Data Processing

Bruno Merín

Herschel Lessons Learned Meeting

ESAC, 28-30 May 2013

Topics

- User requirements
- Scope of the software: internal and external/HIPE
- Selecting the programming language
- HIPE: Software, documentation, user friendliness and support
- Management of the development process
- Model of data processing: centralized/integrated or ICC-driven?

User requirements

Involve **external users** (HUG?/funded DPUG?) in the development from the beginning

- Spend a few months at the beginning of the development writing down good requirements for IA
- Internal astronomers (calibration scientist) cannot provide all the requirements you need
- Once there is software, use scrum sessions to have user inputs for all the early decisions
- H34, H51, H102, H113, H140, H179

Keep a constantly evolving set of user requirements available through development

- An alternative origin of ever-updated requirements could be open JIRA tickets from external users.
- H51, H81, H92, H140

Base the user requirements repository in use cases as realistic as possible

- It is important to get different inputs from the various kinds of users: archival users, beginner users (just use reduced data), `hard' users (they process the data in HIPE) and developers/contributors (internal project astronomers).
- H7, H92, H140

Scope of the software: internal vs external

Define well in advance of development the scope (internal or external) of the packages

- H8, H118

Plan **extra resources** for software that is going to be given to external users

- For internal consumption software (like the SPG infrastructure) the human-power needed, as well as the documentation and testing is certainly smaller than for an external consumption software (like HIPE).
- Scheduling both systems in parallel translated to the non-operational one (HIPE) being under-prioritized.
- But in the end it became a very visible (and not a very good) face of the project.
- H8, H21, H118, H140

Selecting the programming language

Make a prospective study involving external users of emerging technologies/languages/ libraries before selecting a common programming language for the project

- H12, H140

Choose a language that allows professional integration and scaling for large projects

- H12, H140

And choose a language that users will like for external consumption unless you plan them to do everything via GUIs (as in e.g. Hspot).

- H12, H140

Be ready to **change** that specification for parts of the software to adapt to changes (like e.g. porting/aliasing now to numpy/scipy/pyfits in HCSS).

HIPE: s/w, doc, user friendliness, support

For designing external consumption software, have constant external user consultation

- H34, H72, H112, H113, H123, H140, H155

Plan releases with incremental functionality and only release to external users basic, well tested and well documented software

- H21, H110, H123

Simplify the **user workflows**: search and retrieve data, analyse it, reprocess it and save the results

- H63, H80, H110, H123, H150, H155

Late attempts to communicate with users other than via the helpdesk do not work well

- H102, H125

Enable supported user sessions on virtual machines to overcome memory needs for large maps H68

Management of the development process

Enable as much **communication** as possible within the project

- CSDT meetings, co-locations, project plans, Twiki, JIRA, webex, webinars, Skype

Decouple where needed software packages with very different needs and goals

- H8, H21, H35, H155

Give astronomers/developers training on advanced java computing techniques

- H73

Try to avoid instrument/software delivery schedule clashes (e.g. the pools for production)

- H73, H150

Enforce SQA standards from the beginning (at integration time?)

- H98,

Have a centralized integration authority that schedules deliveries based on large-scale science goals (empower the DPUG?/HUG?)

- H112, H138, H117, H155

Reuse software to reduce its cost and help new users (e.g. HerschelSpot)

- H116

Model of DP: centralized or ICC-driven

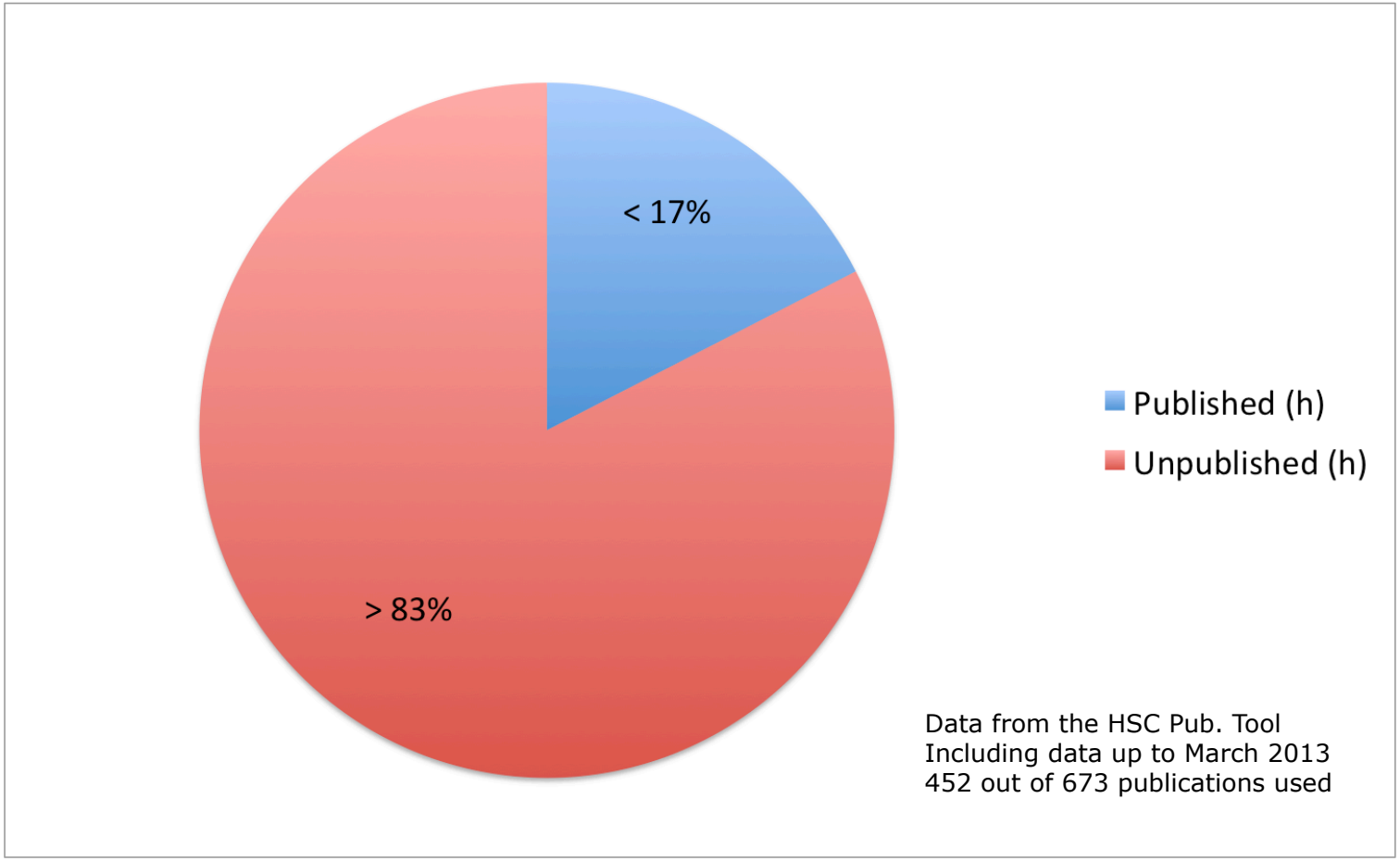
For a project with little chance of generating usable products by sytematic pipelines during operations, the option to have ICC-driven DP might be better, although it requires more work by the end user for different instruments

- H116, H102

For a project with chances of generating usable products by sytematic pipelines during operations, the option of centralized/integrated DP might be the best, since it makes life easier to end and archival users (although it might be more expensive?)

- H116, H102

Model of DP: centralized or ICC-driven



- Most data are still to be published: so then look after KPs and beginner/archival users
- ICC/NHSC post-operations phases are smaller than the HSC one and the HSA will stay

Lessons learned on data processing

- User requirements -> external users
- Scope of the software: internal and external/HIPE -> human-power
- Selecting the programming language -> change
- HIPE: Software, documentation, user friendliness and support -> workflows
- Management of the development process -> communication
- Model of data processing: centralized or ICC-driven? -> timescales

Get end users in the loop