CA Plex Technical Overview

Architected RAD for .NET, Java and IBM i

www.ca.com

Table of Contents

2 CA Plex Overview

CA Plex is an Architected RAD (ARAD) tool that uses the techniques of model-based development, code generation and patterns to improve the delivery and maintenance of business software applications. Applications are developed in a Windows environment and compiled and tested on the target .NET, Java or IBM I platform. A wide range of business applications can be developed, including web services, web applications, batch, client-server and character-based applications.

CA Plex is proven in small, medium and large enterprises and is routinely used to develop line-of-business and packaged applications that comprise hundreds or even thousands of database tables along with the associated programs, services and user interfaces. Such applications often have a lifetime of many years and Plex provides efficiencies in the area of application maintenance as well as initial design and development. Independent Software Vendors (ISVs) find Plex valuable as it enables a single application design to be deployed to be multiple platforms enabling the ISV to target different markets and be more responsive to changing customer requirements.

2.1 Architected RAD

The term "ARAD" was coined by Gartner Group to describe development tools that complement RAD techniques with model-based development, patterns and code generation. The creation of Plex pre-dates the term "ARAD" but it describes Plex very well and Gartner papers on ARAD have specifically highlighted Plex.

CA Plex takes two very powerful methods and combines them to get the best of both worlds. The first is Information Engineering, where entity relationship diagrams (ERDs) are used to drive development from a data perspective. The second is Object Orientation (OO). Application development with CA Plex typically starts with creating an ERD-type data model and applying object-oriented techniques such as abstraction and inheritance.

CA Plex offers:

- A Windows-based visual development environment, complete with GUI screen designers, an implementation-neutral action language editor, a diagrammer and impact analysis tools;
- A multi-developer repository with built-in configuration management for storing design models across multiple versions, languages and platforms;
- Code generators that automatically create 100% of the C#, Java, C++, HTML, Java, RPG or SQL code required, together with RIA, HTML and GUI clients, 5250 host screens, server programs and database objects.

Patterns

CA Plex includes hundreds of reusable business objects called *patterns*, grouped into libraries. Additional pattern libraries are available from CA Partners and users also create their own. The CA-supplied patterns provide most of the basic functionality required by a typical business application. This fact means that new users of Plex can quickly become productive without needing the relatively advanced product knowledge required to actually design patterns of their own.

3 CA Plex Functionality

3.1 Introduction

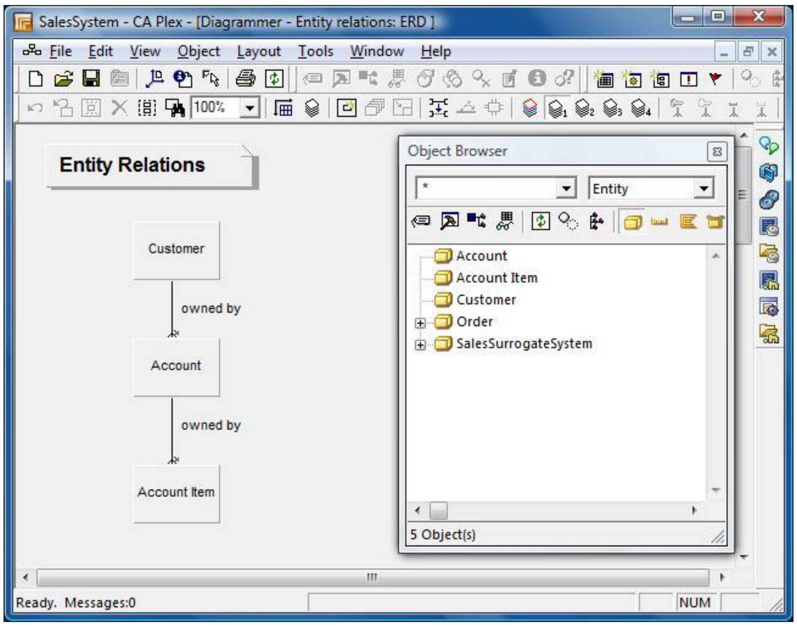
This section describes the primary functionality and features offered by CA Plex. This is achieved firstly by stepping through a simplified example of the development of an application. In this case, we have used a simple 'Customer, Account and Account Items' scenario. Thereafter the techniques used to model and generate services are examined.

3.2 Modeling For Construction

The application development process using CA Plex is essentially a three-step process -involving Data Modeling, Pattern Matching and Customization – followed by automatic code generation. The developer will follow a path of iteration through these three steps, using an Architected RAD process to achieve the desired application. CA Plex uses the concept of a model to hold the application design. The model is held within a repository containing both the model as well as all reusable objects.

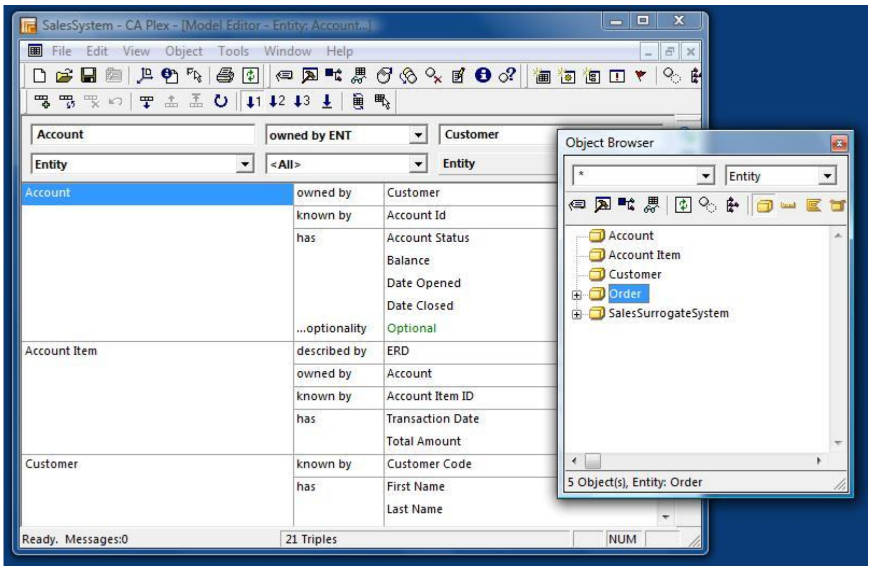
Data Modeling

CA Plex is used to capture the entities, attributes and relationships of the application data model. Shown here is the result, displayed using the Entity Relationship Diagrammer. A range of diagrams is available for both capturing and displaying design information held in the model. Inset is the Object Browser used to navigate through all objects held within the repository.



The structure of a CA Plex application, reflecting the Data Model design but also the structure of other aspects of the application functionality (including functions, components and panels) can be captured using a set of statements defined in a "Subject, Verb, Object" form. This set is known as a Triple. They are captured within a part of the toolset known as the Model Editor. Data models can be entered manually, as is typically the case when developing new applications from scratch or they can be reverse-engineered from existing database schema or other modeling tools such as CA ERwin.

Here, the Model Editor is display and edit the Attributes for each Entity.


SalesSystem - CA Plex - [Model Editor - Entity: Account...]
File Edit View Object Tools Window Help
Object Browser

Account	owned by ENT	Customer
Entity	<All>	Entity
Account	owned by	Customer
	known by	Account Id
	has	Account Status
		Balance
		Date Opened
		Date Closed
	...optionality	Optional
Account Item	described by	ERD
	owned by	Account
	known by	Account Item ID
	has	Transaction Date
		Total Amount
Customer	known by	Customer Code
	has	First Name
		Last Name

Ready. Messages:0 | 21 Triples | NUM

Object Browser: * | Entity

- Account
- Account Item
- Customer
- Order
- SalesSurrogateSystem

5 Object(s), Entity: Order

Pattern Matching and Inheritance

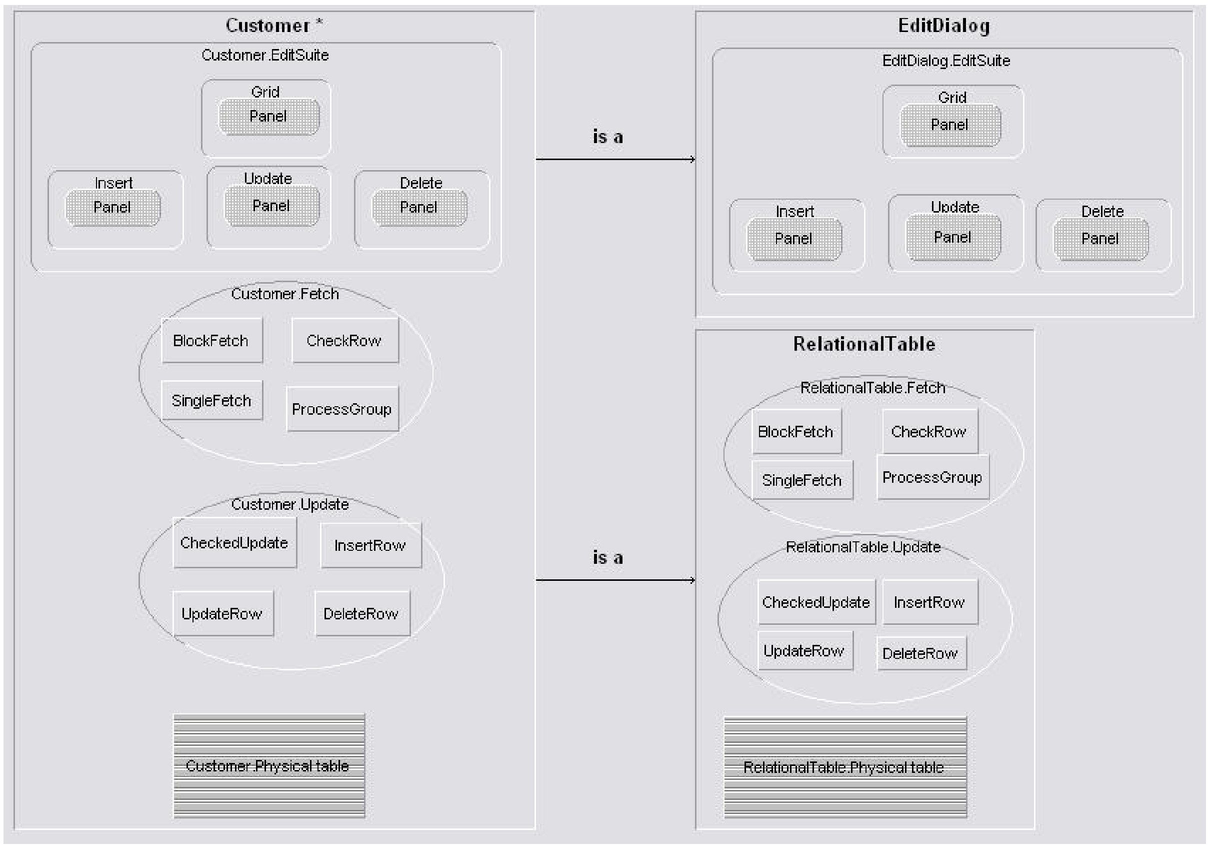
The second part of the development process is to consider the functionality that the application is required to deliver. Examples of this could be Use Cases or the product of a process modeling tool. The idea is to match the functionality required against a set of Design Patterns held within the repository. CA supplies a large set of these Patterns, and part of the power of CA Plex lies in constructing your own set of Patterns to reflect the types of functionality specific to your organization.

Inheritance is the mechanism by which an object or set of objects can acquire the characteristics or design of another object or set of objects. This is the means by which Patterns are implemented in the application design and is always expressed using an ‘is a’ verb in a triple. Multiple inheritance is a key part of the way applications are developed in CA Plex.

Here, a diagram expresses the way that an entity, representing a Customer, has rapidly acquired simple functionality. The entity acquires:

- a table to hold the data about the Customer;
- views of the data;
- access functions to retrieve and update the table; and
- a suite of user functions to enable the display and capture of Customer data.

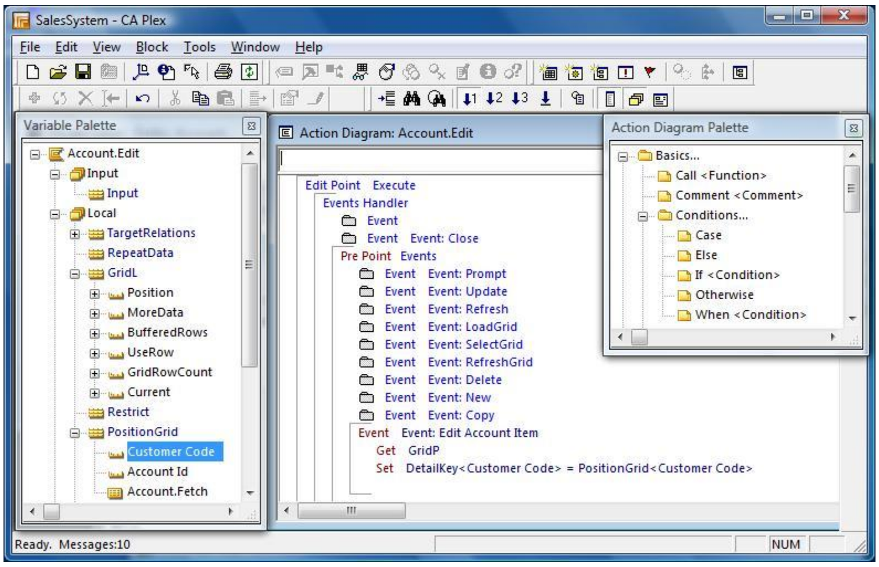
Note the use of the 'is a' statements:



Customization

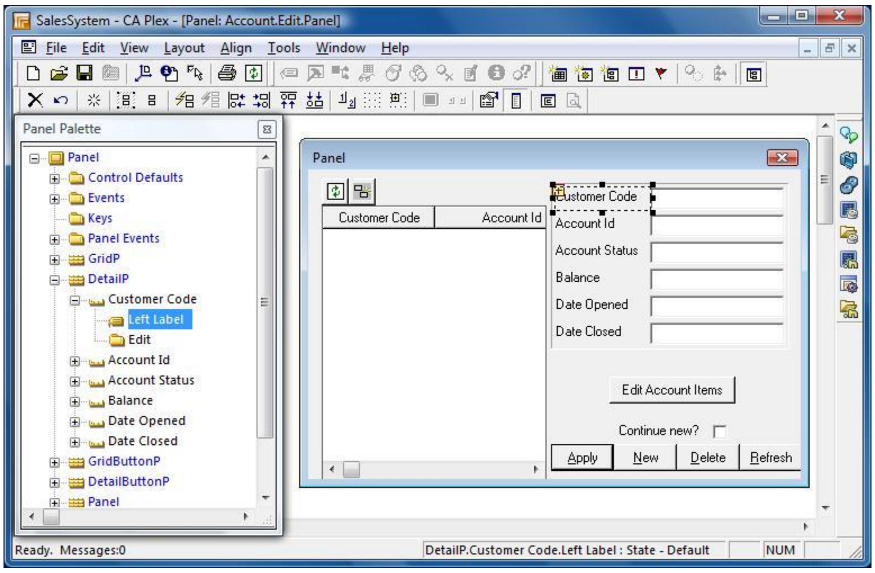
The third phase of constructing the application is to customize the implemented Patterns. If we think of the Patterns as classes of behavior, held as design information, then our implemented Patterns need to be sub-classed in order to suit the application requirements. Examples of this could be overriding elements of the Pattern structure, replacing part of the Pattern design with our own design or adding logic to the application to add behavior to the design. Still further customization involves user interface design or construction of new functionality.

This is the action language editor (or Action Diagram Editor) where logic can be added or changed within the application design. Essentially, each function will have logic expressed as procedural statements held within an action diagram. Most of this logic will be automatically inherited from the design pattern. Appropriate points are made available for the developer to insert custom logic. Statements are explicitly type checked to ensure no syntax errors occur.



```
Edit Point  Execute
  Events Handler
    Event
    Event  Event: Close
    Pre Point  Events
      Event  Event: Prompt
      Event  Event: Update
      Event  Event: Refresh
      Event  Event: LoadGrid
      Event  Event: SelectGrid
      Event  Event: RefreshGrid
      Event  Event: Delete
      Event  Event: New
      Event  Event: Copy
    Event  Event: Edit Account Item
      Get  GridP
      Set  DetailKey<Customer Code> = PositionGrid<Customer Code>
```

With the Panel Designer, the developer is able to make changes to the screen design of the application under construction. The model-driven development process means that Plex automatically provides a default panel layout based on the meta-data in the model. As a result Plex developers rarely need to design a panel from scratch. Note the Panel Palette where all the visual controls making up the panel are available, as well as logical objects such as events.



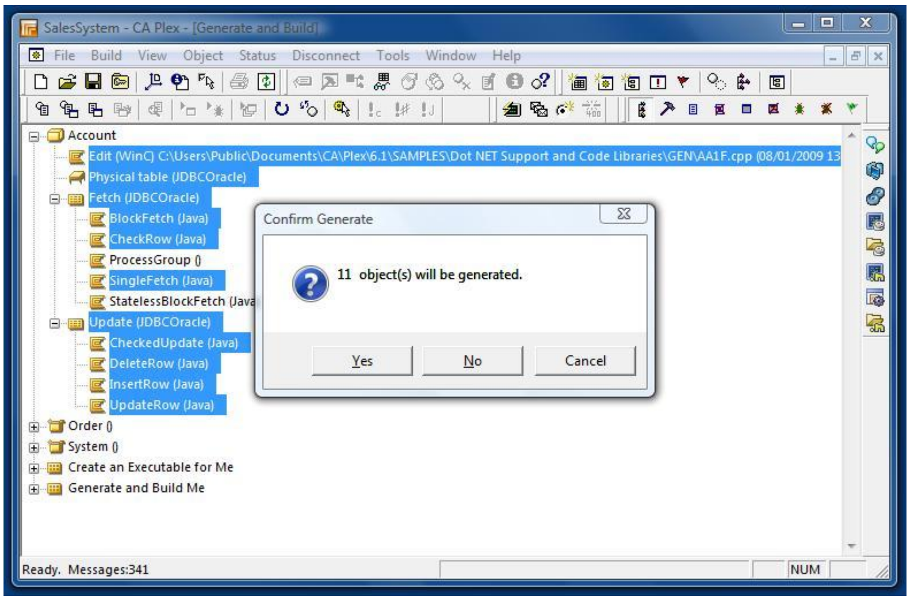
3.3 Application Code Generation

Once the developer has passed through one iteration of the preceding three steps, the application code can be generated, compiled and unit-tested.

The term "application generator" rather than "code generator" is sometimes used to describe Plex. This is to emphasize the fact that Plex generates all of the code for the application – not incomplete fragments, stubs or frameworks. To maintain the source code for an application you edit the model and then re-generate the code rather than editing the code directly. Plex developers rarely modify the generated code and even then typically only to assist in debugging or as a temporary workaround. Where there is a need to use custom source code, the typical approach is to store the code in the model and then inject it into the appropriate point within the generated code. Plex provides a "source code" object type specifically for this purpose.

Once compiled, the application can be run directly from the Generate and Build window.

This picture shows the Generate and Build screen. The developer selects the required objects, generates the code and submits the build. Plex automatically invokes the appropriate build tool. For example, Java code is compiled with Apache ANT whereas C# is compiled with MSBuild. The associated build scripts can be customized to meet the requirements of development and change management processes.



```
Account
  Edit (WinC) C:\Users\Public\Documents\CA\Plex\6.1\SAMPLES\Dot NET Support and Code Libraries\GEN\AA1F.cpp (08/01/2009 13
  Physical table (JDBCOracle)
  Fetch (JDBCOracle)
    BlockFetch (Java)
    CheckRow (Java)
    ProcessGroup ()
    SingleFetch (Java)
    StatelessBlockFetch (Java
  Update (JDBCOracle)
    CheckedUpdate (Java)
    DeleteRow (Java)
    InsertRow (Java)
    UpdateRow (Java)
Order ()
System ()
Create an Executable for Me
Generate and Build Me
```

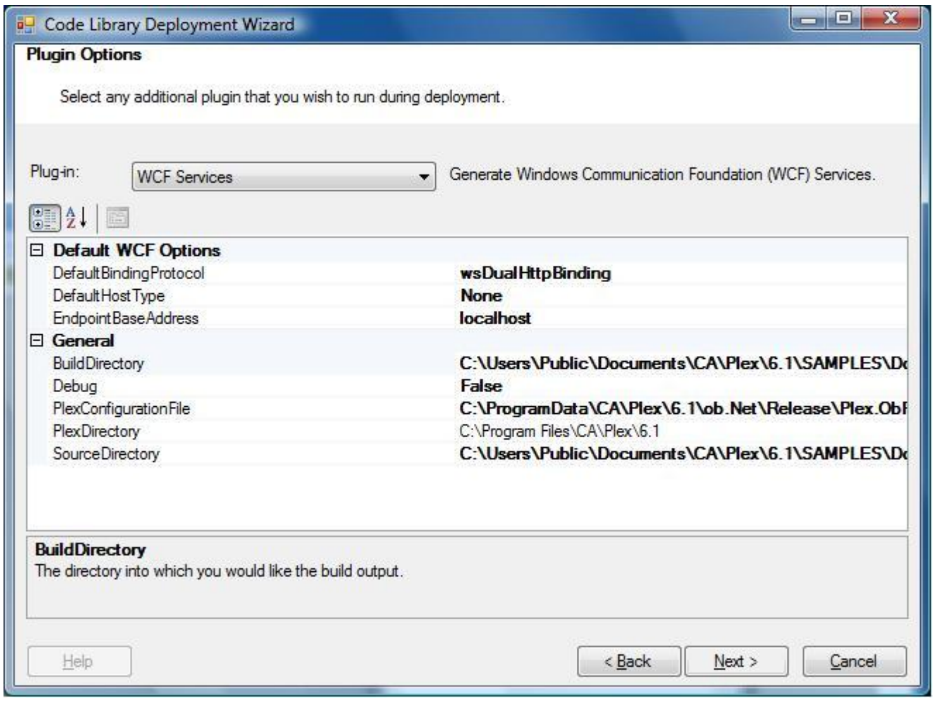
11 object(s) will be generated.

3.4 Service-Oriented Development and Application Integration

New applications are rarely built in isolation. Typically a new application must reuse or integrate with existing applications and packages. Increasingly this includes ERP packages from vendors such as SAP. CA Plex provides a rich set of service development, application integration and reverse engineering capabilities including:

- An abstract Service Modeling language capable of defining service interfaces that can be generated in multiple deployment technologies.
- .NET WCF service generator for generating services based on Microsoft's Windows Communication Foundation technology (see screenshot below)
- EJB Connectors that provide an interface to Java and J2EE applications.
- Database schema import to allow new applications to be built upon an existing databases;
- COM, .NET Connectors and OLE Automation interfaces that allow CA Plex-generated applications to be consumed by external COM and .NET-compatible environments such as Visual Basic;
- COM Import capability that allows Plex developers to consume existing third-party COM components using the native Plex action language;
- Ability to embed hand-coded 3GL code within the generated code without compromising platform independence. This allows the developer to accomplish any programming tasks not directly supported by the native action language, including calling of existing Java, .NET or Windows programs written with other development tools.
- Special capabilities to easily call existing IBM i (OS/400) programs.
- A Model XML Import/Export capability that allows meta-data to be interchanged with other development tools such as CA ERwin.

This picture shows the wizard interface that is used to generate services that have been previously modeled within Plex. In this example, the WCF Service plug-in has been selected for generation. WCF supports a number of deployment options including IIS web services, Windows operating system services and MSMQ.



Plugin Options

Select any additional plugin that you wish to run during deployment.

Plug-in: WCF Services — Generate Windows Communication Foundation (WCF) Services.

Default WCF Options	
DefaultBindingProtocol	**wsDualHttpBinding**
DefaultHostType	**None**
EndpointBaseAddress	**localhost**
General	
BuildDirectory	**C:\Users\Public\Documents\CA\Plex\6.1\SAMPLES\D**
Debug	**False**
PlexConfigurationFile	**C:\ProgramData\CA\Plex\6.1\ob.Net\Release\Plex.Ob**
PlexDirectory	C:\Program Files\CA\Plex\6.1
SourceDirectory	**C:\Users\Public\Documents\CA\Plex\6.1\SAMPLES\D**

BuildDirectory
The directory into which you would like the build output.

3.5 More on CA Plex modeling

This section describes some additional aspects of the CA Plex approach to application development and modeling. It explains how change is managed in the CA Plex model and how the CA Plex approach relates to industry standards such as MDA and UML.

Configuration Control

CA Plex Configuration control involves managing different versions and implementations within a single model. It can be used in parallel with conventional change and source code management systems which address the management of source code and implementation objects.

The CA Plex configuration management facilities are designed to cater to the needs of the most demanding users (typically software product vendors), while imposing no overhead on the in-house user for whom configuration is less of an issue.

A software application product can comprise one logical design, running on several hardware platforms, in several human languages, in several countries (each with its own statutory requirements), and at many release levels. From time to time, new features have been developed for specific customers that add value to the mainstream product, and need to be integrated into some or all of the current versions.

The CA Plex configuration management system enables CA Plex models to:

- Support many implementations of the same logical design, for different hardware platforms and national languages.
- Allow new features and functions to be developed and tested with current live versions, while maintaining the integrity of those versions.
- Allow any live version to be rebuilt at any time.
- Allow bugs in code which are common to many product versions to be fixed once and once only at the source, or to allow bug fixes to be included selectively in live versions. (When users have developed effective workarounds, fixing the underlying bug can cause far more disruption than letting it continue.)

CA Plex has three complementary configuration control systems:

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

A two-dimensional mechanism is provided to manage versions, and a simpler one-dimensional mechanism to manage variants and national languages.

Model-Driven Architecture, UML and CA Plex

Model-Driven Architecture is a standard promoted by the Object Management Group (OMG) that addresses the abstract modeling, design and implementation of software systems. The CA Plex modeling approach meets the MDA objective of separating business and application logic from the underlying platform technology while also fully automating the generation of application code. MDA defines a series of models and the transformations between them: from the Computation Independent Model (CIM) business model, to the Platform Independent Model (PIM) and finally the Platform Specific Model (PSM). CA Plex provides an equivalent approach with, for example, the CA Plex variants described in the previous section playing a similar role to the transformation marks defined by MDA.

A related OMG standard is UML (Unified Modeling Language). MDA does not require UML although it is utilized by many MDA tools. Unlike UML, CA Plex does not aim to provide a general-purpose modeling language. For example, CA Plex is not suited to the development of real-time or embedded systems. Instead, CA Plex is better characterized as a "domain-specific" modeling and development tool where the domain is that of business-transaction-processing applications using relational databases for persistent data storage.

4 CA Plex Platforms

4.1 Introduction

This section gives an indication of the development and target environments, covering clients, servers and data sources.

4.2 Development Platforms

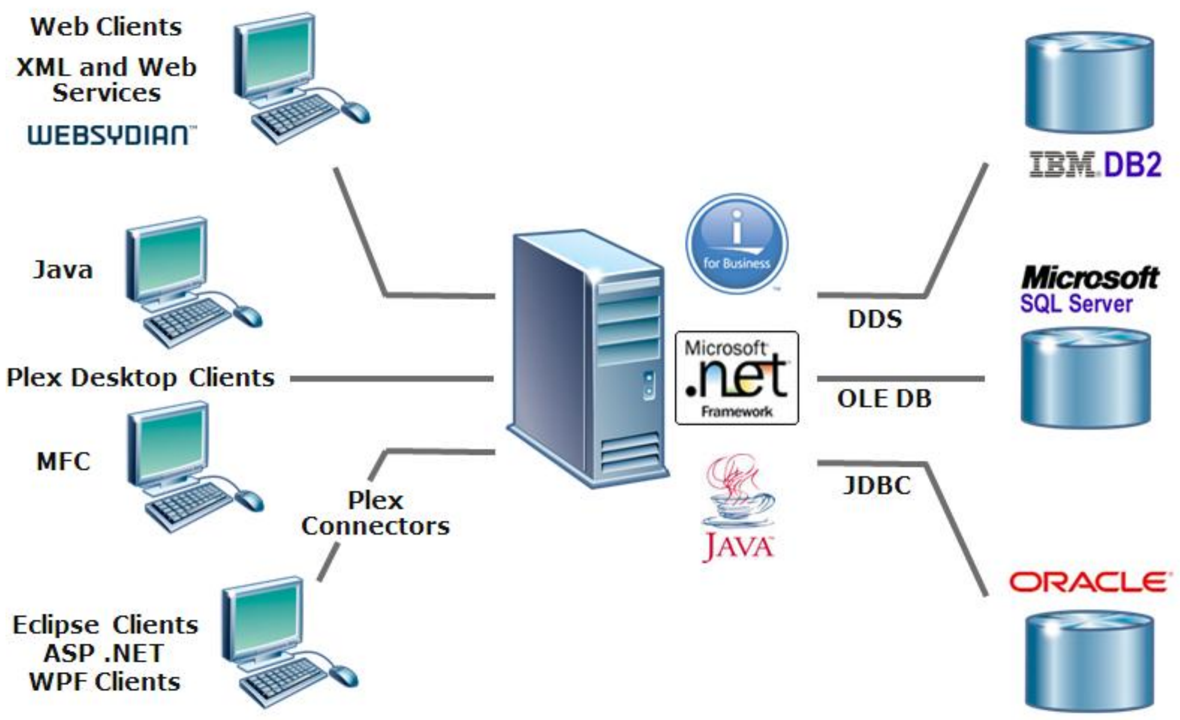
The development toolset is Windows-based graphical environment that runs on Windows Vista or Windows XP. In addition, an appropriate target platform will be required to compile and test the application. For example, generated RPG IV code must be compiled on the IBM I platform.

The development toolset is customizable through an extensive API and XML Import/Export capability. These facilities enable the creation of add-ins to automate and extend the functionality of the tool and support the interchange of meta-data with other tools.

4.3 Target Platforms

CA Plex provides a rich set of options for deploying applications on Windows, Linux, IBM i and other platforms. A key benefit of the tool is that the development process is very similar regardless of the target platform. This means that developers can create and maintain applications for multiple platforms without being an expert with each platform. For example, a CA Plex Java programmer could generate RPG programs without needing to code or even understand RPG syntax (and the opposite is also true – a CA Plex RPG developer can create Java programs without writing a single line of Java code). A well-balanced CA Plex project team normally includes at least one developer with good knowledge of the underlying platform. For example, a CA Plex Java project team should include at least one experienced Java developer. This is because most non-trivial development projects require the use of some hand-coded source code in the model or some level of integration with platform-specific systems, APIs or third-party products.

The deployment options are summarized in the diagram below and described in more detail in the sections that follow.



Server and Database Options

CA Plex supports a number of code generation languages for server-side business logic and data access. CA Plex provides full support for the design and generation of database schema, not only server logic.

The three principal server-side code generation languages are listed below. These servers can be combined in a variety of n-tier and dynamic application partitioning scenarios. For example, a CA Plex .NET application on Windows can make calls though to an RPG IV application running on the IBM i.

- **Java.** Platform-independent Java code can be generated and deployed in both Standard Java (Java SE) and Enterprise Java (Java EE) configurations. JDBC is used for data access.
- **.NET.** Microsoft's .NET platform is supported through C# server code generation with OLE DB data access.
- **RPG IV.** For native IBM i applications, RPG IV code generation is provided. See later section for more on IBM I support.

Because CA Plex applications use standard data access APIs such as OLE DB and JDBC they can target a wide variety of database management systems. CA officially certifies the industry-leading commercial databases Microsoft SQL Server, DB2 and Oracle. Many CA Plex customers have successfully deployed applications beyond this core set of supported databases.

RIA, AJAX and Thin-Client Options

CA Plex provides a number of options for deploying applications via a web browser including modern so-called Rich Internet Applications (RIAs). These options include:

- Using the Websydian WebClient product to generate a thin client based on HTML and AJAX. WebClient can be used for many different types of web applications including "Business to Consumer" e-commerce applications. This option integrates web development within the CA Plex model-driven approach.
- Using the CA Plex Java generator to create Swing-based Java Webstart clients or Java Applets. Java applets provide rich graphical interfaces that are suited to environments where high bandwidth can be guaranteed, such as a corporate Intranet. This option is not normally used for e-commerce applications.
- Hand-coding the web presentation layer in technologies such as ASP.NET or JSF. This option may be appropriate where the web development team is separate from the CA Plex team developing the core business application. In this scenario, CA Plex WCF service and EJB Connectors provide a convenient way for the CA Plex developers to define a programmer-friendly interface to be used by the web developers.

Websydian WebClient

The Websydian WebClient is developed by CA Partners ADC Austin and Websydian A/S. It provides the CA Plex model-based software patterns and runtime systems required for dealing with all the challenges of browser-based software development. This includes user and session management, data integrity protection and role-based user access control. Using panel layouts created in the CA Plex model, WebClient automatically generates AJAX-enabled web applications using open source technologies such as Eclipse and the Dojo Javascript library. Because it is based on the CA Plex Java generator, WebClient supports a wide variety of web application servers, including IBM Websphere.

Desktop GUI Client Options

CA Plex enables the design and implementation of Desktop GUI clients (sometimes called Rich Clients). These clients can communicate with any of Java, .NET or RPG IV servers listed in the previous section. Dynamic application partitioning is also supported so, for example, a single CA Plex Java Client can be switched dynamically at runtime between Java, .NET or RPG IV servers with no code re-generation or re-deployment being required.

- **Desktop Client (Java SE)** Deployable on Windows, Linux or any other operating system that supports the Java SE platform. CA Plex Java clients are based on the Swing library and may be deployed as standalone Java client applications or as downloadable applets executed in a web browser. They are also compatible with the Java Webstart deployment technology.

- **Desktop Client (C++)**: Native Win32 clients based on Visual Studio and MFC that can be deployed on Windows Vista and Windows XP. CA Plex Win32 clients provide a core set of GUI controls plus the ability to utilize ActiveX controls to support a rich user experience.

CA Plex and the IBM i platform

CA Plex provides very strong support for the IBM Power Systems platform and its IBM i operating system (formerly System i, iSeries, AS/400 and OS/400 respectively). This includes:

- RPG IV and RPG/400 code generation
- Native DDS code generation for database objects, panels and reports
- A remote build environment that allows IBM i builds to be submitted and monitored without leaving the Windows-based CA Plex development environment.
- Java and SQL generation for IBM Power Systems, including Linux as well as the IBM i operating system.
- "Drag and drop" 5250 screen designer. Green screen (5250) generation is a separately-licensed feature.
- A TCP/IP "Dispatcher" service to manage communications with CA Plex-generated applications on other platforms.
- Specific support for calling existing IBM i programs including operating system programs, hand-coded programs, and programs generated by CA 2E and third-party development tools.
- Reverse engineering of IBM i databases and applications, including those generated by CA 2E.
- Reverse engineering of COBOL and RPG code into CA Plex though partner services and tools.

Additional information on supported platforms

CA publishes detailed version information regarding the operating systems and databases supported by specific versions of CA Plex. This information is provided in a Compatibility Matrix that is accessible from the CA Plex product page on the CA Support Online web site (www.ca.com/support).

5. Other Information

5.1 Introduction

This section covers an outline of the CA Plex benefits, CA consultancy and training support available, and CA Partner products.

5.2 Product Benefits

The benefits achievable through CA Plex-based development are summarized as:

- Increased productivity through working at a level of abstraction rather than at the 'nuts and bolts' level;
- Productivity is enhanced through the use of pre-existing model components – Patterns;
- Supports rapid application development;
- Increased application quality and flexibility provided by a model-based and pattern-based approach;
- Increased maintenance efficiency, again achieved through working at the model level;
- A high degree of technology independence as different generators are provided for the different supported environments;
- Elimination of the need to have teams of experts in C#, Java, RPG, C++, DBMS, HTML, etc. – a single skill-set is required.

These benefits may track through to a range of business benefits, depending on usage and the nature and scope of the generated application:

- Reduced development costs
- Reduced total cost of ownership (including maintenance)
- Improved 'time to market'
- Improved customer satisfaction
- Improved employee satisfaction
- Improved development staff satisfaction and retention.

5.3 Related CA Products

- CA ERwin for extended data modeling and support
- CA Wily Introscope for performance management of Java and .NET applications in production.
- CA 2E shares a common history with CA Plex and many CA 2E users have adopted CA Plex alongside CA 2E or migrated their 2E models to CA Plex.

Comparing CA 2E with CA Plex

CA 2E is an enterprise application development tool that is widely used on the IBM i. CA Plex was designed by the same team as CA 2E and CA Plex uses many of the same design concepts as 2E but extends them to incorporate object-oriented techniques and multi-platform deployment. This enables 2E developers to more easily learn CA Plex.

Similarities

- Data modeling - both 2E and CA Plex implement a data-driven approach in exactly the same way.
- Modeling language - CA Plex allows developers to describe their data models using exactly the same grammar and verbs as 2E (refers to, owned by, known by and has). CA Plex extends this familiar and effective language.
- Design objects - CA Plex uses most of the same design objects as 2E – entities (files), functions, fields, views (access paths) and so on. CA Plex has additional object types to support functionality not provided by 2E.
- Action diagrams - CA Plex uses action diagrams to describe procedural logic in a manner very similar to 2E.

Differences

- Multi-platform code generators – CA Plex supports Java, C# and C++ code generators as well as RPG IV.
- GUI panel design – CA Plex supports the design of graphical user interfaces, not only 5250 screens.
- Object orientation - CA Plex supports multiple inheritance through class hierarchies of unlimited depth.
- Diagramming - CA 2E has no diagramming facility. CA Plex does.
- Pattern libraries - CA 2E's function types and templates provide some of the functionality of a pattern library, but it supports only one such library which is built-in to the tool. CA Plex supports any number of pattern libraries, each of which can be optimized for a particular environment or industry. CA Plex patterns are built using CA Plex. Thus you can modify any aspect of them or create your own patterns at any time.

5.4 CA Partner Services and Products

CA Plex benefits from a strong eco-system of partners around the world who provide services and training for the product. For a listing of partner companies, go to the CA Plex product page on www.ca.com and click the Partners tab (or click this direct link: http://www.ca.com/us/products/collateral.aspx?cid=193923)

In addition, several partners produce complementary product offerings. A partial list is provided below.

- ❖ ADC Austin and Websydian A/S provide the Websydian WebClient for CA Plex as described earlier in this document as well as automated migration from the CA 2E to Plex.
- ❖ Websydian A/S provides Websydian TransacXML for XML document processing and web services capabilities such as WSDL import. In addition Websydian provides the Websydian Express web deployment solution.
- ❖ CMFirst provides change management and model management solutions for CA Plex.

- Jumar Solutions provides a bridge with CA's popular CA ERwin data modeling product.
- Desynit provides the YouEye patterns for developing rich graphical user interfaces with a modern look and feel.

5.5 References

The following web sites provide further reading and additional information on CA Plex:

- CA Plex product page. Go to www.ca.com and look for Application Development and Databases under the Products menu. CA Plex can be found in the Application & Business Rules Development section. Or click this direct link: http://www.ca.com/us/products/product.aspx?id=258
- CA Plex support page (www.ca.com/support).
- Message board on caforums.ca.com (http://caforums.ca.com/ca/?category.id=caplex)
- CA Plex Wiki. (www.plexinfo.net) A user community web site that anyone can edit. Note that CA is not responsible for the content of this site.