THE ATLAS DATA MANAGEMENT SOFTWARE ENGINEERING PROCESS

Mario Lassnig (CERN PH-ADP)
on behalf of the ATLAS collaboration

mario.lassnig@cern.ch
ph-adp-ddm-lab@cern.ch

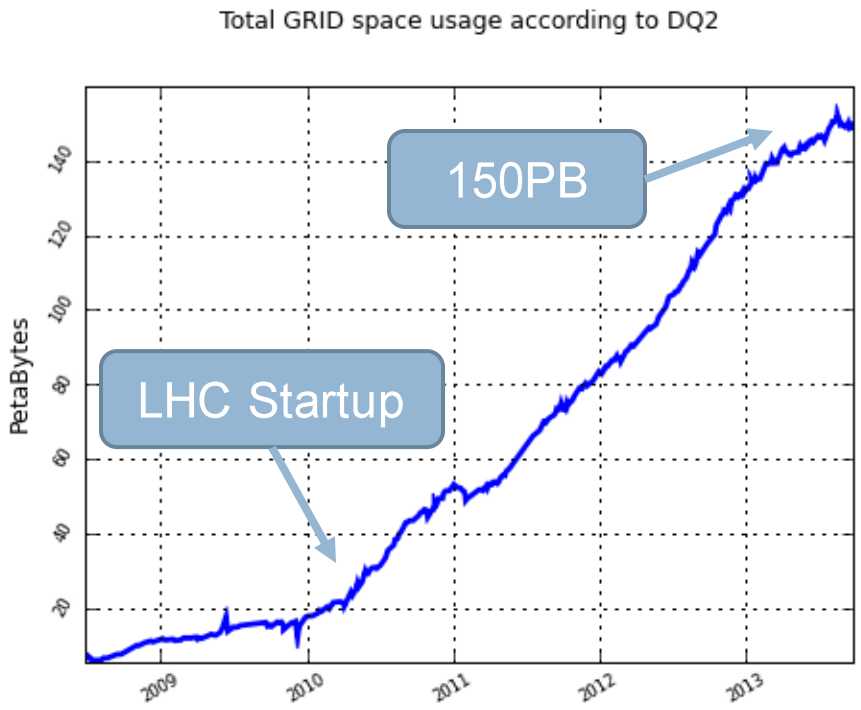
CHEP, 2013-10-17, Amsterdam, NL

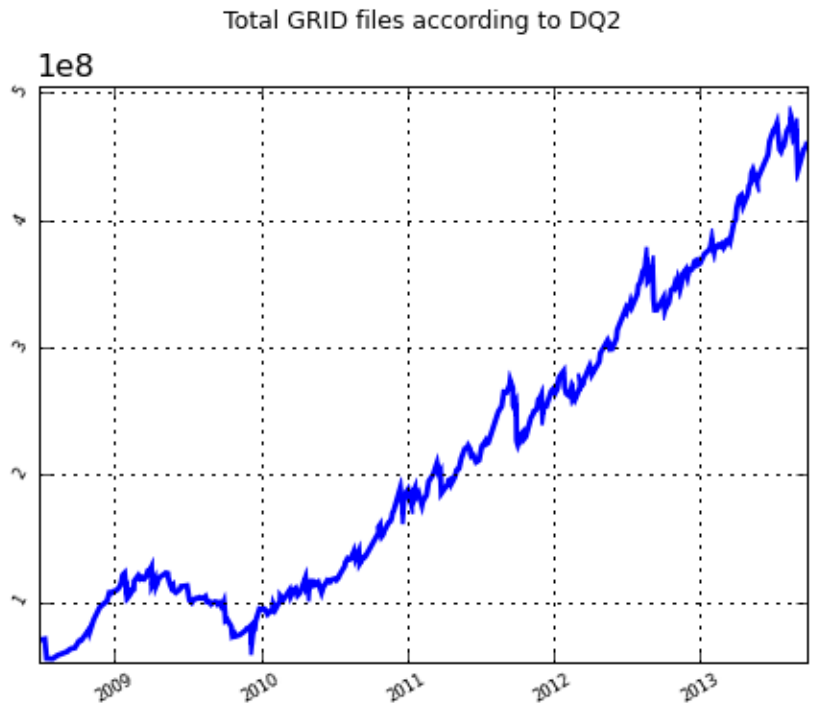
Overview

- ATLAS Data Management
- Requirements engineering
- Implementation and test
- The human factor
- Conclusion

ATLAS Data Management

- Currently accumulating 40PB of data annually
  - Data is written to files, and aggregated into datasets
  - Distributed to data centres in the WLCG





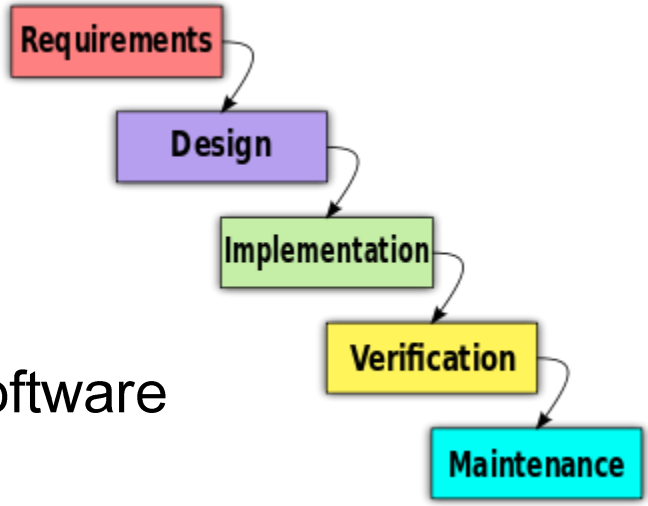
A new data management system?

- Next-generation data management system
  - Rucio: cf. Vincent Garonne's talk from Tuesday
- Ensure system scalability beyond LHC Run-2
- Reduce operational overhead
- Support new ATLAS use cases
- New technologies, paradigms, middlewares

- Building Rucio required a new approach to software engineering
  - Existing Domino/Jenga approach not feasible

Software process

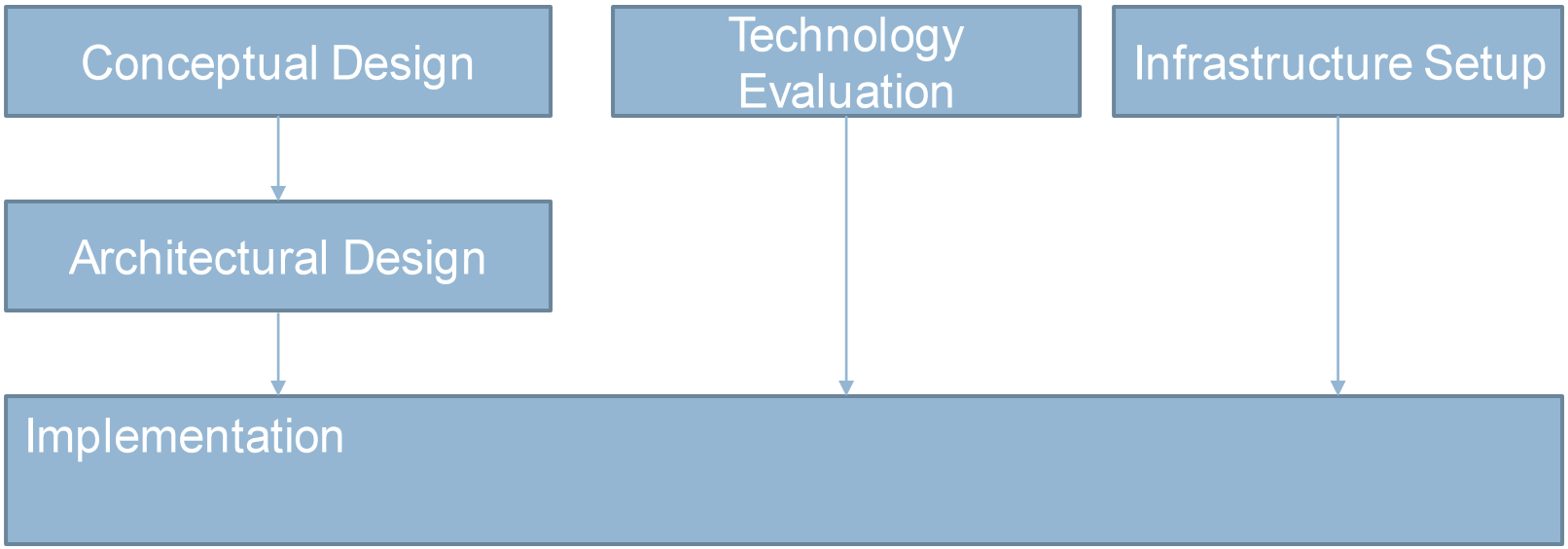
- Classic model: Waterfall
  - Fully complete each step
- However, not really applicable
  - Clients are required to state explicitly what they want upfront!
  - Critical dependencies on third party software e.g., a file transfer service



- Alternatives, such as Agile, Spiral, or Prototyping also have their own problems
  - Too consumer/business-focused, etc..
- Instead, we opted for our own modified waterfall model

Modified software process

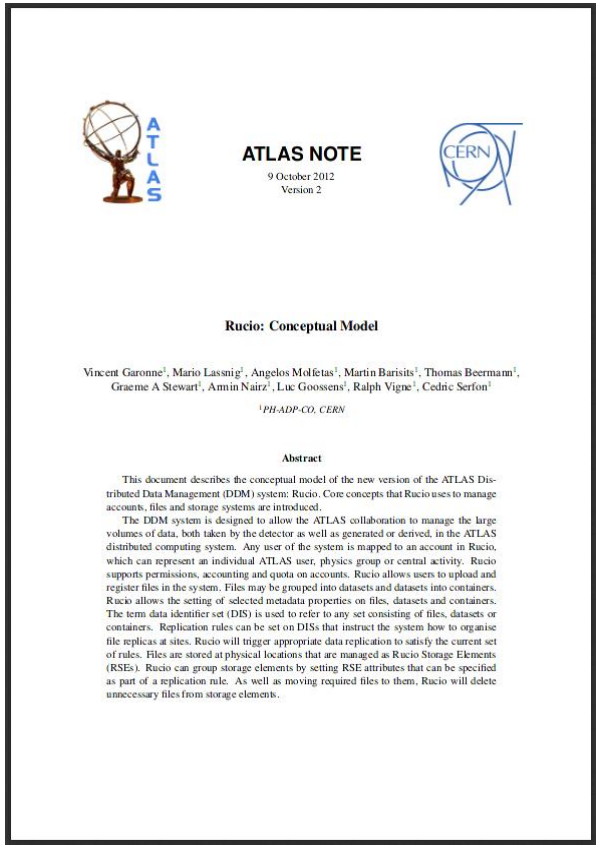
- Serialise specification and design
- However, in parallel, construct the foundation for the later stages of the process
  - Evaluation of technology – proof-of-concept
  - Setup of the infrastructure
- Couple the implementation and verification
  - Maintenance follows naturally



Requirements engineering

- Started with a whitepaper
- Sent out surveys to our clients
  - Tier-0, AMI, data preparation, PanDA, endusers…
- Focused sessions with each client separately
- Technical meetings with other LHC experiments
- Workshop on the evolution of ATLAS Data Handling
- Explicit use case descriptions

- Resulting in the Rucio conceptual design
  - Signed off sentence-by-sentence



ATLAS NOTE

9 October 2012
Version 2

Rucio: Conceptual Model

Vincent Garonne[1], Mario Lassnig[1], Angelos Molfetas[1], Martin Barisits[1], Thomas Beermann[1], Graeme A Stewart[1], Amnin Nairz[1], Luc Goossens[1], Ralph Vigne[1], Cedric Serfon[1]

[1]*PH-ADP-CO, CERN*

Abstract

This document describes the conceptual model of the new version of the ATLAS Distributed Data Management (DDM) system: Rucio. Core concepts that Rucio uses to manage accounts, files and storage systems are introduced.

The DDM system is designed to allow the ATLAS collaboration to manage the large volumes of data, both taken by the detector as well as generated or derived, in the ATLAS distributed computing system. Any user of the system is mapped to an account in Rucio, which can represent an individual ATLAS user, physics group or central activity. Rucio supports permissions, accounting and quota on accounts. Rucio allows users to upload and register files in the system. Files may be grouped into datasets and datasets into containers. Rucio allows the setting of selected metadata properties on files, datasets and containers. The term data identifier set (DIS) is used to refer to any set consisting of files, datasets or containers. Replication rules can be set on DISs that instruct the system how to organise file replicas at sites. Rucio will trigger appropriate data replication to satisfy the current set of rules. Files are stored at physical locations that are managed as Rucio Storage Elements (RSEs). Rucio can group storage elements by setting RSE attributes that can be specified as part of a replication rule. As well as moving required files to them, Rucio will delete unnecessary files from storage elements.

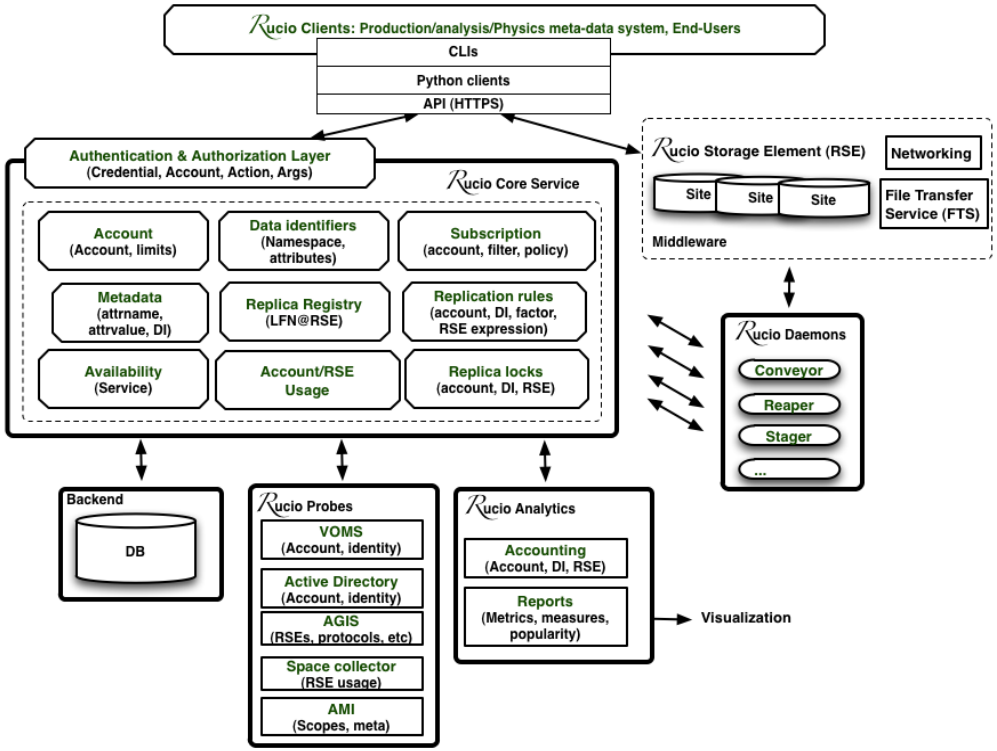
Meanwhile…

- Several technologies were evaluated
  - No specifications yet as the conceptual model not finished
  - However, experience from DQ2 leads to educated guesses

- Database management systems
- Database abstraction layers
- Communication protocols
- Source code management
- Queuing and notification
- Integration with external systems
- Testing and analysis
- Deployment

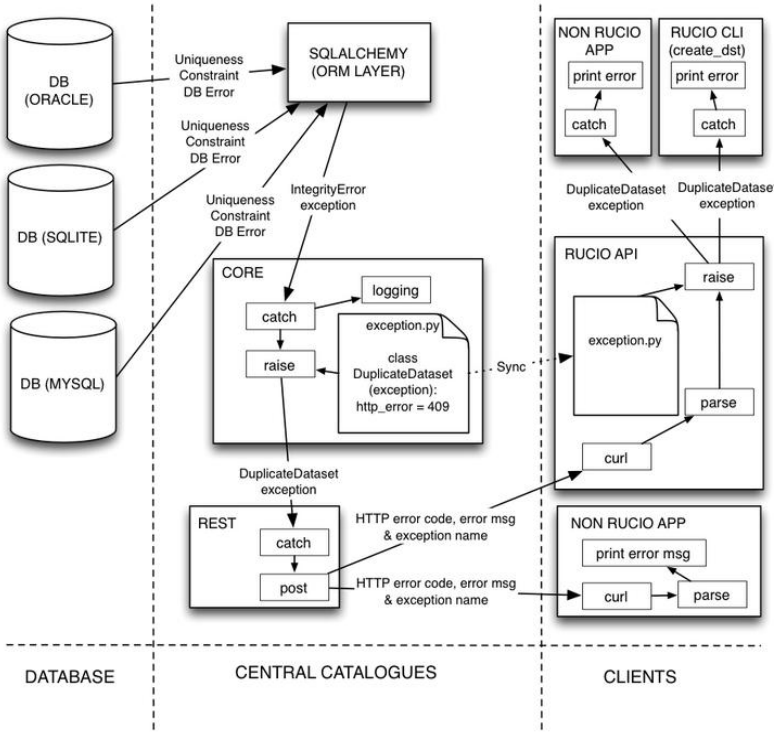
Design and architecture

- Only after the conceptual model was finished
- First results from the technology evaluation
- Component interaction based design
  - Sequence/Flow diagrams for the most critical parts



Design and architecture

- Only after the conceptual model was finished
- First results from the technology evaluation
- Component interaction based design
  - Sequence/Flow diagrams for the most critical parts



Implementation

- We decided to follow a single rule:

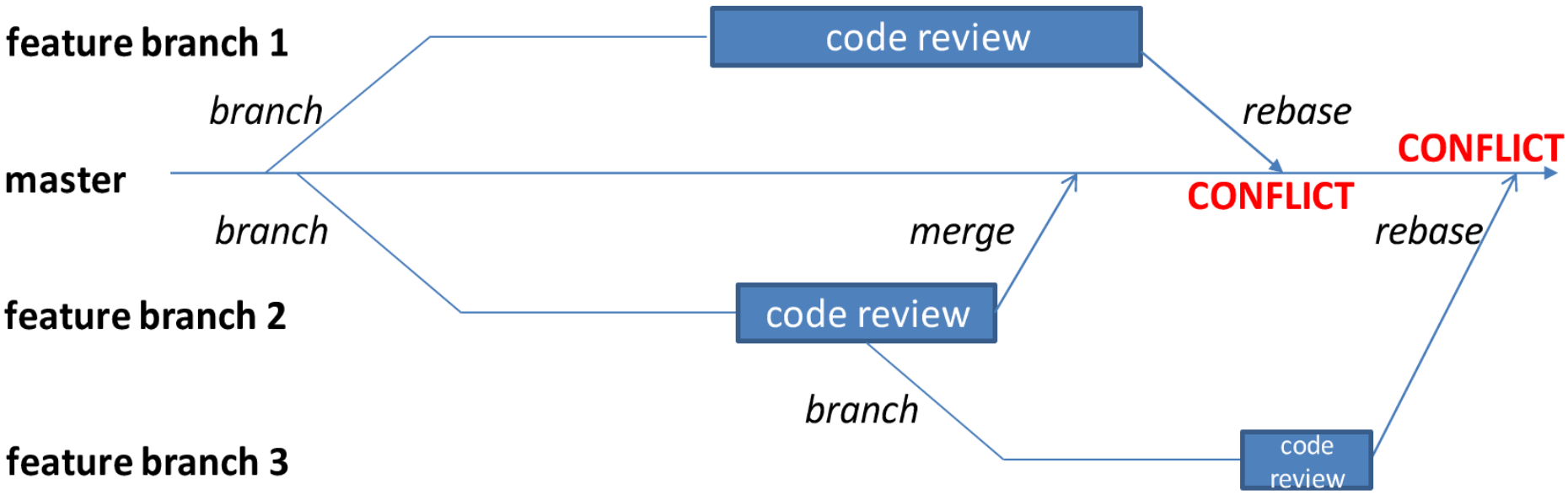
  Rucio must be *always releasable*

- This poses several requirements
  - We must be confident that the software works
  - No one is allowed to modify the software unsupervised
  - New features are implemented separately

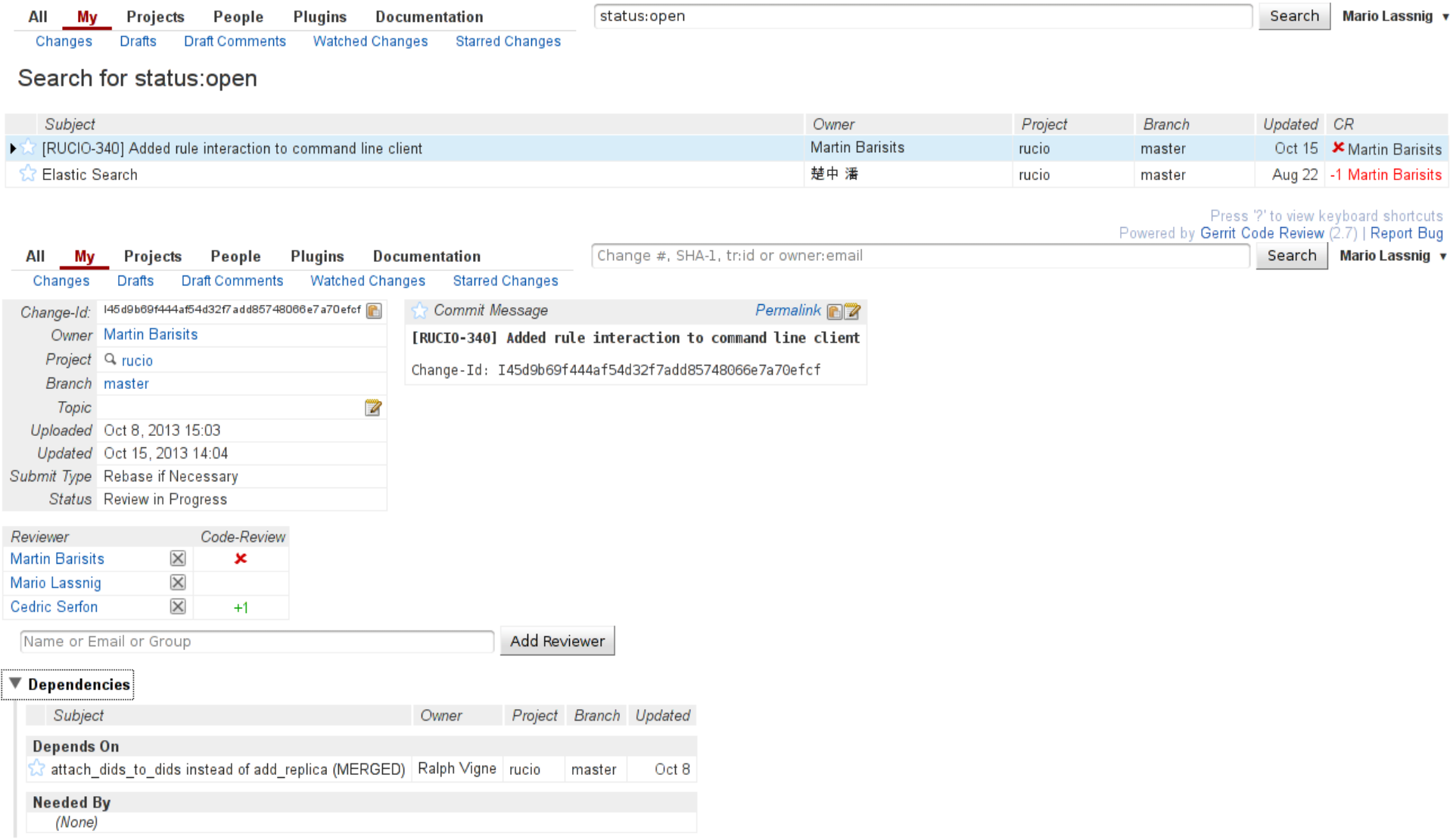
Implementation

- Test-driven development
- Approval of features
  - Visual scanning of the source code
  - Local test of the source code
  - Explicit vote
- Feature branches are rejected without review, in case of
  - Missing test cases
  - Missing documentation or comments
  - Wrong coding style
- We use *git* (distributed version control) and *gerrit* (code review) to enforce this workflow
  - *http://git-scm.com/*
  - http://code.google.com/p/gerrit/

The branching strategy



Gerrit – Project overview



Gerrit – Patch overview

Reference Version: Base

Patch Set 2 5c4173eb8cd6d54a203c93029be2827736f71541

Author	Martin Barisits <martin.barisits@cern.ch> Oct 8, 2013 15:03
Committer	Martin Barisits <martin.barisits@cern.ch> Oct 9, 2013 16:58
Parent(s)	9776fc31ec166d95c6827a9670686f7eff1d9ae4 attach_dids_to_dids instead of add_replica
Download	checkout | pull | cherry-pick | patch | Anonymous HTTP | SSH | HTTP |

```
git fetch ssh://mario@atlas-gerrit.cern.ch:29418/rucio refs/changes/30/830/2 && git cherry-pick FETCH_HEAD
```

Review | Abandon Change | Rebase Change

	File Path	Comments	Size	Diff		Reviewed
▸	Commit Message			Side-by-Side	Unified	
M	bin/rucio	**2 comments**	+155, -1	Side-by-Side	Unified	
M	lib/rucio/api/rule.py		+4, -0	Side-by-Side	Unified	
M	lib/rucio/client/subscriptionclient.py		+1, -1	Side-by-Side	Unified	
M	lib/rucio/core/rule.py		+24, -18	Side-by-Side	Unified	
M	lib/rucio/tests/test_rule.py		+0, -1	Side-by-Side	Unified	
M	lib/rucio/web/rest/did.py		+1, -1	Side-by-Side	Unified	
M	lib/rucio/web/rest/rule.py		+2, -0	Side-by-Side	Unified	
M	lib/rucio/web/rest/subscription.py		+1, -1	Side-by-Side	Unified	
			+188, -23	All Side-by-Side	All Unified	

Patch Set 1 20925691d9dbca76197c91eda2c9055267d6184b

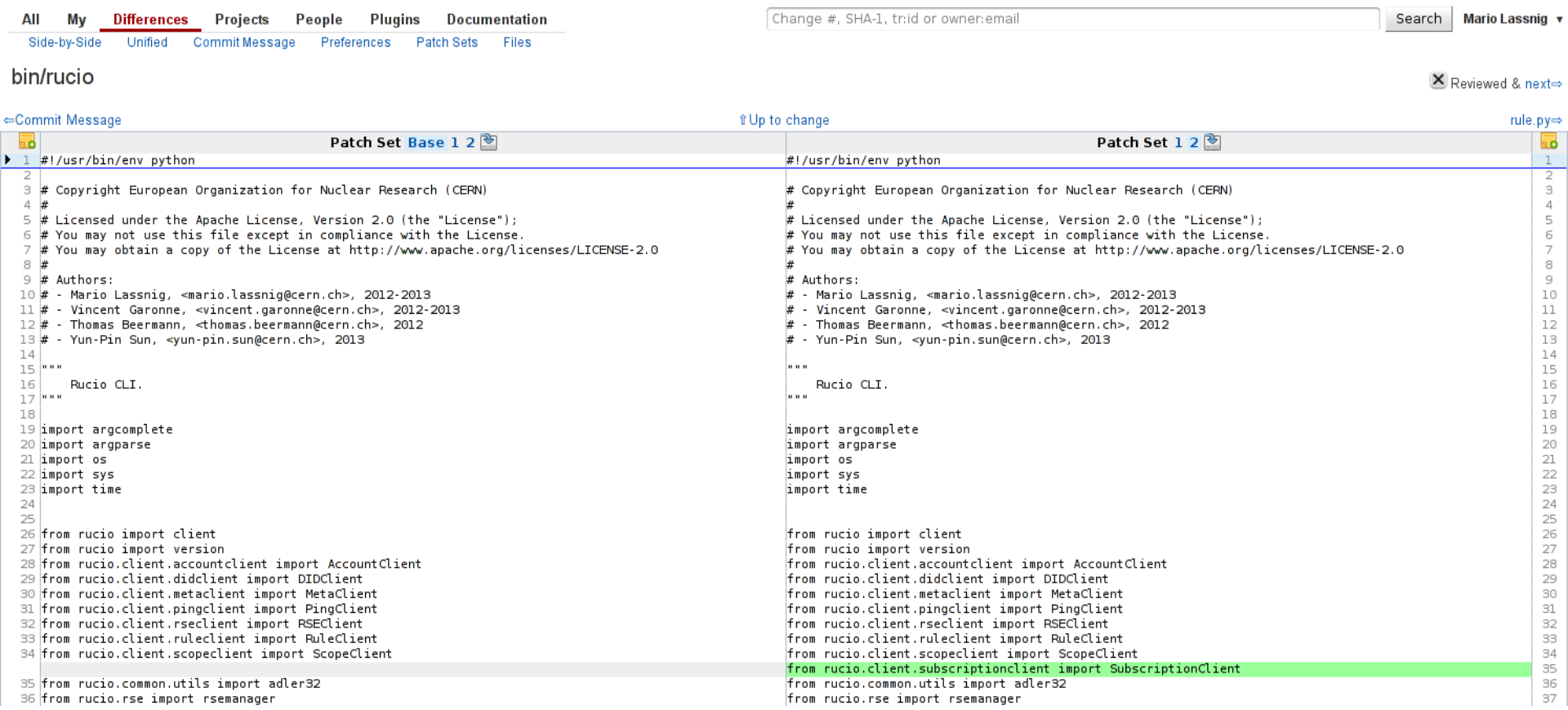
Comments Expand Recent | Expand All | Collapse All

Mario Lassnig Patch Set 1: (2 comments) typo	Oct 8 15:13
Martin Barisits Uploaded patch set 2.	Oct 9 16:58
Cedric Serfon Patch Set 2: Code-Review+1 (1 comment)	Oct 10 16:06
Martin Barisits	Oct 15 14:04
Patch Set 2: Code-Review-2 (1 comment)	

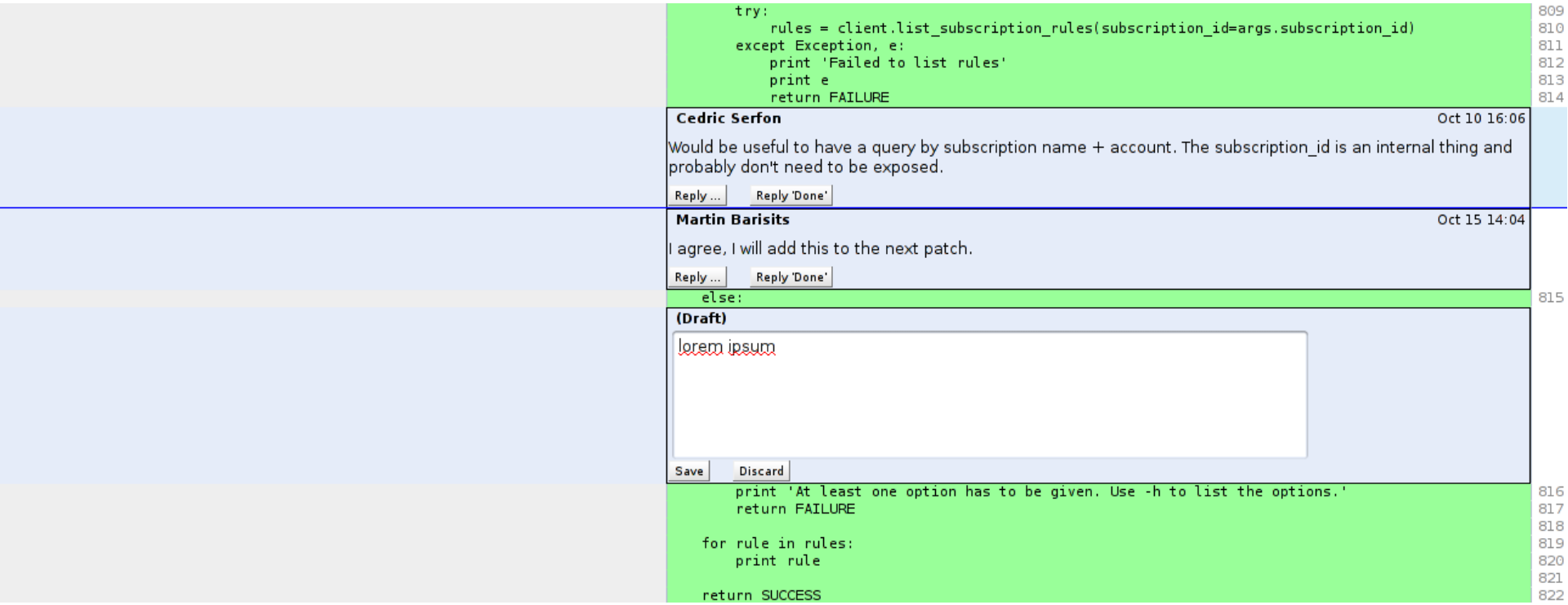
Add Comment

Press '?' to view keyboard shortcuts
Powered by Gerrit Code Review (2.7) | Report Bug

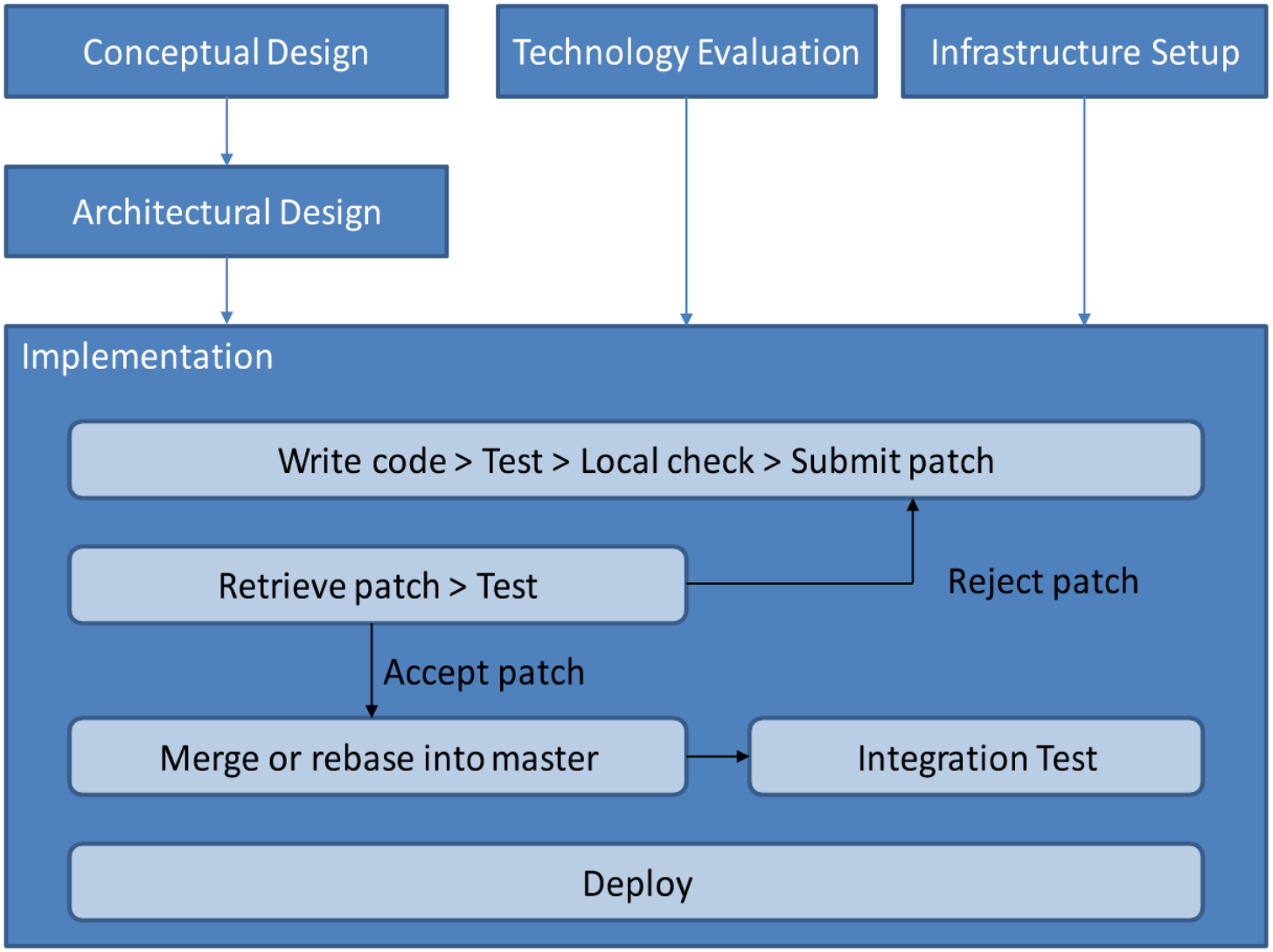
Gerrit – Patch comparison



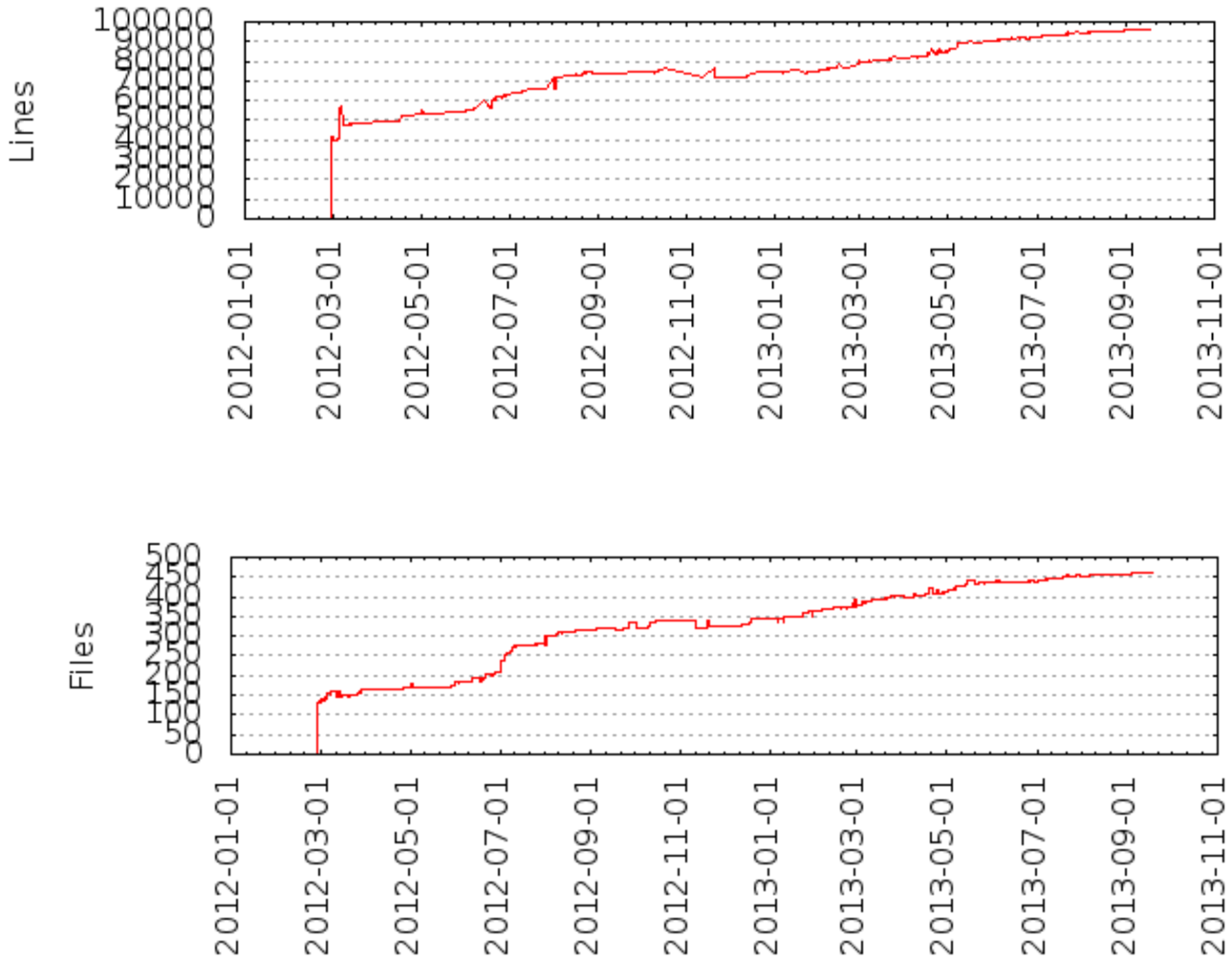
Gerrit – Patch comments



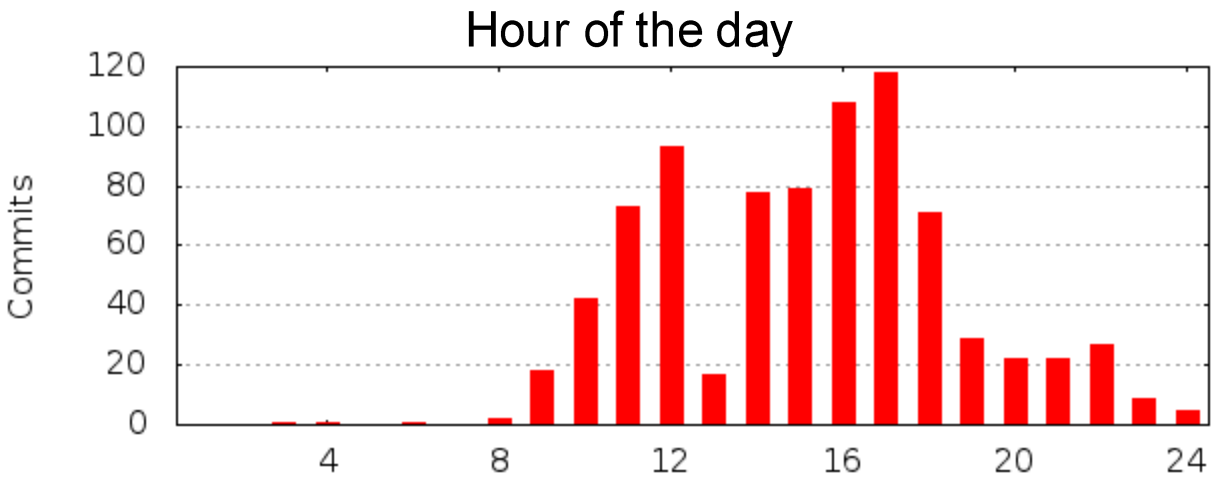
The full process

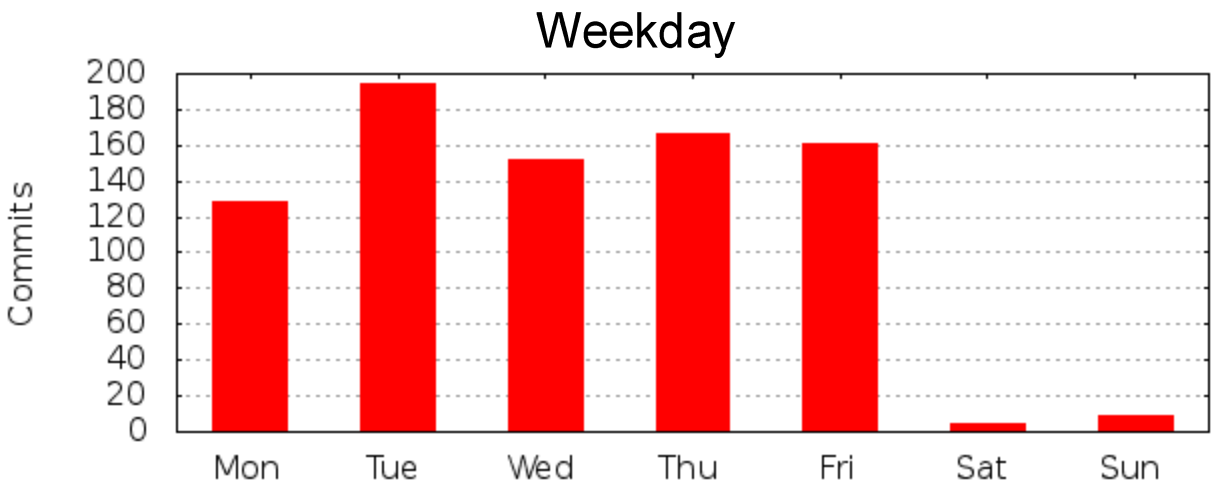


Some development statistics

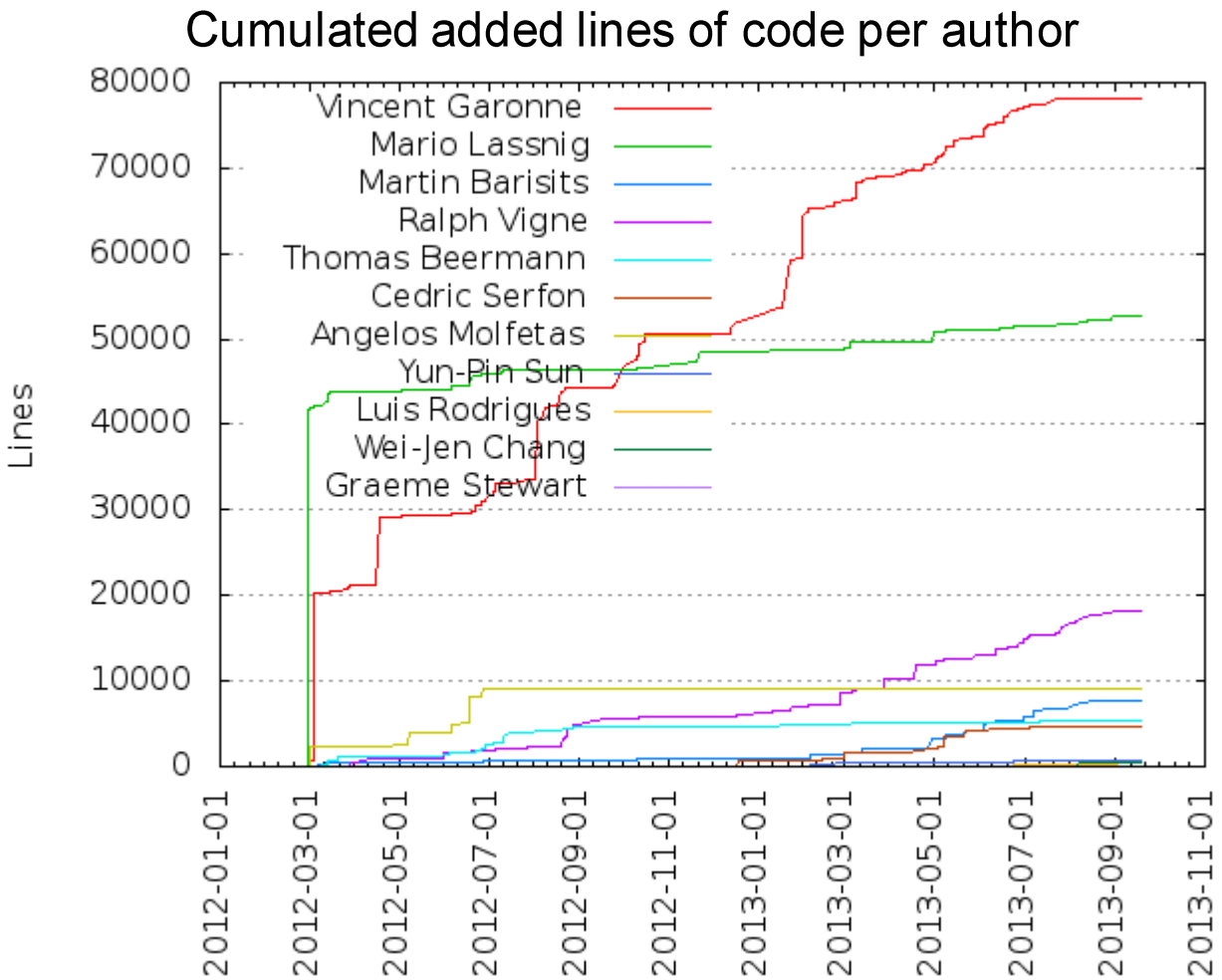


Some development statistics

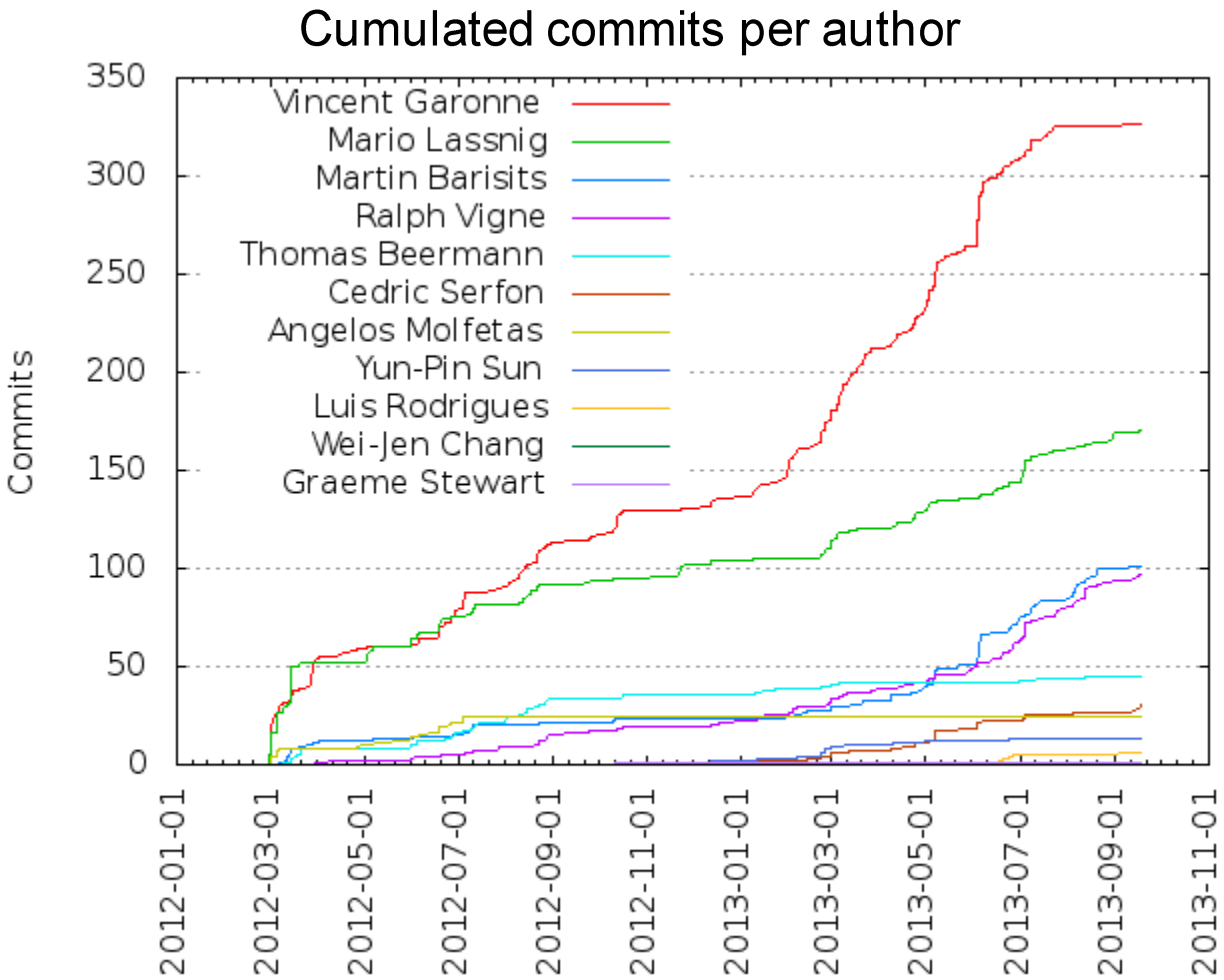




Some development statistics

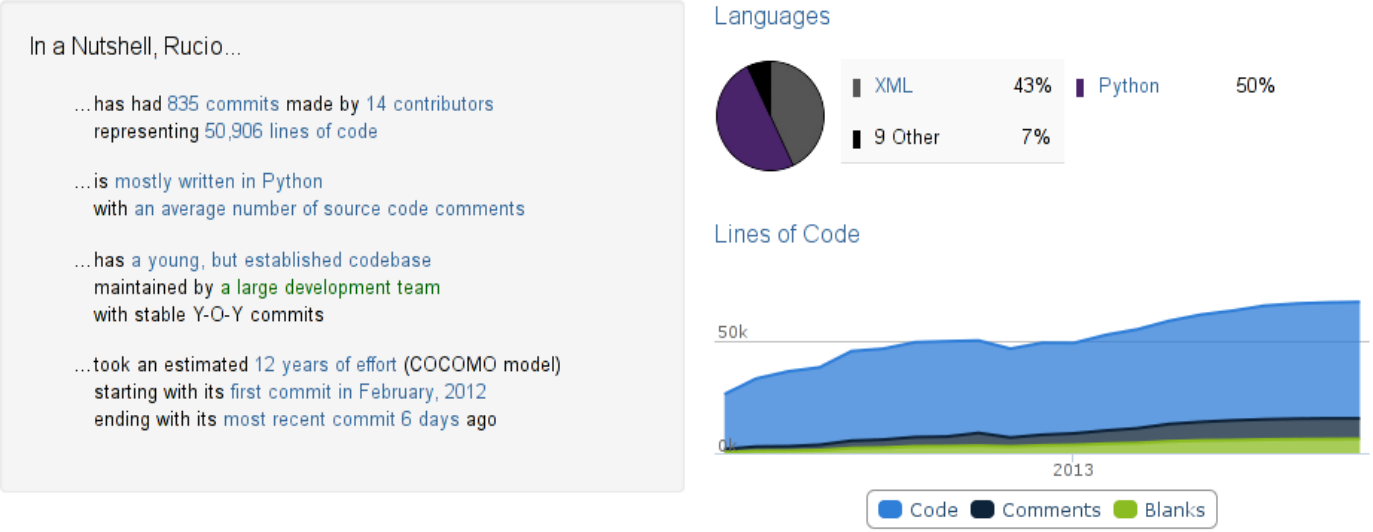


Some development statistics



What others think

- http://ohloh.net/
  - Open source software directory
    - Including Firefox, Apache HTTP, Linux, …
  - Automatically collects code and analyses projects



The human factor 1/2

- Won't this take a long time to get my changes into the source code?
  - Only ramp-up in the beginning
- What happens if I need to quickly fix something?
  - Self-approval of patches (public embarassement)
- I really don't like that someone else is judging my work.
  - Tough luck.

The human factor 2/2

- Writing test cases is bothersome.
  - If you don't have time now, then you will not have time later anyway.
- We will never agree on a common coding style.
  - We did and software now enforces it.
- Branching? Merging? What is this?
  - Git tutorials
- How do we deal with new members in the team?
  - Up to date documentation
- How can we deal with outside contributions?
  - Submit patches to gerrit

Conclusion

- Rucio is the new data management system of ATLAS
- Its development follows a modified waterfall process
  - From conceptual design
  - Via architectural design
  - To test-driven and peer-reviewed implementation
- Initial inhibition threshold well overcome
  - Social uncertainties almost negligible
  - Conduct enforced where possible by software
- Resulted in
  - High throughput of essentially error free code
  - Easy injection of new engineers into the team

THE ATLAS DATA MANAGEMENT SOFTWARE ENGINEERING PROCESS

Mario Lassnig (CERN PH-ADP)
on behalf of the ATLAS collaboration

mario.lassnig@cern.ch
ph-adp-ddm-lab@cern.ch

CHEP, 2013-10-17, Amsterdam, NL