

Automating Software Builds

Rationale and Case Studies

**by Kevin Alons
Kinook Software, Inc.
May 1, 2003**

The Challenge

Today's software development environments span a vast spectrum of technologies, complexity and size. In every case, the software is transformed from source code written by developers into an application that can be utilized by end-users. This process is typically referred to as a "build process." This build process might be as simple as compiling a standalone application written in a single programming language or as complicated as checking code out of a source control system, generating code and content, building multi-component, multi-language, multi-lingual, multi-platform applications, updating web servers, and more.

While most development platforms provide at least rudimentary support for building applications, even a single-developer shop can find these tools inadequate to meet real-world challenges. The limitations and inefficiencies presented with these tools grow exponentially with the size of the development environment. In addition, most shops of any size use more than one development platform, necessitating the integration of multiple processes and tools, which further complicates the entire process.

Most "home-grown" efforts at build automation consist of batch files, scripts, and small applications typically written in a RAD tool or script language. This can achieve some of the goals of automating a build process. However, it typically requires continual efforts by individuals highly knowledgeable in the intricacies of the appropriate development tools, compilers, source controls tools, etc. In addition, the efforts to maintain this type of build process are hampered by difficult debugging, changes to development tools as they mature and evolve, and substantial efforts to accomplish the logging, error handling, reporting, and other requirements that are common to every attempt at build automation.

An automated approach to the build process is usually created incrementally over time in a reactive way. That is, when a need arises, a quick solution is achieved and the immediate need is met. However, this type of approach rarely provides a reliable, consistent, or robust solution, so continual ongoing maintenance is required. Advanced features (as mentioned above, such as logging, error handling, etc.) are often not implemented, leaving these issues either to manual supervision or worse, are never addressed at all. In all but the most simplistic scenarios, this is at best very inefficient, and can be very costly due to wasted time, effort, and reliability.

The Solution

There is an effective answer to this challenge facing every software developer. It is Visual Build Professional from Kinook Software, Inc. Visual Build Pro is a purpose-built tool that addresses the build automation needs of companies spanning the globe from single-developer Visual Basic shops to large software development shops utilizing the latest tools and techniques. It is compatible with virtually all major development tools and much more. This versatile product will enhance your build automation capability in a very cost-effective way and will pay for itself in a short period of time.

Visual Build Pro accomplishes all of this by integrating with the development tools you use, providing documented, proven, point-and-click automation of those systems, along with flow-control, file manipulation, notification, extensive logging and other vital build-related capabilities. Visual Build provides a central location for all build-related logic, rules, and configuration, along with all the support functionality necessary to keep you in control of every aspect of your software build cycle. It does this in an intuitive way, eliminating the tribal knowledge and experience that is otherwise associated with a typical build process.

Visual Build Professional will:

Save your company time and money. Many hours of manual effort will be eliminated paying for itself in a very short time (often less than one month!).

Improve product quality. Your team's software quality will improve and bugs will be found and repaired more quickly.

Eliminate key-person dependencies. "Tribal knowledge" about the build procedure (often kept in people's heads) will be replaced by a documented build framework.

Create a permanent build record. XML project and log files will become part of your project's source code, providing a persistent record of how builds are performed.

Visual Build Professional Features:

- Custom actions provided for many major development tools
- Integrated with Microsoft Visual Studio.NET and 6.0
- Extensive logging capabilities, including XML log files
- COM automation interface allowing full access to application and build components programmatically
- Full notification features, including email, FTP, etc.
- Scheduling of builds within the application
- Fine-grained debug capability
- Incredible flexibility – almost every feature can be extended with the script engine of your choice

Case Study #1: CFL, LLC

Calls For Less, LLC

South Dakota, USA

- National payphone telephone service provider

At CFL, a single software developer, Ben Robertson, manages a Visual Basic 6.0 code base in use by 25 users. Five applications are being maintained that consist of roughly 35 support components in use at the corporate office. An additional application is used at secondary sites, each of which must be kept up-to-date with all application changes. Ben is also responsible for managing the databases and all other IT resources, including the servers, LAN, email and workstations.

The build process at this location consists of the following steps:

- 1) Checking the code out of Perforce
- 2) Building several large Visual Basic project groups
- 3) Checking the code back into Perforce
- 4) Copying the compiled binaries to the server for production use
- 5) Stopping/starting an IIS server before loading compiled dll files
- 6) Assisting in binary compatibility issues related to Visual Basic 6.0

While these steps can be accomplished manually (and were before Visual Build), the introduction of Visual Build streamlined the build process, providing the following advantages:

- 1) Efficiency – saved approximately 30 minutes per week in build times
- 2) Consistency – no more forgotten steps when the pressure is on
- 3) Reliability – Removed the dependency on the Visual Basic IDE for project group compilation

“Visual Build Pro has paid for itself many times over by saving me time and simplifying the build process. I inherited a significant code base from the previous developer. Having the build automated in Visual Build truly assisted my learning curve and made my life a lot easier! We are even using Visual Build as part of our disaster recovery plan.”

-- Ben Robertson, Senior Developer

Case Study #2: Insurance Technologies, Inc.

Insurance Technologies, Inc.
Colorado, USA

- Point-of-sale management solutions for the financial services sector

Insurance Technologies produces Windows and web-based point-of-sale software for the financial services industry. Customizations of the software are provided for each customer implementation. The software is developed with Microsoft Visual Studio 6.0 and .NET, and many other technologies are utilized, including Microsoft Access, SQL Server, COM+, and DHTML/JavaScript.

Due to the complexity of the software and number of clients, the build process is very complex. Prior to using Visual Build, builds were performed manually, resulting in lengthy build times, and the error-prone process often caused delays in product releases and quality problems with releases.

By utilizing Visual Build Pro, Insurance Technologies was able to automate the entire build process for all of their Windows and web projects, including server deployment of web projects. Build times have been reduced from several hours to under 30 minutes for the average build, giving developers more time to focus on the design and development process. The builds are scheduled to run automatically at night and are ready for testing and deployment the next morning. Steps included in builds are:

- 1) Checking code in and out of SourceSafe and labeling builds
- 2) Building Visual Basic and Visual C++ solutions
- 3) Generating object wrappers from database schema
- 4) Creating Wise installation executables for Windows applications
- 5) Deploying web sites onto IIS servers
- 6) Deploying components into COM+ Services

Beyond the time savings, many other benefits were also realized by Insurance Technologies:

- 1) Visual Build's RAD build development environment greatly reduces the work required to create and maintain build projects.
- 2) The people in charge of builds actually enjoy the process of completing project builds and creating installation packages.
- 3) Because builds are performed more often, testers are in the loop and are able to turn around their work more quickly, leading to improved product quality.

"Visual Build Pro allows our developers to focus on core product development and design, rather than spending hours on each build. By using this product, we are saving time and money while increasing employee productivity."

-- David Fenimore, Senior Vice President, Product Development



Kinook Software, Inc.

<http://www.kinook.com>

+1 (719) 481-4128

P.O. Box 63413

Colorado Springs, CO 80962