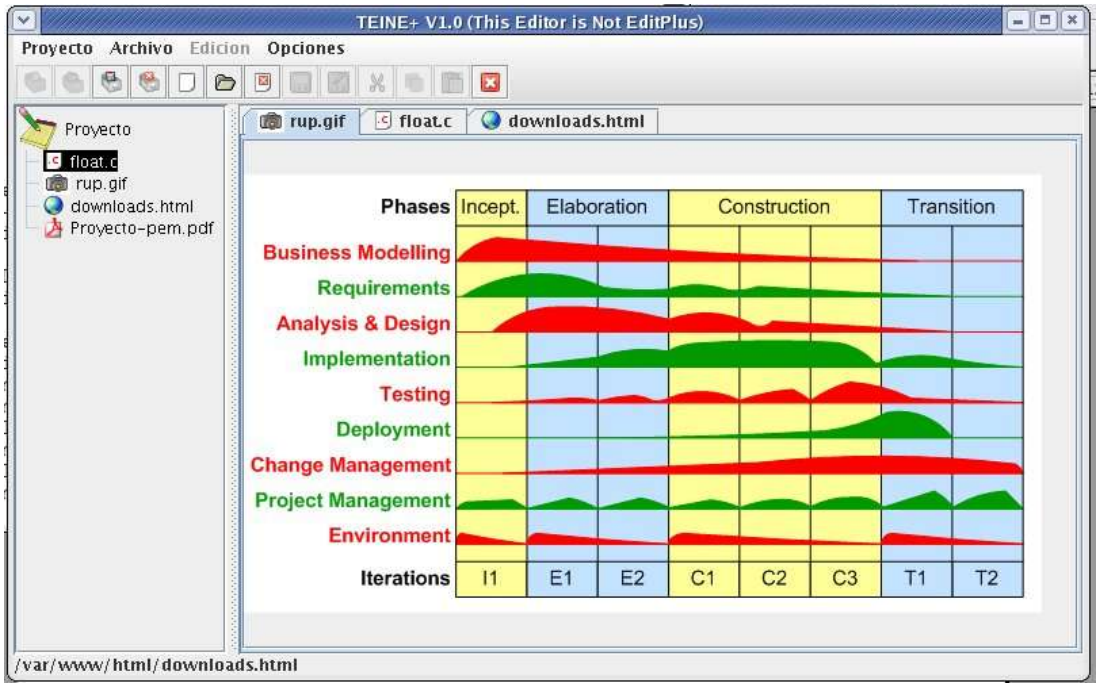
Curso de ingeniería de software: Arquitectura ejecutable

Humberto Cervantes Maceda

Octubre 2005

Introducción

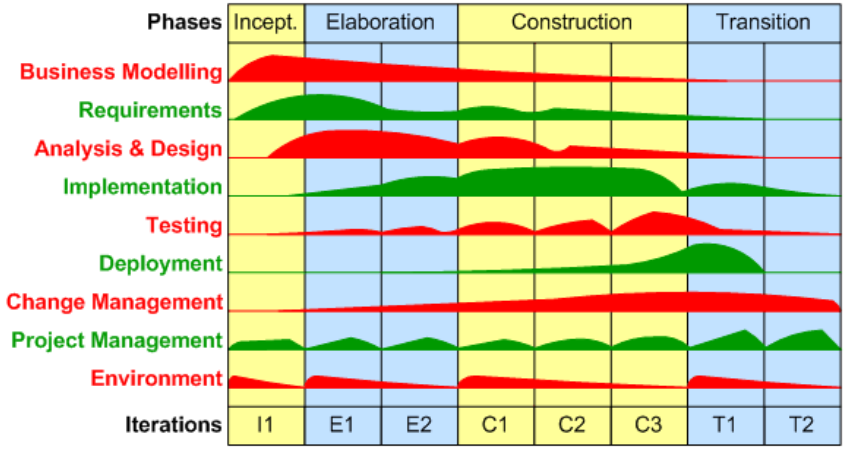
- **Durante el curso de I.S. se realiza un proyecto de desarrollo**
  - El proyecto propuesto es un editor genérico extensible



  - El curso del día de hoy presenta las bases para este desarrollo

Las etapas del proceso unificado

- **Concepción**
  - Comprensión general del problema
    - ~20% de los casos de uso
  - Análisis de riesgos asociados a realización
  - Propuesta inicial para implementación



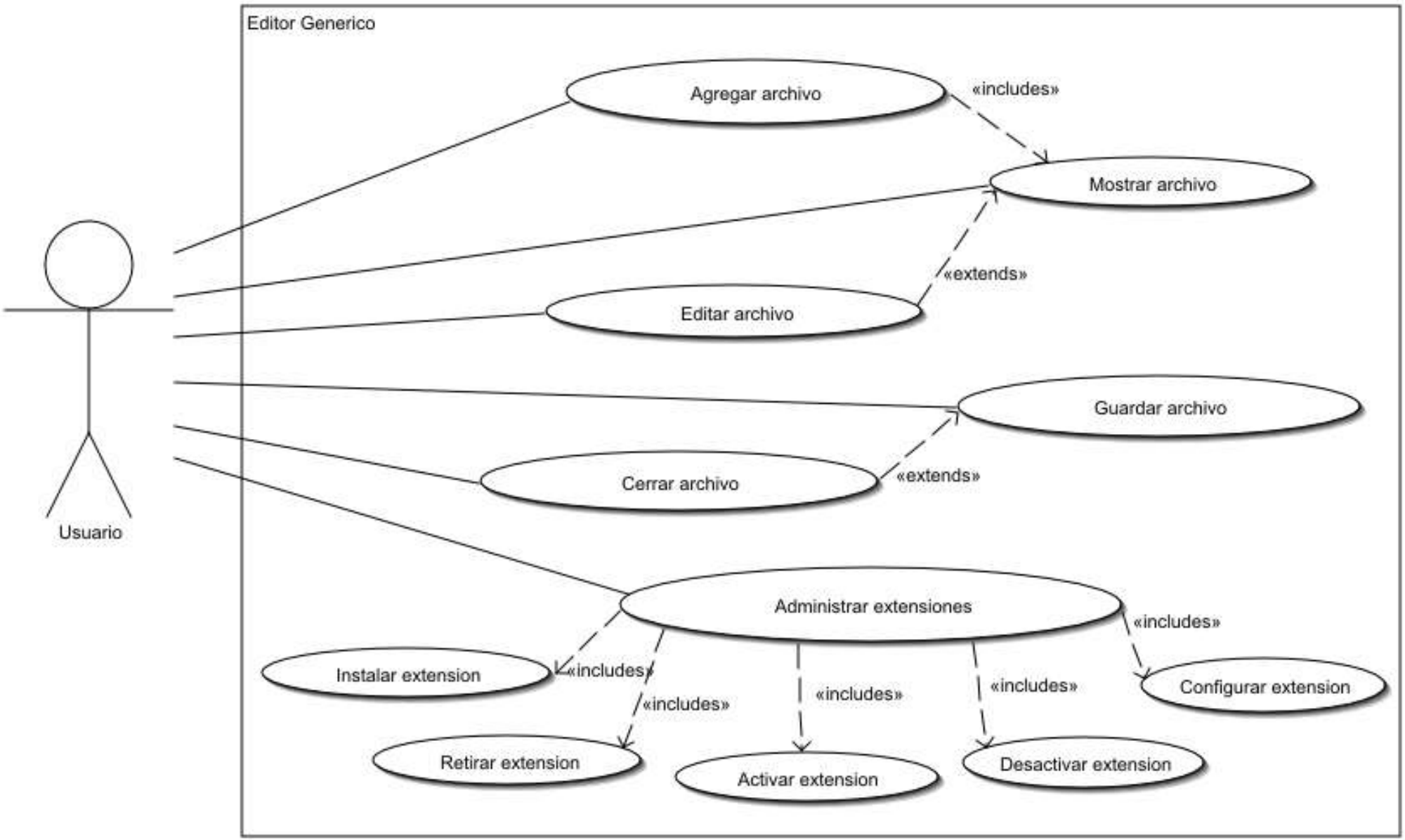
- **Elaboración**
  - Comprensión más detallada del problema
    - ~80% de los casos de uso
  - Realización de una **arquitectura ejecutable**

Arquitectura ejecutable

- **La arquitectura ejecutable provee los "cimientos" de la aplicación y permite estimar los riesgos asociados a la tecnología**

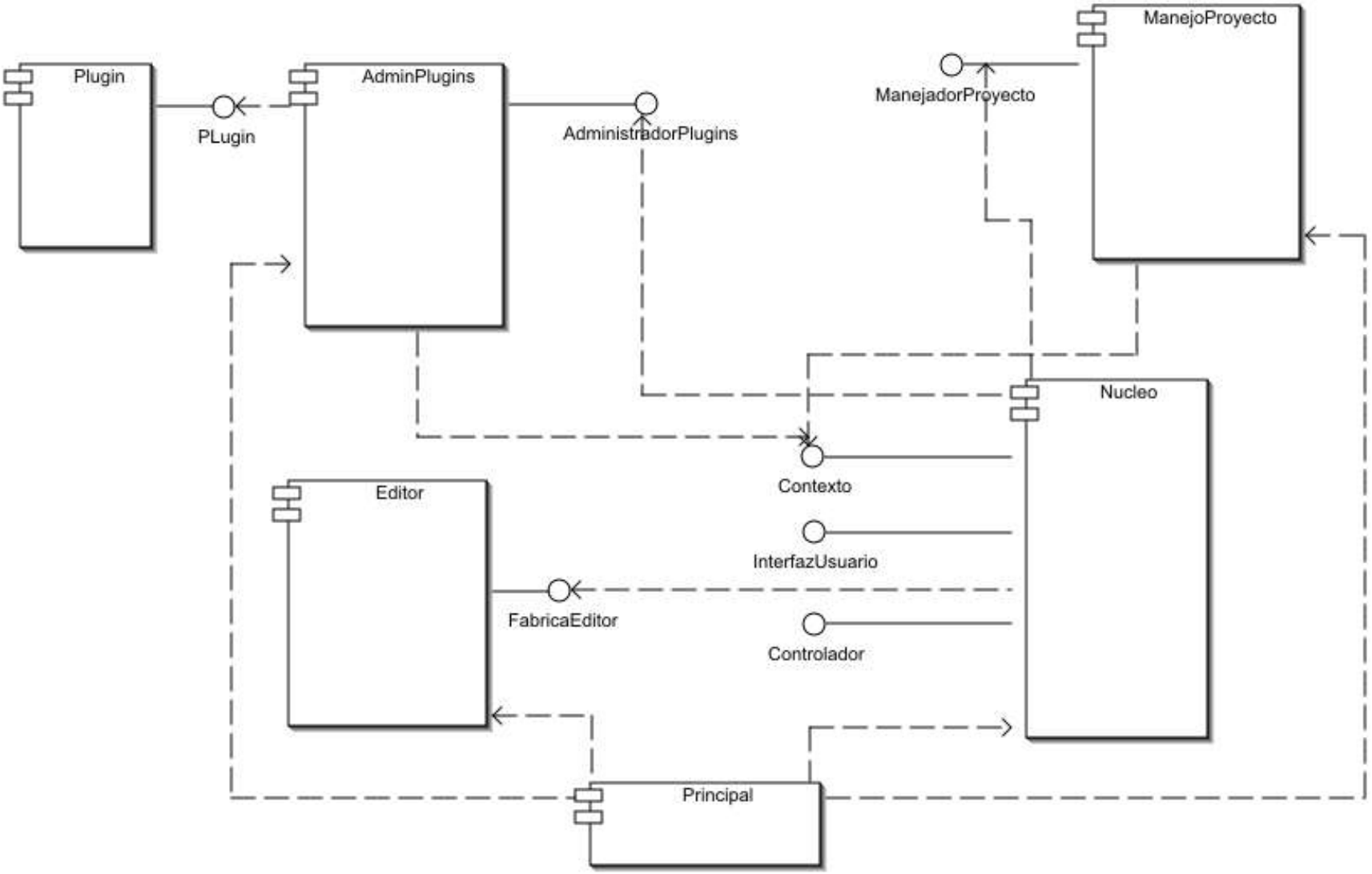
- **La arquitectura propuesta para el proyecto**
  - Permite reutilizar una plataforma de base
  - Permite realizar desarrollo basado en componentes
  - Permite soportar desarrollo en paralelo
  - Permite realizar pruebas unitarias de forma automática
  - Permite realizar integración automática
  - Permite preparar la implantación

Vista de casos de uso



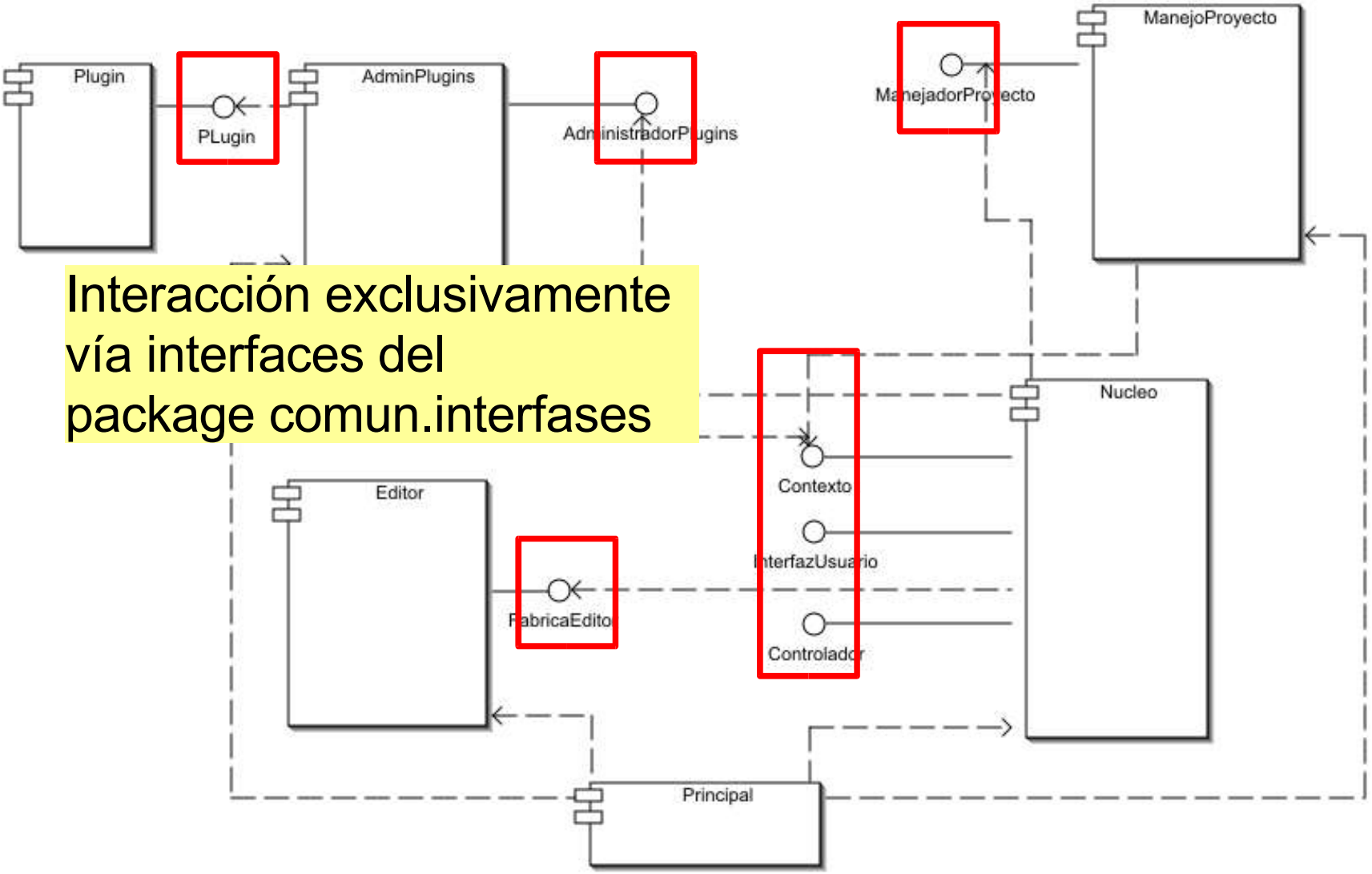
Vista de implementación

- Componentes principales de arquitectura



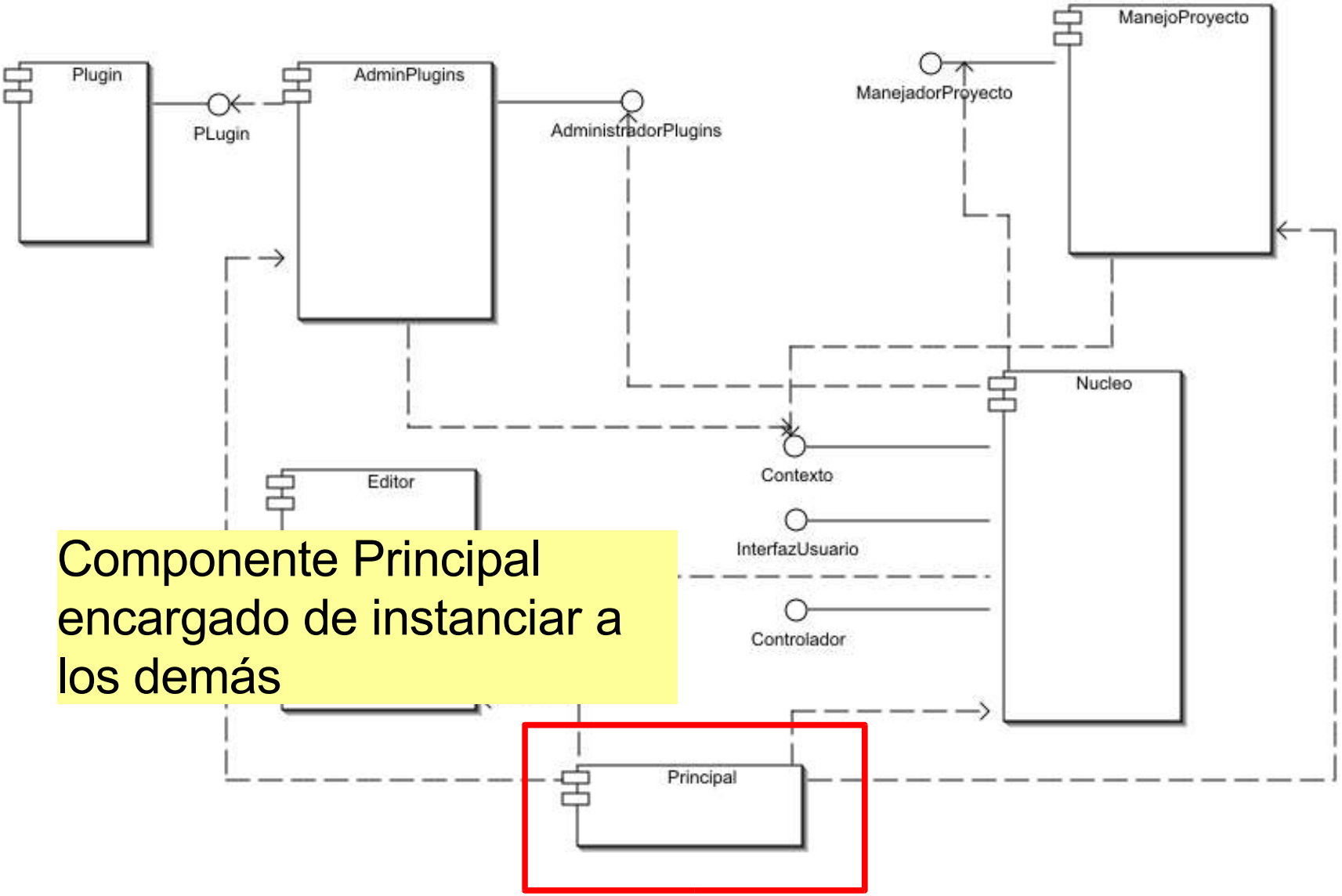
Vista de implementación

- **Componentes principales de arquitectura**



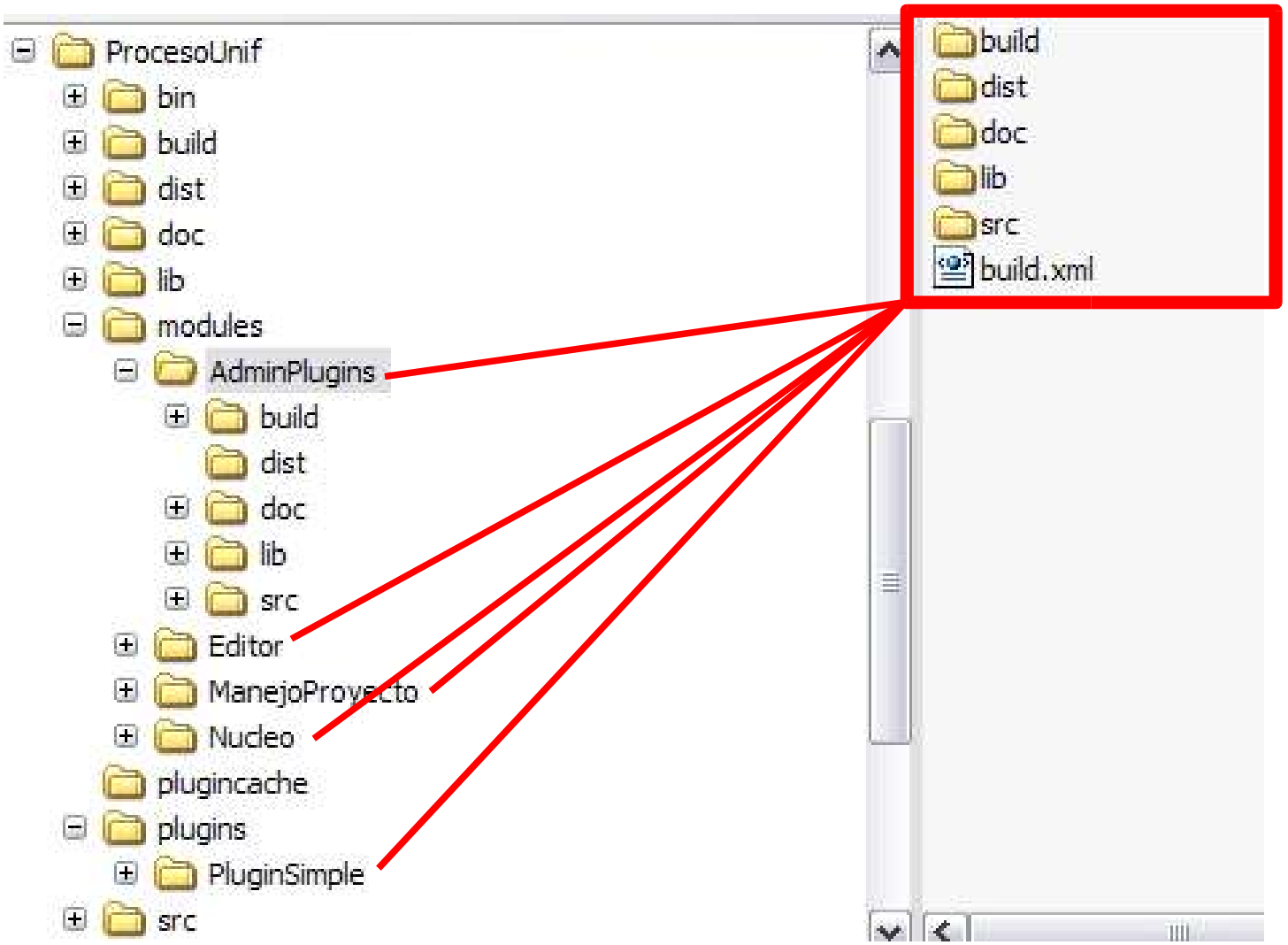
Vista de implementación

- **Componentes principales de arquitectura**



Estructura del proyecto

- **Cada módulo es un proyecto independiente**
  - Permite soportar desarrollo en paralelo



Pruebas unitarias

- **Basadas en Junit**
  - Se aplican a las interfases expuestas por los componentes
- **Cada componente tiene dos espacios de nombres**
  - Espacio de nombres asociado a los fuentes
  - Espacio de nombres asociado a pruebas
    - Comienza con test.xxx
- **Módulos se pueden probar de forma independiente gracias a *stubs***
  - Exportados por componente principal

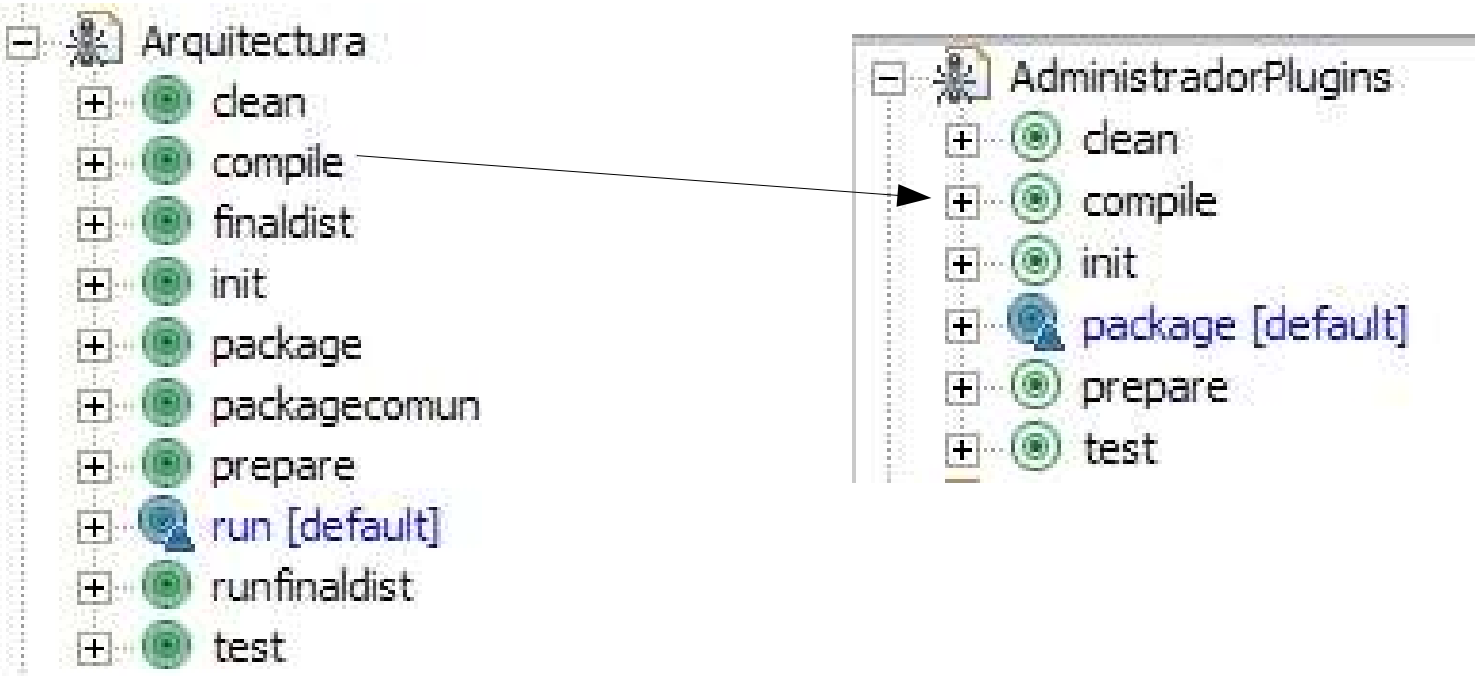
Pruebas unitarias (2)

- **Ejemplo de prueba unitaria de modulo**

```
[junit] Running test.mx.uam.ingsoft.pluginadmin.AdministradorPluginsImplTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Time elapsed: 1.281 sec
    [junit] Testsuite:
test.mx.uam.ingsoft.pluginadmin.AdministradorPluginsImplTest
    [junit] Tests run: 7, Failures: 0, Errors: 0, Time elapsed: 1.281 sec
    [junit] Testcase: testAdministradorPluginsImpl took 0.031 sec
    [junit] Testcase: testGetListaPlugins took 0.016 sec
    [junit] Testcase: testGetPlugin took 0.172 sec
    [junit] Testcase: testInstalaPlugin took 0.093 sec
    [junit] Testcase: testRetiraPlugin took 0.094 sec
    [junit] Testcase: testActivaPlugin took 0.781 sec
    [junit] Testcase: testDesactivaPlugin took 0.094 sec
```

Automatización construcción

- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal y script de módulo**



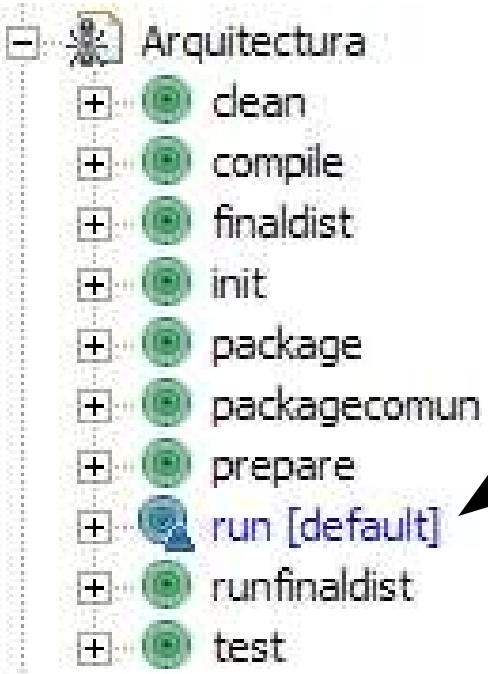
Automatización construcción

- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal**

Realiza la compilación de las interfaces así como la inyección de la librería resultante dentro de todos los módulos, previo a compilación general

Automatización construcción

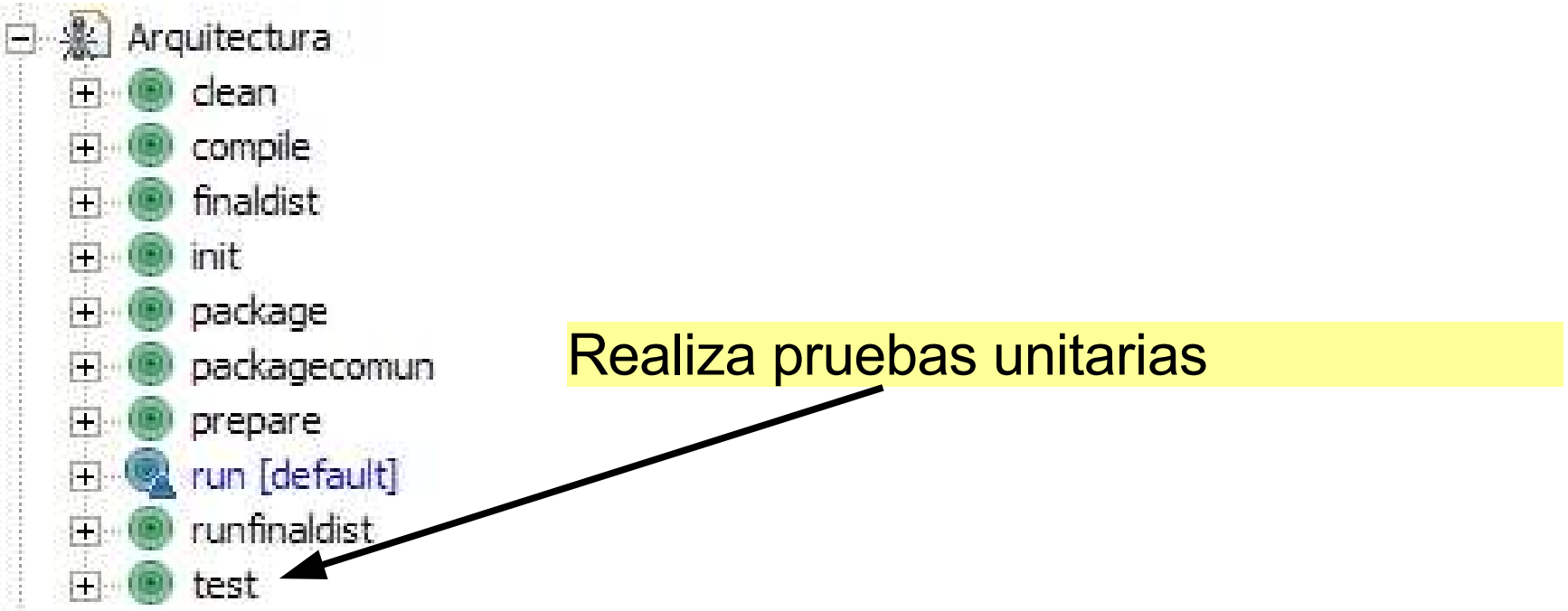
- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal**



Ejecuta sin realización de pruebas unitarias

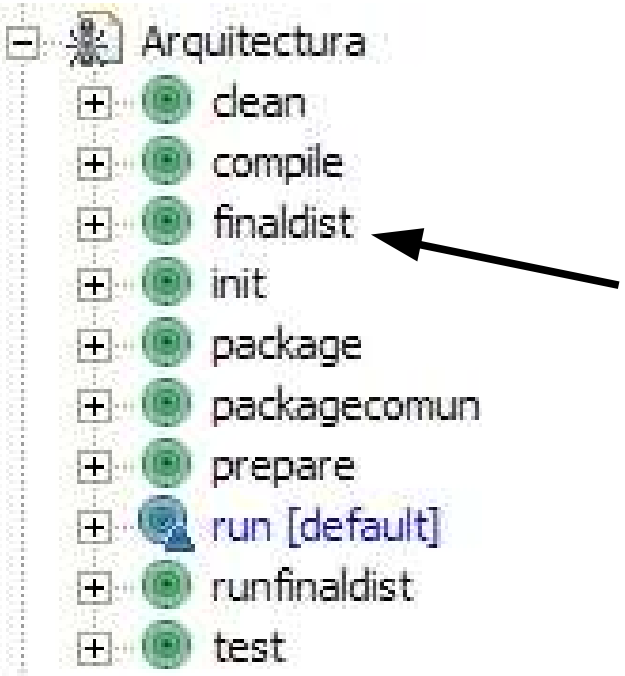
Automatización construcción

- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal**



Automatización construcción

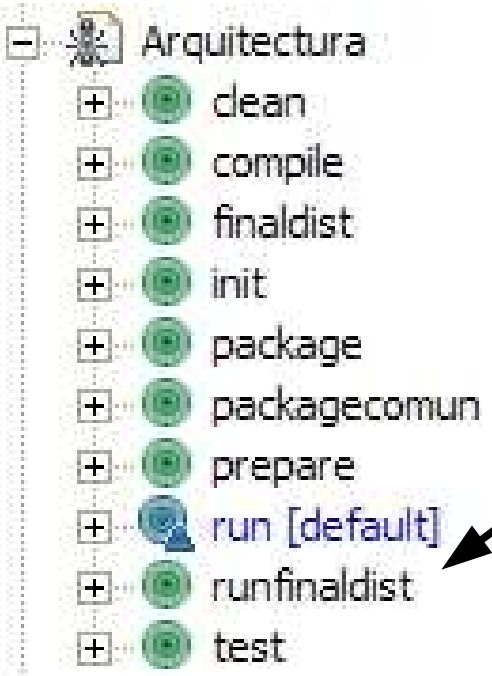
- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal**



Integración automatizada. Primero se realizan las pruebas unitarias y posteriormente se realiza el empaquetamiento del Principal y de cada módulo. Los paquetes resultantes son copiados en un directorio particular.

Automatización construcción

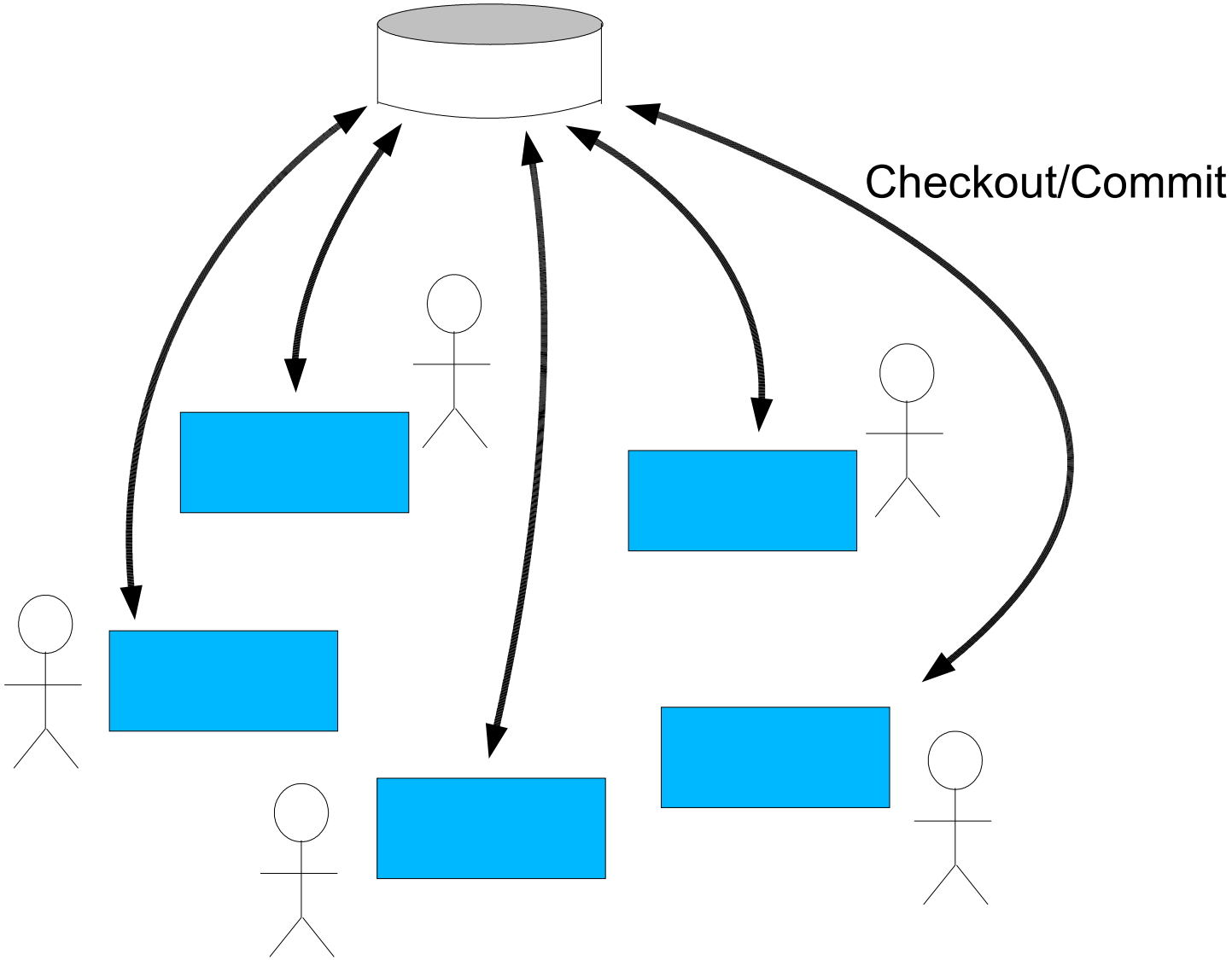
- **1 script Ant principal y 1 script por cada módulo**
  - Las tareas del script principal invocan las tareas en los directorios modules y plugins
    - Se pueden agregar componentes sin modificar script de construcción (subant)
- **Script principal**



Ejecución a partir de la distribución final en preparación para la transición

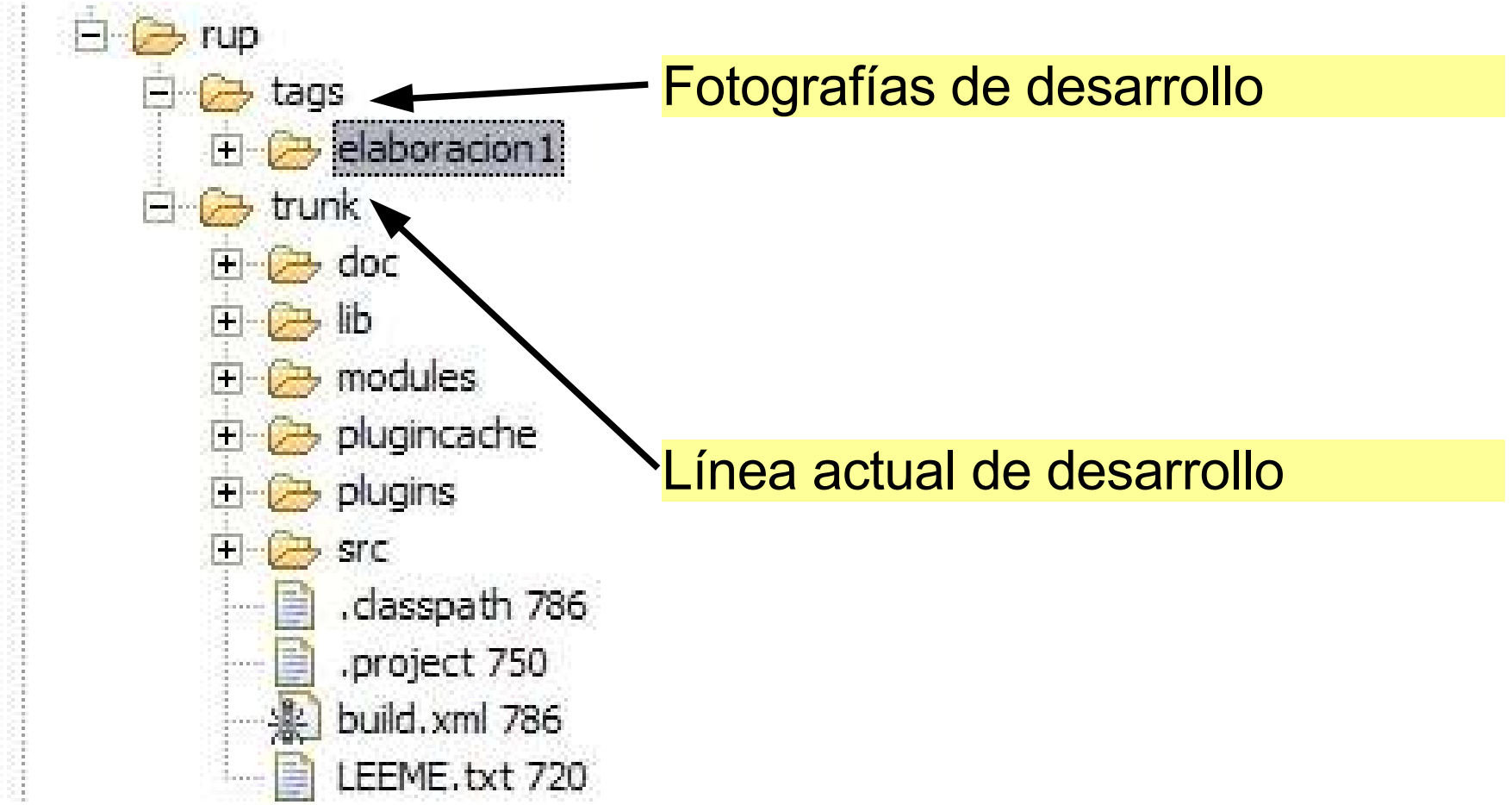
Organización del desarrollo

- **El código se encuentra en una base centralizada**
  - Guarda todas las versiones



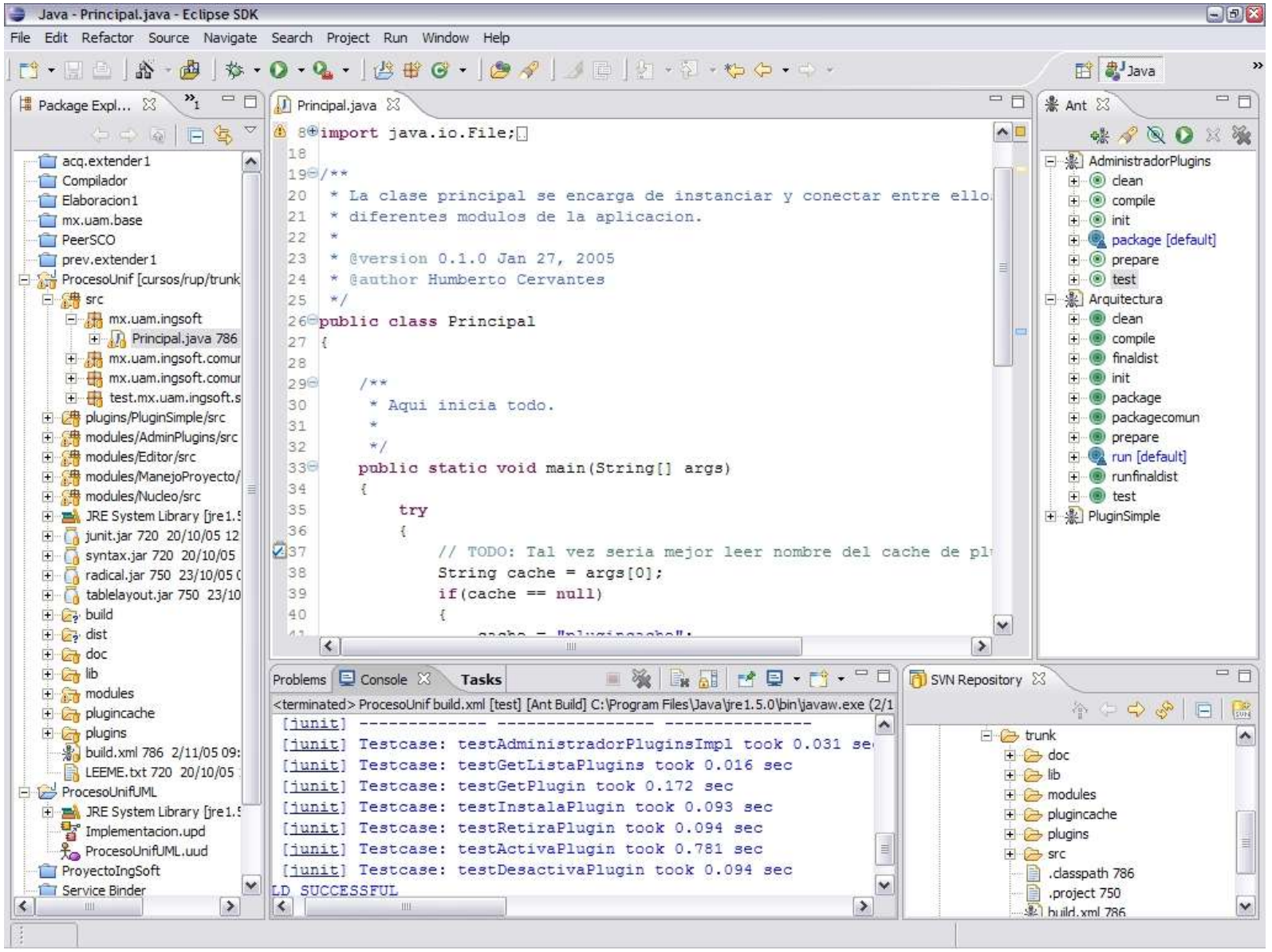
Organización del repositorio

- **El proyecto entero se guarda en un repositorio de código (subversion)**
  - Organizado de la manera siguiente



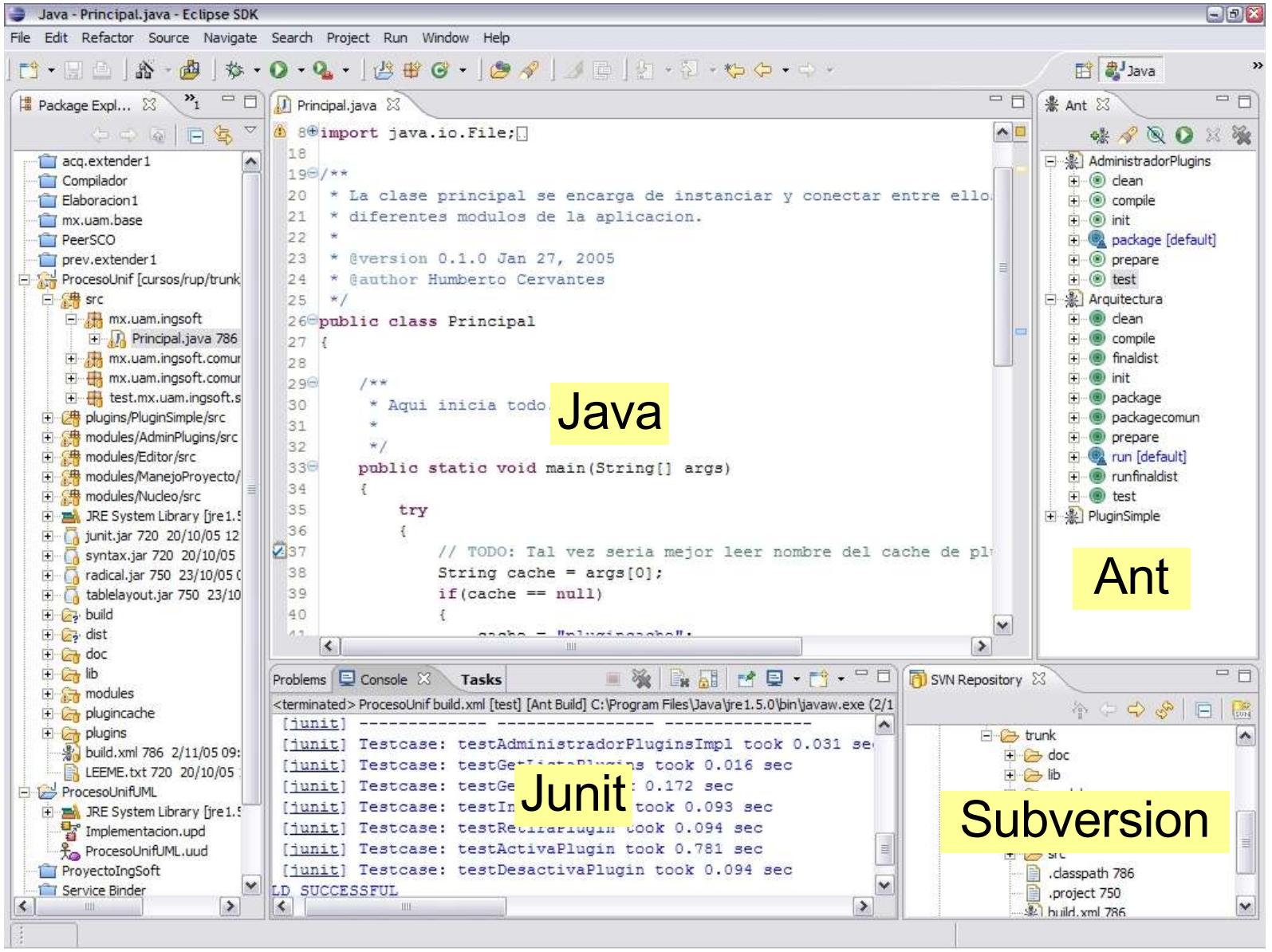
Entorno de desarrollo

- Todas las herramientas se integran en Eclipse



Entorno de desarrollo

- Todas las herramientas se integran en Eclipse



Desarrollo de un componente

- **Se necesita crear un proyecto nuevo bajo modules o plugins**
  - Módulo: no se introduce dinámicamente
  - Extensión: se introduce dinámicamente

- **Proyecto debe tener estructura estándar y archivo build.xml**
  - src – dos packages: fuentes y pruebas
  - lib – librerías: compile y runtime
  - res – recursos adicionales

Desarrollo de un módulo

- **Sólo se deben importar clases de espacio de nombres común**
  - Se puede obtener una referencia al Controlador que da acceso a los demás componentes
  - Se pueden agregar elementos a la interfaz de usuario vía InterfazUsuario

- **La clase Principal debe crear la instancia del módulo**

Desarrollo de extensión

- **Una extensión tiene acceso al Contexto que le permite introducir elementos a la interfaz de usuario**
  - Debe proveer una clase que implemente Plugin
  - Puede proveer una clase de configuración

- **Se debe proveer un manifest**

```
Nombre-Plugin: PluginSimple
Clase-Plugin:
mx.uam.plugin.PluginSimpleImpl
Autor-Plugin: Humberto Cervantes
Version-Plugin: 1.1.0
Inicia-Activo: false
```

Para terminar...

- Realizar un checkout de la arquitectura
- Explorar los distintos puntos cubiertos aquí
- Desarrollar una extensión que introduzca un nuevo menú y que sea configurable