Existen herramientas que se pueden agrupar en distintas fases cada herramienta tiene sus características: lenguaje de generación, análisis estructurado u orientado a objetos, etc.

4.Objetivos

La tecnología CASE supone la automatización del desarrollo del software, contribuyendo a mejorar la calidad y la productividad en el desarrollo de sistemas de información a la hora de construir software se plantean los siguientes objetivos:

- Aumentar la productividad en las áreas de desarrollo y mantenimiento de los sistemas informáticos. (Reducir tiempos y costes).
- Mejorar la calidad del software desarrollado.
- Mejorar la gestión y dominio sobre el proyecto en cuanto a su planificación, ejecución y control.
- Mejorar el archivo de datos (enciclopedia), de conocimientos (know- how) y sus facilidades de uso, reduciendo la dependencia de analistas y programadores.
- Automatizar:
    - El desarrollo del software.
    - La documentación
    - La generación del código
    - El chequeo de errores
    - La gestión del proyecto
- Permitir:
    - La reutilización (reusabilidad) del software
    - La portabilidad del software
    - La estandarización de la documentación

- Integrar las fases de desarrollo (Ingeniería del software)
- Facilitar la utilización de las distintas metodologías que desarrolla la propia Ingeniería del Software.

5.Ejemplos

- SAP PowerDesigner: Herramienta que soporta la modelización de datos (soporta los modelos de datos conceptuales, lógicos y físicos) y modelización de procesos de negocio y de aplicación (UML y ofrece un mapeo relacional/de objeto) actualmente propiedad de SAP. También soporta, entre otros, la generación de código y de reportes.

- System Architect: Unicom System Architect es una herramienta de arquitectura empresarial que permite la modelización de procesos de negocios, aplicaciones y datos.

- Visual Paradigm: Herramienta que soporta la modelización de sistemas (con diagramas UML, DER, DFD, etc) gestión de proyectos, prototipado, generación de código, etc.

- ERwin: Erwin es una herramienta de diseño de base de datos. Permite el modelado lógico de los requerimientos de información hasta el modelado físico perfeccionado para las características específicas de la base de datos diseñada. ERwin permite visualizar la estructura, los elementos importantes, y optimizar el diseño de la base de datos. Genera automáticamente las tablas y miles de líneas de procedimientos almacenados y disparadores para los principales tipos de base de datos.

6.Clasificación:

No existe una única clasificación de herramientas CASE y, en ocasiones, es difícil incluirlas en una clase determinada. Podrían clasificarse atendiendo a:

- Las fases del ciclo de vida del desarrollo de sistemas que cubren.
- Su funcionalidad.

Según fases del ciclo de vida del desarrollo:

Herramientas Upper CASE - Se usan en las etapas de planificación, análisis y diseño.
Herramientas Lower CASE - Se usan en la implementación, las pruebas y en el mantenimiento.
Herramientas Integrated CASE - Son de utilidad en todas las fases del ciclo de vida, desde la reducción de requisitos y las pruebas hasta la documentación.

Según la funcionalidad:

Herramientas de diagramas: los componentes del sistema y el flujo de datos y de control entre estos componentes se representan mediante herramientas de diagrama usando gráficos. Ejemplo: Flowchart Maker, ArgoUML, Visio, Giffly, Día.
Herramientas de modelado de procesos: El proceso mediante el cual se crea un producto de software se representa en un modelo utilizando las herramientas de modelado de procesos. Estas herramientas ayudan a los gerentes a elegir un modelo de procesos o modificarlo según los requisitos del producto de software. Ejemplo: Eclipse Process Framewok
Herramientas de gestión de proyectos: Los diferentes pasos involucrados en la gestión de proyectos de software deben ser cumplidos por los gerentes para ejecutar el proyecto de software. Estas herramientas de gestión de proyectos realizan las actividades relacionadas con el proyecto, como la planificación, la estimación de costos, la programación del proyecto y la planificación de recursos. Ejemplo: Microsoft Project
Herramientas de documentación: La documentación abarca todas las fases del ciclo de vida de desarrollo de software. Estas herramientas ayudan en la generación de documentación para usuarios finales y usuarios técnicos según estándares. Ejemplo, Doxygen, DrExplain
Herramientas de análisis: ayudan a la captura de requisitos, verifican automáticamente cualquier inconsistencia, inexactitud en los diagramas, redundancias de datos y errores si los hay. Ejemplo: CaseComplete, VisibleAnalyst.
Herramientas de diseño: ayudan a los diseñadores de software a diseñar la estructura de componentes (módulos) del software, que pueden desglosarse en otros más pequeños utilizando técnicas de refinamiento. Estas herramientas proporcionan detalles de cada módulo e interconexiones entre los mismos. Ejemplos: Autodesk KeyShot
Herramientas de gestión de la configuración: una instancia de software se lanza bajo una versión. Las herramientas de gestión de configuración tratan con:

- Gestión de versiones y revisiones. Ejemplos: Git (sistema de control de versiones distribuido), AccuRev (sistema de control de versiones centralizado)
- Gestión de configuración de línea de base: una "línea de base" es una descripción acordada de los atributos de un producto, en un momento dado, que sirve como base para definir el cambio.
- Gestión de control de cambios. Se ocupan de los cambios realizados en el software después de que se fija su línea de base o se lanza el software por primera vez. Automatizan el seguimiento de cambios, la administración de archivos, la administración de códigos y más. Ejemplo: iTop

Herramientas de programación: consisten en entornos de programación como IDE (Integrated Development Environment y brindan una ayuda integral en la creación de productos de software e incluyen características para simulación y pruebas. Ejemplos: Cscope para buscar código en C o Eclipse.
Herramientas de prototipos El prototipo de software es una versión simulada del producto de software previsto. Un prototipo proporciona una apariencia inicial del producto y simula algunos aspectos del producto real. Las herramientas CASE de creación de prototipos esencialmente vienen con bibliotecas gráficas para crear interfaces de usuario. Estas herramientas nos ayudan a construir prototipos rápidos basados en la información existente. Ejemplos Por ejemplo, Mockup
Builder.
Herramientas de desarrollo web ayudan a diseñar páginas web con todos los elementos que se puedan necesitar como formularios, texto, gráficos, etc. Las herramientas web también proporcionan una vista previa en vivo de lo que se está desarrollando y cómo se verá una vez finalizado. Ejemplos, BlueGriffon, Brackets
Herramientas de aseguramiento de la calidad El proceso de desarrollo del producto debe asegurarse de que mantiene la calidad de la organización, y el proceso y los métodos adoptados deben ser monitoreados para asegurar la calidad. Esto se hace mediante las herramientas de garantía de calidad. Las herramientas de prueba de software, la herramienta de control de cambios y las herramientas de configuración juntas también constituyen herramientas de garantía de calidad. Ejemplo, Jmeter, Selenium.
Herramientas de mantenimiento el mantenimiento del software incluye modificaciones en el producto de software después de su entrega. Ejemplo, Bugzilla para el seguimiento de defectos.