Microsoft® Visio® 2000

Microsoft® Visio® 2000 Developer Tools for UML White Paper

Table of Contents

Published: April 2000

Visualizing Software Development Projects

Successful software development goes beyond writing good code; it starts with effective communication between information technology teams and business decision-makers. Software design that is captured in flowcharts, diagrams, or words can communicate how the software will work, what it will do, when it will do it, and how people, data, hardware, and other software will interact with it.

Four industry-standard tools make it possible to depict and share this information graphically. These tools are:

- **Unified Modeling Language (UML)**, a common notation for visually describing and interpreting the pieces, relationships, and actions that a software application comprises.
- **Microsoft® Visio® 2000 Professional Edition and Microsoft Visio 2000 Enterprise Edition**, flexible IT design and development tools that speak the UML notation fluently.
- **Microsoft Repository**, a tool that enables developers to store and retrieve vital model components so that the information can be shared across teams.

This paper discusses the value of a visual approach to software design, summarizes the Unified Modeling Language, presents a detailed scenario of Visio 2000-based UML diagramming, describes the role of Microsoft Repository in software design, and explains how Visio 2000 developer tools facilitate easy, accessible software design and documentation.

The Value of a Visual Approach

A *software model* is a collection of diagrams that describe how a software-based system behaves and interacts with the world around it. A software model can depict precise interactions between the system and entities such as users, PCs, mainframes, servers, and other software. A software model can also depict object-oriented components such as events, objects, and classes. Some examples of software systems that can be modeled using UML include computer games, library information systems, and a process for assembling microchips.

Build better software

A single model can offer different views that provide a big-picture look at how complex components interrelate. Or it can help developers focus on one aspect of development. The simple act of abstraction required to specify the design up-front can enable developers to think more clearly about a problem and visualize unexpected solutions. Whether used as part of a formal process or informally on a case-by-case basis, Visio 2000 developer tools—with built-in, comprehensive support for UML-based software design—can expose better ways to build applications, reveal errors and mistaken assumptions, and help developers clarify their ideas and understand all parts of a project.

Meet project requirements

Graphical models are a proven means for conveying complex concepts to various audiences. Like flowcharts, software diagrams act as visual checkpoints to ensure that a system's design meets requirements and customer expectations. Using the Visio 2000 developer tools for UML, teams can create complete documentation that supports reusable code components, enhances project planning, and helps achieve buy-in from management, marketing, and others. Initial and ongoing diagramming can enhance communication with end users and team members to ensure that project requirements are met.

Save time and money

Software designs can be created before coding begins and they can be updated and extended throughout development. Because it enables developers to capture requirements early, a visual view of the software can shorten development cycles and save money by preventing costly design errors. Powerful tools like Visio 2000 can even reverse engineer legacy code into a visual model. This enables developers to more rapidly understand the structure of existing code and more easily enhance earlier versions. In addition, Visio 2000 provides dynamic, language-specific error checking for more efficient compiling.

Enhance collaboration

Another feature of high-performance modeling and documentation tools like Visio 2000 Enterprise Edition is the ability to export models and their component parts to a storehouse called a *repository*. Once in the repository, the model can be easily accessed by other developers using any compatible tool. Visio 2000 program's support for documenting and diagramming software in the major object-oriented notations further ensures that developers divided by department, function, time, or geography can easily share diagrams and work together.

Make diagramming easy

Drag-and-drop shapes and automated, intuitive drawing tools enable any developer to start diagramming instantly. Intelligent shapes—organized by diagram type for quick access—are preprogrammed to behave like the objects and classes they represent. When you change the attributes in one instance of a shape, its attributes change in all instances automatically. The UML Navigator window shows elements in a tree structure for quick overviews and easy drill-downs. The Microsoft Visio 2000 interface integrates standard Microsoft Windows® commands and features. Comprehensive online help for the Visio 2000 application and its UML solution is built-in.

Microsoft Visio 2000 developer tools make it possible

The Visio 2000 developer tools provide full-fledged support for documenting, designing, and developing software applications using UML. Although Visio 2000 drawing tools are renowned for their ease of use and informality, Visio 2000 diagrams can also be built upon complex underlying models and used to convey rigorous, precise meanings. Developers who need a quick solution for visualizing existing and proposed systems—as well as developers who want a modeling solution that enables them to design first and then generate code—will find the ideal solution in Visio 2000.

The Unified Modeling Language in Microsoft Visio 2000

The Unified Modeling Language (UML) is used to specify, construct, visualize, and document a software system. A *model* is an abstraction that reveals all or part of the system from a specific perspective. A model typically refers to a particular phase of development. Like blueprints, UML models help teams visualize a system's architecture. One system may include many diagrams. UML notation, meanwhile, is a kind of visual vocabulary. Its shapes and symbols allow the user to create diagrams, each of which provides a unique view of a model. UML notation enables the creation of these eight diagram types.

Diagram Type	Purpose
Use Case	Shows someone or something interacting with the system.
Static Structure (Class)	Describes the static view of a system in terms of classes and relationships.
Sequence	Shows how objects interact with each other over time.
Collaboration	Shows the relationships and interactions between particular objects.
Statechart	Shows the sequence of states that an object can go through.
Activity	Captures actions and their results in terms of changes in objects' states.
Component	Shows how high-level software components make up the system's structure.
Deployment	Shows how hardware and software in the system are physically configured.

Visio 2000 Professional Edition and Enterprise Edition are ideal for modeling, diagramming, and documenting software using all UML diagram types. The following scenario demonstrates how the structure of a particular software system can be more effectively visualized using UML diagrams created in Visio 2000.

UML Design Scenario: A Car Rental Software System

This scenario discusses how the eight UML diagram types might be used to model a car rental agency's software system. Beginning with three simple use cases, the examples capture the core processes in the system.

Use case diagram

A *use case* specifies an interaction between an actor and the system. One property of a use case is that it achieves a goal for an actor. A typical software system might include hundreds of simple use cases. Some use cases applicable to the rental agency's system are:

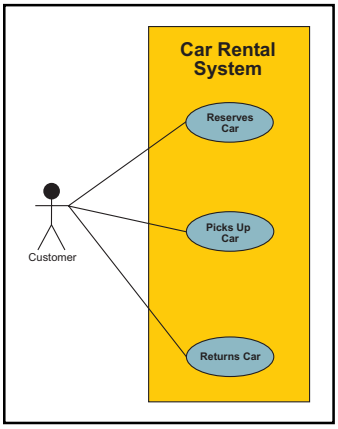

Figure 1: Use case diagram. *The Visio 2000 UML solution provides a complete library of SmartShapes symbols for all eight UML diagram types. Users drag and drop the intelligent shapes onto the page to represent elements in UML notation. The shapes are programmed to behave in ways that are consistent with UML semantics.*

I. Customer reserves car

Before obtaining a car, a customer must make a reservation. The customer contacts the rental agency and makes a request. The agency accepts or declines the request based on a number of criteria, such as the availability of cars or the customer's rental history. If the reservation is accepted, the agency completes a form containing customer details. Payment of a deposit completes the reservation.

II. Customer picks up car

When the customer arrives at the agency, the model of car the customer requested is allocated, depending on current stock levels. After paying the full fee, the customer receives the car.

III. Customer returns car

The customer returns the car to the agency on the day specified in the rental agreement.

Static structure (class) diagram

The next task is to classify the objects involved and their relationships. Examining use cases helps identify classes. Classes of objects are modeled using static structure diagrams that show the overall structure of the system, as well as relational and behavioral properties.

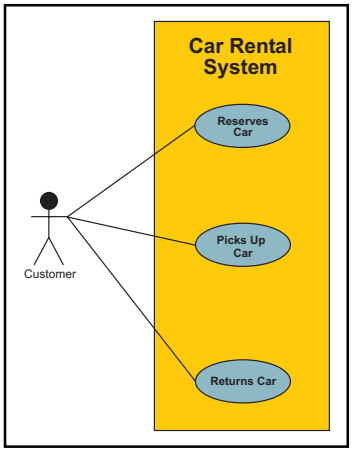

Figure 2: Static structure (class) diagram. *Visio 2000 supports the creation of integrated system models based on UML 1.2 notation. UML Properties dialog boxes allow the user to easily add or change attributes, operations, and other properties of UML elements. In Microsoft Visio 2000 Enterprise Edition, developers can generate fully customizable Microsoft Visual Basic, C++, and Java code skeletons from class diagrams.*

In a class diagram (Figure 2), the objects involved in the car rental system are grouped into classes. Each class contains a name section and an attribute section. Some classes also include an operations section, which specifies how objects within that class may behave.

In the Customer class, attributes include name, phone number, and address. The date of birth is required to determine that the customer meets the minimum age requirement to rent a vehicle. The rental company must also record each customer's driver's license number. The Customer class also stores operations, such as *reserves*.

Class diagrams support inheritance. In Figure 2, for example, the Mechanic and Rental Agent classes inherit attributes like *name* and *address* from the Employee class.

Sequence diagram

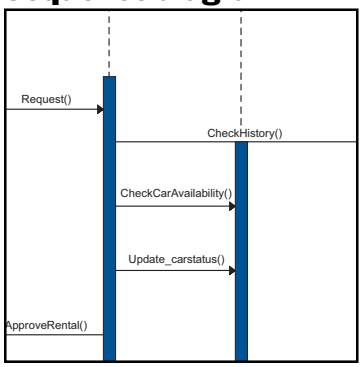

Figure 3: Sequence diagram *showing one use case (Customer reserves car) over time. The sequence of events is read from top to bottom. Objects are shown in boxes across the top. Each object's lifetime, shown as a dotted line, extends downward through time. The time period in which an object is performing an action is shown as a vertical bar; messages flow horizontally between objects.*

A sequence diagram shows an interaction arranged in a time sequence. The participants are shown as lifelines with messages passing between them. Sequence diagrams are drill-down views of use cases, but they show the flow of messages from a different viewpoint. Sequence diagrams help to document the flow of logic within an application. In a comprehensive software system, the sequence diagram can be quite detailed and can include thousands of messages.

Suppose that a customer wishes to reserve a car (Figure 3). The rental agent must first check the customer's records to ensure that the customer may do so. If the customer has rented a car from the company before, his or her rental history will already be recorded and the agent need only ensure that all previous transactions ran smoothly. For example, the agent can confirm that the customer's previous rental cars were returned on time. Once the customer's rental status is approved, the car can be rented.

Collaboration diagram

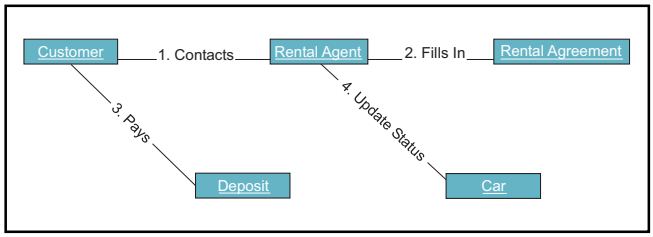

Figure 4: Collaboration diagram. *As in a sequence diagram, interactions between objects are shown in order of occurrence. Yet in a collaboration diagram, time is less important than demonstrating which objects interact with each other.*

A collaboration diagram is another type of interaction diagram. Like a sequence diagram, it shows how a group of objects in a use case collaborate with one another. In a collaboration diagram, each message is numbered (Figure 4).

Statechart diagram

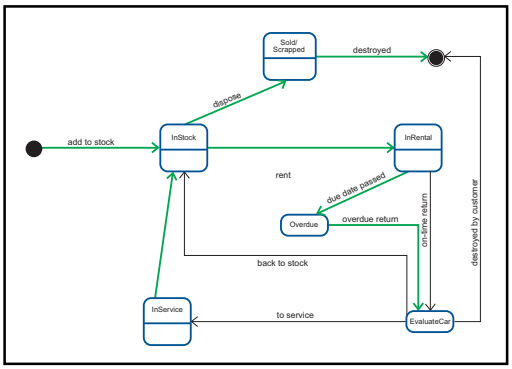

Figure 5: Statechart diagram. *The life of a car as it might appear in a UML model of a rental agency's software system. At any given moment, the car must be in one of the states shown; arrows represent transitions between states. Microsoft Visio 2000 Enterprise Edition checks UML models for semantic errors, including circular references and missing data.*

An object's *state* is defined as its attributes at a particular moment. Objects move through various states as they are influenced by outside stimuli. In this example, the object is a car (Figure 5). As it moves through the rental system, its many states produce a complex but illuminating diagram.

For example, a car is first added to the fleet. It remains in the state *InStock* until it is rented. After renting, it is returned to the fleet and to the InStock state. At various times in its commercial life, the car may require repairs (*InService*). When the car reaches the end of its usefulness, it is either sold or scrapped to make way for a new vehicle. The statechart diagram maps these states, as well as the triggering events that cause the object to be in a particular state.

Activity diagram

An activity diagram is essentially a flowchart showing the logic that occurs in response to internally generated actions. An activity diagram relates to a specific class or use case, showing the steps involved in carrying out a particular operation (Figure 6).

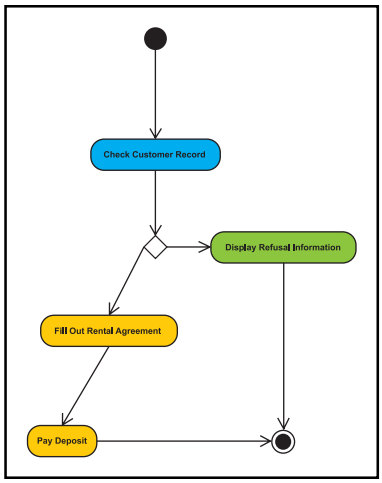

Figure 6: Activity Diagram. *The logic flow starts at the black dot at top. A customer requests a rental car, and the rental agent checks the customer's details and history. If the customer has been identified as one the company will not rent to, a refusal screen is displayed to the rental agent. The customer does not rent a car and the logic proceeds to a final state, shown by the black dot at lower right. If the rental is approved, a rental agreement is completed, the deposit is paid, and the logic again proceeds to the final state.*

Component diagram

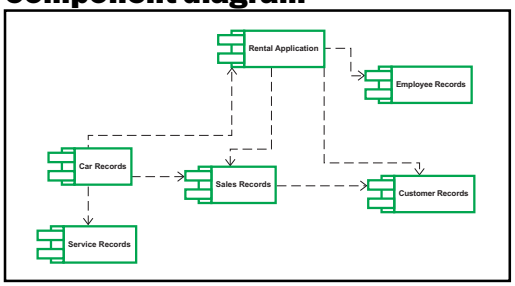

Figure 7: Component diagram. *These subsystems comprise the rental software system.*

The software system is built on a centralized database containing past rental records, car details, service records, and customer and employee details. It is critical that this data be centralized in one database, because stock levels vary by the hour and all parties must have up-to-the-minute information. Keeping the data current requires real-time updating of information by all parties. A component diagram shows how various software subsystems make up the overall structure of the system (Figure 7). The subsystems include Car Records, Service Records, Sales Records, Customer Records, and Employee Records.

Deployment diagram

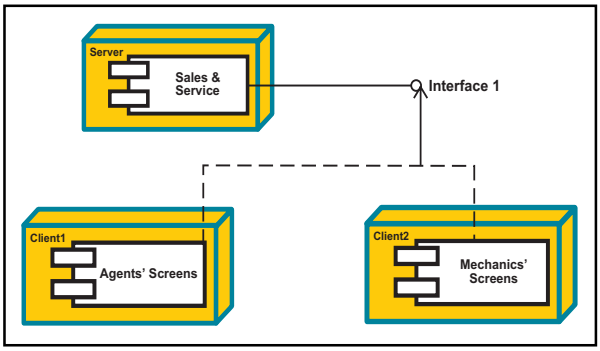

Figure 8: Deployment diagram. *The client-server architecture makes it possible for different classes of users to access and update the data in a central database.*

A deployment diagram shows how the hardware and software in the system are configured. The rental agency needs a client-server system with a central database of records that the staff can access. Rental agents need access to the data on vehicle availability. Meanwhile, mechanics need to be able to flag a particular car as being InService.

Microsoft Visio 2000 Developer Tools: Two Levels of UML Support

Visio 2000 offers full support for creating object-oriented software system models. The Visio 2000 solution includes these features to help developers get up and running quickly:

- comprehensive support for UML 1.2
- complete library of SmartShapes® symbols for all UML diagram types
- shapes organized by diagram type for quick access
- automated, intuitive approach to drawing
- standard Microsoft Windows interface, commands, and features
- smooth integration with Microsoft Office
- built-in Microsoft Visual Basic® for Applications (VBA) 6.0 for creating custom solutions
- support for documenting and diagramming systems in all major object-oriented notations, including Booch, Rumbaugh, Jacobson, Shlaer-Mellor, Yourdon and Coad, OLE, and COM.

Two-tiered functionality meets every team's needs

Microsoft Visio 2000 Professional Edition and Enterprise Edition provide IT professionals with two levels of functionality for designing and documenting enterprise-level software applications and with two levels of integration with UML.

- **Microsoft Visio 2000 Professional Edition** is best used for visualizing the structure of existing and proposed software systems. It supports UML 1.2 notation and reverse engineering to UML diagrams.
- **Microsoft Visio 2000 Enterprise Edition** is ideal for designing software and leveraging software diagrams to jump-start development by means of code generation. It includes complete Professional Edition functionality, as well as code generation, Microsoft Repository, and additional UML support.

Both Visio 2000 Professional Edition and Enterprise Edition provide:

UML 1.2 notation support

Predefined templates and nearly 90 SmartShapes symbols representing UML elements enable the construction of all UML diagram types. Also included are more than 250 additional methodology-specific shapes, preprogrammed to behave according to the rules of each methodology.

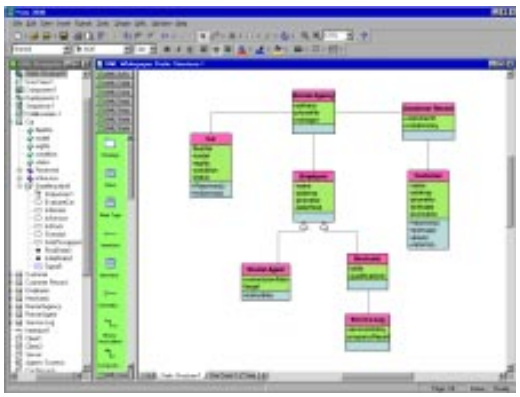

Figure 9: The Visio 2000 interface. *The UML Navigator, at right, gives developers a hierarchical view of all the elements in a model. UML diagram-specific stencils provide nearly 90 intelligent, drag-and-drop shapes—each programmed to behave like the UML element it represents. The easy, Microsoft Windows–based interface reduces ramp-up time and increases productivity.*

UML Navigator

The UML Navigator allows developers to create, view, and manage a system's entire set of models and diagram types by using a hierarchical tree. The UML Navigator enables developers to move quickly from one diagram to another and to view a model's complete structure while working on a single diagram type. Model elements can be added, deleted, and renamed directly in the Navigator, and these changes are immediately reflected in the elements on the drawing page. By dragging and dropping items from the tree onto the drawing page, users can display only the components they want to manipulate. When a developer creates a UML static structure diagram by reverse engineering a Microsoft Visual Studio® project, the model elements appear automatically in the UML Navigator.

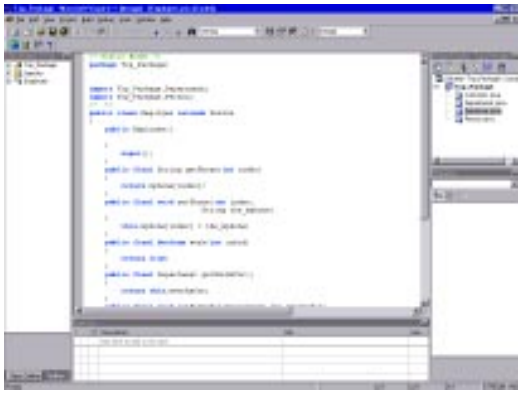

Figure 10: Visio toolbar in Microsoft Visual Studio. *On-demand generation of software diagrams enables developers to get a visual view of their software directly from the integrated development environment.*

Reverse engineering to UML diagrams

The ability to auto generate diagrams from Microsoft Visual C++®, Microsoft Visual J++®, and Visual Basic source code to generate UML static structure diagrams allows developers to easily document their software projects before handing them off, or to understand the structure of legacy systems before embarking on system updates. Following installation of either Visio 2000 Professional Edition or Enterprise Edition, Visual Studio toolbars can be customized to show a Visio 2000 toolbar that provides on-demand reverse engineering (Figure 10). This approach enables Visio 2000 developer tools to access information directly from the integrated development environment (IDE), yielding accurate UML models.

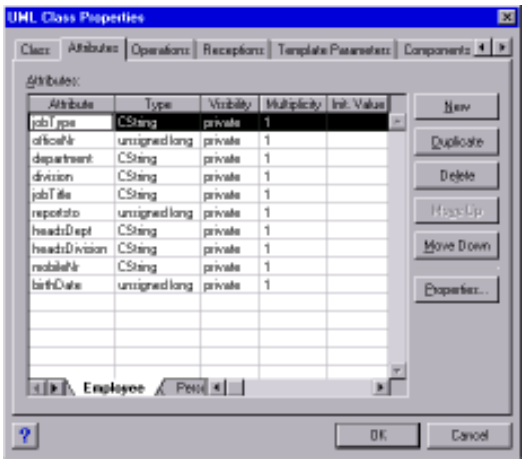

Figure 11: UML Properties dialog box. *Developers can easily add attributes and discretionary values to UML elements.*

UML Properties dialog box

Developers can easily add attributes, operations, and other properties to UML elements, and they can add discretionary values (called tagged values) to any UML element (Figure 11).

Extensive online and UML-specific help

Both Visio 2000 Professional Edition and Enterprise Edition include the entire UML 1.2 specification for printing and reference. The context-sensitive, online Help provides quick access to UML-specific definitions, as well as help on Visio 2000 shapes, specific diagram types, and UML property values.

Built-in Microsoft Visual Basic for Applications 6.0 (VBA)

Developers can use VBA to easily customize existing solutions and build custom applications.

Microsoft Visio 2000 Enterprise Edition also provides:

Microsoft Repository 2.0 support

Visio 2000 Enterprise Edition provides full, integrated compatibility with Microsoft Repository 2.0. Support for Microsoft Repository allows users to compare and exchange diagrams and to import and export information models regardless of the platform or modeling tool. Developers can easily pass UML static structure models to and from Microsoft Repository for use in other visual modeling and CASE (computer-aided software engineering) development tools.

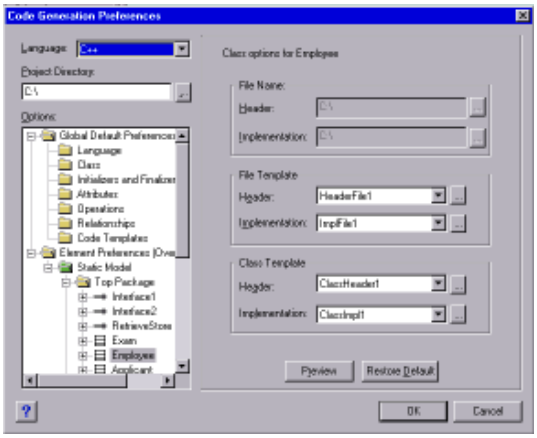

Figure 12: Code Generation Preferences dialog box. *Developers can customize the structure and format of Visual Basic, C++, and Java code generated in Visio 2000 Enterprise Edition.*

Code generation

Microsoft Visio 2000 Enterprise Edition not only allows users to automatically generate diagrams from source code. It also generates fully customizable Visual Basic, C++, and Java code for any diagram that contains class definitions. Fully customizable default templates for code formatting are provided. Developers can select preferences like language, classes, initializers, finalizers, attributes, operations, and relationships to determine the formatting and structure of the resulting code (Figure 12). They can choose whether to customize code on a per-class or full-project basis before it is generated.

Dynamic semantic error checking

Microsoft Visio 2000 Enterprise Edition dynamically checks UML models for semantic accuracy to help developers diagnose and fix errors as they work. For example, Enterprise Edition can detect circular references, missing data, and improper use of UML notation. It can identify two elements that have been assigned the same name, and it can check many other UML syntax rules. When an error is detected, the associated item in the diagram is displayed in red, and its icon in the UML Navigator is selected. An error message automatically appears on the Errors tab in the Status window, guiding the engineer toward a fix.

Language-specific code checking

Users can identify potential compilation errors by checking the syntax in models based on the target language.

Reports

Software designers and developers can create customizable reports from UML static structure, activity, and deployment diagrams, and print or export them as Rich Text Format (RTF) files.

Microsoft Visio 2000 and Microsoft Repository

Microsoft Repository is a software tool in which object-oriented software models can be stored, retrieved, and exchanged. UML models stored in Microsoft Repository retain all the objects, interfaces, data types, properties, and relationships expressed in object-oriented programming languages. The repository manages storage, versions, and access for elements using check-in and check-out procedures. Once a model is stored, engineers can access it with any CASE tool that supports the repository—such as Microsoft Visio 2000 Enterprise Edition, Microsoft Visual Studio, and Rational Rose—to rapidly generate framework code directly from model components. Teams using different modeling or development tools can freely exchange and build on UML components and class diagrams.

In addition, Microsoft Repository 2.0 includes important team-based development features. Teams can maintain multiple versions of the same models to create a historical record of changes. They can freeze components against changes, and they can version or group objects. These features allow individuals to develop a single software application in multiple, independent streams and then recombine the streams when needed.

Conclusion

In addition to the software design, documentation, and code generation described in this paper, both Microsoft Visio 2000 Professional Edition and Enterprise Edition can be used to produce comprehensive supporting documentation for all aspects of software system development, including project time lines, data flow diagrams, database models, business process diagrams, and Microsoft Windows interface design. The resulting diagrams can be printed, pasted into Microsoft Office documents, published in HTML, or saved in a number of graphics formats to be included in specifications, reports, and presentations.

For More Information

To see Visio 2000 in action, visit **www.microsoft.com/office/visio**.