

Preface

This technical document discusses the current status, roadmap and change in requirements of the upcoming Xbase⁺⁺ release termed Visual Xbase⁺⁺.

The Initial Strategy: About Visual Xbase⁺⁺ 2.0

In 2001 Alaska Software set out to provide a new, state-of-the-art integrated development environment (IDE) for its Xbase⁺⁺ product. The IDE, together with the additions and changes on the kernel itself, was destined to be released as a new product: Visual Xbase⁺⁺ 2.0.

The goals of Visual Xbase⁺⁺ 2.0 (VX) were to boost overall developer productivity by introducing an integrated debugger, intelligent help system and an intelligent and seamless project management module to the Xbase⁺⁺ development environment. Furthermore, it was to provide integrated design capabilities via 2-way form, database and project designers.

To achieve sophisticated design capabilities, the various designers of Visual Xbase⁺⁺ were modelled around a new component library to be introduced with Visual Xbase⁺⁺ 2.0.

Project Status

Development efforts put into the VX product have resulted in various prototypes and case studies, some of which have been shown on various conferences and usergroup meetings. The project has progressed nicely and in fact, various parts of the integrated development environment have been completed.

Various kinds of the IDE's designers, especially the form designer module, and the component model have been developed and tested against usability and featureset requirements of Xbase⁺⁺ developers. These aspects of the project, however, have been temporarily put on hold in December 2002 due to new design considerations.

Invisible Motor: The Component Model

One of the core technical aspects of Visual Xbase⁺⁺ 2.0 – and probably the most visible to the Xbase⁺⁺ developer – is its new component model. Generally speaking, a “component” is a piece of software providing certain services and functionality to other components and the application in general.

Component Services

- **Component Interface**

Interfaces allow a component to be interrogated about its characteristics and capabilities. This also allows for interoperation of components written by other parties without having knowledge about a component's internals. A component can define properties, methods and events that can be made publicly available to other components or applications. An interface may or may not contain additional information about either the component or its interface.

- **Event Handling Model**

A generic event and event handling model allows to integrate and extend component behaviour. Furthermore, it allows components to interact with each other, irrespective of the component designer or vendor. The event handling model also contains a specification of how to map operating system events such as a mouse click for example, to the piece of code implementing the component's behaviour to the event.

- **Persistence**

Components and their state can be saved and restored between different sessions.

- **Packaging Model**

Although comparable to a class in many respects, a component also provides extended information and functionality. It may comprise an icon or information on the component's purpose and author for access in the IDE, for example. The component model therefore incorporates specifications on how to provide, distribute and access such meta data.

Similar to the Xbase Parts (XBP) in previous Xbase⁺⁺ revisions, the component library of Visual Xbase⁺⁺ provides extensive services to the developer in the areas of graphical user interface, interconnectivity and data binding. It is through component interfaces that the IDE achieves its visual design capabilities, eg. the capability to design and inspect various parts of an application.

The component library also introduces new principles which result from existing requirements on the Xbase⁺⁺ product, especially in the area of the OOP model, and demands determined by research activities.

Generally speaking, streamlined interfaces allow a much more intuitive approach that will especially make the first steps easier for beginners. The extended event-handling mechanism building on the existing callback/callback slot-model, for example, makes it possible for the IDE to treat events such as mouse clicks as simple properties that can be manipulated via a visual designer. This allows to use simple drag & drop operations for designing forms or database connectivity. Coding a function for handling a key press, for example, is just a matter of selecting the

appropriate event from the respective component's properties and adding code to a handler function automatically injected into your project by the Visual Xbase⁺⁺ IDE.

We must admit that the work necessary for design, development and testing was grossly underestimated. Furthermore, every requirement brought up by market requirements had to be weighed against backwards compatibility. Additionally, recent market developments and emerging platforms, such as Linux and .NET, introduced new demands that Visual Xbase⁺⁺ and its component technology must be able to address.

Market Demand: The Effect of Linux and .NET on the Visual Xbase⁺⁺ Project

In the first quarter of 2003, a serious internal debate took place regarding the strategic positioning of the Xbase⁺⁺ product. The slip of the Visual Xbase⁺⁺ project and the changes in the worldwide IT market in terms of emerging platforms, namely Linux and .NET, gave us the opportunity to review our initial goals and requirements. This is especially true in the context of the visual design components and the next generation graphical user-interface library. These reviews and discussions resulted in the belief that to ensure the future success of Xbase⁺⁺, it has to support both the Linux and .NET platforms.

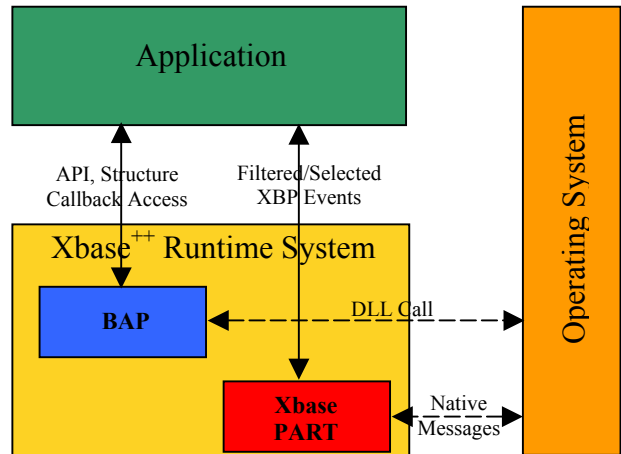
The strategic decision to target the Linux and .NET platforms also affects the Visual Xbase⁺⁺ project. For instance, it adds the multi-platform factor to the equation, something that, as we'll see later, the initial specification of Visual Xbase⁺⁺ 2.0 was ill-equipped to address.

Achilles' Heel: The Component Library

On the technical level, the decision to add support for the Linux and .NET platforms has a huge impact on the Visual Xbase⁺⁺ project. For instance, the initial strategy put a strong focus on compatibility with previous Xbase⁺⁺ releases. Consequently, Visual Xbase⁺⁺ 2.0 was initially specified to utilize a component library built around Xbase PART (XBP)-technology. For the application developer, the library was to give transparent and easy access to the underlying technology while adding advanced functionality, such as self-describing objects, full introspection-support and more, for the component designer.

The XBP technology was originally drafted at a time where developers began to move from text-based application interfaces to full GUI. Consequently, a central design goal of the XBP library was to aid developers in making this move by introducing an abstraction from the operating system's internal mechanisms. By reducing the overwhelming number of messages normally generated by a GUI application to selected notifications, simplicity

was achieved through a slim, easy-to-explain event model. Furthermore, other technologies, such as the unique Hybrid Mode and implicit thread safety, provided Xbase⁺⁺ developers with a unique migration path for their applications.



Original Xbase⁺⁺ Operating System Abstraction

However, despite the fact that the Xbase PARTs have reached these creditable goals, they also posed a serious hurdle for many developers. Because of its proprietary nature, transferring knowledge gained through other sources such as Windows programming books, for example, proved difficult.

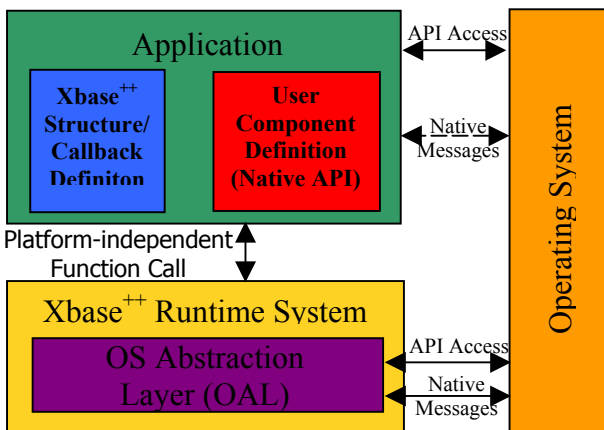
For the experienced developer, on the other hand, the aforementioned operating system abstraction became a limiting factor, which was further aggravated by the progress that was made in the graphical user interface sector in recent years. Because of the need to adapt the proprietary Xbase PART technology to each new invention, its featureset began to fall behind compared to the functionality available from the operating system.

Evaluation: The Xbase PART Technology

- Simple and easy to understand
- Asynchronous event processing, allows database and business logic operations to be performed independent of GUI and/or timing requirements
- Automated management of GUI and OS resources
- Complex functionality such as thread-safety, for example, instantly available
- Lack of transparency due to "black box" approach (no source code, no internal Xbase⁺⁺ implementation); only limited customizing possibilities for the Xbase⁺⁺ developer
- Necessity for Alaska to adapt Xbase PARTs to new technology introduced by OS vendor introduces time

- delay
- Evolution of the Xbase PART library led to complex interfaces due to inconsistent naming scheme for instance variables and methods, insufficient usage of advanced access strategies such as "ACCESS ASSIGN variables"
 - Makes platform unattractive to non-Clipper developers, especially compared to other development packages
 - Inconsistent with goal to publish library in source code
 - Porting XBPs to Linux or .NET also means porting their weaknesses

For these reasons and the new requirements introduced by Linux and .NET, the Xbase PART technology was found inadequate to serve as the foundation for Visual Xbase⁺⁺'s next-generation component library. Instead, an approach will be taken that builds on simple, easy-to-use and maintain components that will be shipped in source code. At its core, the component library is built on top of a lightweight operating system API wrapper that is easily portable to new platforms. Furthermore, due to transparent access to operating system functionality, including support for structures and callbacks, the next-generation component library will provide Xbase⁺⁺ developers with unprecedented transparency and extensibility and will allow them to customize the library to their specifications.



Revised Visual Xbase⁺⁺ Operating System Access

Revised Goals: Visual Xbase⁺⁺ 2.0 and 3.0

The review of the current and projected future market situation as well as of the technical aspects just discussed resulted in a redefinition of the initial requirements. Consequently, development of the Visual Xbase⁺⁺ component library was put on hold while design goals were being reviewed and the benefits of an IDE on the

Windows platform were weighed against the perspective of having support for Linux and .NET.

Scales were finally tipped by the realization that adding support for .NET and Linux actually requires lesser efforts than developing a new GUI framework supporting visual design capabilities. Especially when considering that the most basic requirements actually constitute major shortcomings in Xbase⁺⁺ today, i.e. are issues that must be resolved anyway, continuing to pursue the initial development strategy would be short-sighted.

Consequently, a new development schedule was drafted that is centered around a revised Visual Xbase⁺⁺ product featureset and release schedule. Most notably, the Visual Xbase⁺⁺ 2.0 release has been refocused with respect to the technical requirements and customer benefit. Furthermore, functionality related to the support of the Linux and .NET platforms as well as the IDE's visual design capabilities have been deferred to the next major release termed Visual Xbase⁺⁺ 3.0.

Visual Xbase⁺⁺ 2.0

The goals of Visual Xbase 2.0 have been refocused in order to provide Xbase⁺⁺ developers with access to as much of the benefits of the Visual Xbase⁺⁺ project as soon as possible.

To achieve that goal, Visual Xbase⁺⁺ 2.0 will be shipped with the existing GUI component library (Xbase PARTs) and will not feature visual design capabilities for forms and database connectivity. The incorporation of the next-generation GUI component library as well as the visual design capabilities have been deferred to a later release.

Instead, the Visual Xbase⁺⁺ 2.0 release solely focuses on developer productivity achieved through its integrated development environment featuring an integrated source-level debugger, an intelligent help system and an powerful project management.

Visual Xbase⁺⁺ 2.0 Focus

- Fully integrated Source Code Editor with intelligent help system (IntelliHelp) and other productivity enhancements
- Powerful Project Management, supporting Pbuild-style project files
- Fully integrated source-level debugger, with support for conditional breakpoints
- Command Window for executing arbitrary expressions or commands while coding or debugging

Visual Xbase⁺⁺ 2.0 is targeted to give developers a huge boost in productivity and code safety by reducing turn around cycles and decreasing programming errors.



Visual Xbase⁺⁺ 3.0

Visual Xbase⁺⁺ 3.0 introduces design capabilities to the Visual Xbase⁺⁺ IDE. Building upon a new, platform-independent component library implemented in Xbase⁺⁺, it will provide Xbase⁺⁺ developers with a state-of-the-art rapid application development (RAD) environment.

At least the GUI introduced by Visual Xbase⁺⁺ 3.0 as well as its visual design functionality will be made available on all platforms supported by Xbase⁺⁺ as a language.

Visual Xbase⁺⁺ 3.0 Feature List

- Next-generation component library and GUI controls implemented in Xbase⁺⁺ using native API access via a portable Operating Abstraction Layer (OAL)
- Full platform support for Windows and Linux regarding the user interface components (GUI)
- Shipped with source code
- Transparent access to operating system-specific functionality such as Windows messages
- Transparent access to operating system APIs
- Full support for C-style structures, eliminating the need for BAP or similar add-ons
- Streamlined, easy to use component interfaces
- Complete visual design capabilities for visible and non-visible components, such as entry fields or data tables
- Support for 3rd party vendors plug-ins and extensions
- Support for native Windows, Linux X Windows and .NET forms and controls
- Wizard system, extensible by 3rd party vendors and add-on suppliers
- Editor and designers are fully extensible by 3rd party vendors, allowing to incorporate language add-ons that feature different programming paradigms and/or syntaxes.

tested with respect to Windows 32Bit, .NET and Linux support.

Visual Xbase⁺⁺ 3.0: Will introduce the new GUI component library and visual design capabilities for the Xbase⁺⁺ IDE.

Xbase⁺⁺: as a language and technology platform will become available under .NET and Linux on Intel IA32 and IA64 machines.

The benefits/drawbacks of this revised strategy in terms of time-to-market are as follows:

Visual Xbase⁺⁺ 2.0: Customers will get access to existing productivity enhancements in a relatively short timeframe.

Linux and .NET: Customers will get support of other platforms such as Linux and .NET. This will lead to a better competitive position in their respective markets.

Visual Xbase⁺⁺ 3.0: Visual design capabilities originally announced for Visual Xbase⁺⁺ 2.0 have been deferred to this release.

Conclusion

In total, the existing release plan has been revised to deliver the following results and products:

Visual Xbase⁺⁺ 2.0: Has been re-focused by removing its visual design capabilities from the featurelist. It will, however, provide all features related to developer productivity. The new goal is to provide an unbeaten productivity boost for existing Xbase⁺⁺ developers.

Component Library: A new graphical user interface component library has to be designed, developed and