White Paper

AllFusion® Plex Multi-Platform Development

John D. Rhodes
Principal Architect, ADC Austin

William A. Hunt
Product Manager, CA

October 2006

AllFusion® Plex

Seamless multi-platform application development and deployment has always been one of the "Holy Grail" goals of computer technology. For the software vendor, the goal may be to develop an application that can be sold on most commercially available computer platforms. For a large organization with many enterprise platforms in use, the goal may be to deploy n-tier applications across heterogeneous hardware and operating systems.

Wouldn't it be great if you could model your application in an enterprise development tool, and then have the tool generate application code that is optimized to your target platforms? If this were possible, you could deploy C# / SQL Server code for Windows-based servers, and J2EE Java/RPG/DB2 code to iSeries based servers. Actually, this is something that is not all that futuristic - CA AllFusion Plex is an Architected Rapid Application Development (ARAD) tool that can deliver on this promise today. It does not do magic - there is always some application code that must be customized for each deployment platform. However, it can get you 95% of the way there automatically, and even the last 5% can be handled within the AllFusion Plex managed modeling environment.

ADC Austin is a CA partner and software integrator focused on iSeries and .NET software development. Many of the systems developed by ADC have large Microsoft and iSeries requirements. ADC has repeatedly found that CA AllFusion Plex product offers the ideal solution for many multi-platform focused companies.

The remainder of this white paper is focused on multi-platform development support in AllFusion Plex, from model configuration control through source code generation. For a general background please consult the white paper "Combining Information Engineering and Object Orientation with AllFusion Plex."

One Model, Many Dimensions

AllFusion Plex enables you to model your application in three dimensions, via a set of complementary control mechanisms

- *Versions* are used to control changes in the functionality of an application over time.
- *National languages* are used to control the translation of an application into different languages (French, Spanish, English, and so on).
- *Variants* are used to control the implementation of an application in different hardware and software environments.

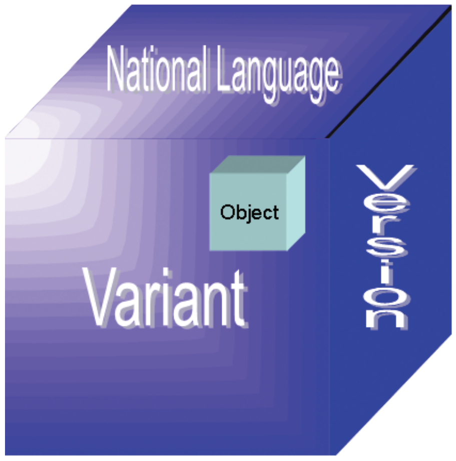
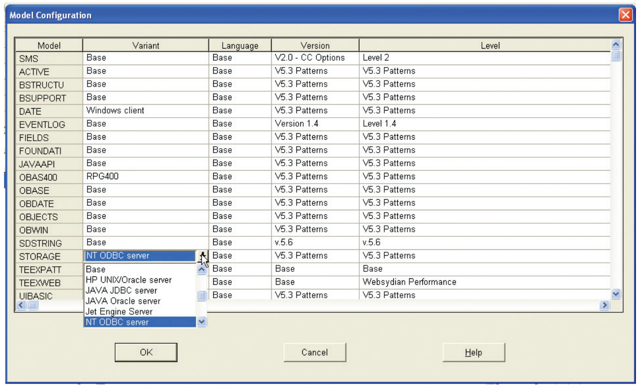


Figure 1. AllFusion Plex models are three-dimensional in concept, with Variant providing developers with the means to easily design platform-specific features.

The first two dimensions of the model, Versions and National Languages, are powerful features in their own right - and are beyond the scope of this white paper. The third dimension, Variant, gives the developer a powerful tool to model an application once and then deploy everywhere. Model objects such as panels, functions, and tables can be optimized and customized to fit the target platform. Components deployed on one server can talk to components deployed other server, even across hardware and OS boundaries.

Controlling the Configuration

The AllFusion Plex model simultaneously contains all configured dimensions, including platform. Variant is set by the configuration control utility. In this example, the application model consists of many linked library models. The configuration can be set independently for each library model. For example, the client application model may be set to generate a Microsoft client application, but the back end storage model can be set to i5 J2EE or RPG IV. This configuration would result in a client server application.



Model Configuration

Model	Variant	Language	Version	Level
SMS	Base	Base	V2.0 - CC Options	Level 2
ACTIVE	Base	Base	V5.3 Patterns	V5.3 Patterns
BSTRUCTU	Base	Base	V5.3 Patterns	V5.3 Patterns
BSUPPORT	Base	Base	V5.3 Patterns	V5.3 Patterns
DATE	Windows client	Base	V5.3 Patterns	V5.3 Patterns
EVENTLOG	Base	Base	Version 1.4	Level 1.4
FIELDS	Base	Base	V5.3 Patterns	V5.3 Patterns
FOUNDATI	Base	Base	V5.3 Patterns	V5.3 Patterns
JAVAAPI	Base	Base	V5.3 Patterns	V5.3 Patterns
OBAS400	RPG400	Base	V5.3 Patterns	V5.3 Patterns
OBASE	Base	Base	V5.3 Patterns	V5.3 Patterns
OBDATE	Base	Base	V5.3 Patterns	V5.3 Patterns
OBJECTS	Base	Base	V5.3 Patterns	V5.3 Patterns
OBWIN	Base	Base	V5.3 Patterns	V5.3 Patterns
SDSTRING	Base	Base	v.5.6	v.5.6
STORAGE	NT ODBC server	Base	V5.3 Patterns	V5.3 Patterns
TEEXPATT		Base	Base	Base
TEEXWEB		Base	Base	Websydian Performance
UIBASIC		Base	V5.3 Patterns	V5.3 Patterns

Figure 2. Model Configuration screen for creation of custom variants.

As you can see in Figure 2, there are many possible deployment configurations. Also, AllFusion Plex developers are free to create custom variants that can cover special circumstances.

Device Designs

Device designs are a major challenge for platform independent development. Most dramatically, the screen design for a 5250 application differs markedly from a Windows-based GUI. Microsoft and Java platform screen designs are generally much closer in appearance, but still contain differences. For example, in the case of a date selection screen feature, a Microsoft panel may contain an ActiveX/COM calendar component, while a Java panel may contain a JavaBean component. Although both components result in similar appearance and functionality, the methods and properties supported by each component can be different.

AllFusion Plex solves this problem with screen modes, and by making the panels dependent on Variant configuration. The panel object is initially designed in a base configuration where all common elements are created. Customized platform implementations can then be created in a seamless manner on top of the base configuration. ActiveX and JavaBean components can exist on the same panel but are only active in their particular Variants.

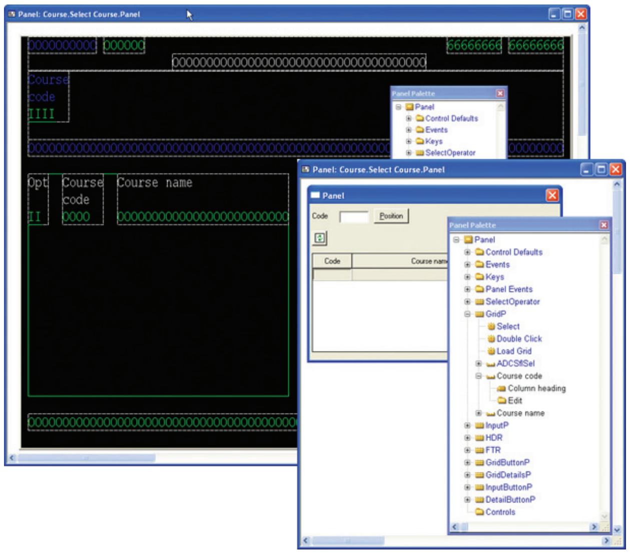


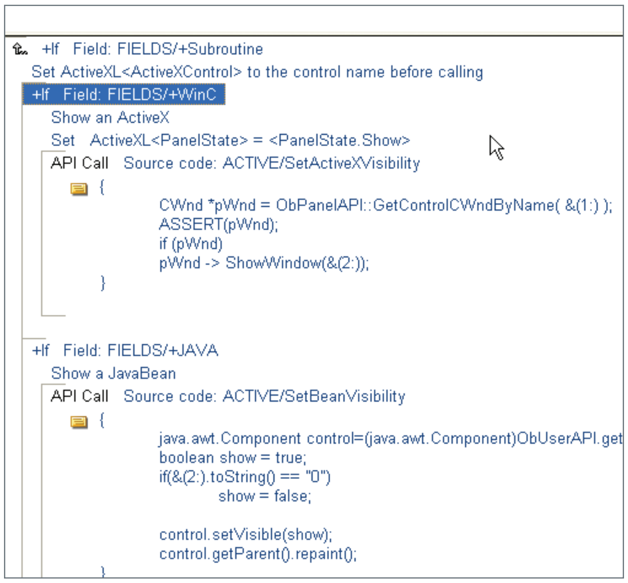
Figure 3. One application design deployed across multiple environments.

Figure 3 shows a single panel object that has separate 5250 and Windows/Java panel designs. Although both designs display the same fields, and have the same basic functionality, the appearance is quite different. However, the business logic is identical - freeing the developer from maintaining two code bases.

Business Logic and Source Code

AllFusion Plex business logic is developed in the action diagram modeler. The software developer models the high level business logic of the program using action diagram statements. The action diagram is an implementation language neutral modeling environment - for example, the statements required to open a table and process the records are identical no matter which implementation language you are targeting. The resulting function is later used to generate platform code such as C# or Java.

However, occasionally a developer may need to access platform specific features. For example, a Windows program may need to manipulate file system objects, or an iSeries program may need to access a data queue. One powerful aspect of the AllFusion Plex model environment is that it allows the developer to incorporate platform specific code. Using meta code to access the specifics of the variant being modeled, the developer can incorporate meta conditional statements, with accompanying platform specific source code objects. When the program is generated from AllFusion Plex, the correct source code for the target platform is selected and is compiled into the application object.



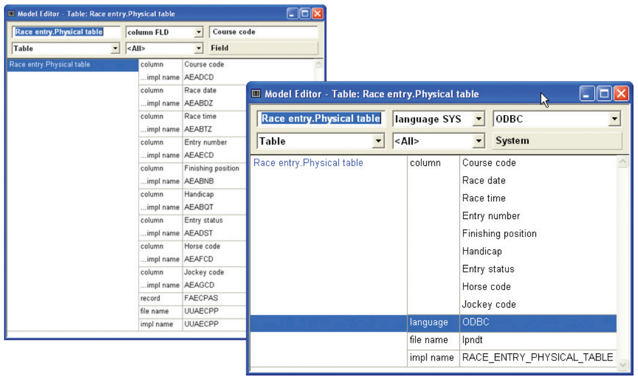
```
+If   Field: FIELDS/+Subroutine
Set ActiveXL<ActiveXControl> to the control name before calling
+If   Field: FIELDS/+WinC
   Show an ActiveX
   Set   ActiveXL<PanelState> = <PanelState.Show>
   API Call   Source code: ACTIVE/SetActiveXVisibility
         {
                    CWnd *pWnd = ObPanelAPI::GetControlCWndByName( &(1:) );
                    ASSERT(pWnd);
                    if (pWnd)
                    pWnd -> ShowWindow(&(2:));
         }

+If   Field: FIELDS/+JAVA
   Show a JavaBean
   API Call   Source code: ACTIVE/SetBeanVisibility
         {
                    java.awt.Component control=(java.awt.Component)ObUserAPI.get
                    boolean show = true;
                    if(&(2:).toString() == "0")
                            show = false;

                    control.setVisible(show);
                    control.getParent().repaint();
         }
```

Figure 4. Multi-platform specific code within an AllFusion Plex action diagram.

Figure 4 depicts an example of multi-platform specific code. In this case, the action diagram contains source code to control both a Windows COM component and a JavaBean component. The "+If language" statement is the key. If this function is deployed to a Java environment, only the code within the +If +Java section will be generated and compiled. In this manner, AllFusion Plex provides an overall framework for managing all application objects regardless of platform. You don't have to worry about managing bits of source code that is spread amongst various code libraries on various different operation systems - all the information necessary to build a multi-platform application is contained within the model.

Implementation Names

Functions and database object implementation names are also variant dependent. Consider a software vendor who wants to produce a software package that will run on both i5 "classic" DB2 and SQL Server. Table names under the classic style of DB2 tables can only be 10 characters long. It would be possible to store the database implementation name triples in the base variant so that they are used by both configurations. This is probably the simplest approach. However the table and column names would be limiting for someone used to the long names supported in SQL Server.



Model Editor - Table: Race entry.Physical table

Race entry.Physical table	column FLD	Course code
Table	<All>	Field

Race entry.Physical table	column	Course code
	...impl name	AEADCD
	column	Race date
	...impl name	AEABDZ
	column	Race time
	...impl name	AEABTZ
	column	Entry number
	...impl name	AEAECD
	column	Finishing position
	...impl name	AEABNB
	column	Handicap
	...impl name	AEABQT
	column	Entry status
	...impl name	AEADST
	column	Horse code
	...impl name	AEAFCD
	column	Jockey code
	...impl name	AEAGCD
	record	FAECPAS
	file name	UUAECPP
	impl name	UUAECPP

Model Editor - Table: Race entry.Physical table

Race entry.Physical table	language SYS	ODBC
Table	<All>	System

Race entry.Physical table	column	Course code
		Race date
		Race time
		Entry number
		Finishing position
		Handicap
		Entry status
		Horse code
		Jockey code
	language	ODBC
	file name	lpndt
	impl name	RACE_ENTRY_PHYSICAL_TABLE

Figure 5. AllFusion Plex implementation names for different variants as defined in the Model Editor.

In this case the software vendor may want to take advantage of the long name allocation feature within the SQL-related generators of AllFusion Plex. These long implementation names are incompatible with OS/400. Therefore, you have a requirement to override the base implementation names in the SQL variant.

The solution is simple - switch to a SQL variant and, in the Model Editor, add or change the required implementation name triples. These changes override the corresponding triples that exist in the base variant. This results in two sets of implementation names, each appropriate to the target platform.

N-Tier Development

AllFusion Plex enables applications to be partitioned dynamically. This means that a call from a client function (Windows, Java, or HTML) or from a server function can be configured to call functions on different servers. These calls can be made on any given server machine of any type supported by the AllFusion Plex runtime. The partitioning is highly flexible because you can configure the location of the call at design, deployment, or execution time. It is not necessary to generate, build, and deploy different versions of the client application.

Useful applications include:

- "Multi-tier" client/server and e-business applications. A Windows server middle tier can provide access to i5 RPG, Java (any hardware platform), or another Windows server. A Java server middle tier can provide access to the i5 and other Java systems including mainframe and UNIX/Linux.
- Performance and backup support. For example, if a server is not currently available your application can switch seamlessly to another server.

- Data transfer. Data can be transferred easily between platforms by reading the data, switching servers, and then writing the data to the new server.
- Large-scale distributed applications. An end user working in a branch office can update data on a local or remote server, as required.

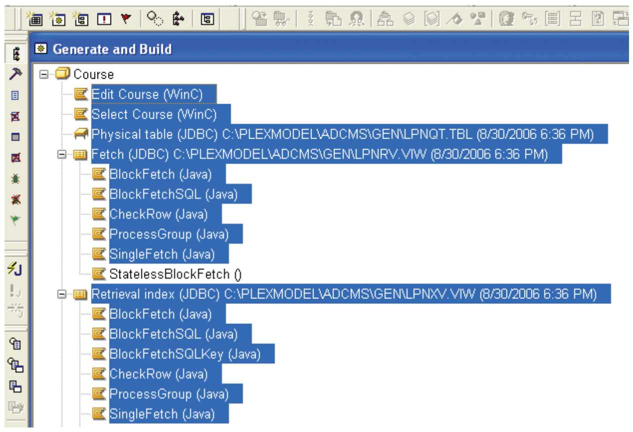
AllFusion Plex has n-tier capability to allow large, complex, multi-platform applications to be deployed with a single skill set. It is not necessary to have separate development teams partitioned by platform.

Generating and Building the Application

Once the application has been fully modeled, it is time to generate and build the various parts and components, and then deploy on the target platforms. The AllFusion Plex Generate and Build tool is a sophisticated facility for generating all source in the correct computer languages, and then directing the source for compilation to appropriate system.

The Generate and Build tool includes the following features:

- Setting up target systems, and defining options like database names and target directories for remote source.
- Configuring pre-built code libraries
- Filtering objects by language and platform
- Setting up deployment packages
- Monitoring status of compiled objects from compilation through testing
- Running, viewing, and debugging source code



```
Course
  Edit Course (WinC)
  Select Course (WinC)
  Physical table (JDBC) C:\PLEXMODEL\ADCMS\GEN\LPNQT.TBL (8/30/2006 6:36 PM)
  Fetch (JDBC) C:\PLEXMODEL\ADCMS\GEN\LPNRV.VIW (8/30/2006 6:36 PM)
    BlockFetch (Java)
    BlockFetchSQL (Java)
    CheckRow (Java)
    ProcessGroup (Java)
    SingleFetch (Java)
    StatelessBlockFetch ()
  Retrieval index (JDBC) C:\PLEXMODEL\ADCMS\GEN\LPNXV.VIW (8/30/2006 6:36 PM)
    BlockFetch (Java)
    BlockFetchSQL (Java)
    BlockFetchSQLKey (Java)
    CheckRow (Java)
    ProcessGroup (Java)
    SingleFetch (Java)
```

Figure 6. Generate and Build screen.

Figure 6 shows the Generate and Build tool, with some of the features mentioned above. It is possible to control most aspects of the build process from this facility.

Real Life Promise

The preceding sections describe the facilities in AllFusion Plex that make up multi-platform application development. Multi-platform development is often a difficult and painful process, and many people are initially skeptical that any development tool can deliver on the promise.

In the case of AllFusion Plex, there are many examples of software vendors and other organizations that have successfully deployed multi-platform applications. As an application integrator specializing in model based development, ADC Austin has worked with many of these groups. One recent example is Indiana University Foundation (IUF). With assistance from ADC Austin, IUF developed an Endowment Trust Accounting (ETA) package using AllFusion Plex. ETA is an enterprise-class application that is used to manage the over 1 billion dollar Indiana University endowment. The initial deployment of ETA was Java client tied to i5 Java server. However, IUF later decided to make their package available to other universities, who may not use Java or have an i5. Using AllFusion Plex, they were able to rapidly deploy their application in other configurations including Microsoft client to i5 server, and Microsoft client to Microsoft server.

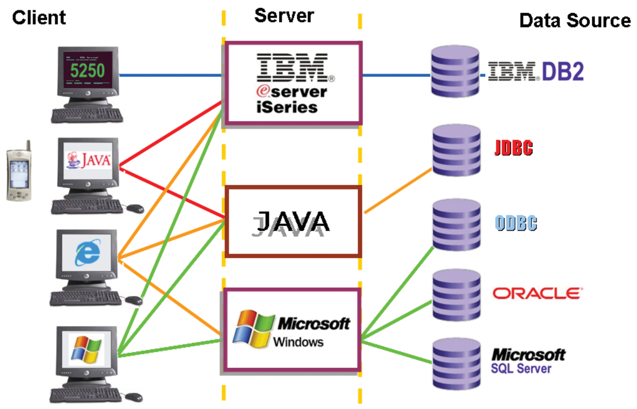


Figure 7. An overall snapshot of different environments that AllFusion Plex targets: client, server and data sources.

For those in the business of deploying applications to multiple platforms, AllFusion Plex is definitely worth strong consideration.

About the Author

John D. Rhodes is the principal architect at ADC Austin,a CA partner for providing midrange application development solutions. ADC has more than 10 years of extensive experience working in the AllFusion Plex development environment.

William A. Hunt is the Product Manager for AllFusion Plex at CA and has served in this role since 2000.

Quality System™
Quality
Endorsed
Company
ISO 9001
Lic. 2443