HarPB Tool Review

BY AL SOUCY

I am currently the administrator of AllFusion Harvest. I manage 100 applications housed in AllFusion Harvest and support the 180 developers using it. The development tools we currently use are PowerBuilder PBV8 and PBV11, Visual Studio 2005, Java and Visual Basic.

As software configuration manager, I provide the administration of the source code management tools. This includes the entire infrastructure of the environment for development, from developing the lifecycles, to providing the best practices, procedures, processes, and training of all the developers on proper source code management with the development tools in our environment.

In 1999, Computer Associates acquired Platinum Products and Phil Wallingford from E. Crane Computing was introduced to the state of New Hampshire. Phil has been an integral part of the state's development team for many years now. We currently use two of his products: PowerGen and HarPB. We use Powergen to do all builds in-house for PowerBuilder development and we use HarPB, which is the interface to Harvest for all PowerBuilder check-outs and check-ins. We have a fully automated process based on PowerGen that operates flawlessly.

Below are definitions of HarPB, a description of the graphical user interface, distinctive features, advantages, an example of HarPB, and my synopsis of the product along with my review. I apologize if there are too many figures for some folks. I find visuals provide clarity and understanding of products unlike language. I hope you find them helpful with respect to HarPB. It is the superman of robust interfaces for PowerBuilder.

HarPB: A Definition

HarPB is a specialized utility for checking PowerBuilder source objects in and out of AllFusion Harvest. It handles the special requirements of checking objects out to PowerBuilder Libraries (PBLs) and checking objects in from PBLs. These operations are non-standard to most source control systems, because the PBL is a proprietary binary format of PowerBuilder.

Due to the PowerBuilder Library structure we use PowerGen to do a mass export of all the PowerBuilder objects as flat files and store those in a repository in Harvest. We then use HarPB to do all the check-outs and check-ins from Harvest by using the import/export activity from within HarPB. The HarPB user interface, its graphical user interface (GUI), is designed for ease-of-use and minimal training time. Standardized icons and toolbars are used throughout the interface.

HarPB is a very easy install. It takes just a matter of minutes and is virtually maintenance-free. Its support is stellar as well and E. Crane Computing is exceptional when it comes to managing any deficiencies, modifications, or enhancements against the product. Once the product is installed and you invoke the exe you get a login screen.

The login screen shown in Figure 1 provides you with a field for your username and password, and the broker name for your Harvest server as well as the HarPB version number (in this case Version 2.1.0.3) and the version of Harvest that this product works with (for CCC/Harvest 5.2 and r7).

After the login is successful the screen view when the product opens up provides you with several panes. The pane on the left-hand side gives you a view of your work Pbl (PowerBuilder Library) and your PowerBuilder application via a target .pbt. I'll explain this in more detail later on. The right-hand pane view lets you see the Harvest Workbench and the lifecycle created in Harvest. Figure 2 provides all the information pertinent to the file you are going to check out.

Figure 3 provides an example of the log screen at the bottom of the overall screen. As you can see this is great for status accounting and auditing. The file is listed as well as all the versions for the file, the package associated with the file, check-out information, check-in information, and the dates of

check-out and check-in. The status of the file is also provided (N) equals Normal status; if there were to be an R for status that would mean that the file has been reserved by someone else. When there's a reserve tag on a file it can't be checked out for update by anyone else until it has been checked back into the repository.

When you click on the top right-hand corner of the previous screen on the top of the menu on application –> Open a new screen appears. This screen lets the user open pbl's, ini's, targets, or a workspace to see the application being worked on. In our environment we work primarily off the .pbt file or the target file. When you navigate to a target that has been created for your application and you open it the target contains all the vital pbl information that HarPB needs to open the application for use in the tool.

Once you have navigated to your .pbt and select it to open (see Figure 4), highlight the pbl on the top line specified in the .pbt and this will show you below all the Pbls that are in that .pbt and associated with the application you are working on (see Figure 5).

Under Options on the top menu you have the Harvest context window that is required as part of the tool setup (see Figure 6). These processes are obtained from the Harvest Workbench lifecycle states. As long as your lifecycle is set up and the processes are there they will be reflected in HarPB. You're going to want to make sure that these fields have been populated with the processes. HarPB even allows you the option of creating new Packages from within HarPB.

Once that is complete, continuing under the Options main menu heading there is the preferences screen that allows you many choices for import or for your Pbl management (see Figure 7). We generally add our work pbl automatically when HarPB opens. The force of Auto Check In is a great function and you can clean your work pbl or work file out after check-in, regenerate on check-out, etc. It's really a wonderfully easy product to use and it has been very successful in our environment because of this easy use.

You can see a PowerBuilder application displayed on the left-hand pane and on the right-hand pane you see the view from the Harvest Workbench including the applications lifecycle flow. All of the same processes available in the Harvest Workbench are also available in this view in HarPB including all access control. You navigate through the lifecycle in the same way that you navigate through the Harvest Workbench view. In each State (e.g., CR Assignment, Development, Development Complete, etc.) is a subdirectory of views (e.g., packages, forms, data views, etc.). The data views lets you navigate down to all of the data housed on your Harvest server. When you've located a file you want to check

Figure 1

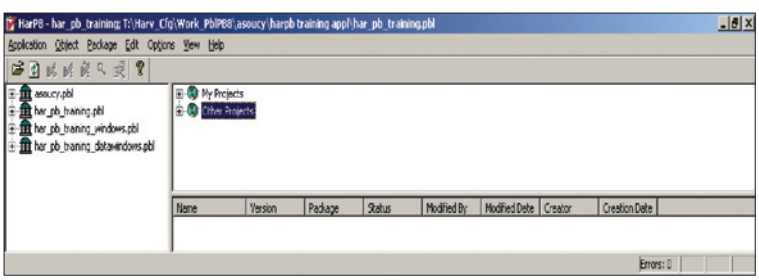

Figure 2

"If you want the best seamless interfacing capability into PowerBuilder for source code management then you want HarPB managing your assets"

out you can use your right mouse click to do it.

You're going to want to highlight the pbl you intend to check out to and select your mode (Browse – read only, Update, or Concurrent). We do not use Concurrent check-out in our shop because 1.) we do linear development and 2.) merging and branching files is no easy task. It also takes more resources because it takes more time to ensure the integrity of the file(s). Check-in is done by right mouse-clicking the file and the Auto Check In does the rest.

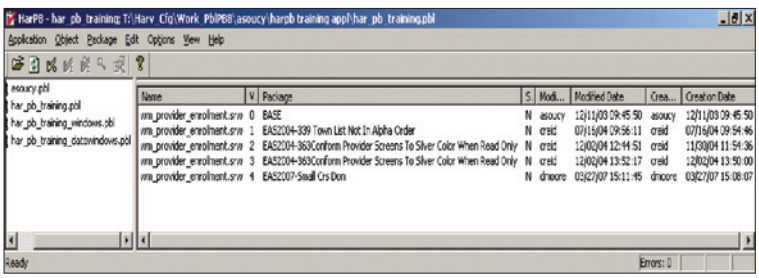

Figure 3

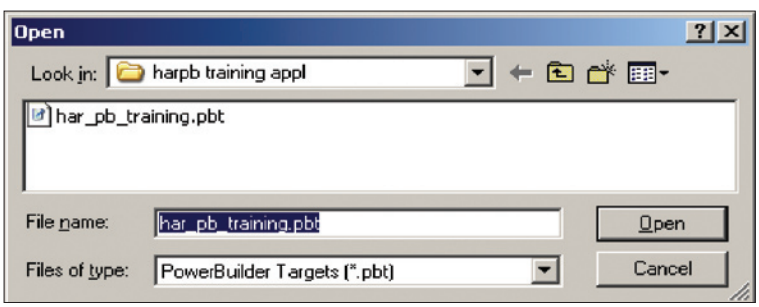

Figure 4

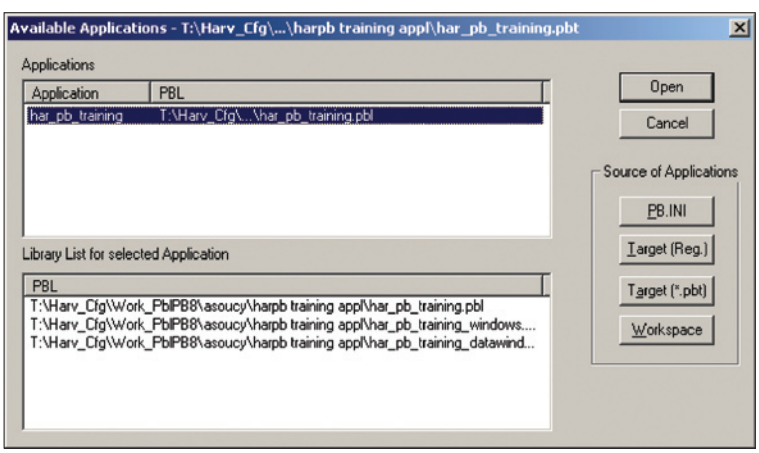

Figure 5

Using Work PBLs

As an option HarPB will automatically add a "work" PBL to an application's library list, when the application is first opened. Work PBLs are commonly used in PowerBuilder development as a place to make changes to objects without overwriting the original object. The work PBL is always inserted at the beginning of the library list so that the objects in it override the objects that occur further down in the library list.

The benefits of the work PBL are that when the object is removed from the work PBL, the application returns to a previous know state. Also all of the development activity for objects that occur in multiple PBLs can take place in the work PBL, so that its contents form a simple indication of what objects are under development.

When you choose to add a work PBL, you may browse to the PBL you want to use as the work PBL. This becomes the default work PBL for all applications until you change this option.

The PowerBuilder IDE interface to source control (since v8.0) doesn't support this very useful concept of work PBLs. You are constrained to check out objects to the PBLs where they were originally registered.

Split View/Drag and Drop

HarPB provides a complete graphical view of both the PowerBuilder application and the Harvest Repository. Each is represented as a tree view in which you can drill down as deeply as you want. The context of the view is preserved from one session to another and the state of the views is restored to its previous condition whenever you start HarPB.

The spit view enhances your understanding of the location of the objects in PowerBuilder and the repository and supports drag-and-drop operations for checking objects in and out.

Handling Object Dependencies

HarPB provides many options to resolve dependencies between objects (which is essentially an import operation).

You may choose to have HarPB regenerate all objects dependent on an object you are checking in.

Also when checking out multiple objects at a time, you can choose to use E. Crane's "Boostrap Import" technology to achieve an error-free check-out when there are complicated dependencies between the objects you are checking out.

- It readily adapts to your development process through interface into the Harvest Workbench.
- Lets you track information associated with every change. The status accounting of this product is truly a great feature. There is nothing you can't find out about a file. When it was checked out or checked in, by whom, on what day, what time, what was done, the size of the files(s), etc. This feature is superior.
- Lets you navigate efficiently.
- HarPB supports all PowerBuilder versions.
- Easy to use. This is one of the most important aspects of this product. I can have a user trained in an hour and using the product productively the same day. The ease of the product allows for more developer productivity.

Tool Integration

Interfaces from popular integrated development environments (IDEs), such as Visual Basic, Visual C++, IBM Visual Age for Java, and IBM WebSphere, let developers perform routine CCM functions without leaving the IDE.

This product provides one the best tool integrations I've worked with and the product is still evolving. It provides one of the more seamless integrations.

Our Experience

We develop and maintain 40 PowerBuilder applications at New Hampshire's Department of Health and Human Services. We have migrated a few applications to PowerBuilder 11. The applications are used extensively in our welfare and health service delivery agencies. Example applications are for child-care licensing and managing adult and elder care. Throughout the state the applications are used by hundreds of users.

Using HarPB has made securing and controlling PowerBuilder objects for source code management an easy task to do.

My Synopsis and Review of HarPB

After all that I have said about HarPB from the environment use, processes built into the product, interfacing capabilities with PowerBuilder, and ease of use, I conclude that if you are in an environment where you require a product that encompasses the best seamless interfacing capability into PowerBuilder for source code management then your environment cannot be without HarPB managing your assets.

About the Author

Al Soucy is software configuration manager at the State of New Hampshire's Office of Information Technology (OIT). In that role Al manages software configuration for dozens of PowerBuilder applications as well as applications written in Java, .NET, and COBOL (yes, COBOL)..

asoucy@dhhs.state.nh.us

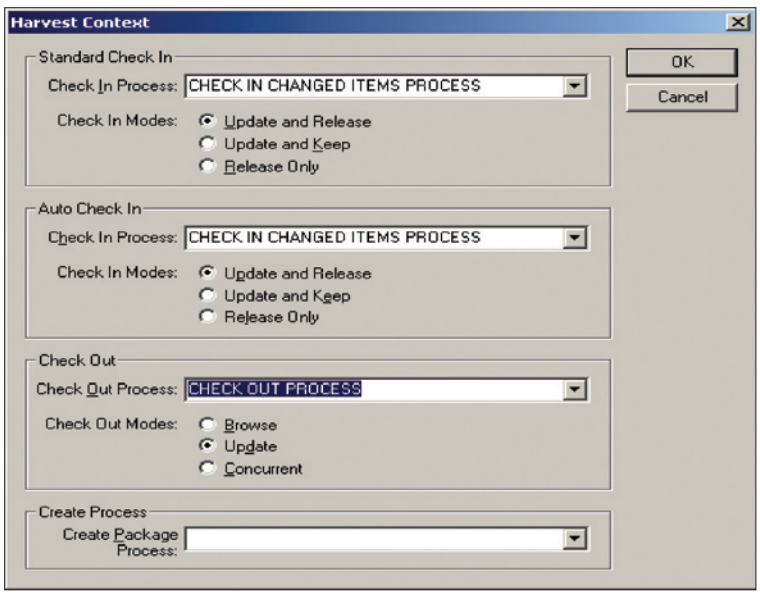

Figure 6

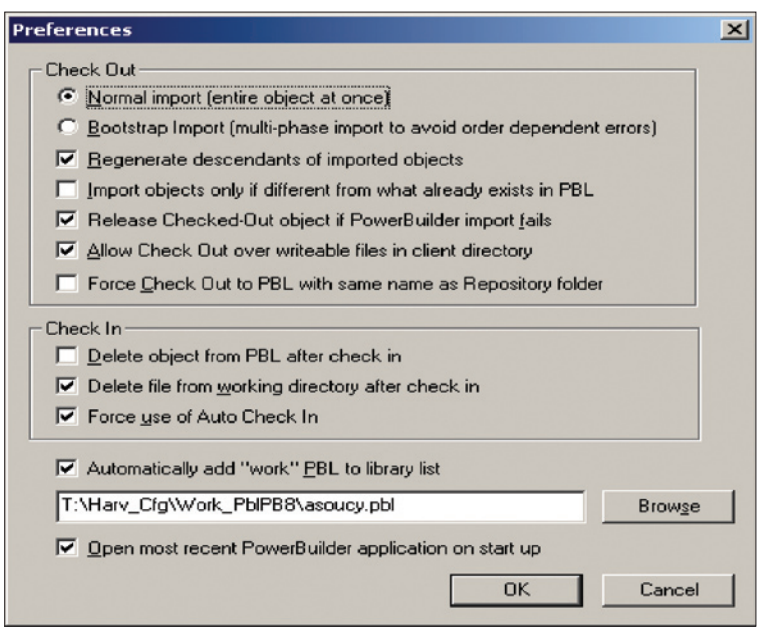

Figure 7