Toolmaking

Julien Goodwin — @LapTop006
jgoodwin@studio442.com.au

Image: https://www.flickr.com/photos/steve-wilson/102518219/ CC-BY-NC

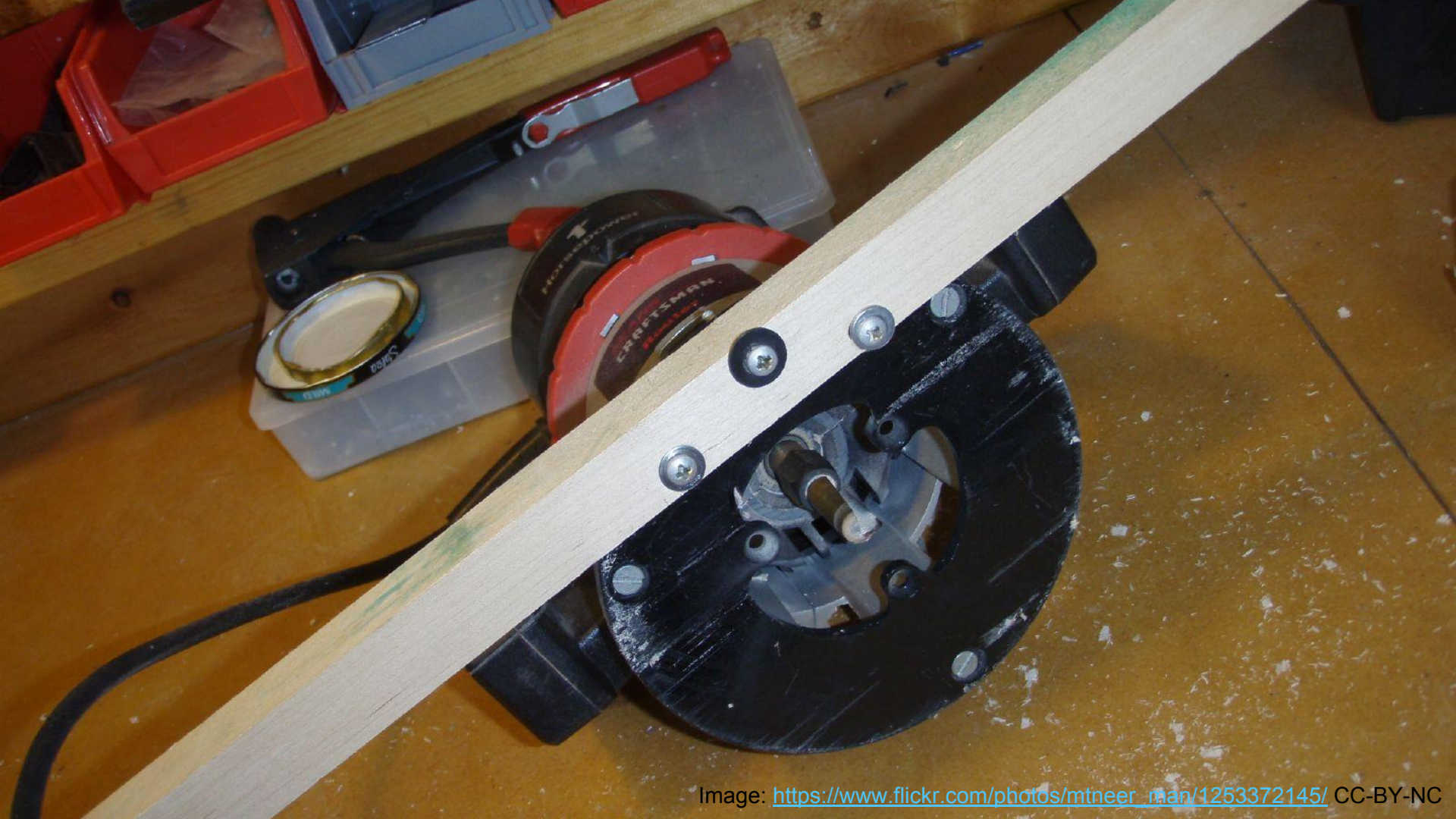

Image: https://www.flickr.com/photos/mtneer_man/1253372145/ CC-BY-NC

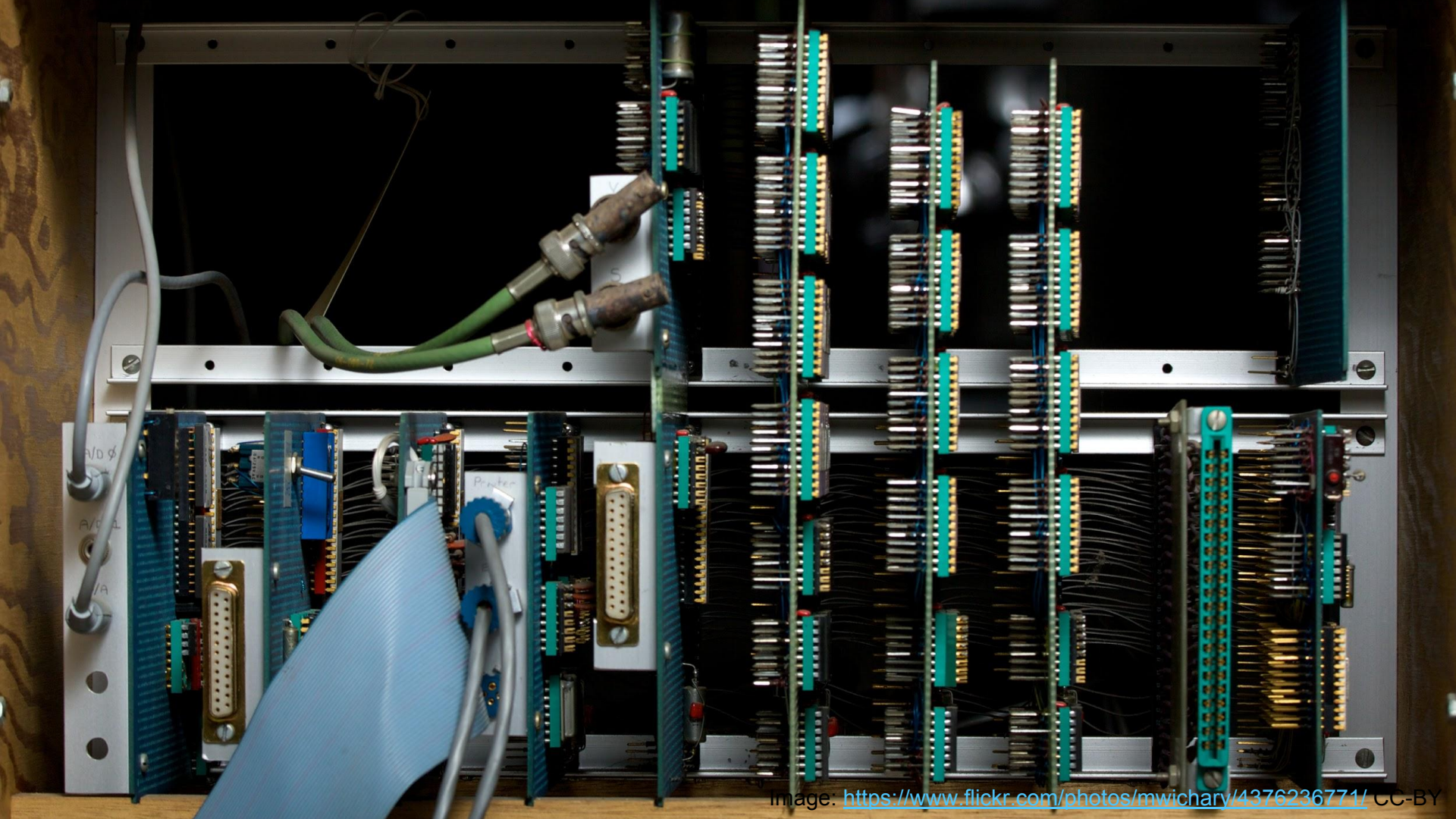

Image: https://www.flickr.com/photos/mwichary/4376236771/ CC-BY

Toolmaking in Operations

What for? (1)

Answer ad-hoc questions

- “Do we have any of these recalled components”
- “What does this BGP route look like on all our core routers”
- Sometimes a rough answer quickly is more valuable than correct but slow

Replace manual processes with consistent automated (even if in part) ones.

- Deployment
- Test

What for? (2)

Simple examples for future work

- Real, but simple uses of libraries

Building a *toolkit* for later uses

- Things you can plumb together are great

What in?

Shell Pipelines

Shell Scripts

Editor macros

Perl / Python / Ruby / Powershell / etc. scripts

C++ / Java / Go / Rust tools

How complex? (1)

Shell Pipelines — A few lines, one-offs

Shell Scripts — A few dozen lines at most

Editor macros — A few lines

How complex? (2)

Perl / Python / Ruby / Powershell / etc. scripts

C++ / Java / Go / Rust tools

— No inherent limits, practice sensible modularity & testing.

Validation & Testing

Testing large operations tools can be hard.

Splitting elements into testable modules, with their own fake implementations for larger system tests can work well.

Try to keep a standard, shared library for constants.

User Experience

Once a tool is ever to be run by someone other than the initial author.

(Or the initial author in six months … weeks … days … hours)

- Some trivial documentation
  - “Tool does X”
  - “Does not do Y or Z”
  - “Will need an update if Project Fremont happens”
- Input validation
  - Doesn’t need to be perfect, catching gross errors and bailing with a usage message is fine

Distribution

No personal tools for team problems

Have a way to distribute so all team members can run

... and update

Scaling Toolmaking

Scaling from zero...

Some companies “don’t allow” ops engineers to “write software”.

There’s often an end run (script exception) that’s used in practice.

Management cover a “read-only” rule may help to start.

Scaling to one.

In many ops teams hiring a single toolmaking-focused engineer can be the biggest marginal value hire possible.

That toolmaker still needs to be a member of the team.

Scaling to zero...

A separate team of toolmakers is often bad.

If scaling to a separate team is needed have that team work on supporting infrastructure.

Questions?

Julien Goodwin — @LapTop006
jgoodwin@studio442.com.au

QT / KDE dev, or work with BGP? Please come talk to me.