Asignatura: Entornos de programación
Entornos basados en combinación de herramientas

Características. Integración de herramientas

1. Entornos "*toolkit*". Características

Los entornos concebidos como una colección de herramientas (*toolkit environments*) consisten en una combinación de diversas herramientas capaces de interoperar entre ellas de alguna manera. Presentan las siguientes características:

- Son un conjunto de elementos relativamente heterogéneos.
- Son algo más que la simple reunión de herramientas. Requieren elementos adicionales para integrarlas en un conjunto coherente.
- Son bastante flexibles. Se pueden ampliar o adaptar a las necesidades de los usuarios mediante nuevas herramientas.
- Ofrecen poco control de uso de cada herramienta individual. No facilitan una disciplina de desarrollo.
- El elemento frontal (*front-end*) para interacción con el usuario suele ser un editor configurable, con llamadas a herramientas externas.

2. Integración de herramientas

- Criterios de integración:
  - Integración de datos
    - Las herramientas pueden compartir e intercambiar información
  - Integración de control
    - Una herramienta puede invocar a otras
  - Integración de presentación
    - Todas las herramientas usan los mismos mecanismos y modelos de interacción
  - Integración de proceso
    - La operación del conjunto de las herramientas sigue las normas y políticas de desarrollo que se hayan establecido

2.1 Concepto de integración de datos

- Interoperabilidad entre herramientas
  - Para que dos herramientas cooperen entre sí debe poder pasar información de una a la otra
- Evitar redundancias (duplicaciones)
  - En general, conviene evitar tener información duplicada, ya que requiere más espacio de almacenamiento y pueden producirse inconsistencias
- Consistencia, evitando incoherencias
  - Si se decide tener información repetida por algún motivo (acceso rápido a datos, p.ej.) hay que garantizar que todas las copias son consistentes
- Base: paso de datos entre herramientas
  - El requisito básico es algún mecanismo de comunicación o paso de parámetros.

2.2 Técnicas de integración de datos

- Transferencia directa
  - Ideal, pero muy restrictivo
  - Las herramientas comparten las mismas variables en memoria o se invocan como subprogramas y reciben los datos como argumentos
- Transferencia mediante ficheros
  - Una herramienta graba datos en un fichero y otra los lee de ese fichero
- Transferencia mediante comunicación
  - Similar al anterior, usando un canal de comunicación, que de hecho equivale a un fichero secuencial
- Repositorio común
  - En lugar de simples ficheros, se usa un repositorio bien organizado, equivalente a una base de datos de objetos. Puede hacer al mismo tiempo funciones de control de versiones

- Conversión de formatos
  - Permite la interoperación de herramientas heterogéneas
  - Hay lenguajes de *script* apropiados: AWK o Perl (texto), XSLT (XML).
  - Se facilita usando marcado estándar, p.ej.. XML

2.3 Integración de control

- Concepto
  - Permite invocar servicios o herramientas desde otras
  - Requiere integración de datos
- Técnicas
  - Se pueden usar los mecanismos nativos del sistema operativo: procesos, mensajes, llamadas
  - Se puede usar un lenguaje de órdenes (*shell*) para automatizar operaciones complejas combinando llamadas a herramientas que realizan funciones más sencillas
  - A veces hay que desarrollar envoltorios (*wrappers*) o intermediarios (*middleware*) para adaptar las interfaces entre herramientas (el equivalente en términos de control a la conversión de formatos para los datos)

2.4 Integración de presentación

- Se aplica sólo a las herramientas interactivas
- El objetivo es disponer de una interfaz amigable y uniforme para el conjunto de las herramientas
  - No es suficiente que cada herramienta por separado tenga una interfaz razonable
- Recomendaciones:
  - Limitar las formas diferentes de interacción
  - Usar formas de interacción adecuadas al modelo mental del usuario
  - Garantizar tiempos de respuesta adecuados
  - Mantener información disponible, evitando que el usuario tenga que recordar nombres o valores para introducirlos luego, o simplemente tenga que repetir los mismos parámetros cada vez que invoca una acción
  - etc.

2.5 Integración de proceso

- Consiste en que el conjunto de las herramientas implemente o dé soporte a un modelo de desarrollo determinado
  - Debe ser posible imponer una disciplina de uso de cada herramienta
    - Autorización de acceso (sólo el responsable de realizar una tarea debe poder invocar la herramienta que la realiza)

    - Prerrequisitos (por ejemplo, no se puede guardar en el repositorio una versión del código que no compila o no pasa ciertas pruebas)
- Debe ser posible definir operaciones programadas para automatizar ciertas tareas de desarrollo (por ejemplo, hacer un *rebuild* completo del proyecto cada noche, y ejecutar pruebas de regresión)

3. Clases de Herramientas

Aplicaremos el término *herramienta* a un producto CASE que da soporte a una tarea concreta dentro de las actividades de desarrollo de software. Dicho soporte consistirá en una serie de *servicios*, cada uno de los cuales automatiza una operación individual. Podemos clasificar las herramientas según los servicios que ofrece y/o la tarea a la que da soporte. A continuación se describen algunas clases de herramientas o grupos de funciones que podemos encontrar en un entorno de programación:

- Edición y examen del código (*editor* / *browser* / *navigator*)
- Codificación
- Verificación y validación
- Gestión de configuración
- Métricas
- Otras herramientas

Otras herramientas de desarrollo no incluidas en la relación anterior se salen del marco de lo que hemos denominado *entorno de programación*, y dan soporte a otras fases del ciclo de vida de desarrollo. Por ejemplo:

- Gestión del proyecto
- Análisis y diseño (metodología concreta - "herramienta" CASE)
- ... etc. ...

3.1 Edición y examen del código

- Editores de texto
  - Editor de texto simple
  - Editor orientado al lenguaje
- Editores gráficos
  - Editor de diagramas
  - Editor de iconos para GUI
- Editores de estructura
  - Gráfico (diagramas)
  - Texto (código)
- Facilidades de *navegación*
  - Referencias cruzadas (definición / uso)
  - Plegado/desplegado (*folding*)
  - *Class wizzard*

3.2 Codificación

- Herramientas de codificación y depuración
  - Ensamblador
  - Compilador

    - Depurador
    - Compilación cruzada
    - Macroprocesador
    - Montador de enlaces (link)
    - Intérprete
- Generadores de código
    - Generadores de esqueleto de código
    - Generadores de interfaz de usuario
    - *Application wizzard*
    - Generadores de compiladores
- Reestructuradores de código
    - Reforma (*prettyprint*)
    - Refactorización

3.3 Verificación y validación

- Análisis estático
    - Análisis de consistencia
    - Detección de código no usado
    - Grafo de flujo de llamadas
    - Referencias cruzadas
    - Diagramas de estructura (dependencias entre módulos)
    - Comprobador de normas
- Análisis dinámico
    - Perfil de ejecución
    - Traza de ejecución
- Comparadores
    - Diferencias entre ficheros o directorios
    - Mezcla de ficheros (*merge*)
    - Visualizadores de diferencias

- Ejecución simbólica
- Emuladores / simuladores
  - Ejecución cruzada
  - Máquinas virtuales
- Comprobación de corrección
  - Ayuda a la demostración formal
- Prueba de programas
  - Generación de casos prueba
  - Ejecución automática de pruebas (pruebas de regresión)

3.4 Gestión de configuración

- Identificación de elementos
- Gestión de versiones
- Gestión de configuración
- Gestión de cambios
- Repositorio, archivo
- Configurador de aplicaciones

3.5 Métricas

La herramientas de obtención de métricas son en realidad un caso particular de las de verificación y validación, aunque tienen entidad propia.

- Métricas de código (análisis estático)
  - Complejidad, calidad
- Capacidad de proceso (análisis dinámico)
  - *Performance*, *benchmarks*
- Otras
  - Estimación/medida de costo, productividad

3.6 Otras herramientas

- Hoja de cálculo
  - Acumulación de datos, estadísticas, resúmenes
- Preparación de documentación
  - Procesadores de texto
  - Visualización de datos
  - Generación de diagramas
  - Extracción de documentación de código
- Sistemas de hipertexto
  - Documentación tipo hipertexto
  - Generación de ayuda en línea

- Ayuda sensible al contexto

4. Editores configurables

El elemento frontal (*front-end*) para interacción con el usuario suele ser un editor configurable, con llamadas a herramientas externas. A veces estos editores configurables se designan también con las siglas IDE (que debería reservarse para el entorno completo). Estos editores ofrecen las siguientes facilidades:

- Pueden servir como medio general de interacción con el usuario
  - Facilita la integración de presentación
- Ofrecen como funciones principales:
  - Edición de código, datos, etc.
  - Modos dependientes del tipo de fichero
- Facilitan la integración de control
  - Invocación de órdenes externas
  - Captura y análisis de resultados

4.1 Ejemplos de editores configurables

- Emacs - quizá el más conocido en el mundo UNIX
  - Personalizable con e-lisp
  - Sirve como herramienta universal
- Vim, Gvim - también muy usados en UNIX
  - Personalizable con macros y descripción de sintaxis (analizador léxico)
- Med - sólo en Windows
  - Personalizable con expresiones regulares y tablas de propiedades
- SciTE, jEdit - editores de uso general
  - Fáciles de usar
- Eclipse - algo más que un editor
  - Plataforma (*framework*) para construcción de entornos
  - Personalizable mediante *plug-ins*
  - Creado inicialmente como entorno Java

5. Discusión

La construcción de entornos mediante la combinación de diversas herramientas ofrece ciertas ventajas, pero también presenta determinados inconvenientes.

- Ventajas
  - Son fáciles de ampliar o adaptar mediante nuevas herramientas
  - Pueden ser construidos en parte por el propio usuario (programador): éste es más o menos el estilo UNIX original
  - Se pueden ir creando de manera incremental, añadiendo nuevas herramientas cuando sea necesario
- Inconvenientes
  - Presentan integración débil
  - Ofrecen poco control de uso de cada herramienta
  - Requieren un esfuerzo significativo para su puesta a punto y mantenimiento