*Ar*lips *D*evelopment *E*nvironment versión 1.0.0

Grupo de Tecnología Informática - Inteligencia Artificial
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia (España)

7 de febrero de 2006

Índice

1. Introducción

El entorno de desarrollo *Arlips* (*ArDE* en adelante) es una herramienta cuyo objetivo es facilitar el desarrollo e implementación de sistemas basados en reglas (SBR) *Arlips*. *ArDE* está integrado en la plataforma de desarrollo eclipse, siendo la primera versión de *ArDE* muy simple, pero capaz de ser ampliado de manera sencilla mediante la utilería ofertada por eclipse. La elección de eclipse se justifica por su filosofía, por ser una plataforma de desarrollo para todo y nada en particular. Eclipse puede ser ejecutado en distintos sistemas operativos y gracias a su estructura de plugins ofrece un entorno de desarrollo adecuado a distintos lenguajes de programación. Esta última característica resulta especialmente útil para *Arlips*, pues el mismo entorno de desarrollo, con los plugins pertinentes, puede ofrecer una adecuada visión al SBR *Arlips*, a la vez que el código generado se puede tratar como un proyecto *C* con la misma herramienta. Mediante la instalación del plugin *ArDE* se facilita la edición y gestión de proyectos *Arlips*. Si tiene instalado un plugin para la gestión de proyectos *C*, podrá revisar y compilar el código *C* generado por *Arlips*. Si así lo desea, podrá ejecutar su SBR o podrá trabajar con un repositorio CVS o SVN sin salir de eclipse.

2. Características de *ArDE*

ArDE ofrece las utilidades básicas que usted esperaría encontrar en cualquier editor de textos gracias al entorno eclipse. La gestión del fichero, las funciones básicas de edición, copiado y pegado, las utilidades de deshacer y rehacer la edición o la búsqueda son algunas de éstas.

El editor también ofrece resaltado de sintaxis de forma automática, distinguiendo entre palabras reservadas, tipos, constantes, comentarios y cadenas de literales.

Finalmente se ha implementado una pequeña utilidad de ayuda al alcance de los paréntesis. Al realizar un doble clic tras una apertura de paréntesis o antes de un cierre de paréntesis, se selecciona el texto afectado por el contexto del paréntesis, facilitando así la detección de errores. Este comportamiento también está disponible con llaves y comillas.

3. Instalación de *ArDE*

Para la instalación y ejecución de *ArDE* siga los siguientes pasos:

1. Si no dispone de eclipse, descargue e instalelo. Para ello, revise el repositorio de software de su sistema operativo o descárguelo directamente de `<http://www.eclipse.org/>`[1].

[1] *ArDE* ha sido desarrollado y probado en la versión 3.1.1 de eclipse. Se desconoce la

2. Extraiga el contenido del fichero comprimido que contiene *ArDE* en el directorio raíz de la instalación de eclipse[2]. Podrá observar que tanto el fichero comprimido como el directorio raíz de la instalación de eclipse contienen un directorio llamado *plugins*. Al realizar la extracción de la forma indicada, se añadirán los nuevos plugins de *ArDE* en eclipse, instalándose éstos en el directorio de plugins.

¡Felicidades! Ya puede ejecutar *ArDE*.

4. Ejecución de *ArDE*

El editor proporcionado con *ArDE* se ejecuta de manera semejante al resto de editores de eclipse. Basta con que se abra un fichero con la extensión de ficheros reconocida por *ArDE* para que éste se active. A continuación se muestra como iniciar un proyecto *ArDE* desde cero.

1. Inicie eclipse.

2. Seleccione el espacio de trabajo (*workspace*) que desee usar[3].

3. Una vez eclipse termine de arrancar, cree un proyecto simple nuevo.

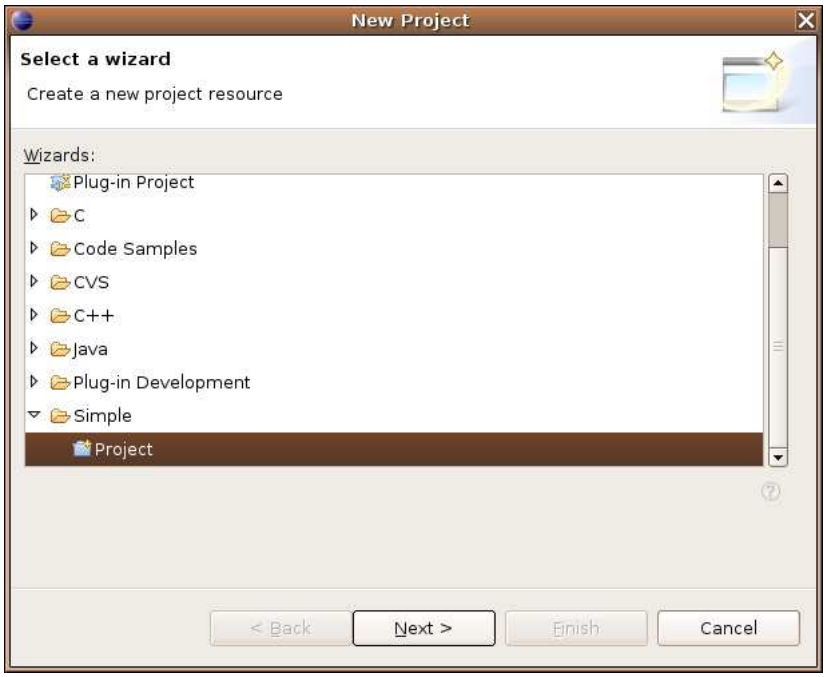
   a) `File -> New -> Project...`

   b) En la ventana que se mostrará despliegue la la opción *Simple*, seleccione la opción *Project* y continúe con el asistente pulsando el botón *Next*.

compatibilidad con versiones anteriores de eclipse.

[2]Puede que necesite permisos de administrador o superusuario.

[3]Recuerde que para cada espacio de trabajo se guarda la configuración del mismo, pero que dos espacios de trabajo distintos no comparten configuración



c) Indique el nombre de su nuevo proyecto, así como si desea cambiar la ubicación por defecto y finalice el asistente.

4. Tras crear su nuevo proyecto, genere en el mismo un fichero cuya extensión sea *.arl*

   a) `File -> New -> File`

   b) Seleccione el proyecto anteriormente creado y dé un nombre a su fichero. No olvide que la extensión del mismo debe ser *.arl*. Tras ello podrá finalizar el asistente.

Tras estos pasos podrá observar que su nuevo proyecto contiene un fichero cuyo icono es (A). Si es así, ha conseguido instalar el plugin adecuadamente.

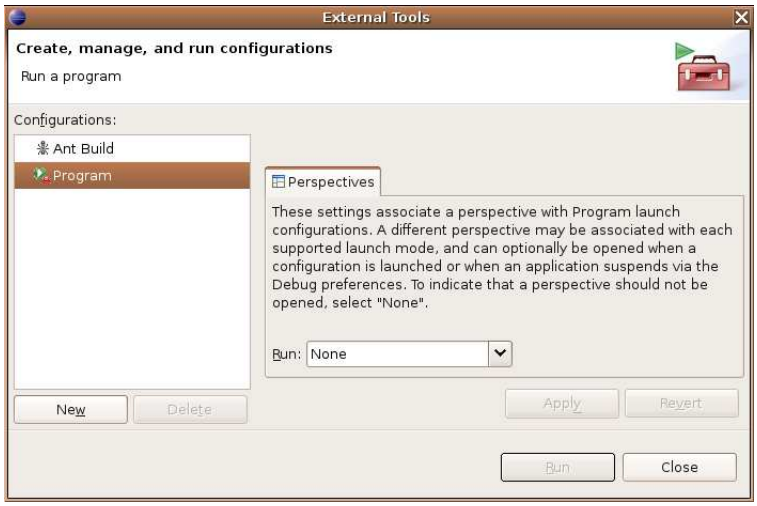
5. Compilación de un SBR *Arlips*

Eclipse ofrece una interfaz que facilita la llamada a programas externos. De esta forma es posible compilar el SBR *Arlips* sin salir de eclipse. Las distintas configuraciones para las llamadas a programas externos son almacenadas por eclipse en el propio espacio de trabajo, por tanto, distintos proyectos podrán compartir dicha configuración siempre y cuando todos ellos compartan el mismo espacio de trabajo. A continuación se describe la configuración recomendada de eclipse para llamar al compilador *Arlips*, de forma que sea genérica a cualquier proyecto.

1. Genere una nueva configuración para la ejecución de aplicaciones externas dedicada a *Arlips*.

   a) `Run -> External Tools -> External Tools...`

   b) Se mostrará una ventana nueva. Seleccione *Program* de la lista y a continuación pulse el botón *New*.

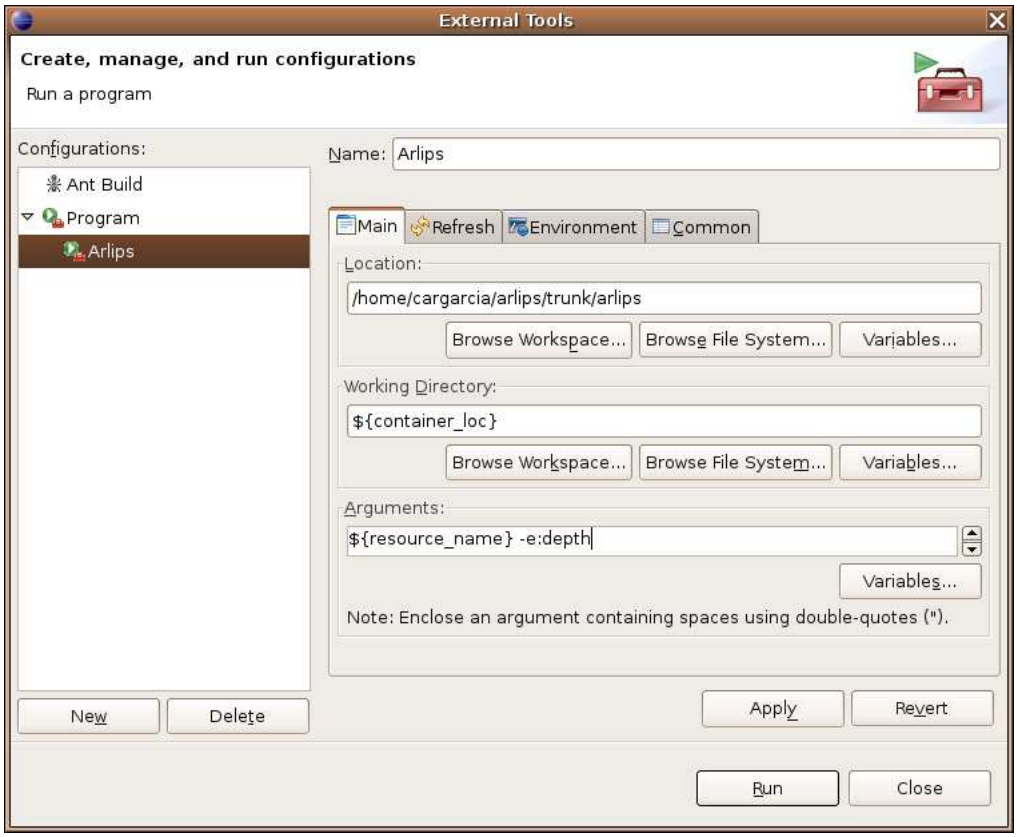


2. En el campo *Name:* introduzca un nombre descriptivo para la configuración recién creada, como por ejemplo *Arlips*. A continuación, configure las opciones de manera pertinente.

3. En la ficha *Main*

   - En *Location:* introduzca la ubicación del compilador *Arlips*.
   - Establezca el valor de *Working Directory* a la variable *container_loc*
   - En *Arguments:* establezca como primer argumento la variable *resource_name*. Recuerde que este campo es el que debe utilizar si desea especificar alguna opción de compilación. Por ejemplo, si desea que su SBR emplee una estrategia de selección en profundidad (*depth*), deberá introducir como argumento adicional *-e depth*[4].

[4]Nótese que esta opción es neutra, pues es la que el compilador usa por defecto

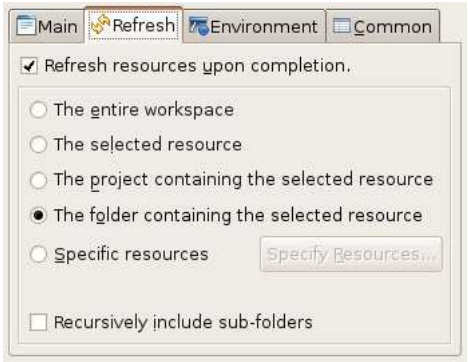


4. Se recomienda que configure la pestaña *Refresh* de la siguiente forma.

   a) Active la opción *Refresh resources upon completion.*.

   b) Escoja la opción *The folder containing the selected resource.*

   c) Desactive la opción *Recursively include sub-folders.*

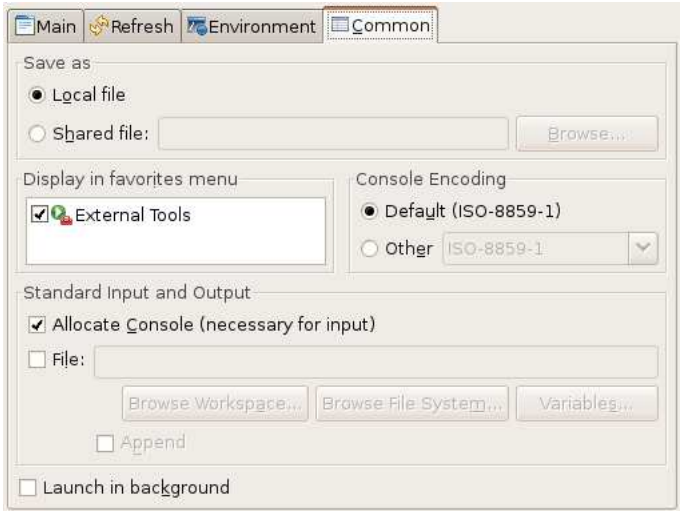


Mediante estos tres pasos se actualizará el contenido de su proyecto, incluyéndose automáticamente en el mismo los ficheros de código *C* generados.

5. No modifique la pestaña *Environment*. *Arlips* no necesita ninguna variable de entorno especial.

6. En la pestaña *Common*:

   - Asegúrese de activar la opción *Allocate Console (necessary for input)*. Eclipse asociará una de sus vistas de consola a la salida que emita el compilador *Arlips*[5]
   - Decida si quiere que su nueva configuración sea añadida a un menú de favoritos o si desea lanzar el compilador en background.



Pulsando el botón *Apply* guardará la configuración realizada. Si pulsa el botón *Run*, la ejecutará. Si configuró el editor de la manera indicada, recuerde abrir y seleccionar el fichero que contenga el código *Arlips* a compilar, pues con la configuración indicada el compilador compilará el fichero que se encuentre abierto en primer plano. Si guardó su configuración como una de las favoritas la podrá ejecutar siempre que desee y que no cambie de espacio de trabajo. Para ello, tendrá que localizarla dentro del menú *Run*, en el submenú *External Tools*.

[5]En ocasiones la consola que vincula *Arlips* con eclipse termina antes de que pueda mostrar algún mensaje. Si usted no observa mensaje de error alguno y tampoco se ha (re)generado ningún fichero de código, es posible que esté sufriendo este problema. Cuando ocurra este caso, podría revisar los mensajes de error que el compilador genera activando la opción *File:* de la pestaña *Common*, en el grupo *Standard input and output*. Seleccione mediante variables y/o cadenas de literales la ruta y nombre de un fichero en el que se escribirán los mensajes de error que genere el compilador.

6. Combinación de *ArDE* con otros plugins

Como ya se anunció en la introducción, eclipse es un entorno de desarrollo para todo y nada en particular. Es por ello que existe una gran diversidad de plugins que pueden facilitar la vida al programador. A continuación presentamos algunos de los que consideramos más útiles para su integración con *ArDE*.

6.1. CDT

CDT es el acrónimo de *C/C++ Development Tools*. CDT es un conjunto bastante completo de herramientas para la edición, gestión, compilación y ejecución de proyectos *C* y *C++*. La integración de CDT con eclipse permite realizar las siguientes fases de un proyecto *Arlips* sin necesidad de cambiar de herramienta. Para obtener más información acerca de CDT, y de cómo instalarlo y usarlo, visite la URL `<http://www.eclipse.org/cdt>`.

6.2. Control de versiones y trabajo en grupo

6.2.1. CVS

La instalación de eclipse incorpora por defecto un plugin para el control de versiones con CVS. Si en su proyecto o institución usan CVS, considere la posibilidad de gestionar su proyecto desde eclipse.

6.2.2. Subclipse

Si por el contrario en el desarrollo del proyecto en el que usted está colaborando usan Subversion, subclipse es el plugin que le permitirá realizar el control de versión de su repositorio svn desde eclipse. Para obtener más información de Subclipse visite `http://subclipse.tigris.org/`.