PatchFactory 3.2

Product reviewed by Craig Murphy

We've all been there: we've spent long hours, days, weeks and months analysing, designing, coding and testing our fine application so that version 1.0 is ready to ship...and perhaps we use an application installer such as InnoSetup or InstallShield to build a polished setup.exe? However, soon after the release of 1.0, your users either find bugs or ask for modifications...so you prepare 1.1. How do you issue updates or patches to your users? Well, in the past I have issued fresh setup.exe files, taking care to ensure that they don't overwrite database files or other critical files. However, the size of these setup.exe files is virtually identical to the original distribution setup.exe (which often exceeds the size of some corporate e-mail inboxes). So what can we do to reduce the size of the 1.1 update? Can we produce a small patch file that moves our application from 1.0 to 1.1?

Enter PatchFactory: it claims to offer "up to 99% file compression". Does it live up to its claim? Read on to find out.

When I reviewed PatchFactory 2.x in Issue 116 of The Delphi Magazine (April 2005), I thought it was one of those applications that I could have made use of years ago. In that review, I noted that PatchFactory 3 was in beta and that a release was imminent. Now that PatchFactory 3.2 is with us, what has changed in this rather useful tool?

What is PatchFactory?

From the PatchFactory web-site:

"PatchFactory provides software developers with a comprehensive solution for creating secure multi-version software update modules, ensuring that end-users always have the latest version of your software or other all-important/critical data. It provides a robust feature set that gives software developers the ability to tailor the update process to their specific needs and to significantly reduce end-user support costs.

Enhanced with features like a powerful byte-level adaptive patching engine incorporating integrated compression technology, helping generate compact patch packages, multi-version patching to bring all in-field versions up to date using a single self-extracting executable, a familiar wizard-style runtime interface, customizable dialogs, MD5 checksums to ensure accuracy, international language support, extensive system editors including registry and shortcut icons, conditions, variables and much more, PatchFactory will most definitely become a valuable asset for both individual software developers and software development companies as an efficient and complete software update solution."

Initial Thoughts

PF3 appears to have enjoyed a ground-up re-write. The user interface now sports an Outlook-style appearance (treeview on the left and a detail pane on the right). When I first ran the application it did look a little sparse. I was slightly surprised to see the standard File, View, Help menu had been "justified" such that the Help menu was right-aligned. At the bottom of the application's main form is an "Output" section - on first glance it looks similar to Delphi's Compiler/Search message area.

And End-to-End Example

I have to admit, it took me a little longer to get going with PF3 than I expected, especially given that I found PF2 very easy and quick to use. What has changed that meant it took me some extra time? Well, PF3 now has a lot of new functionality; I had to work around this new functionality. PF3 now supports a much more complex patching model; this too had to be understood before a simple demo could be created.

As luck would have it, one of my larger applications was due a very small update. The current version of the product is 1.09; my small updates nudge us up to 1.10. The 1.09 .exe is 5231KB in size; the 1.10 .exe is 5232KB in size. Rather than send out the full 1.10 .exe, which would be rejected as being too large by at least one e-mail server, I thought I would put PF3 though its paces.

PF3 works on the premise of Groups and Products. Groups, perhaps obviously, contain Products. Products themselves comprise of Versions. If you take a look at Figure 1, it presents a screenshot of PF with a single Group called Estimating Applications (the user interface is a little less sparse than it was!) Whilst PF3 makes good use of XP-style toolbars, all of the functionality is complimented by menu items or right-click context sensitive menus.

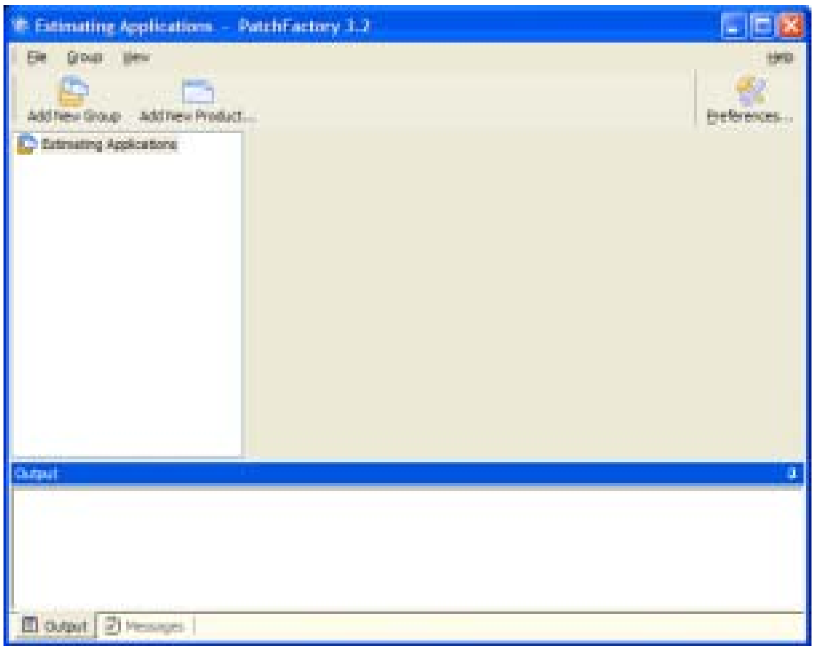
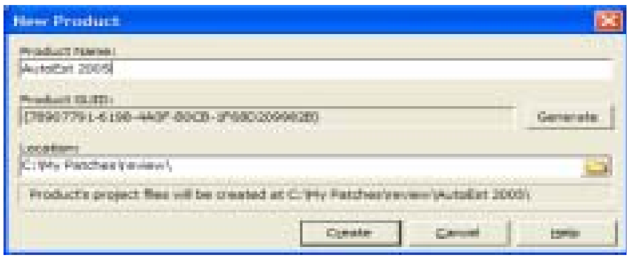


Figure 1 - Adding a New Group

Clicking on the Add New Product toolbar button or right-clicking on the Estimating Applications node on the treeview leads to Figure 2. The most meaningful and useful piece of information that Figure 2 requires is the product name. I am not very sure why we need the Generate button for the Product GUID. After all, when this dialog box first appears it has a Product GUID automatically generated, so why would we need to generate another one?

Useful as it is, the Product's Location field defaults to C:\My Patches. I would prefer this to have picked up the local user account C:\Documents and Settings\CraigM\My Patches for example.



Product Name: AutoEst 2005
Product GUID: [illegible]
Location: C:\My Patches\review\
Product's project files will be created at C:\My Patches\review\AutoEst 2005\

Figure 2 - Adding a New Product

Once a Product has been added, PF3 presents us with a tab control containing product information, as shown in Figure 3. This is the first exposure that we have to PF3's new property-editor – something that many applications appear to be adopting. Generally this kind of control works, however there do appear to be one or two issues relating to the use of the Tab key – on one property-editor using the Tab key doesn't have the desired effect. That said, I am coming around to the property-editor way of doing things, especially in applications that are directed at technical folks/developers (who are used to such editors).

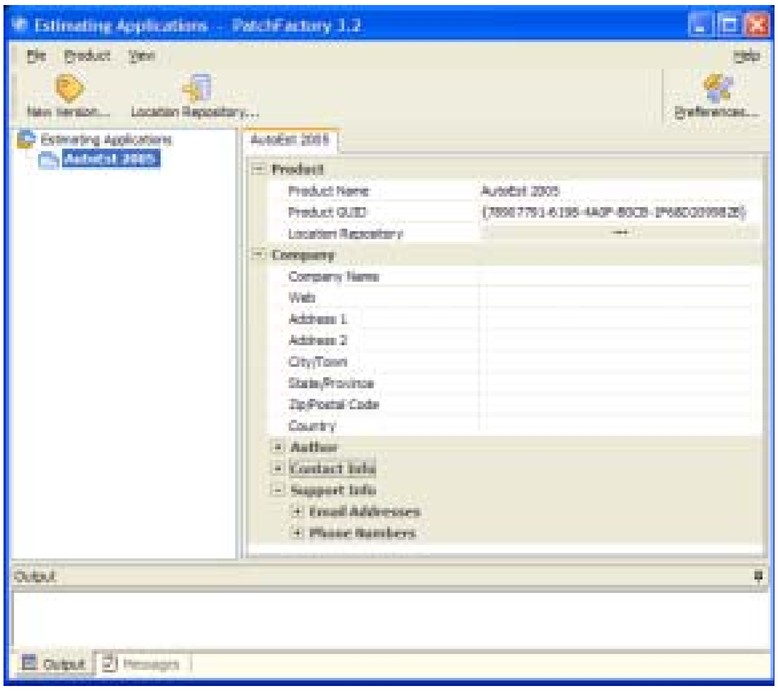
In addition to the Product Name and the Product GUID that we saw earlier, Figure 3 offers us much more. I am not going to explain what the Company, Author, Contact Info and Support Info nodes do: hopefully they should be fairly obvious.

What is less than obvious is the makeup of the Location Repository field. PF3 relies on a previous version of your application being installed on the end-user or client machine. The Location Repository is used to tell PF3 precisely where the original installation actually is. This can be achieved using any of five methods: the Registry, an INI-file, a special folder (e.g. Program Files, Common Files, Windows, System, System Drive, Fonts), an environment variable or a fixed path. My program is installed in C:\Program Files, so I set up an entry for that, specifying the folder name AutoEst2005.

PF3 makes use of the Location Repository to provide you with two elements of protection.

Firstly, it verifies that the original product is installed on the end-user's machine: if it's not there, the patch cannot be applied. Some patching mechanisms involve shipping the update along with the new and current version, therefore folks can install it anywhere they wish. By limiting the patch such that it only works on machines that have your product installed, you can be confident that your application isn't being used in places outside of your control. Similarly, if your application has been altered in anyway, perhaps to circumvent a software protection mechanism, PF3 can detect this alteration and will refuse to apply the patch.

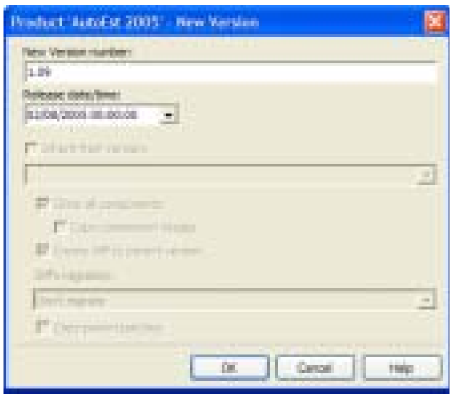
Secondly, it verifies that the previous version exists and can accept this patch. There's nothing worse than a patch that is effectively a retrograde step. PF3 will only patch files if the previous versions match those that you have told PF3 about.



Product
Product Name: AutoEst 2005
Product GUID: [illegible]
Location Repository
Company
Company Name
Web
Address 1
Address 2
City/Town
State/Province
Zip/Postal Code
Country
Author
Contact Info
Support Info
Email Addresses
Phone Numbers

Figure 3 - Product Information

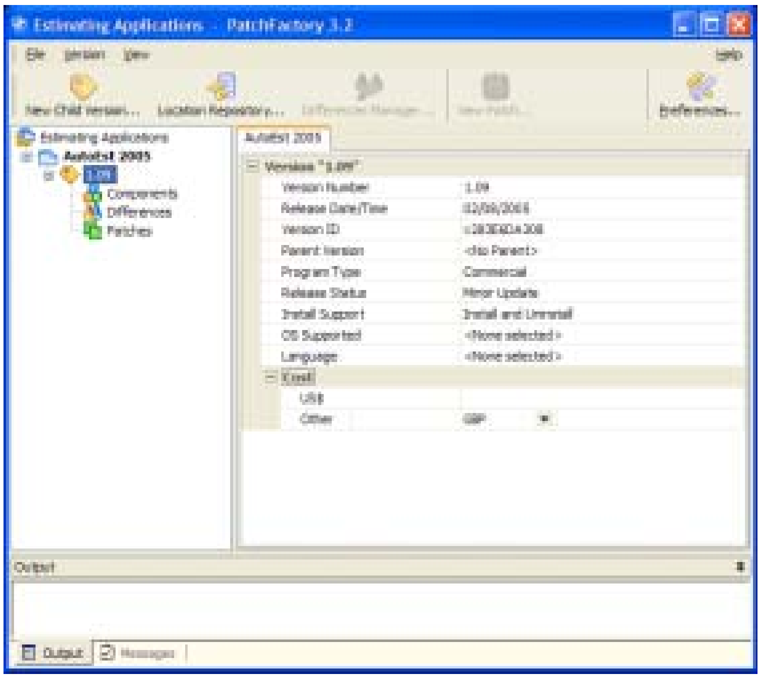
Now that the Product appears in the treeview, we need to add a New Version – again, either by right-clicking on the AutoEst 2005 node or by clicking on the New Version toolbar button. Initially, the "base" version requires us to enter a version number and a release date, as Figure 4 demonstrates. Later, we'll see how to enable the controls that are greyed out.



New Version number: 1.09
Release date/time: 01/08/2005 00:00:00

Figure 4 - Adding a New Version (1.09)

Adding a new version leads us to Figure 5 which presents a lot of information about the version. The Language property relates to spoken language rather than programming language. The OS Supported property is rather comprehensive and includes the obvious flavours of Windows along with a number of exotics like Linux Gnome, OS/2, Win 3.1x, Mac and Palm. The jury's still out on the usefulness of the Cost node.

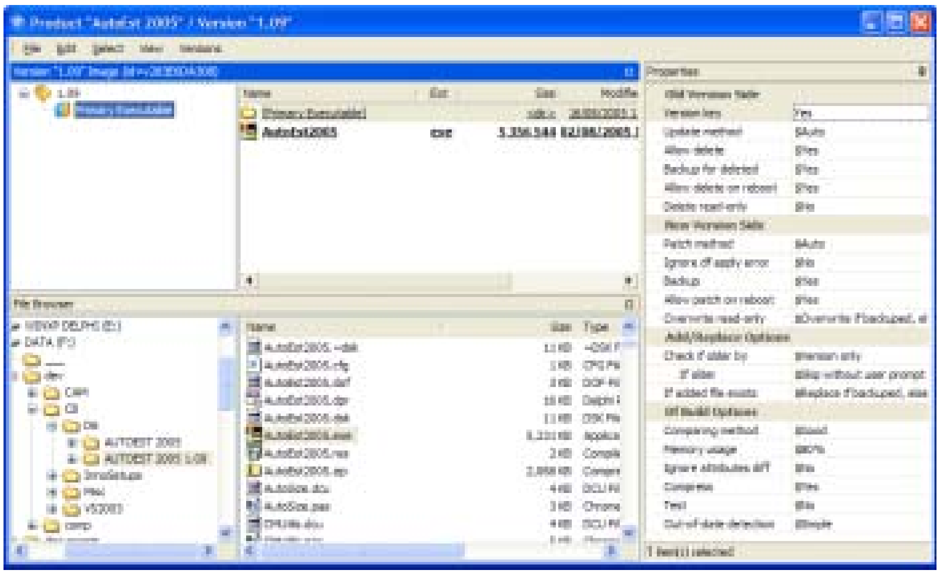

Estimating Applications - PatchFactory 3.2
File Version View Help
New Child Version... Location Repository... Differences Manager... New Patch... Preferences...
Estimating Applications
AutoEst 2005
1.09
Components
Differences
Patches
AutoEst 2005
Output
Output Messages

Version "1.09"	
Version Number	1.09
Release Date/Time	02/09/2005
Version ID	\283E6D4308
Parent Version	<No Parent>
Program Type	Commercial
Release Status	Minor Update
Install Support	Install and Uninstall
OS Supported	<None selected>
Language	<None selected>
Cost	
US$	
Other	GBP

Figure 5 - New Version information

A product version then has one or more Components added to it, in this case by right-clicking and choosing Add Component(s). I named this new component “Primary Executable”. Skipping a little bit of PF3 detail, each component should have one or more files associated with it. In this case, referring to Figure 6, I have told PF3 about the file AutoEst2005.exe.

Once again, a property-editor is used to specify a plethora of PF3 configuration properties – this is the form where the Tab key is a little erratic! That’s not a major problem as we need only adjust one property in order for our simple example to work: Version key is set to Yes.

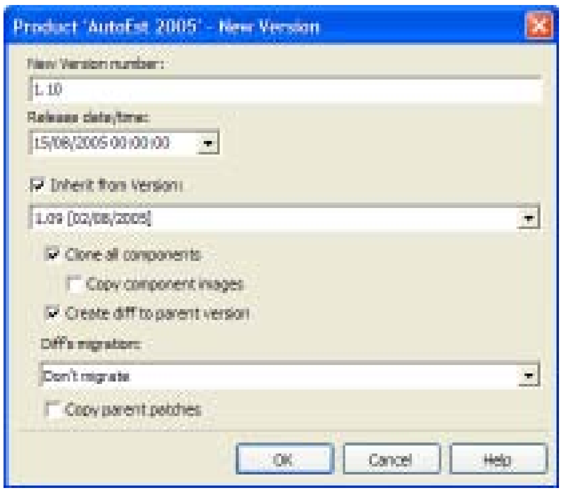

Product "AutoEst 2005" / Version "1.09"
File Edit Select View Versions
Version "1.09" Image Id=\283E6D4308
1.09
Primary Executable
File Browser
Properties
1 item(s) selected

Name	Ext	Size	Modified
[Primary Executable]		<dir>	28/08/2005 1
AutoEst2005	exe	5,356,544	02/08/2005 1

Properties	
Old Version Side	
Version key	Yes
Update method	$Auto
Allow delete	$Yes
Backup for deleted	$Yes
Allow delete on reboot	$Yes
Delete read-only	$No
New Version Side	
Patch method	$Auto
Ignore of apply error	$No
Backup	$Yes
Allow patch on reboot	$Yes
Overwrite read-only	$Overwrite if backuped, el
Add/Replace Options	
Check if older by	$Version only
If older	$Skip without user prompt
If added file exists	$Replace if backuped, else
Diff Build Options	
Comparing method	$Good
Memory usage	$80%
Ignore attributes diff	$No
Compress	$Yes
Test	$No
Out-of-date detection	$Simple

Figure 6 - Adding Files to a New Component

Now that version 1.09 has been specified, let’s create a new version: 1.10. This can be achieved by right-clicking on the AutoEst 2005 product node and choosing New Version. Figure 7 reveals the previously disabled controls that we saw in Figure 4. Given that version 1.10 is a patch from 1.09, it makes sense that it should inherit information from a previous version, in this case 1.09.



Product 'AutoEst 2005' - New Version

New Version number: 1.10

Release date/time: 15/08/2005 00:00:00

[✓] Inherit from Version: 1.09 [02/08/2005]

[✓] Clone all components

[ ] Copy component images

[✓] Create diff to parent version

Diff's migration: Don't migrate

[ ] Copy parent patches

Figure 7 - Adding a New Version (1.10)

With two versions, and two different executables, PF3 is now able to create a patch file for us. In order to build a patch that will take a version 1.09 executable up to version 1.10, we need to right-click on the Patches node of version 1.10. This reveals Figure 8, New Patch. I've called the new patch "to 1.10" and left it as a version-to-version patch, which is the default.

PF3 supports two types of patch: v2v and Cumulative.

v2v: Building this type of patch leads to the creation of independent patch modules that can be used to perform an update from one particular version to another. The contents of any old versions being updated using this patch are defined by differences which exist in the new version.

Cumulative: This patch type is used to generate an update module capable of performing updates from several versions up to one another version. All necessary actions are taken to minimize the size of this patch module during the building process. The contents of any old versions being updated using this patch are defined by differences which exist in the new version.

Figure 7a presents a graphical overview of these two methods.

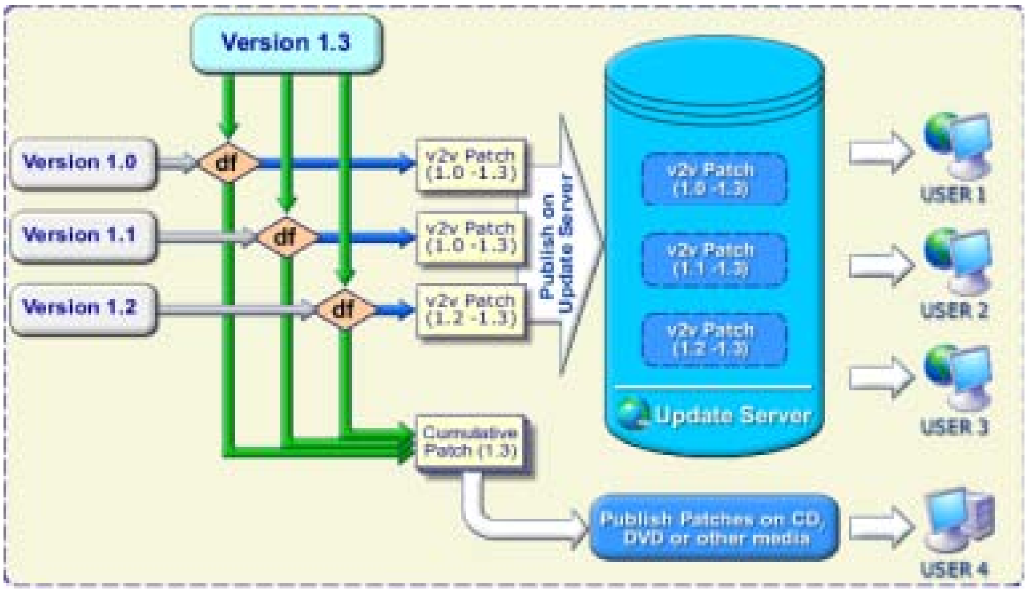
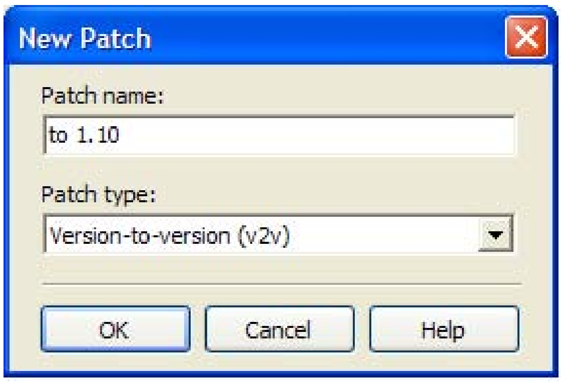


Figure 7a – PatchFactory's patching methodology

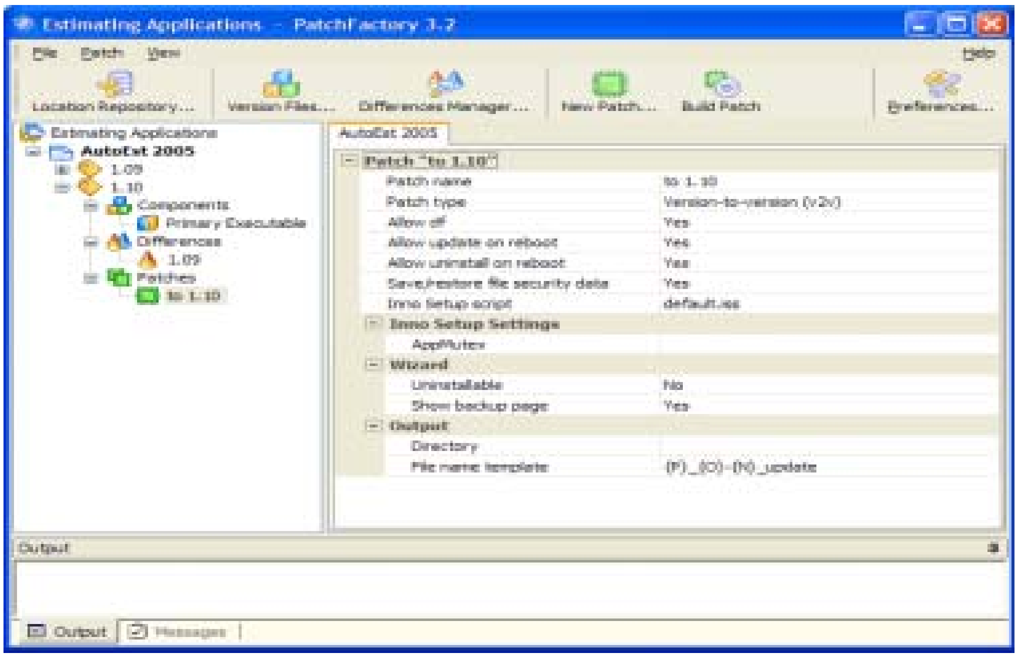


Patch name: to 1.10

Patch type: Version-to-version (v2v)

Figure 8 - Adding a New Patch

Figure 9 presents information about the to 1.10 patch, including a handful of properties that most of us are probably very used to seeing, e.g. allow update on reboot (files are replaced on restart).



Patch "to 1.10"	
Patch name	to 1.10
Patch type	Version-to-version (v2v)
Allow diff	Yes
Allow update on reboot	Yes
Allow uninstall on reboot	Yes
Save/restore file security data	Yes
Inno Setup script	default.iss
Inno Setup Settings	
AppMutex	
Wizard	
Uninstallable	No
Show backup page	Yes
Output	
Directory	
File name template	{P}_{O}-{N}_update

Figure 9 - New Patch Information

By default PF3 creates a patch with a filename AutoEst 2005_1.10-1.09_update.exe, or {P}_{N}_{O}_update in PF3 terminology (more about this in a moment). I would prefer to see patches named as AutoEst 2005_1.09-1.10_update.exe, i.e {P}_{O}_{N}_update. Luckily Figure 9 gives us that chance.

Patches use replaceable parameters delimited by { and }. Here's the full list:

{P} : Product Name.

{U} : Patch Name.

{O} : Old version name (empty string for Cumulative patch).

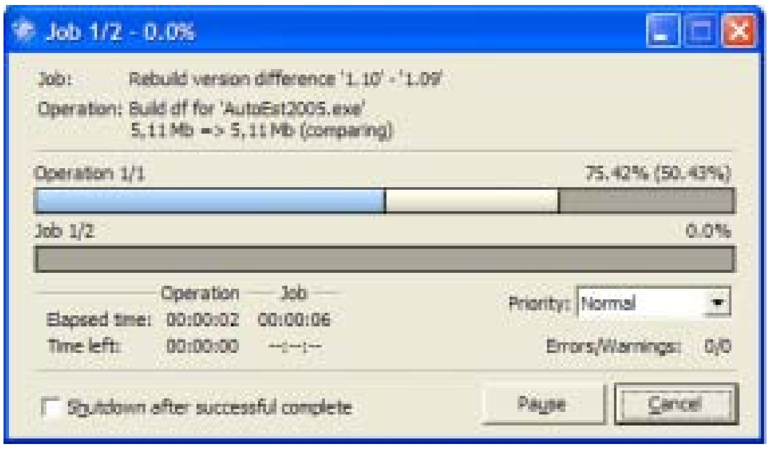
{N} : New version name.

{PID} : Product ID.

{OID} : Old version ID (empty string for Cumulative patch).

{NID} : New version ID.

To complete the patch building exercise, we can click on Figure 9's Build Patch toolbar button. Figure 10 presents the "progress dialog" – the whole patch creation process took a mere 11 seconds…


Job 1/2 - 0.0%

Job: Rebuild version difference '1.10' - '1.09'
Operation: Build df for 'AutoEst2005.exe'
5,11 Mb => 5,11 Mb (comparing)

Operation 1/1 75.42% (50.43%)

Job 1/2 0.0%

	Operation	Job
Elapsed time:	00:00:02	00:00:06
Time left:	00:00:00	--:--:--

Priority: Normal

Errors/Warnings: 0/0

Figure 10 – Building the Patch

… and created the file shown in Figure 11: all 430KB of it.

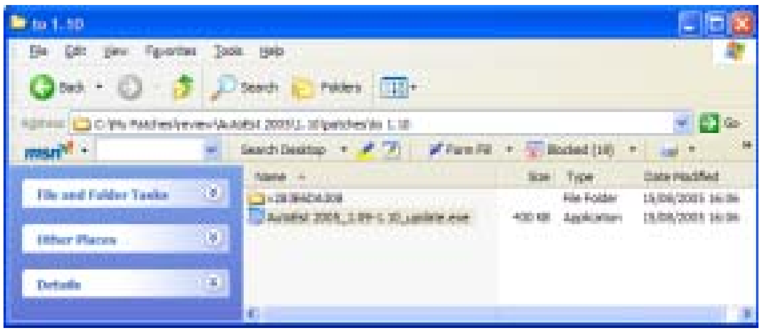


Figure 11 - The Patch - all 430KB of it

Building The Patch

In my PF2 review, I noted that PF2 could create a patch .exe weighing in at 40k. PF3 now integrates [seamlessly] with InnoSetup 5 (review in July / August 2004 issue of this fine publication) and as such now creates patches that are around 400k in size. This is still a remarkable saving over the 5232KB mentioned earlier, and the patch has a pleasant graphical user interface that can be tweaked, customised and modified to suit your needs.

During the patch building process, the Output window at the bottom of the PF3 user interface created Listing 1 (where we can see yet more evidence of the InnoSetup integration). Use of InnoSetup provides us with fine-grained control over the patching process – it has PascalScript built into it, so we essentially gain programmatic control over the patch itself.

```
Start batch processing (2 job(s)), [15/08/2005 16:05:52]

Product “AutoEst 2005”, version “1.10” (id=v5B9A461E08)
================================================================================
Compile diff: ‘1.09’ -> ‘1.10’
OK

Job 1/2: Build version difference ‘1.10’ - ‘1.09’

Build df for ‘AutoEst2005.exe’
5,11 Mb => 5,11 Mb
Max. virtual memory usage: 409Mb (80.0%)
Old md5: 37c4bc9003a8a1f88533e0735cbf753e
New md5: bc067140959f3aa56ddd65cab7577651
Df size: 83464 bytes
Reduction: 98.442%
Compression ratio: 2.811%
Build time: 00:00:07.870
```

```
— Diff ‘1.09’=>’1.10' building report —
Old version ‘1.09’: 5,356,544 bytes in 1 file(s), in 0 dir(s)
New version ‘1.10’: 5,357,568 bytes in 1 file(s), in 0 dir(s)
Added+Replaced: 0 (0) bytes in 0 file(s)
Identical: 0 bytes in 0 file(s)
Identical by content: 0 bytes in 0 file(s)
Df content: 83,464 bytes in 1 file(s)
Reduction (abs/rel): 5,274,104 bytes (98.4%/98.4%)
————————————————————————————

Job 2/2: Build patch ‘to 1.10’
Compile Inno Setup script: default.iss
Inno Setup 5 Command-Line Compiler
Copyright (C) 1997-2005 Jordan Russell. All rights reserved.
Portions by Martijn Laan

Compiler engine version: Inno Setup 5.0.8 (ISPP 5.0.6.0)

[ISPP] Preprocessing.


[ISPP] Preprocessed.

Parsing [Setup] section, line 1074 [snip]
Parsing [Setup] section, line 1166
Reading file (WizardImageFile)
   File: C:\Program Files\PatchFactory3\Scripts\Setup.bmp
Reading file (WizardSmallImageFile)
   File: C:\Program Files\PatchFactory3\Scripts\SetupSmall.bmp
Reading default messages from Default.isl
Parsing [Languages] section, line 1169
   File: C:\Program Files\PatchFactory3\Scripts\Languages\Default.isl
Parsing [Languages] section, line 1170
   File: C:\Program Files\PatchFactory3\Scripts\Languages\Russian.isl
Parsing [Languages] section, line 1171
   File: C:\Program Files\PatchFactory3\Scripts\Languages\German.isl
Parsing [LangOptions], [Messages], and [CustomMessages] sections
   Messages in script file
Reading [Code] section
Parsing [Files] section, line 1180 [snip]
Parsing [Files] section, line 1193
Compiling [Code] section
Creating setup files
   Updating icons (SETUP.EXE)
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to
            1.10\v283E6DA308\oldvers.lst
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to 1.10\..\..\..\locrepos
   Compressing: C:\Program Files\PatchFactory3\dfpatch.dll
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to
            1.10\..\..\..\1.10\v5B9A461E08.v
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to
            1.10\..\..\..\1.09\v283E6DA308.v
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to
            1.10\v283E6DA308\v283E6DA308.lst
   Compressing: C:\My Patches\review\AutoEst 2005\1.10\patches\to
            1.10\..\..\diff\v283E6DA308\Primary Executable\AutoEst2005.exe
   Compressing: C:\Program Files\PatchFactory3\Scripts\null
   Compressing Setup stub
   Updating icons (SETUP.E32)
   Updating version info


Successful compile (3.295 sec). Resulting Setup program filename is:
C:\My Patches\review\AutoEst 2005\1.10\patches\to 1.10\AutoEst 2005_1.09-1.10_update.exe
================================================================================
    Total time: 00:00:11.991 [15/08/2005 16:06:04]

    Result: 0 - error(s), 0 - warning(s)
```

Listing 1 – Building the patch

Pricing

PF3 costs €345 for a "Shareware" license, €695 for a "Commercial" license or €2495 for a "Corporate" license. Upgrades from PF1 or PF2 cost either €172.50 for the Shareware license or €347.50 for the Commercial license. From the AgenSoft web-site:

"The discounted Shareware License is offered as a service to the industry, primarily for single person companies with little revenue (such as shareware authors). The software is licensed to the name of the individual purchasing the license.

The standard Commercial License is intended for small companies. The software is licensed to the name of the company purchasing the license. Maximum number of employees using the software is limited to 5. It also entitles an organization to receive high-priority support (with guaranteed answer within 2 business days) via email. If you would like to obtain more extended services or more employees to use the software - consider buying the Corporate license.

The premium Corporate License is intended for large companies and corporates with many employees. The software is licensed to the name of the company purchasing the license. And besides this type of license entitles an organization to receive one copy of the distribution software and to duplicate the software for any number of people or workstations within the corporation. It also entitles an organization to receive high-priority support (with guaranteed answer within 1 business day) via email and free major version upgrades during lifetime of the product."

PF3's authors have targeted the pricing at an appropriate level – it seems that they have listened to public opinion surrounding pricing structures, which seem to be rather emotive. So if you're a small company, you pay less than larger corporates…seems fair.

Conclusions

I thought that PF2 was an excellent tool. PF3 is equally good and does have some features that differentiate it from PF2. The ground-up re-write and the user interface make PF3 a pleasing application to use and it's fast too. With the exception of the small niggles mentioned over the course of this review, I struggle to find anything to complain about. Whilst PF3 is more expensive that PF2, it is still an economical product that could ultimately protect your investment (via reduced software piracy, the patch is useless on its own).

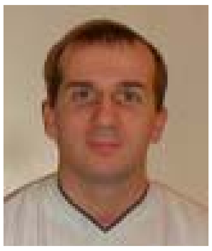
Whilst PF3 inherits the benefits of InnoSetup (which can be included in automated build scripts), PF3 itself does not appear to offer a means of integrating it with build scripts and build tools such as FinalBuilder (another fine piece of software). Having said that, would I expect patch building to be part of the build process? Part of me says yes I would, yet another part tells me it's a separate job that can be done prior to shipping. I'm a man of two parts.

The HTML Help file weighs in at 315KB and brings with it full coverage of PF3. In contrast to PF2, the help is much more detailed, offering scenario-based walk-throughs and wizard-like step-by-step approaches. Given that the author of the help file doesn't speak English as a first language, there are a few oddities to work around (but, since my command of a second language is virtually non-existent, the aforementioned author should be complimented on his multi-lingual skills!)

I'm pleased to recommend PF3. It's a solid product that is robust in use and does what it says it will do, and it does it well. If you are looking for an application patching solution, do consider PF3.

Resources

More information about PatchFactory, including a trial version, can be found here: http://www.agensoft.com/



Craig is an author, blogger, community evangelist, developer, speaker, Certified ScrumMaster and Microsoft MVP (XML Web Services). He specialises in all things XML, particularly SOAP and XSLT. Craig is evangelical about .NET, C#, Test-Driven Development, Extreme Programming, agile methods and Scrum. He can be reached via e-mail at: bug@craigmurphy.com, or via his web site: http://www.craigmurphy.com (where you can also find the source code and PowerPoint files for all of Craig's articles, reviews and presentations).

Craig's blog can be found here: http://www.craigmurphy.com/blog