Conan for LabVIEW

Installing dependencies per project with an open-source package manager

Stefan

@stefan_lemmens
#LabVIEWshortcuts

INTERSOFT ELECTRONICS

Agenda

- What we want
- Handling Dependencies
  - Per project
  - Avoid Relinking
- Package Management
  - Existing processes
  - Package managers
  - What is Conan
  - Creating & deploying a package
- Pro's & Cons

What this is (not)

- Short story of our journey to find a solution to handle dependencies the way we want
- An overview of our findings and opinions
- A demo of the current way of working

- No tutorial on how to install or configure Conan
- No in depth behind the scenes explanation
- No intention to replace existing package managers

What we want

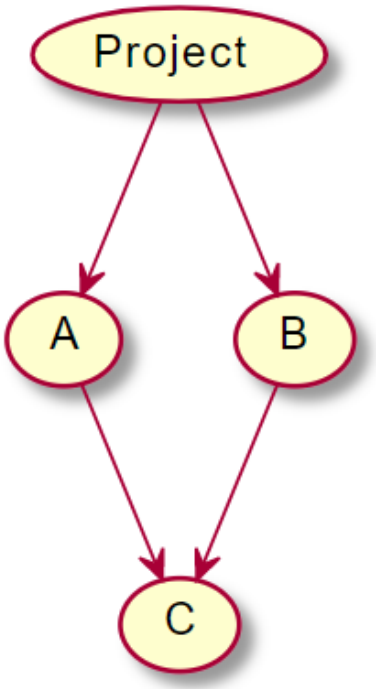
- Dependencies per project
  - Everything a project needs inside the project folder
- Version controlled dependencies
  - Version numbers of dependencies in git
  - Not the dependencies themselves in git
- Easy way to share/deploy
  - Between colleagues for collaboration
  - With CI systems for build automation
- Same workflow for all developers
- Open source (preferably)
- Free (preferably)

Dependencies per project

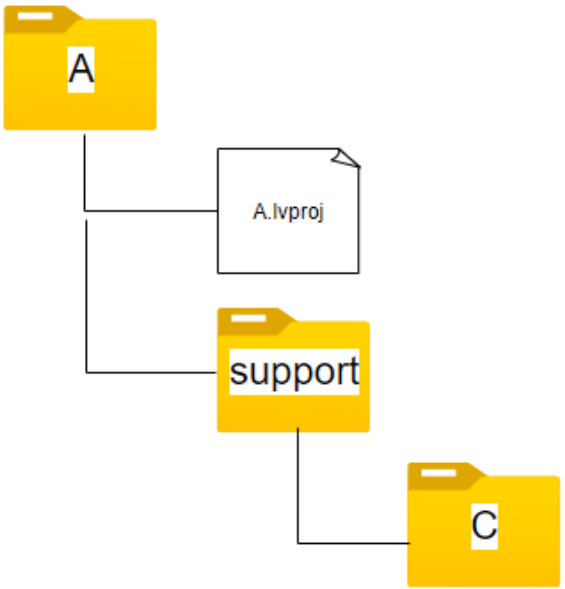
- Everything a project needs inside the project folder
  - Make everything part of project? -> not realy re-use of code
  - Exclude re-use code from projects repo
  - But include “overview” of requirements and their versions
    - e.g. vipc file
- Different projects can have different requirements
  - Also different versions of those requirements
- Not in a common folder like user.lib

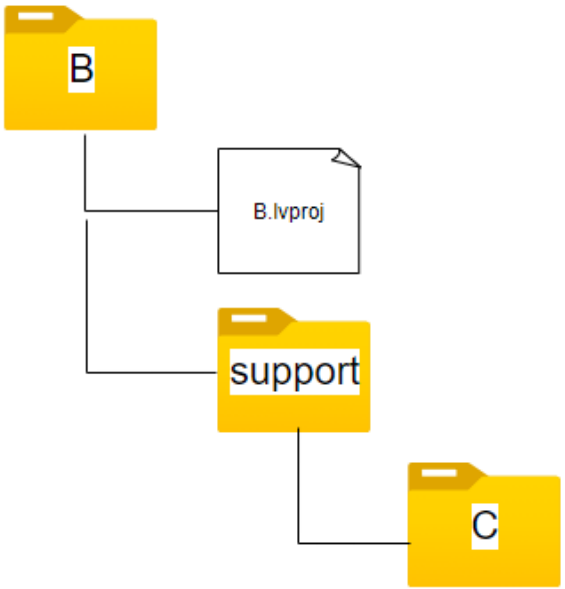
Avoid Relinking

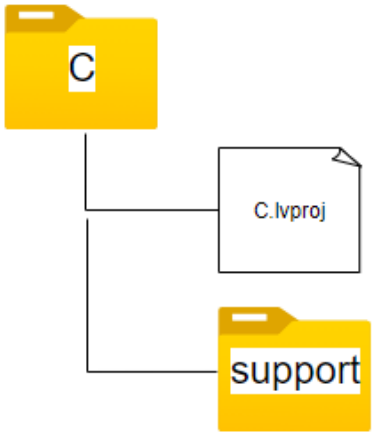
- Putting all dependencies in a support folder in your project might change their relative path to each other therefor they need to be relinked
- We want to avoid the relinking



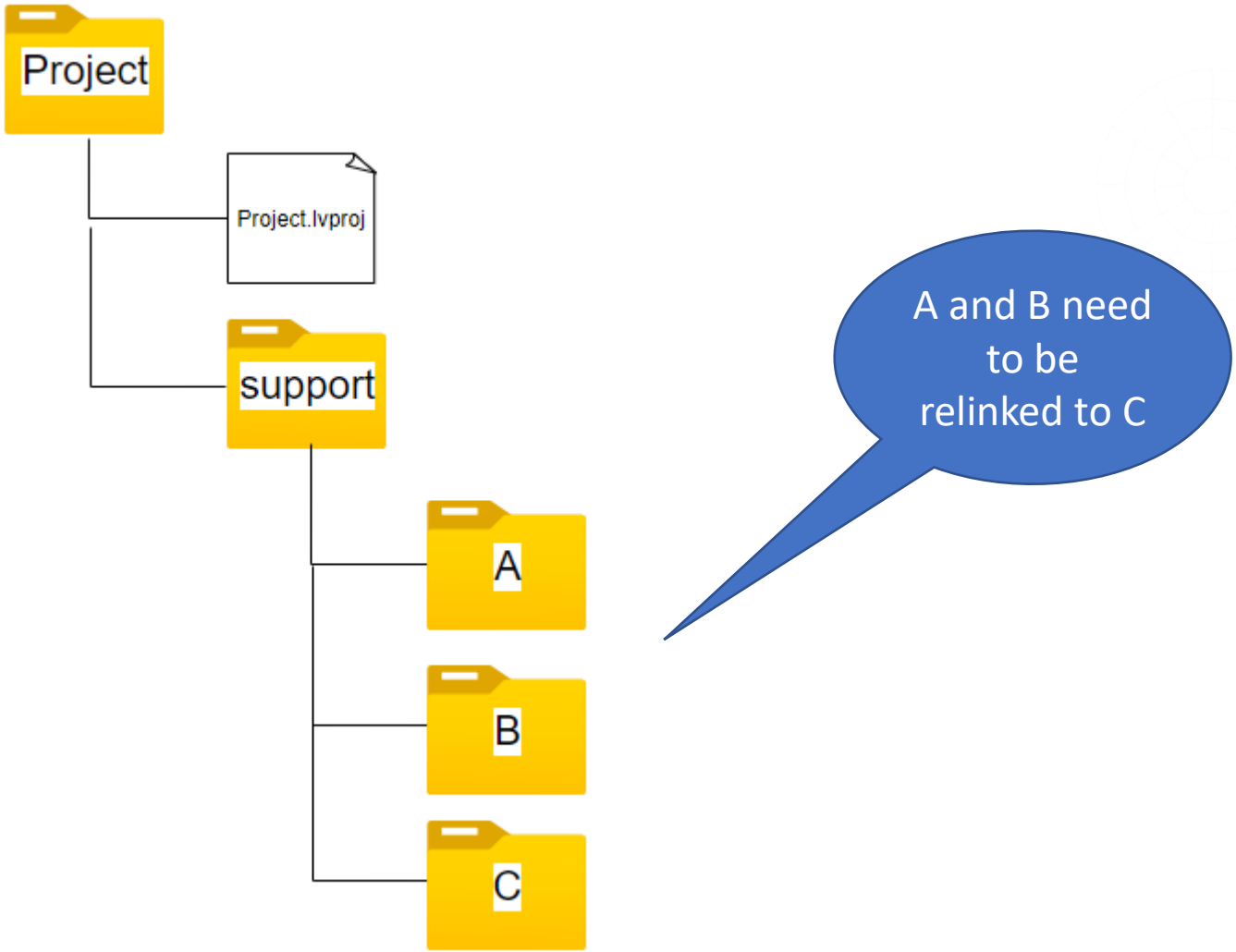
Copying Source







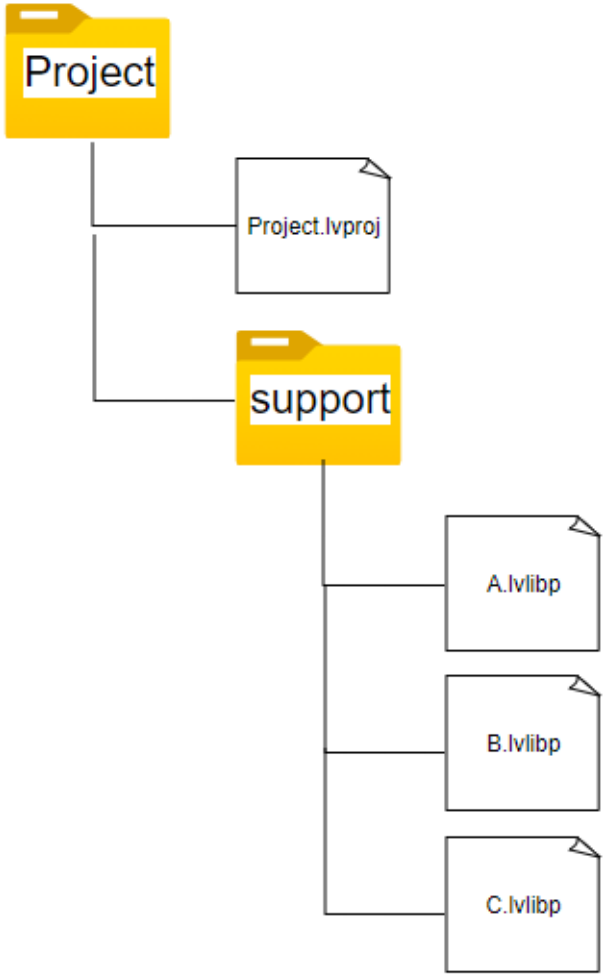
Copying Source



Using ppl's

- Dynamically linked libraries
- Pre-compiled

- Support folder
  - auto-populating folder in A.lvproj
  - ignored by git
  - populated by package management



Relinking Tool

- Dependency Redemption
  - Composed-CI

- How it's done
  - List all dependent packages
  - Sort all dependent packages
  - Add all to temp project (least dependencies first)
  - Save all

Package Managers

Characteristic	VI Package Manager	NI Package Manager	G Package Manager (GPM)
Installs With LabVIEW	Yes	Yes	No
Created By	JKI	National Instruments	MGI (Moore Good Ideas)
Year Released	2006	2017	2018
Open Source	No	No	Yes
Language Written In	G	C#	G
Free To Install	Yes (Community & Free)	Yes	Yes
API	Yes (Community & Pro)	Yes	Yes
Local repository management	Yes (Pro)	Yes	Yes
Supports multiple LabVIEW versions from a single package	Yes	No	Yes
Natively supports mass compile after install	Yes	No	Yes
Scope of package installation	LabVIEW IDE or OS File System	OS File System	Per Project
Supports LabVIEW NXG	No	Yes	No
Create palettes	Yes	No	No
Dedicated package manager app	Yes	Yes	Yes

https://labviewwiki.org/wiki/Package_Manager_Comparison

NIPM vs VIPM

Use...	If you want to...
NI Package Manager	•Install, update, repair, or remove NI software. For example, LabVIEW, modules, toolkits, and drivers. •Distribute deployed applications to your clients. For example, LabVIEW-built executables, packed project libraries, source distributions, drivers, and LabVIEW Run-Time Engine.
JKI VI Package Manager	•Discover, install, and remove LabVIEW add-ons from repositories such as NI Tools Network and JKI Package Network. •Create and distribute LabVIEW add-ons to your clients. For example, libraries of reusable VIs and development tools.

Choosing between the NI Package Manager and JKI VI Package Manager

Existing Processes

- Copy source
- Use zip to distribute possibly with manifest file
- Use git submodules
  - https://www.youtube.com/watch?v=iv7WwDgyb0U
- Dependency Redemption presentation
  - https://www.youtube.com/watch?v=zQBPe0SmjRY
- VIPM (with project Dragon), NIPM, GPM,…

- Why don’t you try Conan?

What is Conan?

- Open source package manager for C/C++
- Can handle binaries
- Python package : pip install conan
- Very good documentation
- Decentralized, host packages on your own server
  - Local cache
  - Connects to remote server
- Already used within Intersoft Electronics

What is Conan?

- Conan the Frogarian
- Jfrog

- conan frogarian

How to use

- Each repo must have a conanfile.py
  - Contains requirements
  - Contains extension or script overrides

- Run on command line
- Each command will execute several methods of the conan class

How to use

- conanfile.py

```
from conans import ConanFile

class LabVIEWBuild(ConanFile):
    requires = "labview_conan_poc_libc/[~=0.*]@intersoft/testing"
    python_requires = "LabVIEWConanExtension/[~=20.0.*]@intersoft/testing"
    python_requires_extend = "LabVIEWConanExtension.LabVIEWConanExtension"
    gitURL = "[redacted]/labview_conan_poc_liba.git"
```

How to use

- conan create .
  - creates a package from the current git repo
- conan install .
  - installs all packages defined as requirement in the correct version and also their requirements
- conan upload * --all –r <remote> -c
  - uploads all packages from the local cache to the defined remote repository

conan create .

- set_name = LabVIEW project name
- set_version = from latest git tag
- requirements
- package_id
- system_requirements
- source = git clone to cache\source
- imports = install requirements@cache\source
- build = run buildscripts in lvproj with G-CLI
- package = copy build to cache\package\libs
- package_info

Version number

- Get latest tag from git = x.y.z

  git describe --tags --match "[0-9]*.[0-9]*.[0-9]*" --abbrev=0’

- Get total number of commits until current

  git rev-list –count <first_commit> <current_commit> = buildnr

- If current branch is master
  - version = x.y.z.buildnr
- Any other branch
  - version = x.y.z+1.buildnr

conan install .

- requirements
- package_id
- package_info
- deploy = install requirements
  - copy from cache\package\libs to \support

conan upload …

- Upload from local cache to remote server
- Supports Artifactory
- Supports GitLab Package Registry

- If install doesn't find the package in the cache it will try to get it from the remote
- If only a recipe is available it will build the package before installing it

Pro's & Cons

- Per project dependencies under SCC
- Same workflow for all developers
- Free & open source with very good doc's
- Easy to use

- No nice UI (yet)
- Not the default LabVIEW way of working
- No IDE integration (yet)
- Internal reuse only, no community reuse

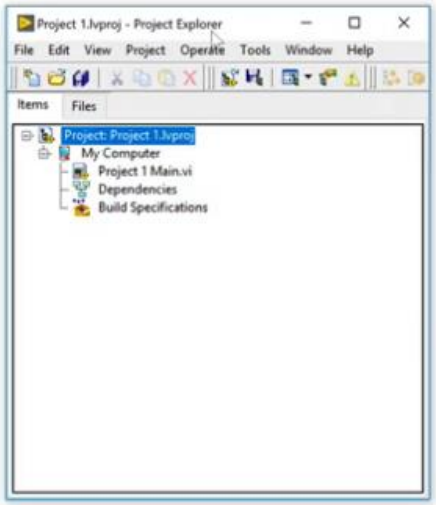
What about project Dragon

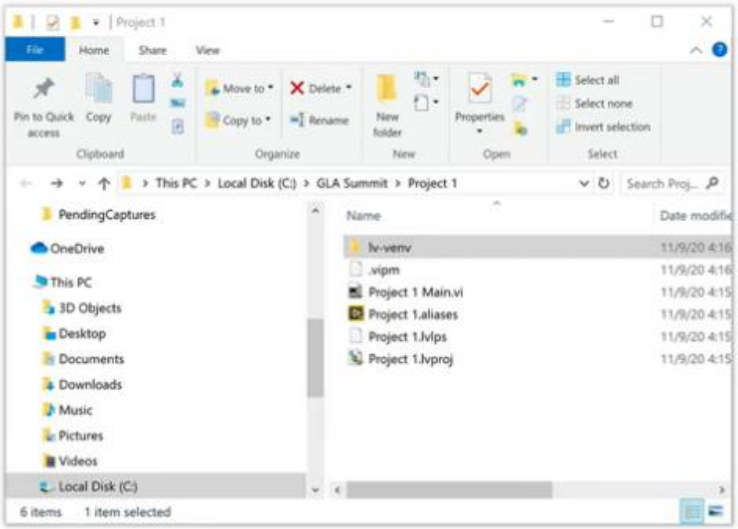
- JKI added this feature to VIPM
  - Managing virtual environments with VIPM
- Will be released with LabVIEW 2023 Q1
- Announced at GDevCON3
  - Create.vi -> NI and JKI partnering

LabVIEW Project Dependency Management (Dragon Features) (Free)

- User Interface for LabVIEW Project Dependency Management
- Per-project Package Installation via LabVIEW Virtual Environments
- Manage Project Dependencies on NI Packages (e.g. NI Software/Drivers)
- Store Project Dependencies in a Text-Based Configuration (e.g. .dragon) File

What about project Dragon





Questions & Answers

Any questions?!?