THE GAP PACKAGE DISTRIBUTION

Max Horn

GAP Days Winter 2022

OVERVIEW OF THIS TALK

- What and why is the GAP package distribution?
- How does it (currently) work...
  - for users / package authors / the GAP team
- Why does it need to change?
- How could things look in the future?
- How do we get there?

WHAT AND WHY IS THE GAP PACKAGE DISTRIBUTION?

- Short answer: a bunch of GAP *packages that get shipped with* each *GAP* release
- All *undergo basic technical review* initially, some a full refereeing process
- Technically very *easy to implement* compared to “package managers” many/most languages employ these days, such as Cargo, gem, npm, Pkg.jl, pip, ...
- (Though there is now the `PackageManager` GAP package...)

WHAT ARE ITS PROS AND CONS?

- Benefits:
  - users get a broad range of functions at once
  - we check that these packages “work together”
    - ... at least in theory ...
  - increased visibility for those package
  - increased recognition (?) for package authors via refereeing process
- Drawbacks:
  - updates and new packages only delivered through GAP releases, can reach users
  - storage wasted on unused packages (do we care?)

HOW DOES IT (CURRENTLY) WORK...

- for users?
- for package authors?
- for the GAP team?

> *Let's review it*

HOW DOES IT WORK FOR USERS?

- They get the packages together with GAP
- They usually also have to compile some packages they need (e.g. `io`, `cvec`, `semigroups`, etc.)
  - Off-topic: should provide compiled binaries
  - We do it for for Windows; also Frank's rsync distro for Linux
- If users need different package versions or different packages: requires non-trivial manual work

> *Status: OK, but really want `PackageManager`*

HOW IS IT FOR PACKAGE AUTHORS?

SUBMITTING A NEW PACKAGE

- upload `.zip` or `.tar.gz` archive with the package on a website they control
- send email to GAP team, requesting either to deposit the package or to submit it to refereeing[1]
  - revise based on feedback, repeat if necessary
- upload final archive, `README`, `PackageInfo.g`
- last one contains URLs of all 3 files, other metadata
- submit URL of `PackageInfo.g` to GAP maintainers

---

[1] package refereeing need its own talk...

HOW IS IT FOR PACKAGE AUTHORS?

SUBMITTING UPDATES

- upload new archive, `README`, `PackageInfo.g`
- the GAP team somehow detects the changes, downloads new version
- GAP team validates metadata, performs tests
- if successful, new version is imported
- otherwise, package author is notified, needs to repeat all steps 😤
  - must increment version to get a new try 😭

PAIN POINTS FOR PACKAGE AUTHORS

- Needs to maintain website with package, metadata
  - used to be normal for all academics, less so today
  - mitigated by **GitHubPagesForGAP**
- Don't forget steps (metadata valid? docs rebuilt? ...)
  - may use `ValidatePackageInfo` (easy to forget)
  - mitigated by **ReleaseTools**
- Will technical tests by GAP team pass?
  - we build package, run its test suite, etc.
  - mitigated by CI (“continuous integration”) tests

Status: under control; room for improvements (e.g. better docs)

GAP TEAM'S SIDE OF PACKAGE UPDATES

- A server downloads `PackageInfoURL` of all distributed packages regularly (via a “cron job”)
- validate and extract metadata
- new version detected? download it, continue
- repeat metadata validation
- run package tests three times: with minimal/all/default set of packages loaded
- run GAP's own tests with the new package loaded
- if anything failed, reject update (and notify someone about it...)
- otherwise, add package to distribution

FURTHER REMARKS ABOUT THE CURRENT SYSTEM

- Some relevant scripts can be found in **https://github.com/gap-system/gap-distribution/**
- New packages are added manually to **.../DistributionUpdate/PackageUpdate/currentPackageInfoURLList**
- Scripts download package archives, extract them, store results in private Mercurial (!) repositories
- The final package distribution archives are uploaded to **https://files.gap-system.org** and nowadays also to **https://github.com/gap-system/gap-distribution/releases/tag/package-archives**

WHY DOES IT NEED TO CHANGE?

- The current system doesn't work anymore...
- It relied on a single person (Alexander Konovalov) to spend tons of time and effort to keep it running
- (By the way: Thank you for all your work, Alex!)
- He doesn't have the time anymore
- Nobody else understands the system well enough,
  - *and* has the access to all involved systems
  - *and* has the time to take care of it
- And even if we had such a person: this is not a sustainable model. We need something else

MORE REASONS WE NEED A CHANGE

- There are other reasons a change is warranted:
- Tests are run on a “Jenkins server” few can access
- Authors can't see test logs, have to debug blind
- Lots of configuration hidden away on a server
- If server fails, will be hard to replicate it elsewhere

A BRUTAL SOLUTION?

- One possible solution: “We can't afford a package distribution anymore, let's simply cancel it.”
- Users should manually install packages (via `PackageManager`?)
- But there are drawbacks...
  - Reduces the out-of-the box usefulness of GAP
  - Some packages are *required* to even start GAP
  - We *benefit* from testing GAP with packages: quickly discover regressions in packages caused by GAP, and vice versa

A MORE MEASURED APPROACH

- *automate* as much as possible
- make it as *transparent* as possible, e.g. package test logs should be publicly readable
- *multiple people* should have the means to keep it running
- (granting) write access should be easy
- *document it thoroughly*, and keep the documentation up-to-date

> *This is to me the main goal of this workshop*

THOUGHTS ON A NEW MODEL (1)

> *Leverage GitHub as much as possible*

- some will complain GitHub could away any day / might cancel their free offering / isn't open source / ...
  - I think this is very unlikely resp. irrelevant
  - worst case: migrate to GitLab / own hardware / ...
- in the meantime, it has great benefits:
  - well-known, well-documented
  - we don't need to worry about security updates etc.
  - free compute resources!
  - easy access control
  - low barrier for contribution

THOUGHTS ON A NEW MODEL (2)

> *Put as much as possible in git repositories*

- Most of us are already familiar anyway
- Give us automatic backups, auditing of changes, trivial to undo bad changes
- Idea: package distro could be a one big git repo!
- `git pull` would update all packages (... well, plus `BuildPackage.sh`)
- But: lots of binary `manual.pdf` files would explode repo size quickly
- we could not ship those; but some really use them

THOUGHTS ON A NEW MODEL (3)

> *Put as much as possible in git repositories (except the distro itself)*

- So perhaps don't store the distro in a big git repo
- ... but we *can* store the list of current packages and their meta data in a repo
- also store a SHA256 *checksum* of each package archive, plus our own copy of that archive (on GitHub servers *and* on our own server)
- then we can re-assemble the package distro for any git revision from scratch if needed

THOUGHTS ON A NEW MODEL (4)

> *Use GitHub Actions to detect package update*

- GitHub provides free compute resources to us, called “GitHub Actions”
- we already use these extensively to perform CI (“continuous integration”) tests on GAP and many GAP packages (more in Wilf's talk)
- so: set up such an “action” which is run regularly and which scans for and handles package updates
- this way all code and config is again in a git repository

THOUGHTS ON A NEW MODEL (5)

> *Package updates = pull requests (PRs)*

- our cron job opens a PR on the package distro repo
- (... unless a PR for that package already exists; then either do nothing, or else update the PR?)
- this triggers CI tests for the update
- if successful, the PR can be merged automatically
- if it fails, then all logs are visible from there
- if we add GitHub usernames to `PackageInfo.g` meta data, we could even notify these authors
- PR can be updated by GAP devs, e.g. to re-run tests

THOUGHTS ON A NEW MODEL (6)

> *Also allow "manual" updates*

- You wrote a new package? Submit a PR!
- Your package moved? Submit a PR!
- For the above, perhaps offer a script or website taking URL of `PackageInfo.g`, opens PR for you
- Update multiple packages in lockstep? Submit a PR!
- Withdraw a package from distribution? Submit a PR!
- "Manual" PRs should require review by a human

HOW DO WE GET THERE?

- Let's discuss all of this today...
- ... and then make some decisions ...
- ... and work on implementing them during the rest of the week

THANK YOU!

Questions?