Software Management & Distribution: Creating Updates&Patches

Preface to the software updating problem...

Onrush of the modern software development, and also wide Internet availability, have made usual the frequent release of new versions of software programs. Prompt bugs / mistakes correction and feature adding are those major factors which compel manufacturers to modify their software products on a frequent basis. And if the overall size of the distribution package of a software product is quite sizeable, then the necessity to download the full new version distribution package can become an expensive and tiresome procedure for end-users.

Most often, the size of the changes from one version to another is significantly less than the size of the full distribution package, so many developers could thought that it would be a good idea to have an opportunity to implement software updating by delivering only those small changes by size which distinguish the new version from the version already installed on the end-user's machine. Well, an attempt to reduce the update data size as much as possible is praiseworthy. However, how to make it really effective? In case of the text data, everything is clear.

There are a lot of excellent algorithms for text files comparing and the text data can be compressed effectively at that. But if you need to build an update package for executable PE-files (exe, dll) then you will find out that these algorithms are completely useless to make something worthwhile. Effective comparison of executable files or any other binary files requires absolutely other approaches.

Aspects of the software updating problem...

A problem of effective binary files comparing is not the only one in the view of the software update task. The most important aspects which are really worthy of notice are automation of patch building and delivery of the update module to end-users.

Let's assume, that one day the number of your program versions has reached 21. It is obvious that the process of patch building for all these versions and the following publishing of patches on the Internet will be quite complicated even if you have some files/catalogues comparing tool.

Let's have a look what an end-user should perform to update a software product:

- define the installed version number;
- go to your web-site to make sure that a newer version exists;
- probably he would like to read the changes history for the newer version comparing to the version he already has;
- find the required version update in the list;
- download and install the update package.

And moreover, active users will definitely like to receive new versions as soon as possible, having applied minimum efforts to the update process. And this wish is reasonable!

So, the main developer's task is to find the most effective, reliable and reasonably-priced solution - and check if it is suitable for his particular application.

Why use a Patching System and who needs it?

Patching simplifies your product management and makes it easy to manage your software releases. There is no easier way to make professional quality, full-history patches for your software and other electronic content.

Possible audience is software developers, network administrators and IT managers, among others. Regardless of the type of data being distributed - executables, documents, databases, videos, etc. - the patching system can figure out what files have been changed, the exact changes within each file and how to update any previous version to the current one.

Difference between incremental and binary patching...

Incremental update contains all the files which have been changed between two versions. So the result patch can be rather large.

Update modules made with the help of the byte-level patching engine consist only of the changes from within each individual file with the help of the byte-level differencing technology used by patching engine, resulting in a significantly smaller update size.

Let's have a look what a patch maker can offer...

Patch building (the main feature) can offer the efficiency and consequently the small size of the result patch module. Other usefull features are cumulative patching to bring all in-field versions up to date using a single self-extracting executable; selection of the patch-building algorithm efficiency and compression ratio, and also the maximum allowed size of the files to be compared.

Patch installation - the main idea is to create fully customizable modules with a wizard-style runtime interface; use of checksums to ensure accuracy; provide the international language support for patches to be useful for interanational audience of your software product; silent install and silent uninstall with no user interaction required, which can be useful for system administrators and other programs calling the Update Installation program; smart automatic version check to determine the installed version; automatic Rollback at patch applying to guarantee that the end-user's system is successfully updated.

The main players in the market of patch building (which provides possibly the most efficient patching engines) are ***RTPatch*** *by PocketSoft*, and ***PatchFactory*** *by AgenSoft* (www.agensoft.com), which are designed to automate all stages of the update cycle for a software product or any other binary data.

Of course these product are different. This difference concerns flexibility of licensing options, the total cost and consequently the variety of solutions and environment functionality provided in the view of software update creation.

You can observe the efficiency of these products' patching engines and the range of offered services on those vendors' websites. Or just download the evaluation version and start patching your software product!