PUSHING THE LIMITS OF LABVIEW

JKI's VI Package Manager Drives Worldwide Collaboration for NI Systems Engineering

By Christian Loew, Systems Engineering Manager, National Instruments

Architecture Consulting for LabVIEW Applications

The Systems Engineering group at National Instruments helps our customers build solutions with NI hardware and software. My team focuses on applications based on our real-time embedded platform, so our expertise spans machine control, embedded systems, data logging, structural test, structural health monitoring applications, and more.

Like most development teams, we follow a structured process from requirements to design, development, test, validation, and deployment. However, unlike many development groups, we have to track these processes for both our own projects and our customers' projects. Our challenge is compounded because the Systems Engineering group has teams all over the world, so we need well-defined collaboration processes to work together on individual projects and share code between projects.

As part of our focus on engineering best practices, we started a more concerted effort to develop and share LabVIEW code for reuse about six years ago.

Distributing LabVIEW Code Before VI Packages – Easier Said than Done

At the time, the Systems Engineering group was heavily focused on creating reference designs, components and libraries of code we wanted to share with our customers. Our goal was to take code we'd created for individual customers – basically proofs of concepts – and turn those VIs into more general libraries we could share online with others.

Originally, we used Windows™ installers built with the LabVIEW Application Builder to package source code for distribution. However, the installer was clumsy and we had trouble with dependencies and versioning. A lot of our components have dependencies on each other, so we'd have to manually run each installer in turn. We also had to document the process in detail for the customer. In one case, we created a reference architecture that comprised eight components, so the customer had to run multiple individual installers before they could open the example.

THE GOAL

Reuse and distribute LabVIEW code to our customers - and within our own internal software group - without the hassles normally associated with code sharing.

THE SYSTEM

- Better engineering process supported by code reuse
- More collaboration between global teams
- Reduced opportunity for installation errors
- Substantial time saved by using VIPM to create and distribute VI Packages
- Team motivated to produce better code, because it will be shared

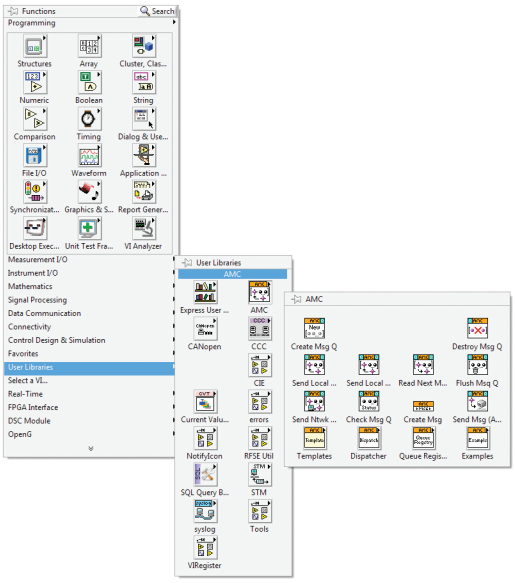
NATIONAL INSTRUMENTS™

11500 N Mopac Expwy
Austin, TX 78759
888.280.7645
ni.com

We tried to mitigate this effect by bundling everything together in a monolithic "master installer," but that approach created a versioning problem. Anytime we made a change to an individual component, we had to rebuild the entire master installer.

Another VI distribution solution was to provide source code in a ZIP file. ZIP files are easier to create than installers, and some customers preferred them because they offer more control over where files are installed. However, installing libraries from ZIP files is a manual – and thus error-prone – process, especially when the location of the installed files is important. Because LabVIEW relies heavily on the relative paths between VIs and components, it is critical to install files in the right locations. If these relationships aren't preserved, LabVIEW can't find the required VIs during loading and the code breaks.

Once we learned about VI Package Manager (VIPM), we realized its VI Packages would be a better method to distribute LabVIEW source code. Originally we wrote our own tool for building VI Packages, but in 2010, JKI's VI Package Manager incorporated Package Building as a free feature in the Community Edition. We switched to using VIPM Community immediately.



VIPM installs your reusable VIs directly into the LabVIEW functions palette

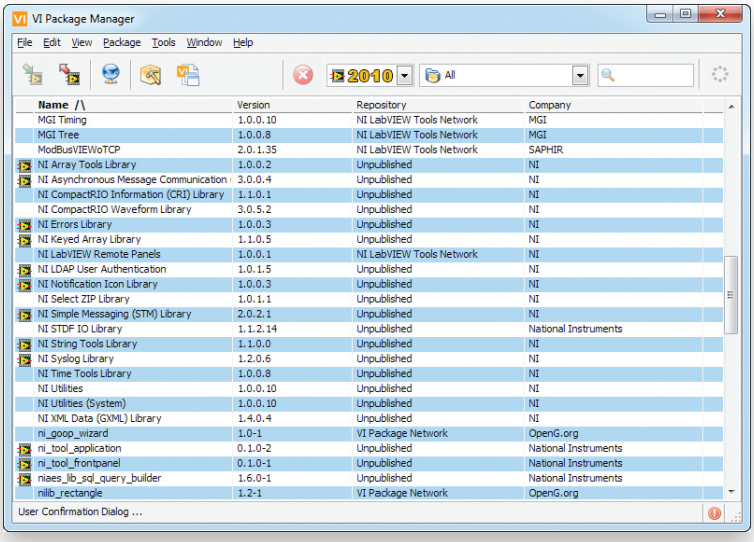
Eliminating Code Sharing Risks and Headaches with VIPM

VIPM circumvents a lot of major headaches and risks normally associated with sharing code with colleagues and customers.

First, VIPM makes it very easy to upgrade and downgrade installed VI Packages. For example, recently we needed to update an old example application that used a previous version of a VI package. With VIPM, I was able to easily downgrade the VI package installed in my LabVIEW environment to an older version, load the example, make the necessary changes, and send it back to the customer. In addition, I was able to switch back to the newer version of the package on my LabVIEW system quickly to continue other development projects. With any another system, making changes in an application using an older library would be very manual, error-prone, and time-consuming.

"Once we learned about VI Packages, they quickly became our preferred way to distribute LabVIEW source code for reuse."

– Christian Loew



NI Systems Engineering group's code library in VIPM

Another big advantage to VIPM is that it works with multiple versions of LabVIEW. Without VIPM, if you have more than one copy of LabVIEW installed, it's very difficult to install a component into multiple copies of LabVIEW. After you install a library into one copy of LabVIEW, the installer thinks it is already installed everywhere, so it won't install again into a different location. Prior to VIPM, the only solution was to manually install libraries into each LabVIEW version – another error-prone time sink.

We also like that VIPM installs VI libraries directly into the LabVIEW functions palette, where our customers can access our add-on libraries as easily as any built-in LabVIEW functions. It's also nice that we can build custom subpalettes that are specific to certain modules such as Real-Time or FPGA, a feature that helps keep us and our customers organized.

Now that VIPM powers the LabVIEW Tools Network and is a preferred method to distribute LabVIEW add-ons, it's an even more convenient solution for us; we can easily recommend it to our customers.

Everyone Contributes; Everyone Benefits

In addition to packaging code for customers, recently we've increased our focus on code reuse within Systems Engineering. My group, as well as the RF (Radio Frequency), Automated Test, and Sound and Vibration groups, all use the free VIPM Community Edition to create components and libraries for team distribution, and we're definitely encouraging other NI Systems Engineering groups around the world to do the same so we can all enjoy the benefits:

1. **Time savings** – VIPM puts all of our reusable LabVIEW code libraries into modular packages and manages the complexities around them, such as versioning and dependencies. It's much faster for us to

"VIPM has helped us set up a more formalized method for sharing and reusing code. We'd highly recommend it for other teams who need to do the same."

– Christian Loew

package code for distribution, and much faster for others to install it. The tool is particularly helpful when we're replicating a development environment for an old project. For example, when I was updating that old example application, it probably took 3-4 minutes with VIPM. Without it, I'd have had to use the installers, and it would have taken 20-30 minutes – that's a 5-10X time savings!

2. **Reduced opportunity for error** – by automating the packaging and installation process, we remove the possibility of user error, so our customers are more likely to get working code on the first try. This benefit is especially important if you have a lot of dependencies where code needs to be installed in the right place; VIPM will even tell you if something is missing.

3. **More team collaboration and code reuse** – global collaboration has been a focus for the Systems Engineering group for a few years. VIPM reduces the barrier to sharing code, and now that it's easy, our teams do it a lot more. NI systems engineers from around the world are now adding libraries and packages to our code library here, and everyone benefits.

4. **Developers are motivated to produce better code** – when developers know their code will be reused and seen by others, they go through the effort to create cleaner code. We've already noticed this effect.

To help us centralize and reuse our code even further, we're about to upgrade to the Enterprise version of VIPM, which provides a private code repository. Enterprise will also give us the ability to use VI Package Configurations* to recreate the exact environment used for past projects.

VIPM is an integral part of our process for sharing and reusing code, and we'd highly recommend it for other teams who need to do the same. VIPM won't create your process by itself – you still need the discipline to create well-designed, clean, reusable code – but it will give you the capabilities you need to build and distributed LabVIEW components and libraries quickly and easily, in a consistent manner, for any LabVIEW environment.

**VI Package Configurations are available in the VIPM Professional Edition as well as the Enterprise Edition. For a list of features present in each edition, please visit jki.net/vipm/compare.*

VI Package™ Manager

VI Package Manager makes it easy to download and install LabVIEW Add-ons, create your own commercial add-ons, and share add-ons with your coworkers, customers, and the LabVIEW community.

Visit jki.net/vipm to learn more!

NATIONAL INSTRUMENTS Certified Alliance Partner

JKI is a National Instruments Certified Alliance Partner.

P.O. Box 2846
Walnut Creek CA 94595
888.891.7821
jki.net
info@jki.net

jki.net/blog
facebook.com/JKISoftware
twitter.com/JKISoftware