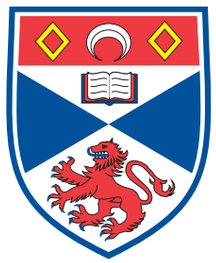
Continuous integration, package update mechanism and release management in GAP

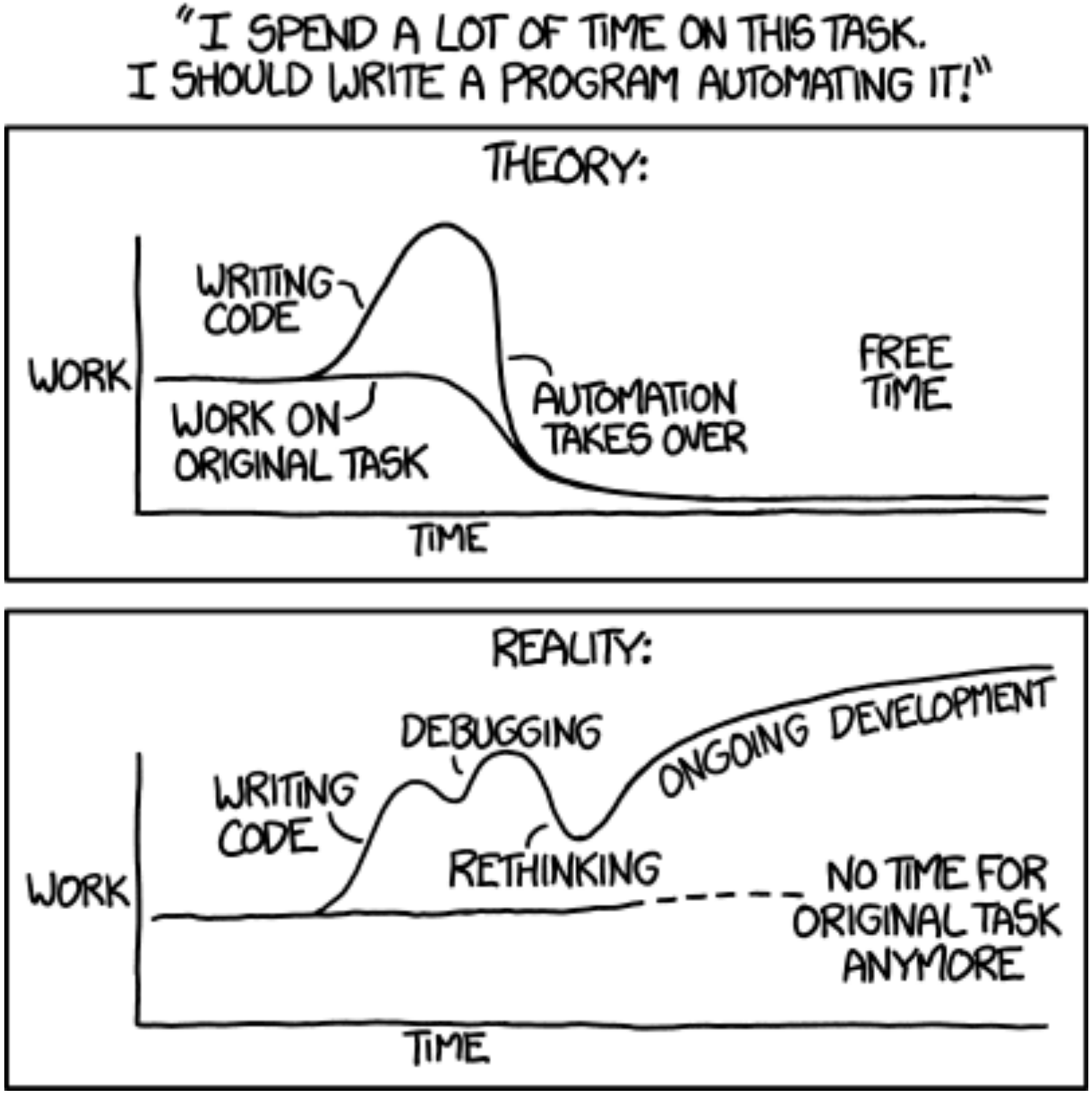
Alexander Konovalov



GAP Days 2014, Aachen, 25-29 August

To users, package authors and contributors to the core system

- A popular wisdom says that you can not get something which is simultaneously
    - reliable
    - quickly delivered
    - reasonably priced
- But we do really need:
    - reliably tested software
    - quick delivery cycle for bug fixes and new features
    - reasonable efforts to maintain it



We should automate it!

http://imgs.xkcd.com/comics/automation.png

Toolkit overview

- GAP Standard Test Suite
- Nightly Jenkins tests

- Package update system
- Jenkins interface to the package update system

- Wrapping and testing the release candidate
- Publishing the release

GAP Standard test suite

- consists of 8 targets **make test<name>**, where name is one of the following
  - **install** fast, uses **tst/testinstall.g**
  - **standard** longer, uses **tst/testall.g**
  - **manuals** extracts examples from manuals
  - **packages** runs tests if they are specified in the **PackageInfo.g** file
  - **packagesload** looks for *failures*, *crashes* and *warnings* while loading packages
  - **packagesvars** reports package variables
  - **updates** extracts and runs tests from dev/Updates entries (GAP.dev only)
  - **obsoletes** performs some simple checks with obsoletes disabled
- could be run anywhere where GAP is installed - cleanly separated from Jenkins CI system
- Also, with **gap -r -A tst/testinstall.g** etc. (e.g. on Windows)

Nightly Jenkins tests

- **test**{**install**/**standard**/**manuals**/**packagesload**} :
  - combinations of 32-bit/64-bit and with/without GMP
  - CentOS Linux i686, x86_64; Ubuntu x86_64; OS X (darwin13.3.0);
  - for the stable-4.7 and default branches
- For Windows, build it on a machine with Cygwin 32-bit, then run **testinstall.g** and **testall.g** without and with default packages on a Cygwin-free machine (just started to look at Cygwin64)
- Compiler tests with **-Werror** to catch compiler warnings:
  - **10 different compilers on 8 machines**: gcc **4.1.2** and **4.6.3** (CentOS), **4.8.1** (OpenSUSE), **4.8.3** (cygwin and cygwin64; OS X via homebrew), **4.9.1** (Ubuntu 14), clang **3.3** (Ubuntu 12) and **3.4** (OS X via Xcode)
  - combinations of 32-bit/64-bit, with/without GMP, with/without readline
  - for **stable-4.7**, **default** and **integration** branches
  - compile and run **testinstall.g** without packages, check that is reached the end and had no diffs

More details

- Demo: exploring the Jenkins interface
- Where are the scripts: **Makefile** and **tst/testutil.g**
- Where and what is documented
  - GAP.dev manual
  - Reference manual
  - GAP package “Example”

Package update system

- Based on ***package release repositories***
- These are different from development versions of packages!
- They keep only the history of “official” releases (changesets are just diffs between the releases)
- A version of a package may be marked as stable
- Easy to assemble merged packages archives giving a specification and test it w.r.t. different branches
- for example, **./mergePackages latest io=3.1 FR=no orb=stable**

Setting up package update system

- See **dev/DistributionUpdate/dist45/pkgupdate** directory
- Clone existing package release repositories
- Set environment variables in **setvarpkg** file
- Check URLs of **PackageInfo.g** files in the **currentPackageInfoURLList** file
- Call the following:
  - **./addPackages currentPackageInfoURLList** - to check for new/migrated packages
  - **./updatePackageInfoFiles** - to check for updated **PackageInfo.g** files
  - **./updatePackageArchives** - to import package archives that pass some initial checks (without actually running the package)
- May call **reportPackageVersions** script to get an overview

Package update steps

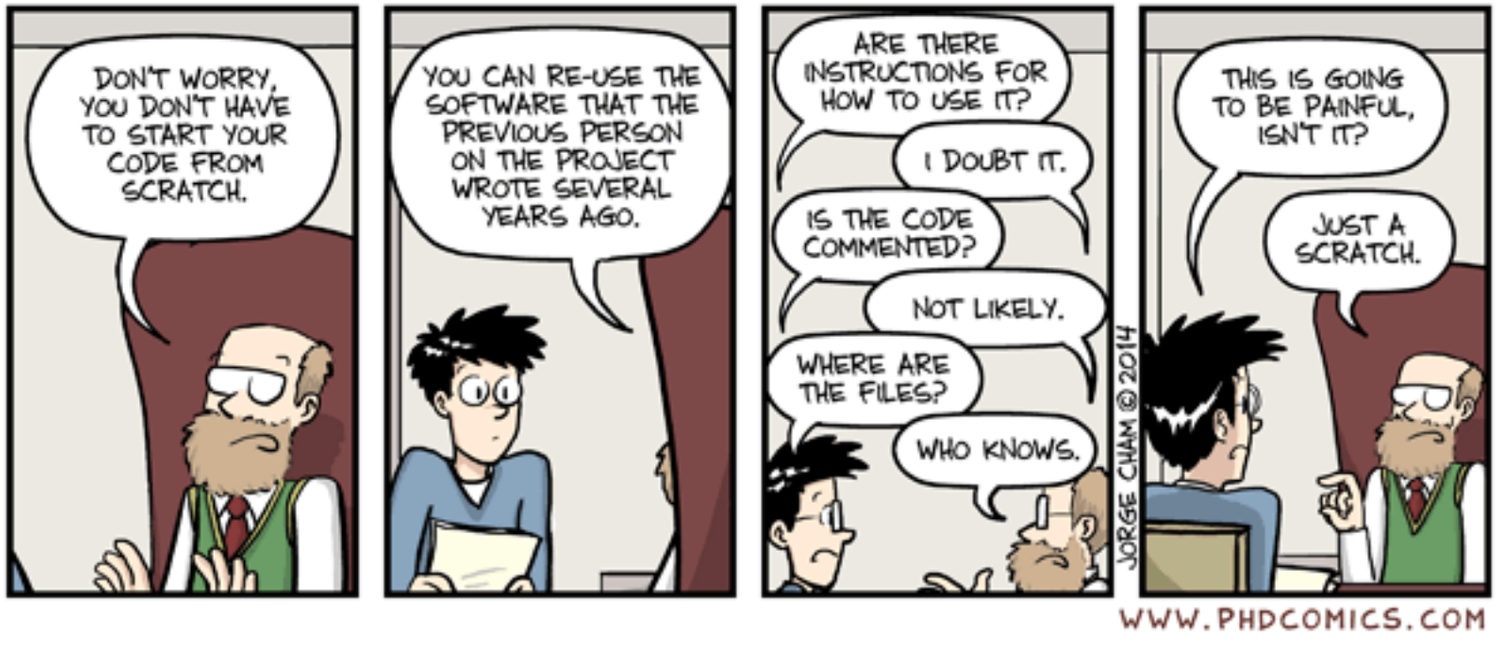
- The rest of the steps implemented in the **dev/DistributionUpdate/dist45/pkgupdate** directory is required only at the release stage
- There is also **storeLegacyPackage** script to retrospectively import old archives
- Further info:
  - **ValidatePackageInfo** in **lib/package.gi** and the Reference manual
  - GAP package “Example”
  - Beware: GAP.dev manual on package update needs revision!

Package updates in Jenkins

- Scheduled in the morning on each weekday
- Produces merged archives for nightly tests and release candidate wrapping
- Performs quick test (64-bit Linux with GMP) of this archive to check for immediate issues, and some “non-matrix” tests
- Jenkins interface demo

Health check:

- **60%** updated since 2013
- **20%** last updated in 2011-2013
- **3** packages **10+** years old



Compliance with failure detection

```
TestMyPackage := function( pkgname )
local pkgdir, testfiles, testresult, ff, fn;
LoadPackage( pkgname );
pkgdir := DirectoriesPackageLibrary( pkgname, "tst" );

# Arrange chapters as required
testfiles := [ "testfile02.tst", "testfileb04.tst" ];

testresult:=TestMyPackageDataLibrary();
for ff in testfiles do
  fn := Filename( pkgdir, ff );
  Print("#I  Testing ", fn, "\n");
  if not Test( fn, rec(compareFunction := "uptowhitespace") ) then
    testresult:=false;
  fi;
od;
if testresult then
  Print("#I  No errors detected while testing package ", pkgname, "\n");
else
  Print("#I  Errors detected while testing package ", pkgname, "\n");
fi;
end;
```

Wrapping and testing releases

- Done using Jenkins during weekends
- Produces update, minor and major release candidates
- Evaluating the tests and deciding to publish archives

- For us, it is the other way round:
  - *“We have too little updates for new release”*
  - *“OK, let’s move the release to a later date”*

Steps of release wrapping

- Several cleanly separated stages with archives as interfaces between them
- Steps may be separated in time and space
- Not tied up to Jenkins at all: may run anywhere, even on a laptop offline
- Only the 1st step depends on the version control system, and may be rewritten if we will switch to another VCS
- See **dev/DistributionUpdate/dist45** directory

Release wrapping scripts

- See **dev/DistributionUpdate/dist45** directory
- There are three targets **make update/minor/major** to wrap the release respectively from:
  - last release tag in the stable branch (update release)
  - tip of the stable branch (minor release, e.g. 4.7.6)
  - tip of the default branch (major release, e.g. 4.8.0)
- each of them reads version numbers from **setvarupdate/ setvarminor/setvarmajor** script respectively, and then calls the **doit** script

How-to doit

- What **dev/DistributionUpdate/dist45/doit** script does:

- Stage 1: checkout and archive the release branch
  - **setvar** - load environment variables (paths, version numbers etc.)
  - **checkouthg** - make a fresh clone of the the repository
  - **classifyfiles** - classify files into {text/binary}x{release/tools} or not shipped
  - **ziprelease branch** - wraps core system and tools archives
  - **zipmetainfo** - wraps metadata

- Stage 2: preparing the GAP core system
  - **unpackreleasebranch** - unpacks archives produced at step 1
  - **updateversioninfo** - insert version number, release date and other depending info
  - **fixpermissions** - dirs to 755, files to 644 and some executables to 755
  - **zipgapcore** - wraps gap core system and tools archives
  - **updatemetainfo** - wraps updated metadata

- Stage 3: merge GAP core with packages
  - **unpackgapcore** - unpacks archives produced at step 2
  - **unpackpackages** - unpack the merged packages archive from package update system
  - **checkpermissions** - check permissions for the content of the **pkg** subdirectory
  - **makedoc** - build main GAP manuals (**tut/ref/changes**)
  - **addmanualfiles** - adding the list of manual files to the metadata
  - **zipgapsourcedistro** - wrap GAP source distribution
  - **finalisemetainfo** - wrap metadata

Release publishing steps

- Evaluate results of regression tests
- Copy archives of the Gap source distribution to the GAP ftp server - now release is **PUBLISHED!**
- Prepare individual archives for *stable* packages and copy them to ftp as well:
    - Using **markAllLatestStable** script to mark all or **markStableRevisions** to mark several packages at a time, set “stable” bookmark (which may be removed with “**hg bookmark —delete …**” command)
    - For each stable release, set a bookmark with the timestamp of the GAP source distribution that contains it, e.g. with **markAllStableWithTimestamp gap4r7p6_2014_08_24-12_12**
        - In doubt, use **reportPackageVersions** to check!
        - Call **./mergePackages all stable** to wrap individual package archives (use “**<pkgname>=no**” if needed)
- Call **./updatePackageDocs** and **./writePackageWebInfos** to update package documentation and generate data for package pages using their **PackageInfo.g** files
- Call **./CopyToFtpServer** to copy individual package archives to the ftp server and **./CopyToWWW2** to copy the autogenerated data to a clone of the GAP website repository
- Copy manuals of updated packages to the public or testing website
- This is documented in the GAP.dev manual: see Chapter “*Preparing GAP Releases*”, Section “*Releasing a new version*”

Prepare the website

- Archives made public => we have the release. Version number can not be re-used!

- Have to update the GAP Website

- This is documented in the GAP.dev manual: see Chapter "*Preparing GAP Releases*", Section "*Releasing a new version*"

- More general description of the GAP website is contained there in the chapter "Maintaining the GAP website"

Spread the word!

- Made the updated GAP website live (**bin/updateGapWWW.sh** on yin)
- Produce alternative distributions (http://www.gap-system.org/Download/index.html#alternatives)
  - rsync-based - for Linux; BOB - for Linux and OS X; Windows Installer
- Announce in the Forum (for major releases - wait for alternative installers to be available too)
- Update package dependencies diagram:

  http://alexk.host.cs.st-andrews.ac.uk/gap/GAP_Packages.html
- Hope to eventually see this version cited at Google Scholar (capable of tracking citations by version): http://bit.ly/gap_citations
- Provide platform for recomputable experiments at recomputation.org
- Support users and encourage them to upgrade in GAP Forum, GAP Support, and also at Mathematics Q&A site from the StackExchange framework:

  http://math.stackexchange.com/questions/tagged/gap
- Tweet at http://twitter.com/gap_system

YOUR LINK GOT A LOT OF INTERACTION.

Save the date: First GAP Days, August 25-29, 2014 (RWTH Aachen University) gapdays2014.coxeter.de

02:53 PM - 04 Jul 14

118 views 1 favorite 1 link visit 2 Retweets

THIS TWEET SPARKED SOME INTEREST.

Primitivity, switching and rigidity: new preprint by Peter Cameron & Pablo Spiga uses GAP arxiv.org/abs/1407.5288 #GroupTheory #combinatorics

11:50 AM - 29 Jul 14

21 views 1 favorite 1 link visit

WELL DONE! THIS TWEET GOT PEOPLE EXCITED.

How to play games with finite groups? See new preprint by @danaernst & N.Sieben arxiv.org/abs/1407.0784 accompanied by jan.ucc.nau.edu/ns46/GroupGenG…

11:57 AM - 08 Jul 14

74 views 4 favorites 2 link visits 2 Retweets

WELL DONE! THIS TWEET GOT PEOPLE EXCITED.

Abstract of Manuel Delgado's talk on GAP packages at @sagemath and #Python meeting (30 May-1 June 2014) in Cáceres eweb.unex.es/eweb/sage/sage…

05:13 PM - 17 Jun 14

21 views 1 favorite 1 link visit

WELL DONE! THIS TWEET GOT PEOPLE EXCITED.

Added to GAP Google Scholar profile: P.Maier, @robstewartUK & P.Trinder, HdpH DSLs for Scalable Reliable #Computation macs.hw.ac.uk/~rs46/papers/h…

11:50 AM - 07 Jul 14

47 views 1 Retweet

PEOPLE REALLY LIKED THIS PHOTO.

Google Scholar is good at finding research software citations! Follow new GAP citations at bit.ly/gap_citations pic.twitter.com/tLDqt2uEIs

11:39 AM

13 View

YOUR LINK GOT A LOT OF INTERACTION.

GAP Forum, today: congratulations @RamonEstebanR @encosllo A.Ballester-Bolinches - acceptance of the package 'permut' mail.gap system.org/pipermail/foru…

12:42 PM - 05 Jun 14

32 views 1 favorite 1 link visit 3 Retweets

NICE. PEOPLE GOT INTO THIS TWEET.

New update for the current 18th edition of Kourovka Notebook (Unsolved Problems in #GroupTheory) has been published: arxiv.org/abs/1401.0300

10:40 AM - 28 Jul 14

12 views 1 favorite 1 link visit

Recomputation.org

Ian Holmes
@ianholmes

Follow

You can download our code from the URL supplied. Good luck downloading the only postdoc who can get it to run, though #overlyhonestmethods

4:52 PM - 8 Jan 2013

313 RETWEETS **98** FAVORITES

It should be as easy to reproduce a computational experiment as to reproduce the chemical reaction from a textbook by mixing baking soda and lemon juice in the kitchen

Recomputation Manifesto by Ian Gent (arXiv:1304.3674)

- Computational experiments should be persistent
- Their recomputation should be very easy, supported by tools and repositories, so it should be easier to make experiments recomputable than not to
- Virtual machines as the way to recomputability

```
Install Vagrant and Virtual Box
mkdir anydir ; cd anydir
vagrant init <experiment_id> <URL>
vagrant up
```

See http://recomputation.org/ecai2014/

Suggestions for today

- Explore all of these in more details
- See what could be improved
- Evaluate the latest release candidates for GAP 4.7.6 and GAP 4.8
- Explore release publishing process
- Look at regression tests for HPC-GAP
- Learn how to find needed information using Jenkins
- Learn how to edit GAP website and try to made some improvements
- ???