Code Co-op

product review by Pat Bell

For years, in my cave in darkest mid-Wales, I have muddled through version control between my main development machine and my mobile laptop installation. Various batch files copy files by date back and forth across my LAN, but only by maintaining both machines as virtual clones of each other. I was the master of the batch file, except for the few times I fell foul of my 'system' and lost several days' edits. Then along came XP and 'security' – now my 'version control' was a two-stage process run on each computer. Still just about manageable for my three or four current projects. But then along comes an assistant developer. "Discipline will prevail" methinks… perhaps not, I think my haphazard methods will not easily work with a collaborator.

So the hunt is on. MS Visual SourceSafe and SourceSafe Offsite Collaborative were recommended but, though well-established, these seem to be somewhat expensive and draconian products ruling like a dragon-lady librarian who won't let me take out 'fmdm1' because my library card is full, or perhaps I have outstanding fines. There are several alternative products out there but this is not a comparative review. I wanted advice about which software to use, Code Co-op was one I had been considering, and when my pleas for advice coincided with a free offer of Code Co-op in return for a review, who was I to argue with destiny.

This, then, is a review of how Code Co-op came into my Delphi programming existence; how it applies to my specific requirements, which are for two Delphi developers to work on the same projects at home and at HQ. If you're a member of a 20 strong, military grade team of Delphi wizards then turn the page, that is unless you want to see how the other 95% of us ordinary mortals muddle through.

First Impressions

Downloading and installing Code Co-op 3.5 was pretty straightforward. The 2Mb file was easily extracted and installed, giving me the option of where to install it and in what program group, options I have come to expect for all Delphi add-ons. Immediately following installation, I was taken automatically (optionally) to the html tutorial. Like many users, I probably didn't read the tutorial and help files as thoroughly as I should have. Just like many developers. Reliable Software knew this and, after several tactful attempts to suggest this as the cause of some of my problems, they acknowledged some 'idiosyncrasies' as I struggled through my introduction to Code Co-op.

At one point I found myself with a disturbing 6.5Gb (yes that's gigabytes!) of messages accumulating in Code Co-op's public inbox – we finally discovered this to be a 'deadly embrace' where I had some non-existent users in a project. Some of my problems were down to my own impatience. But I get ahead of myself: version 3.5 eventually worked but, having encountered some problems, I thought it would be better to review the beta of version 4 which included some fixes resulting directly from my feedback.

Code Co-op has two main components: the project manager, 'Code Co-op itself, and what they call the Despatcher. The Despatcher runs, mostly, in the background, and is the agent for communicating with other developers using Code Co-op either by email or on a LAN.

Initial configuration has been made clear and simple in version 4, though being a user I did manage to find a way of failing to fully configure it by trying to be clever – all in the interest of beta testing, of course <vbg>. However, if you start up the manager (Code Co-op) and try to create a new project or join one before configuration you get

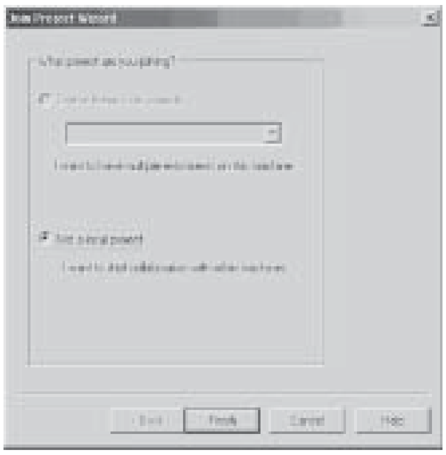
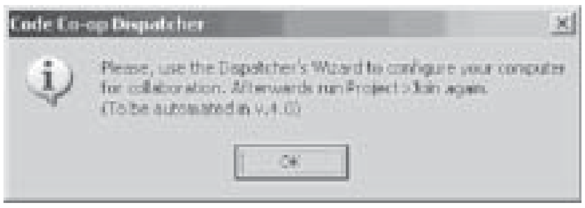


Figure 1

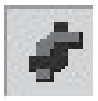
then



Please, use the Dispatcher's Wizard to configure your computer for collaboration. Afterwards run Project > Join again.
(To be automated in v.4.0)

Figure 2

Not automated yet … but right clicking the Despatcher icon



(Figure 3) in the systray and choosing the Collaboration Wizard - which takes you through the steps using prompt and nice diagrams - makes it almost impossible to go wrong.

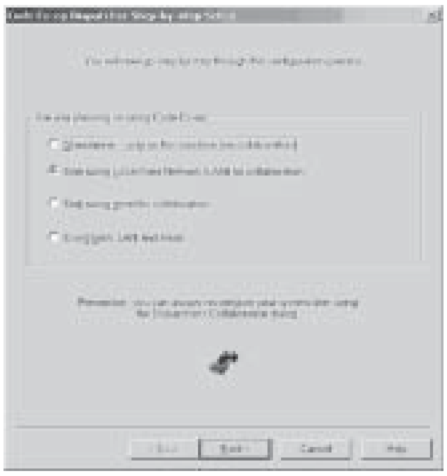


Figure 4

As this dialogue suggests, Code Co-op is for use on a LAN and/or using email. The standalone option is really just for getting to know the program and has no other useful purpose.

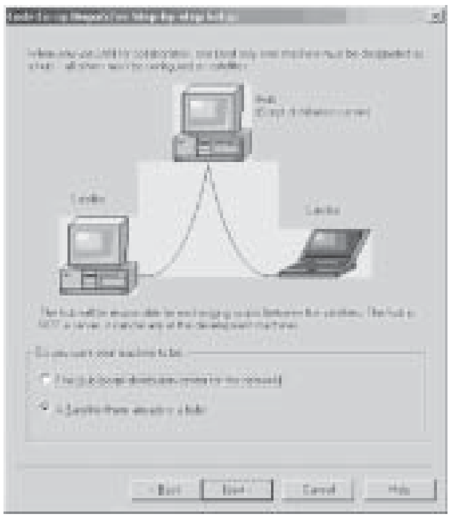


Figure 5

A LAN configuration has to have a single machine (designated as the hub) that distributes the projects and their changes (known as scripts). My own configuration necessitates both email and LAN, and the hub needs to be the machine that receives scripts by email. There are options to forward emailed scripts but I decided that, already having the most complicated configuration, to avoid making things any more so.

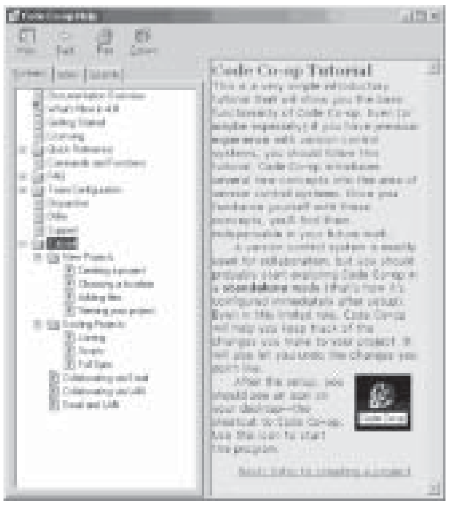


Figure 6

The html help (Figure 6) includes a tutorial, which advises starting with the standalone mode. I believe I caused myself problems with tutorial 'residue', so you might like to run through the tutorial first then uninstall and reinstall. However, the configuration wizard is pretty straightforward so you might like to plunge straight in and follow the tutorial as far as it applies to your own situation.

A quick preliminary glance through the FAQs showed me an issue about Outlook Express security warnings and how to turn them off. I have encountered this issue before when making use of Simple MAPI. OE6 requires an acknowledgement for mail not sent manually. You can't turn this off in Outlook XP, though and you need a third party applet (of which there are several: I found one called ClickYes which works fine).

Adding an existing project is pretty straightforward. Project— >New gets you Figure 7.

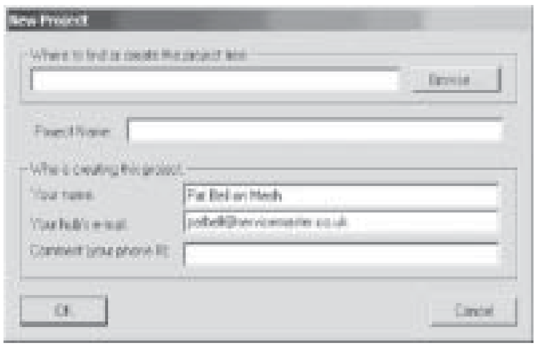


Figure 7

It would have been nice to have had at least a guess made at the project folder, perhaps the Delphi default, but it's easy enough to find with the [Browse] provided. I tried to be smart and typed the path in but missed a folder and so it was expecting me to create a project from scratch, in other words the location of a project yet to be created in Delphi. To correct my mistake, I needed to delete the project from Code Co-op by defecting and being the last member of the project. A rather nice idea, to be a defector and the last one turns the lights out – a feeling of being part of a conspiracy instead of a disobedient schoolboy having lost his library card.

Having located an existing project, the folder is scanned and the extensions found are presented for inclusion (or not) in the files to be managed by Code Coop.

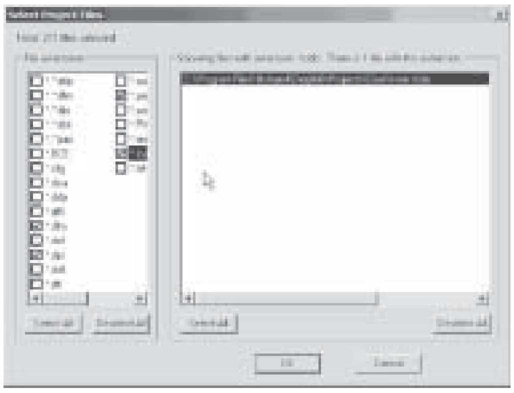


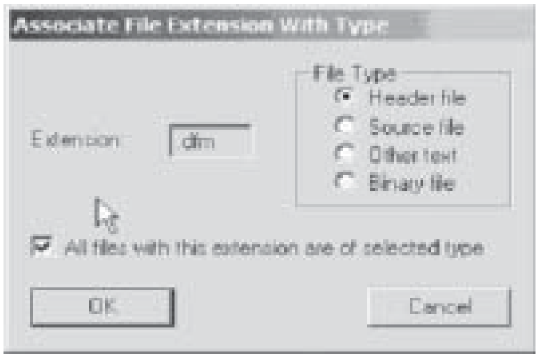
Figure 8

If, like me, you have a lot of other temporary old backup files and dcus, you need to be careful at this stage, since all the files contained in your project folder and sub-folders are initially included in the list. I blundered my way through these setup screens and went ahead and included a whole batch of files I will probably not want to control. I made a few mistakes here first of all and had projects that generated errors on start-up because (I think) I had missed out project system files whilst leaving DCU unchecked. I then checked ddp, dof, cfg, pas and res, leaving all the others including dti and dsk unchecked (who would want to sync their screen layout?).

I wish it were more Delphi-orientated and could use the dpr to identify its constituent units rather than just the content of the folder, but again this is just a niggle not a serious issue..

Anyway [OK] here then takes you through your selected file extensions to identify their type to Code Co-op (Figure 9). Apparently the most important distinction is that between **binary** and other types, something to do with the text comparison that identifies changes in the files. This screen betrays the non-Delphi origin of this product since it refers to header and source. Anyway, I guess dfm, dpr and pas files are source files (well, dfm can be as long as they are saved as text). In fact, this app knows nothing about Delphi at all. It might be useful, for instance, to be able to read the dpr for all files included in a project, or to understand the different kinds of files (perhaps even to recognise if a dfm has been created in text mode or not), also to recognise your Delphi project folder. Useful?. perhaps, crucial?… no.

Anyway, I thought it would be a good idea to convert my dfms to text and …Delphi\bin\convert.exe did that for me.


Associate File Extension With Type

Extension: dfm

File Type
- (●) Header file
- ( ) Source file
- ( ) Other text
- ( ) Binary file

[✓] All files with this extension are of selected type

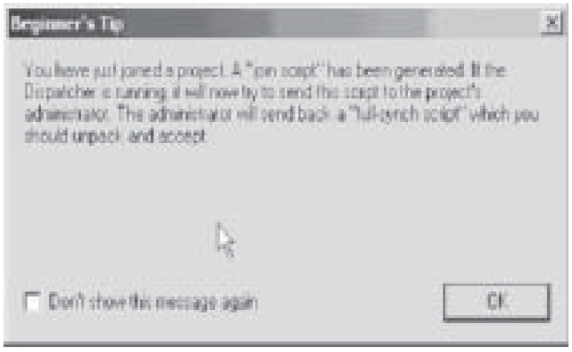
Figure 9

After responding to this prompt for each extension selected in the Project File selection window (Figure 9), Code Co-op then processes the files whose extensions were selected.

Joining a Project

Project | Join gets you a similar dialogue to Figure 9 except with entries for 'where to copy the source tree' and your email address.

Having done this you are shown a Beginner's Tip (Figure 10).


Beginner's Tip

You have just joined a project. A "join script" has been generated. If the Dispatcher is running, it will now try to send this script to the project's administrator. The administrator will send back a "full-synch script" which you should unpack and accept.

[ ] Don't show this message again

Figure 10

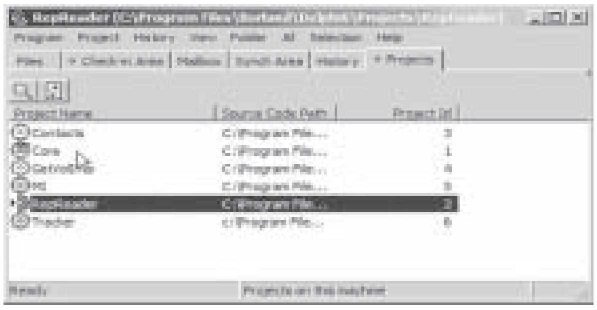
This refers to the Dispatcher which is in the system tray and whose icon changes colour to indicate that it has a script to take care of. These tips often show up with helpful hints.

Whilst it *is* possible to edit files directly with the Code Co-op text editor, obviously the best thing to do is to check out the ones you wish to work on and then use your Delphi IDE. Those not checked out are read-only. After some work you can come back to Code Co-op and check all or some files back in. On my first pass at this, I quickly found some obvious things I had forgotten – check out the project dpr, plus check out the pas and the corresponding dfm files. Using the Delphi IDE 'new forms as text' would seem a good idea, especially having identified the dfms as source, otherwise binary might seem a better definition. Essentially all text files can be source and anything else as binary (such as dpp, res or even third party reports such as raf (Report Builder reports)).

Code Co-op In Action

Having experimented, I made mistakes. The tutorial guides you through the new and the joining process in an example on one machine – don't be misled by this – you start a new project and check out on that same machine: it is other machines that need to join. Also, be very careful about the name of the project you are requesting to join. The wrong one and it is possible to get the despatcher in a loop (or it was in version 3.5; as far as I could tell this loophole has been plugged in version 4).

Of the two Code Co-op components, the despatcher is the one you have least to do with. It runs in the background, just occasionally changing colour when it has a script to process. it is Code Co-op app itself where the work is done.

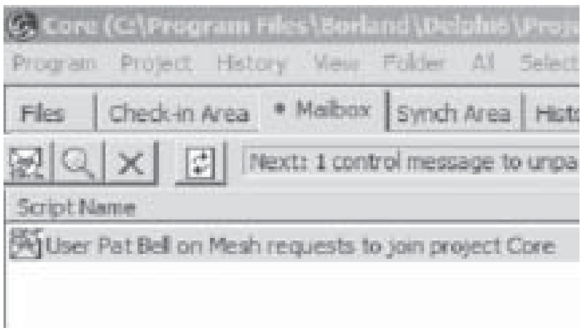

RepReader (C:\Program Files\Borland\Delphi6\Projects\RepReader)

Project Name	Source Code Path	Project Id
Contacts	C:/Program File...	3
Core	C:/Program File...	1
GetVolInfo	C:/Program File...	4
MS	C:/Program File...	5
RepReader	C:/Program File...	2
Tracker	c:/Program File...	6

Ready | Projects on this machine

Figure 11

Each tab shows details for the current project only (which shows as a half cog in Figure 11). How do you know a script is waiting to be processed? Well, apart from the rather subtle change in the despatcher icon, the project icon itself changes as you can see in figure 13.


Core (C:\Program Files\Borland\Delphi6\Proje

Next: 1 control message to unpa

Script Name
User Pat Bell on Mesh requests to join project Core

Figure 12

The button showing an envelope tipping out its contents will process the script. The result of which is shown in the check-in area (Figure 13).

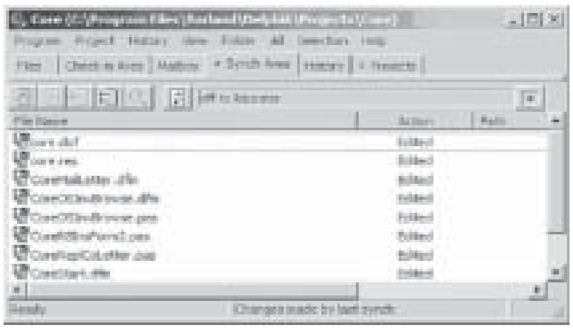


Figure 13

. The first of the buttons showing in Figure 13will accept all the files' changes, the second just the selected ones. Clicking on one of the flagged files shows the diff file. Reliable Software cite the differential display tool as one of the most powerful elements of Code Co-op. I have to say I haven't really looked at this very deeply, nor have I anything to compare it with but it seems fairly straightforward and does the job.

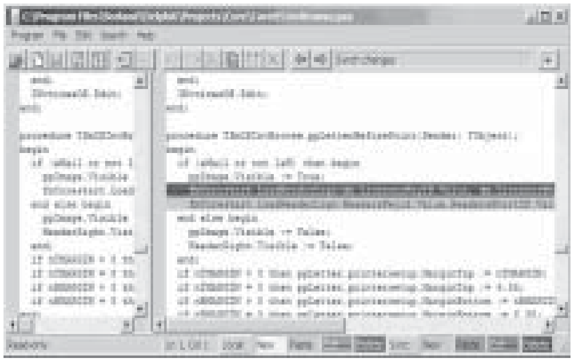


Figure 14

The differential display highlights the lines that were involved in a change.

Your edits are highlighted in shades of yellow and green. An added line is bright yellow, a deleted line is dark green with the text crossed out. Lines that have been moved are shown in more subdued colours. When a chunk of text has been modified, it appears as a combination of deleted and added lines. Sync changes are similarly highlighted in shades of purple.

To jump from one change to the next, click on the forward blue arrow button, or press F4. To jump to previous change, press the backward blue arrow, or Shift-F4. These functions are also available through the *Search* menu.

During a merge, the Differ distinguishes between the edits you've made before unpacking the sync and the ones you've made later. The post-sync additions are highlighted in yellow in the left pane. They **remain highlighted** across different sessions (or *saves*), to help you keep track of your merge progress.

This particular script was the result of my checking in files from another machine on my LAN, prior to my disconnecting my laptop and venturing into civilisation with the results of my retreat (i.e. back to head office for a day). It could equally well have been Alan (the new colleague who triggered all this off) checking in his changes from where he is based - in Portsmouth. Whenever files are checked in, Code Coop asks for a comment, which then can show in the history of that particular project. Figure 15 shows the history of one project.

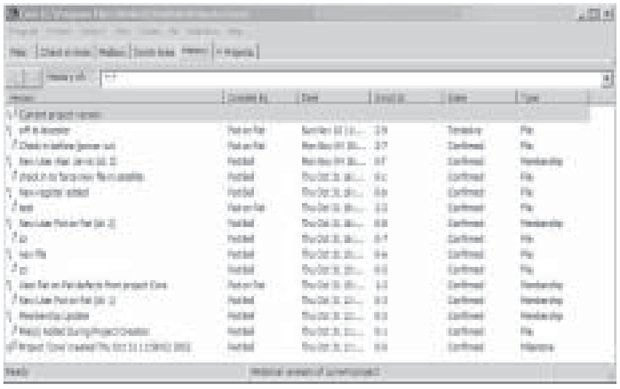


Figure 15

As you can see, some of my comments are not too meaningful – I hope to improve them as I make more use of Code Co-op. The path that a project takes is depicted by the footprint icons. Each script has a state and type. Until a certain script has been confirmed, it is regarded as tentative state. The script I just processed is still tentative because the hub is waiting to acknowledge receipt of the checked in files and also for Alan to acknowledge his receipt of my changes. Emailed scripts can sometimes get lost, so it is useful to be able to resend a script from this screen.

Conclusions

I feel I have only scratched the surface in this review. There is a lot more that can be done to configure and use (or abuse probably too) Code Co-op, and generally make better use than my cursory exploration here. So, do I like Code Co-op? Yes. Will it help me in my special situation? Definitely… it already has. I wonder how I did without it now, even taking into account just my own practice of working with two machines. Does it give me a warm feeling of security? More or less: though I did have some start-up problems, they were more to do with my experimentation and impatience than with any inherent unreliability in the program itself. I now collaborate simply and safely with my colleague and easily pack up my laptop and drive off into the sunset, knowing any program changes I make whilst away can be easily synchronised with whatever other versions of my projects there may be either at home or with my colleague.

Special BUG offer

Code Co-op 3.5 sells at $145 per seat up to 10, $125 each for more than 10. At the time of going to print, we are awaiting confirmation of the price for v4 and the url from which you can buy it. During January and February, it is on offer exclusively to BUG members at a 40% discount. Please contact the BUG office if you're interested.

Pat Bell is a Delphi systems developer, and has been since Delphi 1, latterly freelance but now in waged bondage, having sold his business to his main client. The bondage, however, is pretty loose and he is allowed to live in exile in mid-Wales, subject to monthly treks to Leicester. His home office is in one of the most under-populated areas of Britain this side of the Scottish borders. You can contact Pat on pat@patbell.co.uk.