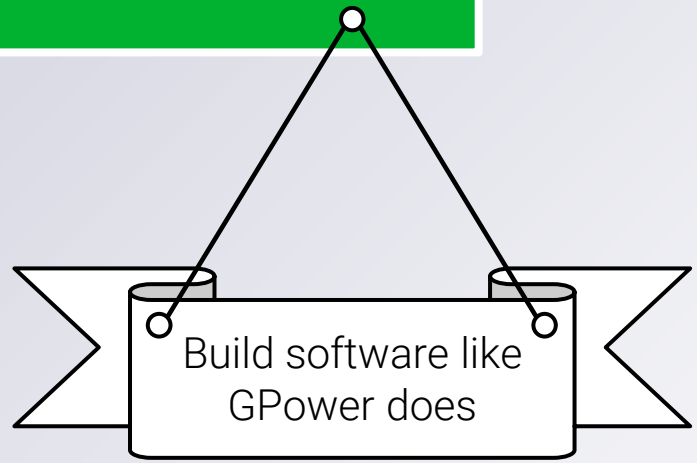
GPower

Low Cost = High Value



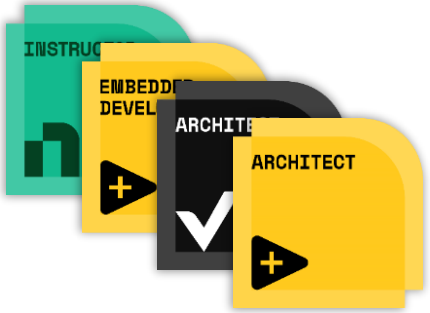
GPower

Instruments

Test & Automation

Consulting

Research & Lab → Product Development → Production → Product Maintenance



x25

x2

Conservative

⬇

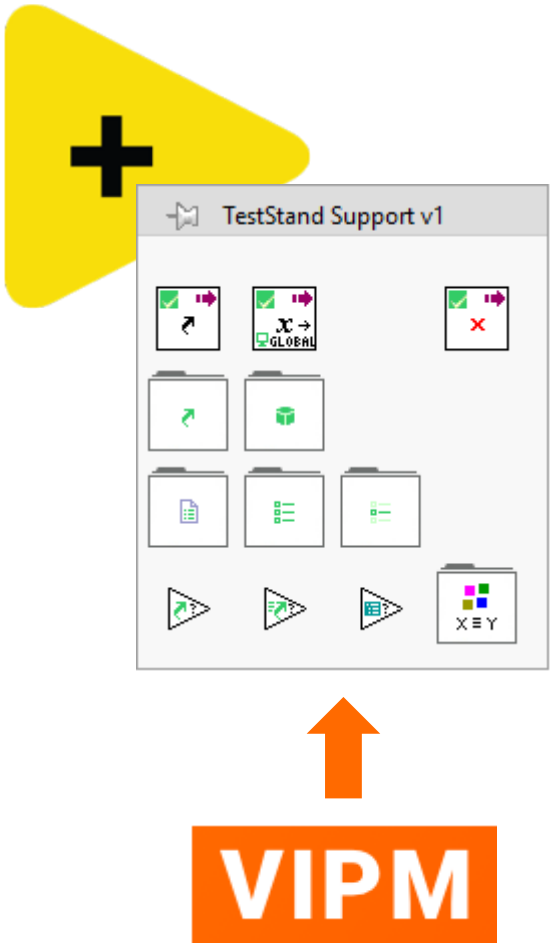
Systems engineering

LabVIEW is expensive …unmaintainable even

⬇

Work smarter

We are limited on time… therefore just one selected case of modular software today (VIPM in LabVIEW IDE)



Let’s connect and follow up

First things first…

Preconditions

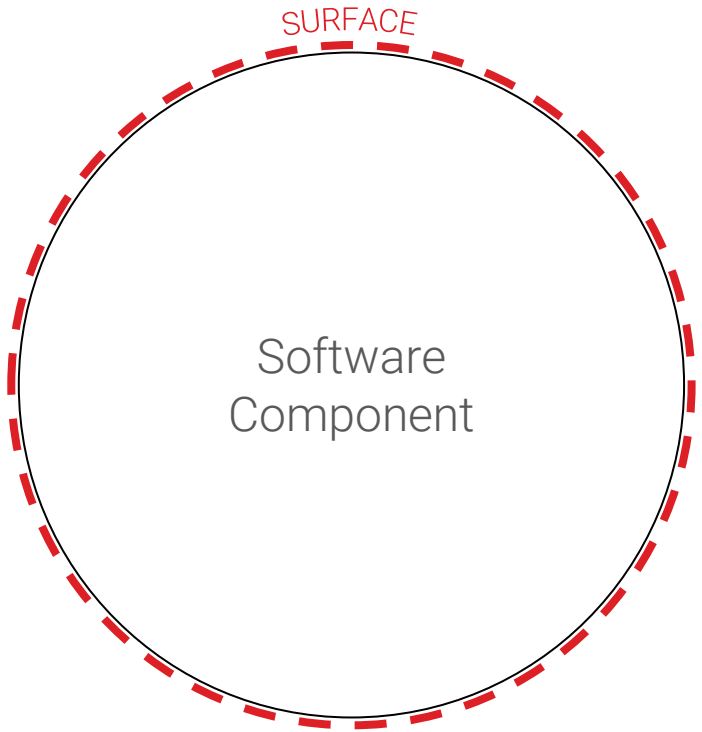
- Package technology is not a preference, it’s proven computer science
  Same as unit tests, source code control, disconnect source from compiled code, no autopop folders etc.

- Version numbering: Major . Minor . Patch . Build

Major	Minor	Patch	Build
Breaking change	**Functional change**	**Bugfix**	**Build**
Increment when you make incompatible API changes	Increment when you add functionality in a backwards-compatible manner	Increment when you make backwards-compatible bugfixes	Increment when you make a new build of a version Reset when you make a new version

- SOLID principles

A software component is…

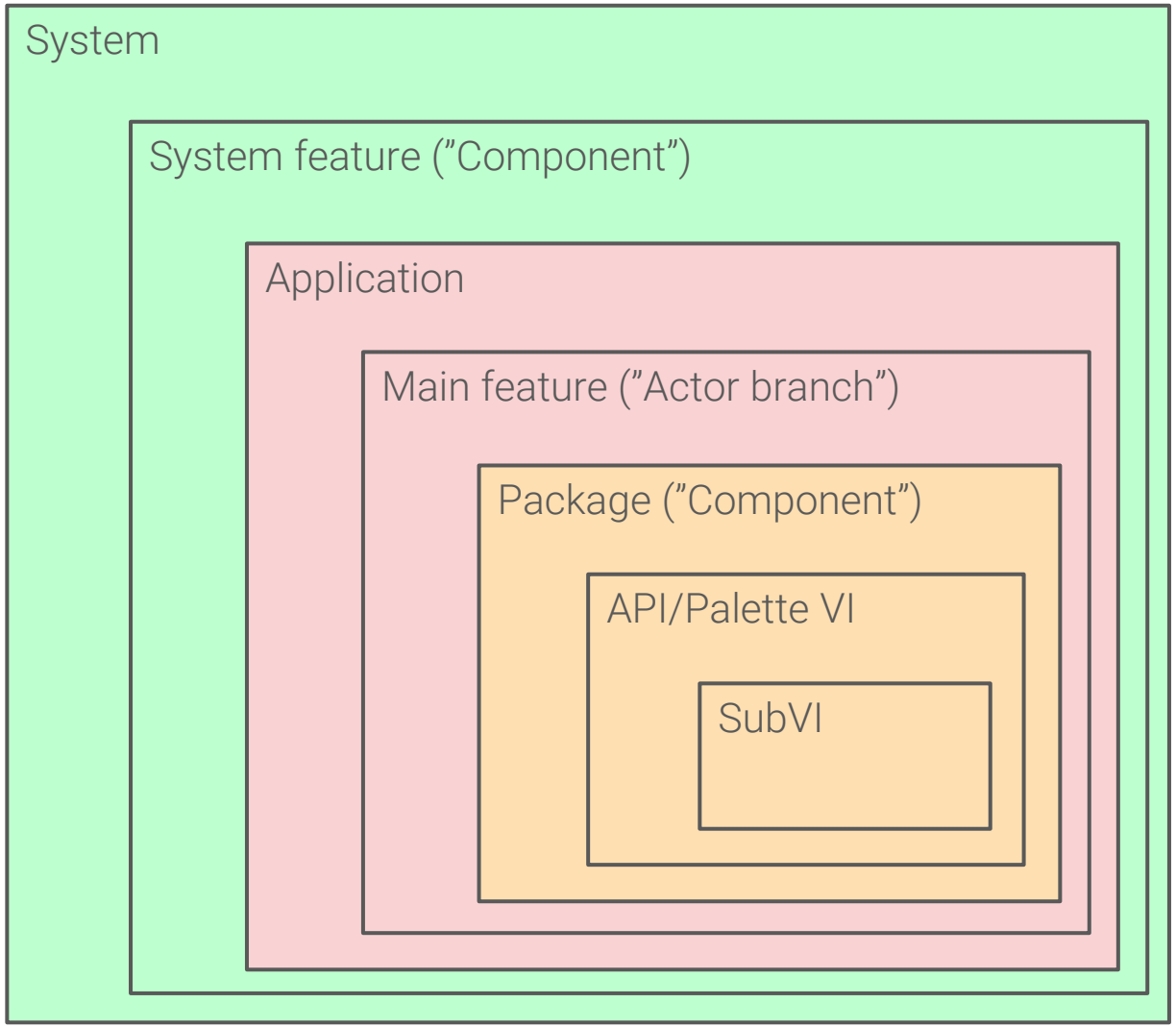


Single responsibility
It does one thing, and it does it good

Understandable
Usable
Maintainable
Works as expected

Sealed
Self contained
Compatible
Replaceable
Stable

Single responsibility is by scope…



But how?


Packages
Low Cost
Reduce development cost
Prepare in advance, SOLID APIs improves efficiency
Reduce downtime
Good code quality, track proven reuse components
Reduce maintenance cost
Single responsibility narrows down what code to touch
High Value
“The package also has that feature…”
Grabbing a ton of great code in a reuse package…
Trustworthy code
Repeated reuse of code done right increases application quality
Code Quality

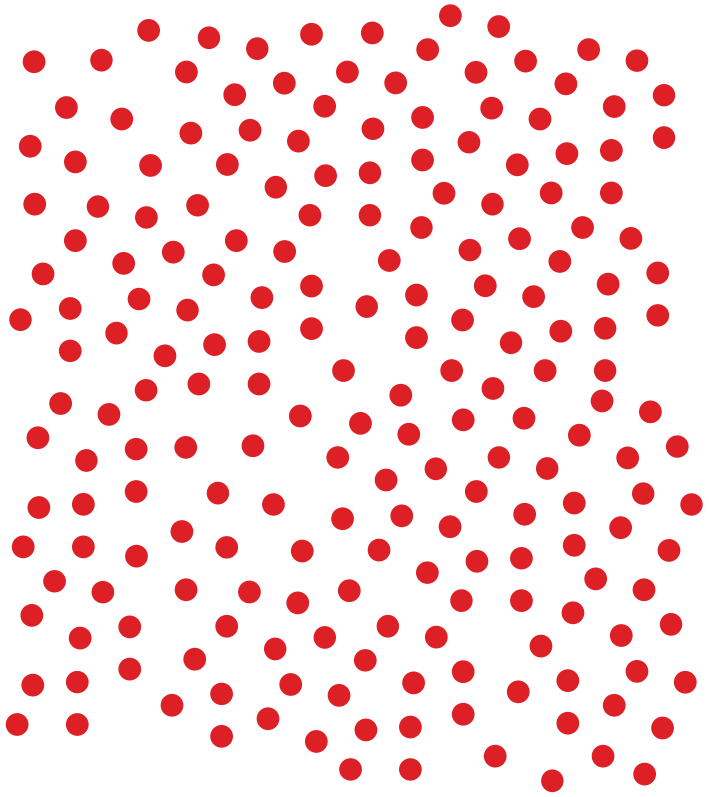
How to make high quality code

- **Do it!**
  It's not enough to *know* how, or *say* the right words, you must *do* it. Every time.
- **Do it right**
  Get all the way to Done, maintain good coding practices (typedefs, naming), refactor when necessary. Don't postpone.
- **Design for reuse**
  Picking out an old project as starting point is the worst kind of reuse. Don't do it!
- **Do it consistently**
  Templates and wizards.
- **Minimize complexity**
  A removed feature is one less thing to implement and maintain. The future must pay for what the future needs.
- **Single responsibility**

What do we want from our packages?

Application code reduction

Application (lvproj)



1500 source files

Application (lvproj)

30 source files

Custom (vi.lib / lvproj)

.vip v1.0.0
.vip v1.0.0
.vip v1.0.0

3 packages
(170 source files)

Reusable (vi.lib / lvproj)

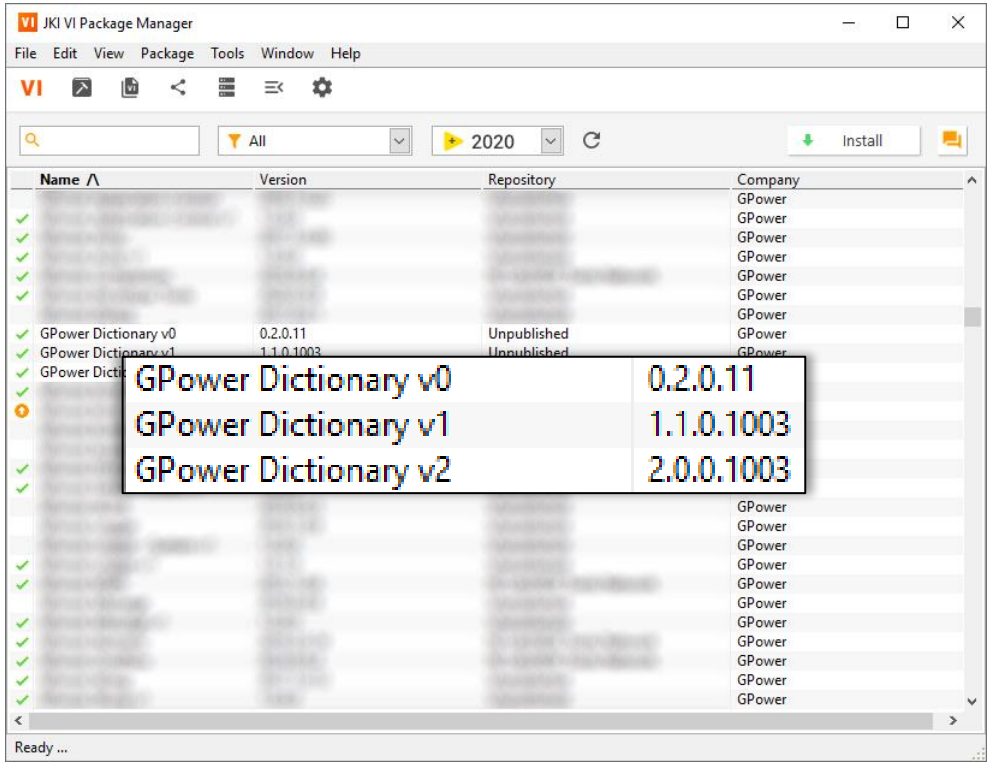
.vip v2.0.1
.vip v1.1.0
.vip v1.0.0
.vip v1.4.0
.vip v1.1.7
.vip v3.1.0
.vip v1.7.0
.vip v1.0.0

8 packages
(1300 source files)

Version flexibility

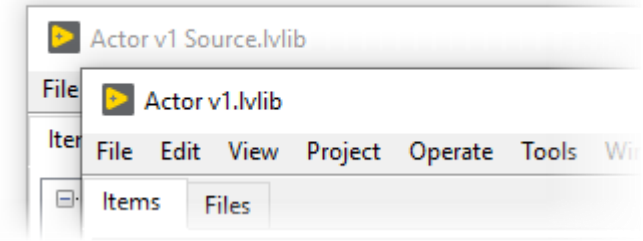
To avoid forced update requirements:

Major versions must be side-by-side installable



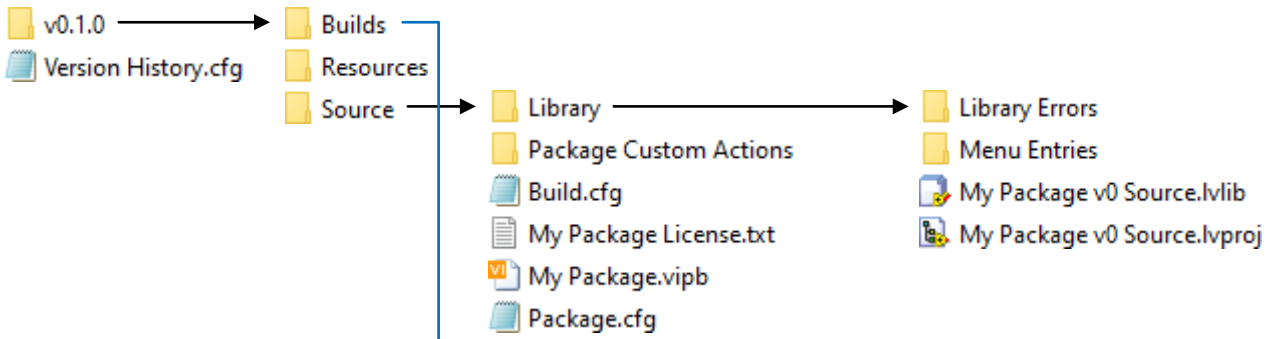
To avoid cross-linking “accidents” and for convenience:

Source must be editable while package is installed

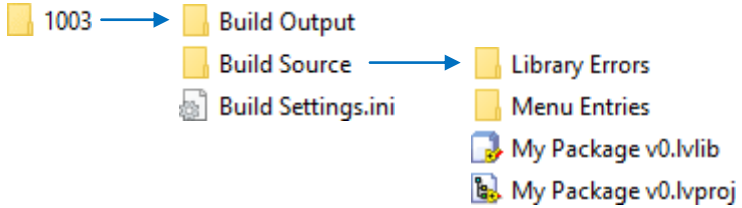


Package folder features

Source:

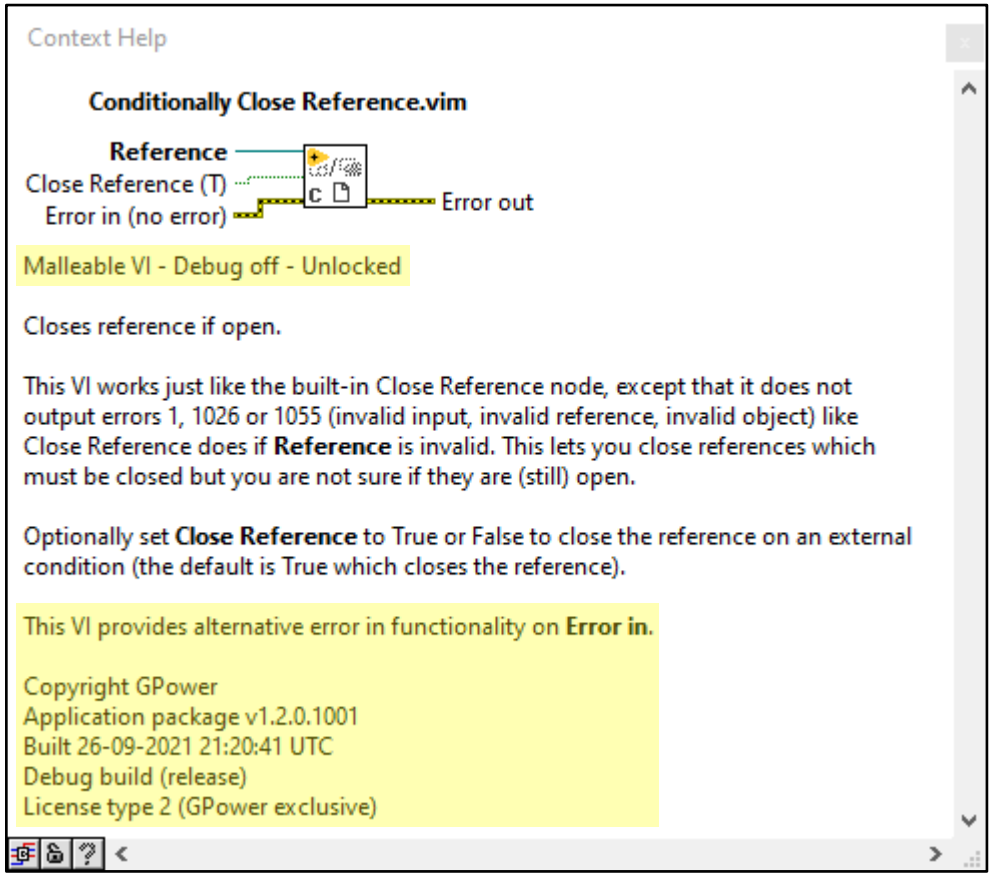


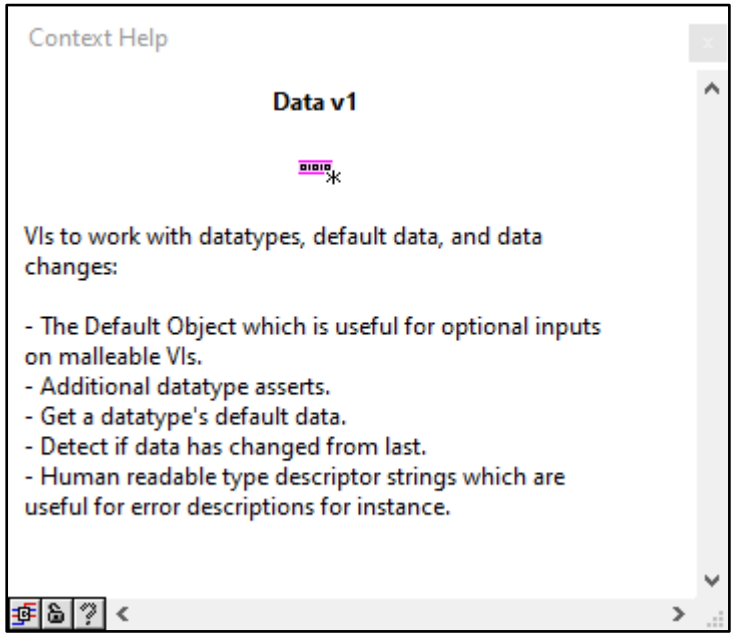
Build source:



- Custom library errors
- Menu entries
- Package configuration
- Build configuration
- Build source traceability

Automatic source code and palette documentation



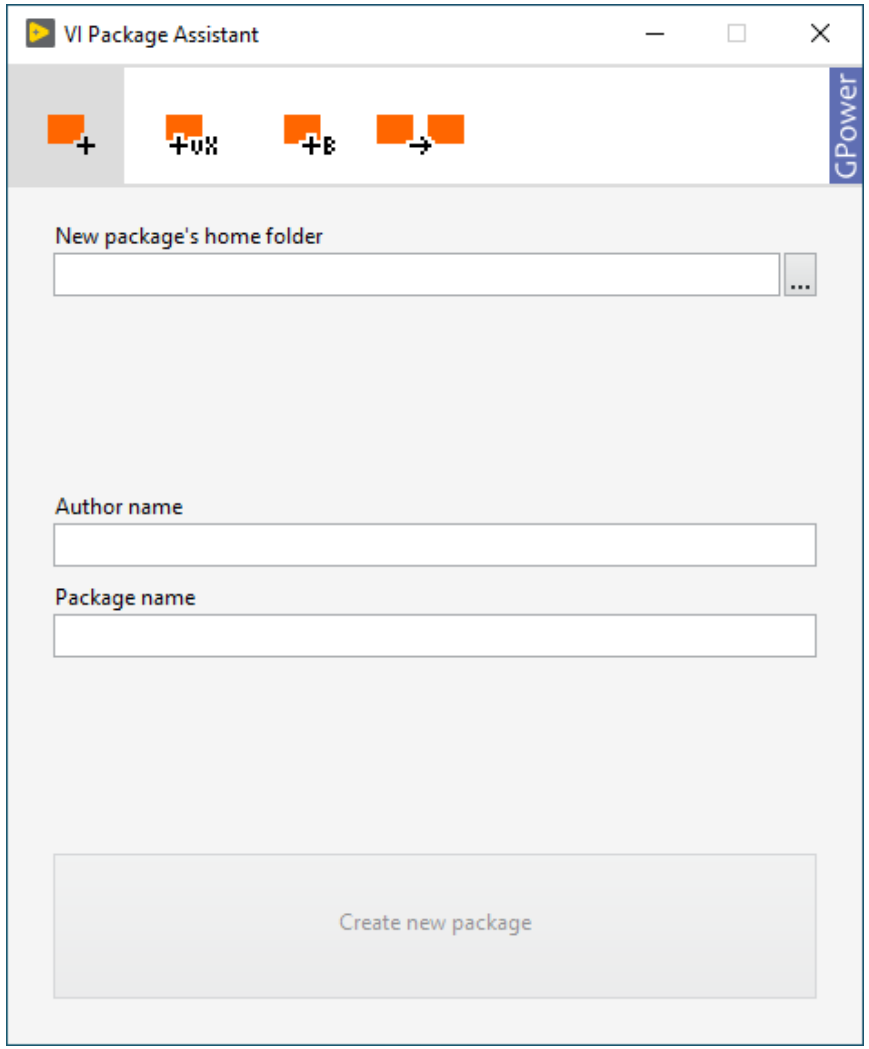


Other selected requirements…

- New Package, New Version, New Build, and Copy Package functions
- Debug, Developer, and Dependency build configurations
- Built for Author, for Customer, or open source license options
- Beta and release maturity build options
- 130 VIPB fields automatically configured
- Automatic handling of unit tests, debug flags, palettes, passwords etc.

How do we do it?

VI Package Assistant



Package description

Palettes

~~Dependencies~~ Probably also auto-calculated in the near future

Save a ton of money by working smarter

Possible when:

- Your entire team and leadership walks the talk
- You understand how and why
- You actually do it

There is no free dinner, but it’s not good business to pay for the same lousy dinner over and over again…

Thank you!

Questions?

Or topics to touch
on the follow up event
(we'll ask on email as well)

Follow up event?