The Synergy of Add-Ins

Everyone has read phenomenal predictions regarding what computing will look like in the years to come. Every corporation has felt intense pressure to bring more sophisticated programs to market faster.

Visual Basic has responded to these trends by providing high-level services for rapid application development. As the product has evolved, its performance and capabilities have increased dramatically. Microsoft designed Visual Basic to be an open design environment that third party vendors could produce extensions for, up to now in the form of a custom controls. These VBX and now OLE components are available from a large number of generalized, or application-specific suppliers. VB allows developers to exploit this component technology by providing a platform that incorporates diverse components into a homogeneous solution.

VB Add-ins provide a new direction to extend the capabilities of Visual Basic. By infusing VB with next-generation tools such as add-ins it will be possible to create an infrastructure that will profoundly change development. These tools will "drive" VB to perform even higher-level tasks. They will dramatically change application development in much the same way that macros changed the way we use spreadsheets. Add-ins however, unlike macros promise much more potential.

What is an Add-In

In VB 4.0 the VBIDE is exposed as an OLE object that can be easily manipulated by specialized OLE servers called Add-ins. They are "specialized" because an OLE server designed as a VB add-in must have two pre-defined methods. One is known as *ConnectAddIn*, which is used by VB to inform the add-in that it is being used. The other is *DisconnectAddIn* which informs the add-in that VB is no longer using it. Once connected to VB an add-in can get information about and manipulate much of the development environment.

Add-Ins can be created by any language that is capable of creating an OLE server, and best of all they can be created by VB itself!

Strategic Uses of Add-Ins

Looking back at VB1, few would have predicted the robust and diverse set of VBXs that have been developed to complement VB. Similarly it's hard to predict what this new capability will deliver—but dream on. In their simplest use, add-ins can automate development tasks, like macros automate spreadsheet tasks. This automation may be incarnated in the form of wizards, or via any type of interface that suits the intended audience. In much more sophisticated implementations, add-ins can provide complete front-ends that allow developers to model systems any number of different ways. These models can then be used to generate an interface and supporting code. The following lists presents possibilities for types of add-ins.

- **Wizards**. Wizards will be created to automate tasks. These can provide easy-to-use wrappers for everything from simple, tedious tasks to the very sophisticated and complex ones.
- **Template Libraries**. Add-Ins will be made to create and deliver libraries of VB object templates. These templates can then be added to any VB project which will accelerate the development process. In addition, libraries will be used to provide complete application frameworks
- **Style Libraries**. In the same way that object templates can be created and used, add-ins will allow the creation of style libraries. Styles are collections of object

properties, which can be captured from existing objects, and applied to over-and-over again

- **Paperless Standards**. Libraries of objects and styles will be used to create paperless standards. Formerly, printed guidelines would be used to explain what a form should look like, or how code is to be commented. Now this information can be retrieved from libraries and added to projects assuring consistent and rapid use of such standards.
- **Forms Generators**. These add-ins will create forms from scratch, or accelerate the process of laying out controls on forms. Custom controls and form templates will automatically be added to projects by these tools.
- **Case Tools**. These tools will allow the graphical modeling of business processes, and then automatically generate code and interface. Such tools may present a building block approach, an outline approach, or some any other approach that can be tailored to a specific audience.
- **Data Modeling**. Similar to case tools, this group of add-ins will address a database-specific set of problems. Project frameworks might be automatically generated after modeling them in high-level interfaces provided by the add-in.
- **Source Code Control**. An add-in can be constructed to be informed of changes to a VB project, such as files being added, saved, and removed. Using these capabilities, source code control systems can provide a much tighter level of integration than was possible before.
- **Expert Systems**. This group of add-ins will provide expert interfaces to VB projects. They might audit projects looking for ways to optimize resource usage, enforce code and interface standards, or examine code for preferred algorithms.