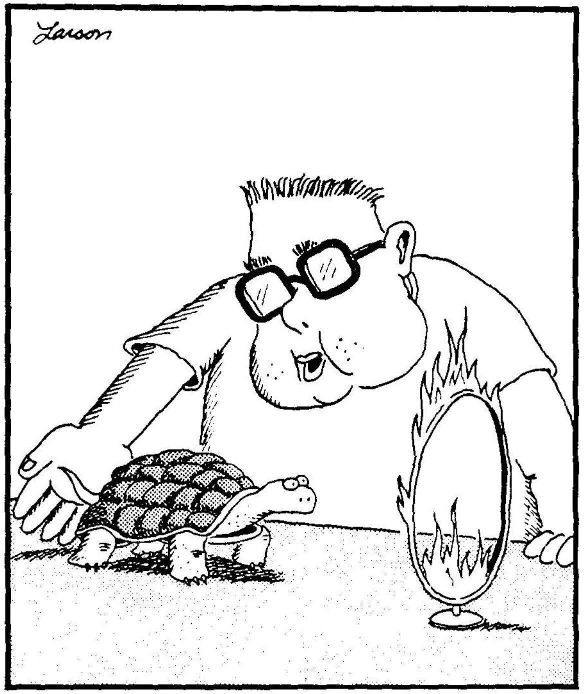
Sometimes you need to think big thoughts

Not all performance problems will be solved with an incremental approach

- "Do we have to do it this way?"
- "Is there a better way to do this?"



"Through the hoop, Bob! Through the hoop!"

- "Do I have to do this at all?"

Traditional example: Sorting a new deck of cards

More

Method 1: Pattern recognition

- There are a finite number of possible arrangements
- Find which one you have, and then reorder
- 52! = $4x10^{66}$ so will need on average $52*4x10^{66}/2$ comparisons

Method 2: Bubble sort

- Scan through, finding the smallest number
- Then repeat, scanning through the N-1 that's left
- Cost is $O(N^2)$ "sum of numbers from 1 to N" = 52*(52+1)/2 = $1.4x10^3$

Method 3: Better sorts - Shell sort, syncsort, split sort, ...

- Even for arbitrary data, better sort algorithms exist
- $O(N \log_2 N)$ = k * 52 * 5.7 = k * 300, where "k" is time per operation
- For N large, important gain regardless of k
- As ideas improve, k has come down from 4 to about 1.1 => 330

Method 4: Bin sort ("Solitaire sort")

- Use knowledge that there are 52 items with unique known labels
- Throw each card into the right bin with 52 calculations: O(N)

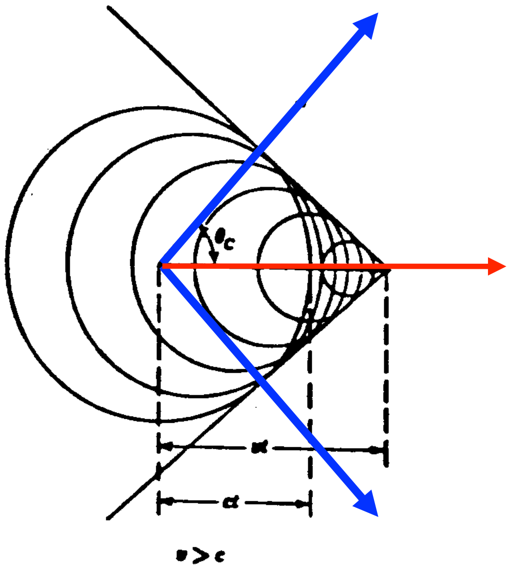
Method 5: New decks are already sorted (No operations!)

Telling pions from kaons via Cherenkov light

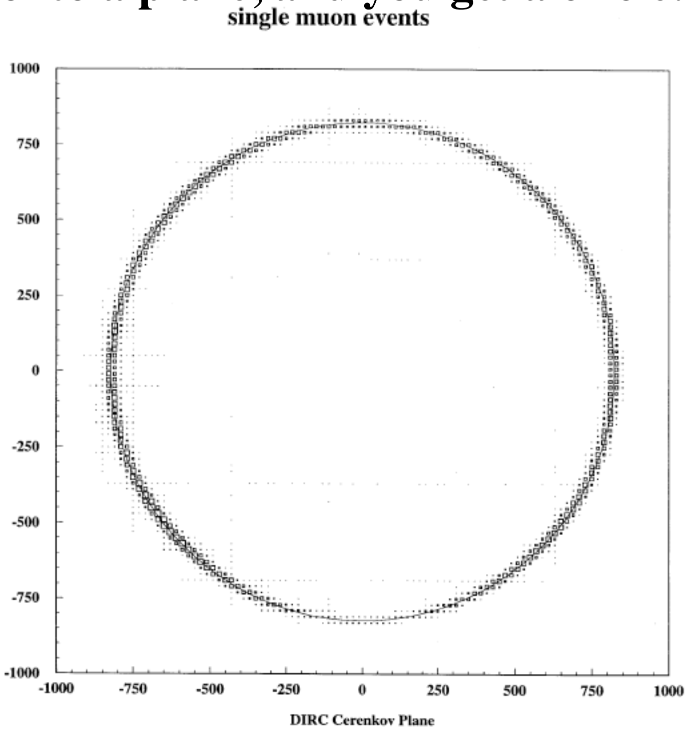
Pions & Kaons have similar interactions in matter, differ in mass

Particles moving faster than light in a medium (glass, water) emit light

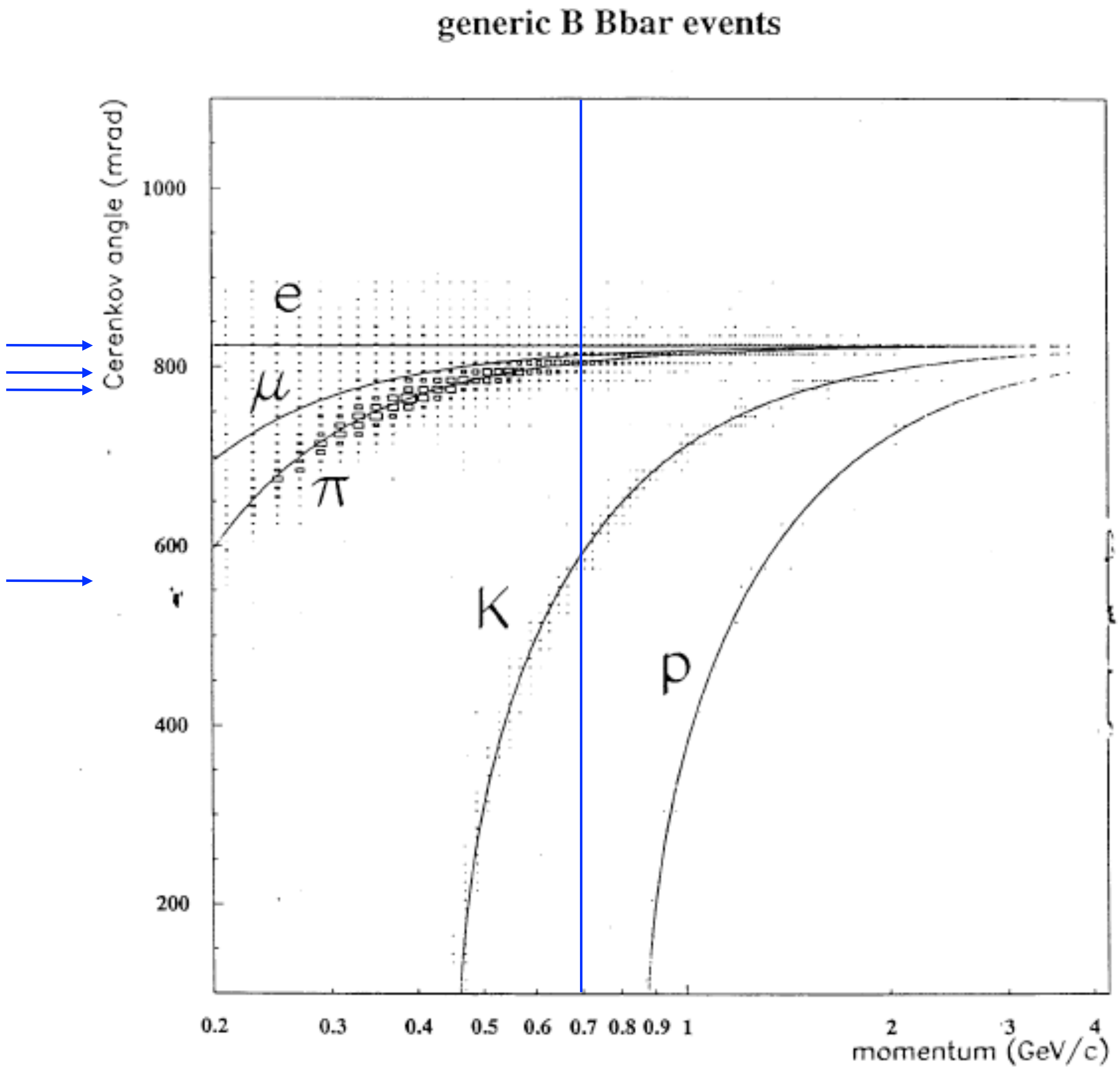
- Angle is related to velocity
- Light forms a cone



Focus it onto a plane, and you get a circle:



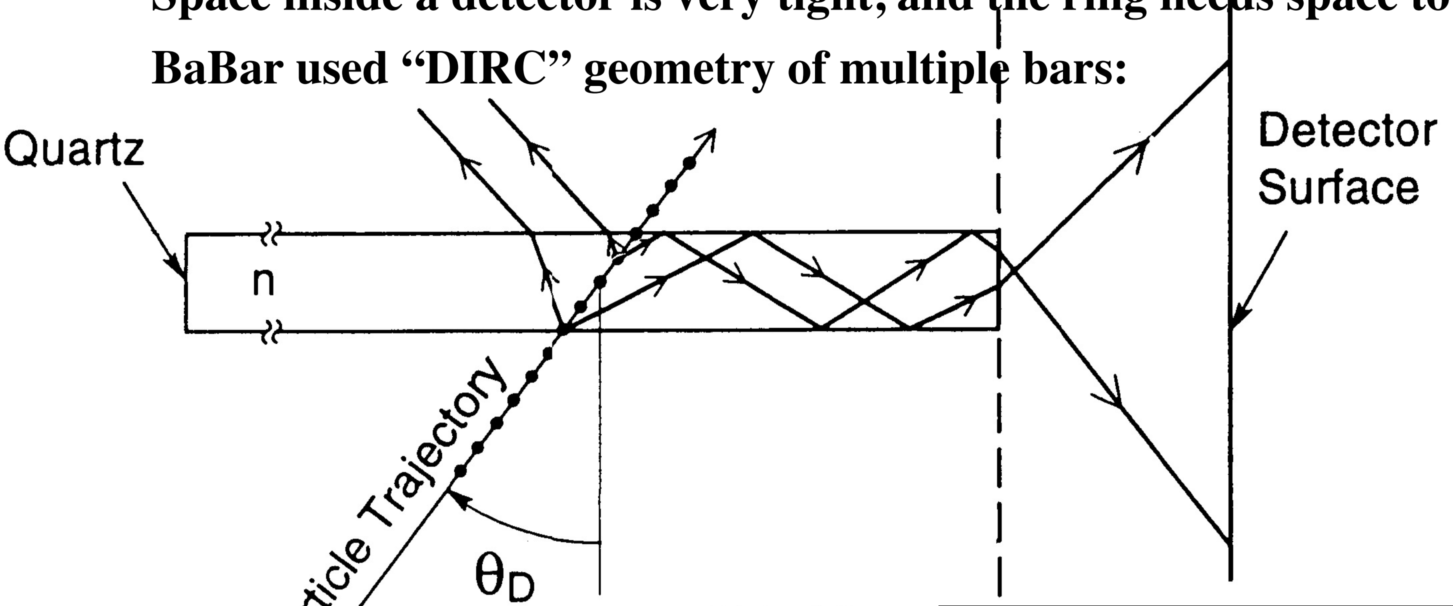
Radius of the reconstructed circle give particle type:

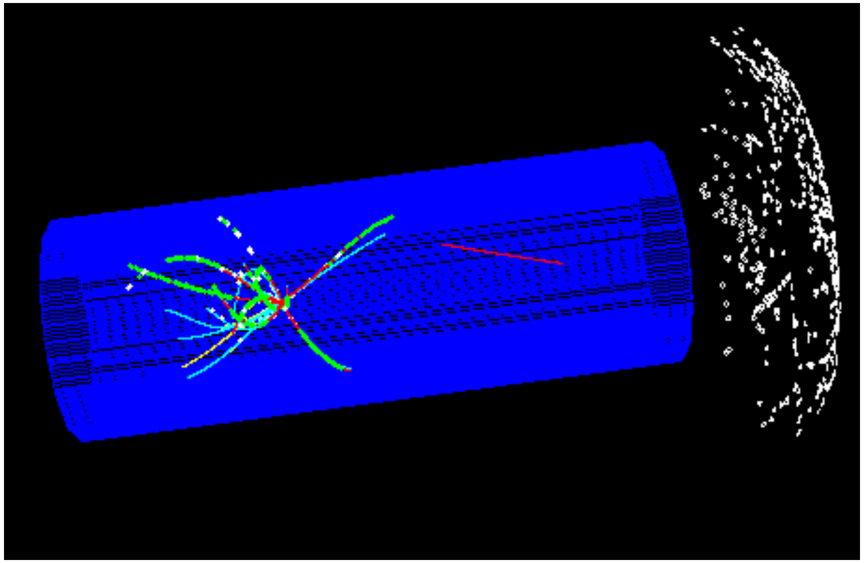


How to make this fit?

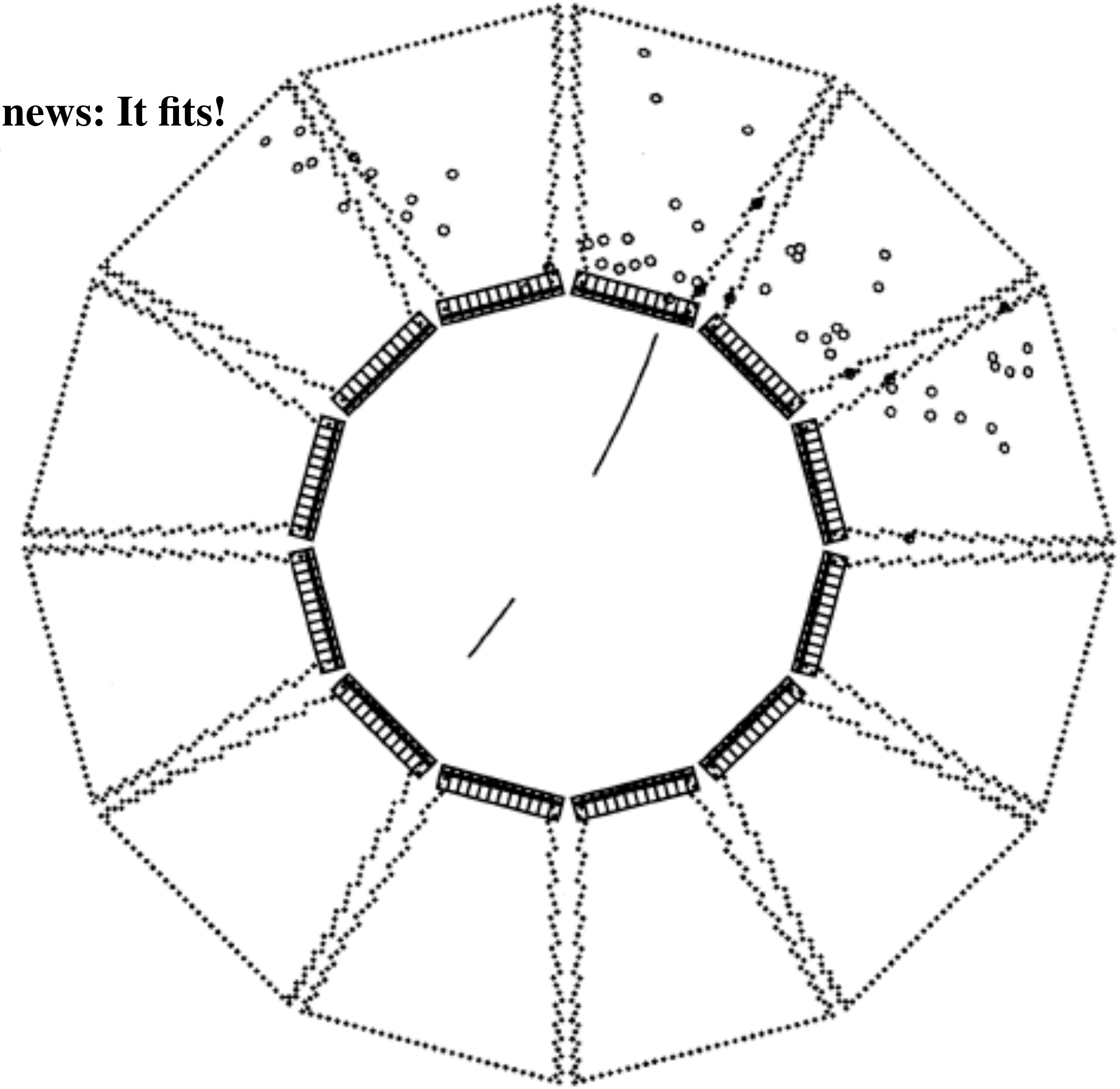
Space inside a detector is very tight, and the ring needs space to form

BaBar used "DIRC" geometry of multiple bars:



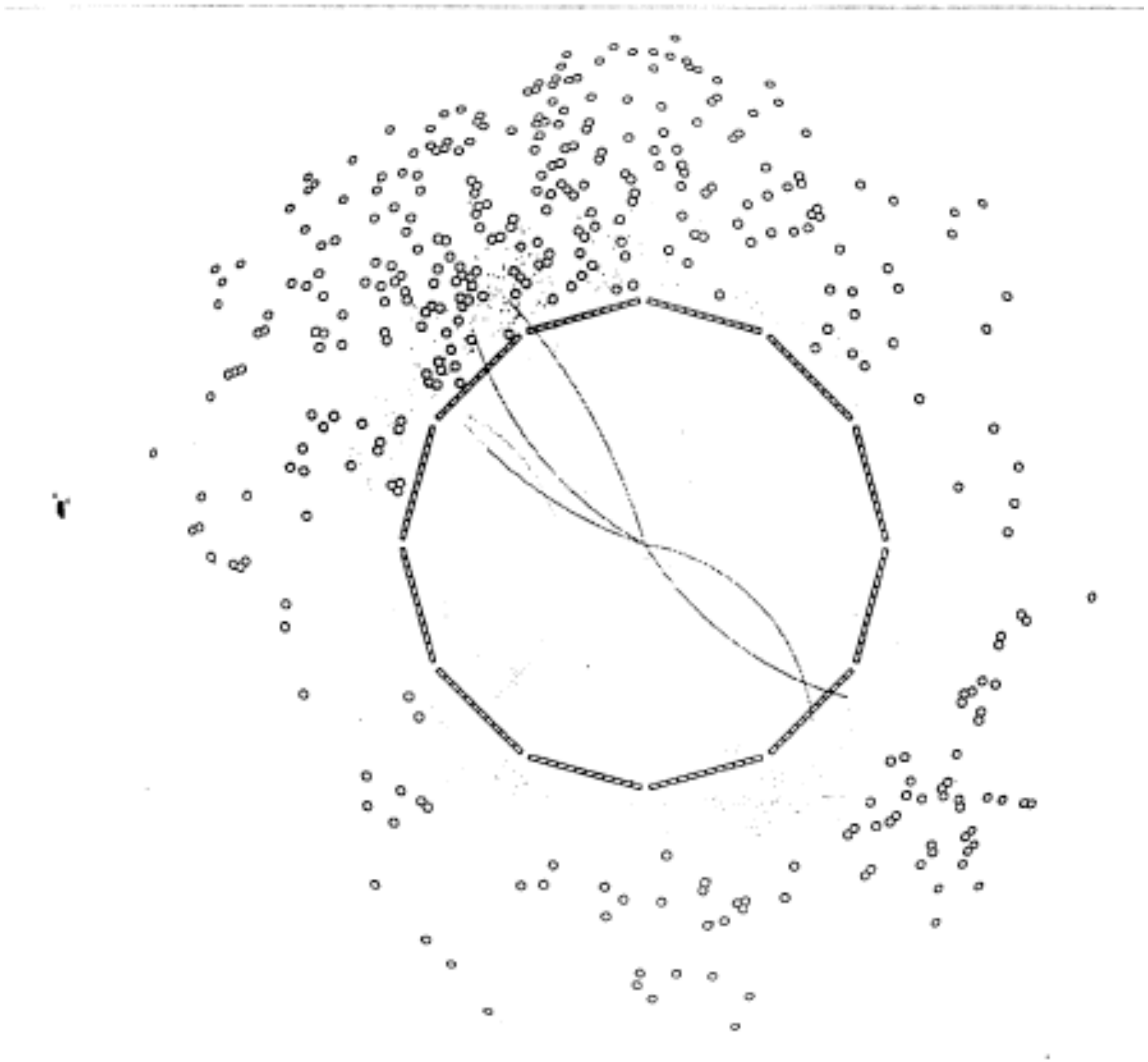


Good news: It fits!



Bad news: Rings get messy due to ambiguities in bouncing

Simple event with five charged particles:



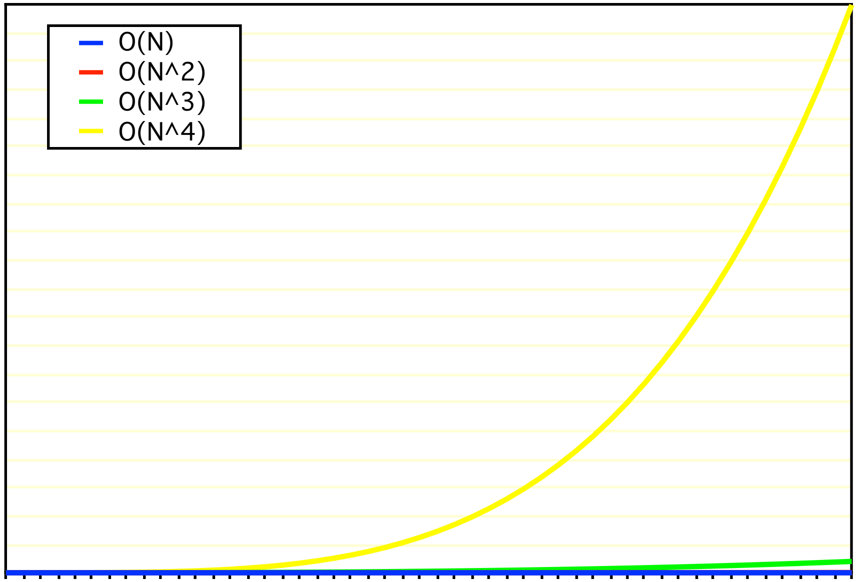
Why is this hard?

More

Brute-force circle-finding is an O(N⁴) problem

- Basic algorithm: Are these four points consistent with a ‘circle’?

Important to understand how cost grows with input size:



Realistic solution for DIRC? (Avoiding O(N^4))

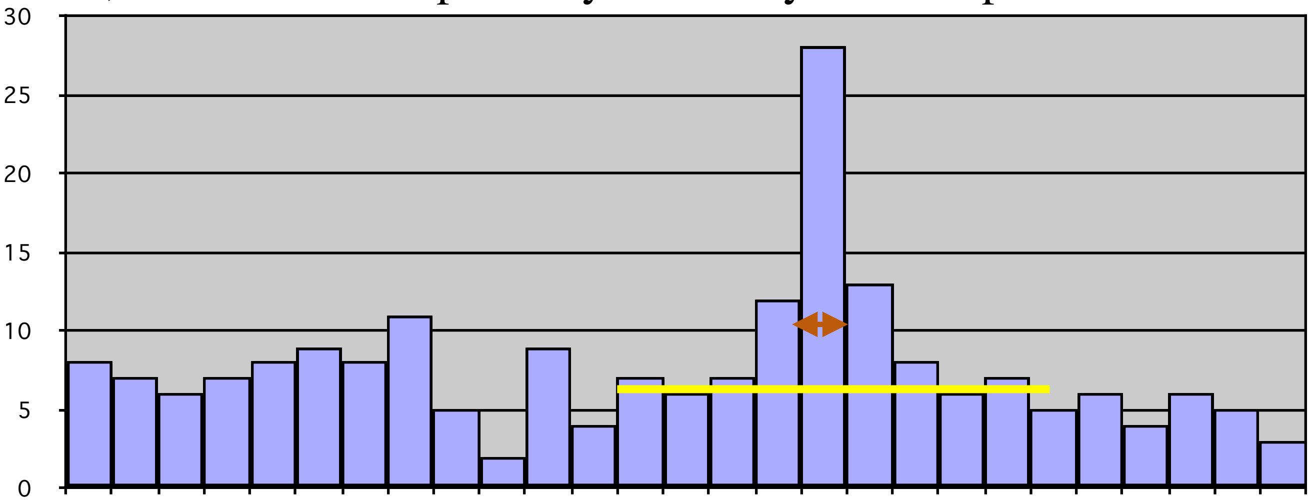
Use what you know:

- Have track trajectories, know position and angle in DIRC bars
- All photons from a single track will have the same angle w.r.t. track

  No reason to expect that for photons from other tracks

For each track, plot angle between track and every photon - O(N)

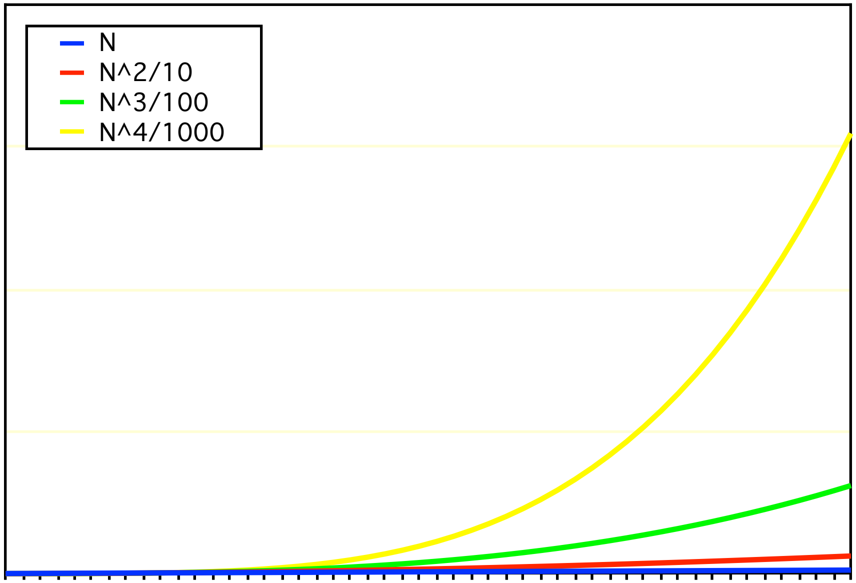
- Don’t do pattern recognition with individual photons
- Instead, look for overall pattern you already know is present



Not perfect, but optimal?

“But each operation is so much slower…”

How do I compare a “fast” $O(N^4)$ algorithm with a slow $O(N)$?

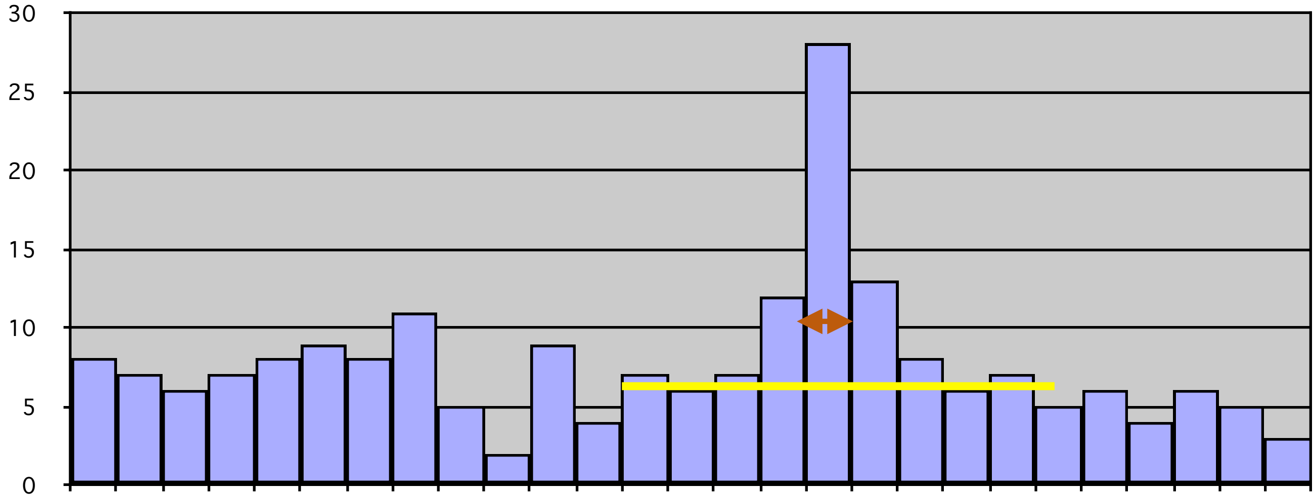


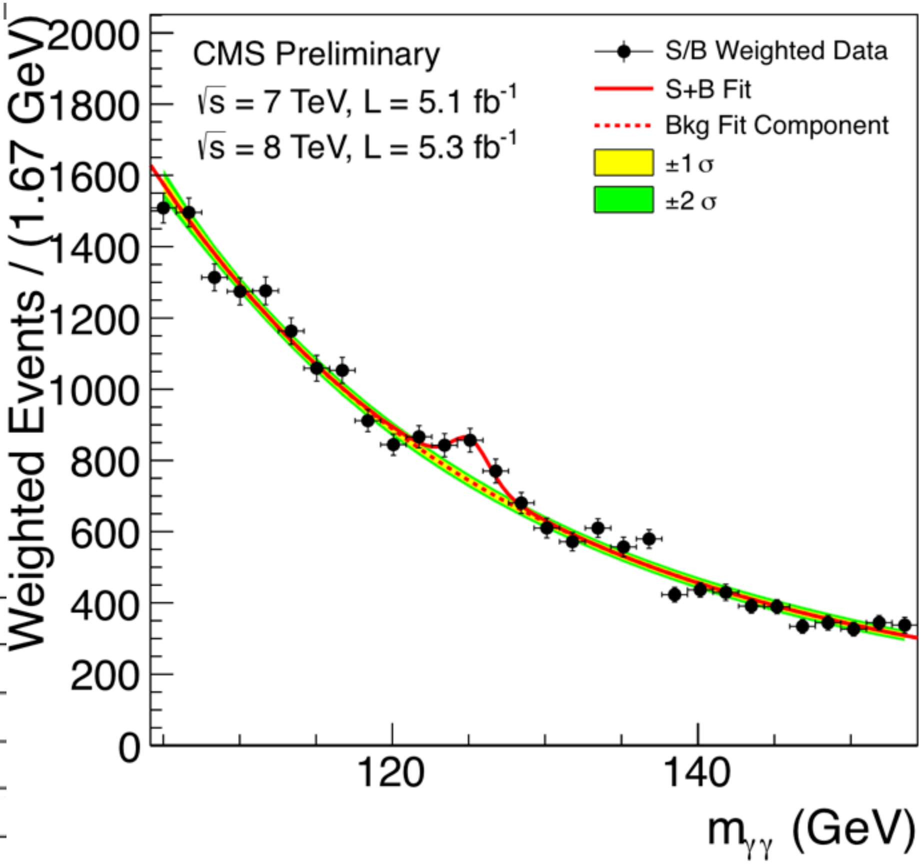
Many realistic problems deal with lots of data items

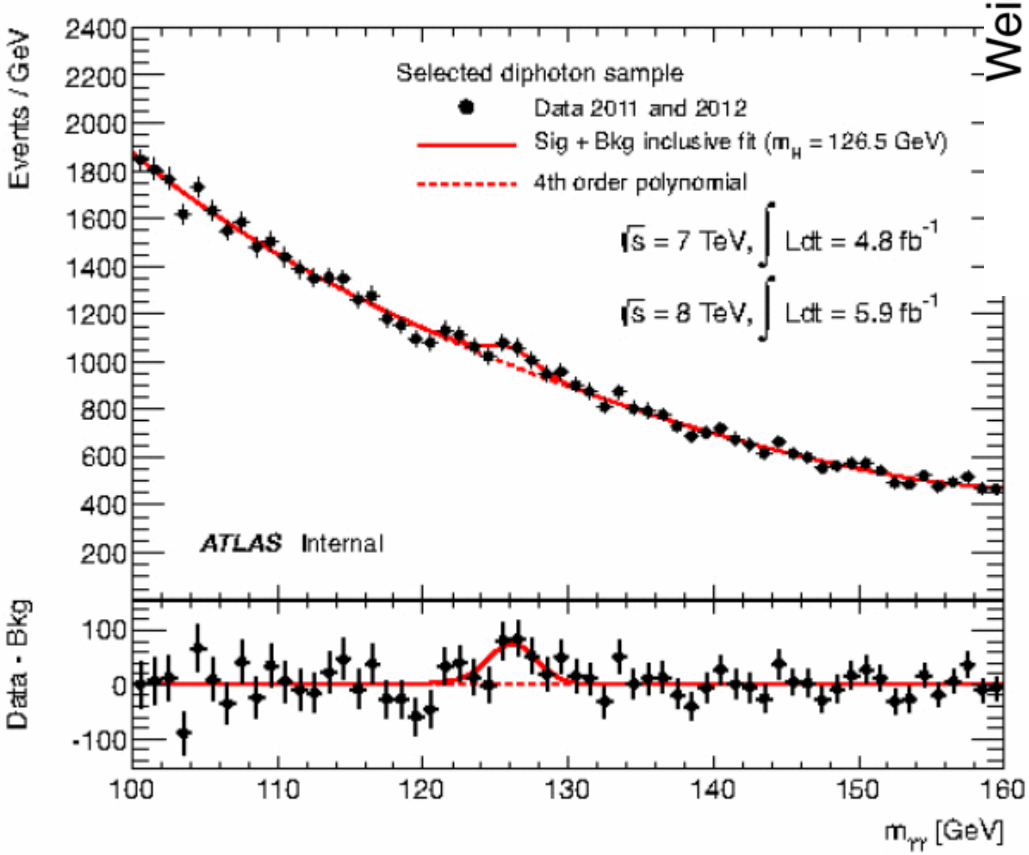
- Sharp coding is unlikely to save you a factor of 50^3 per calculation

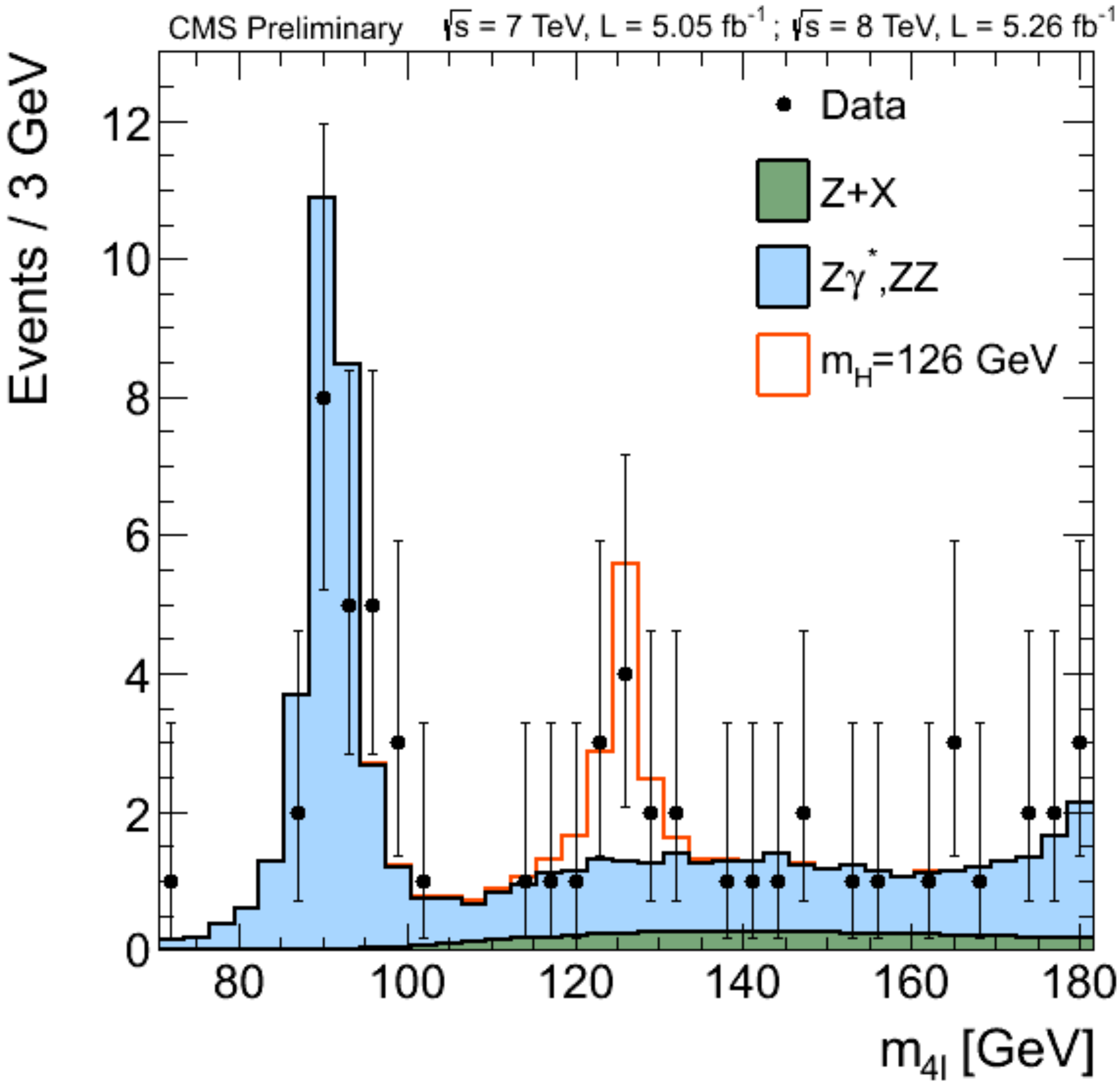
Where else do we see this pattern?

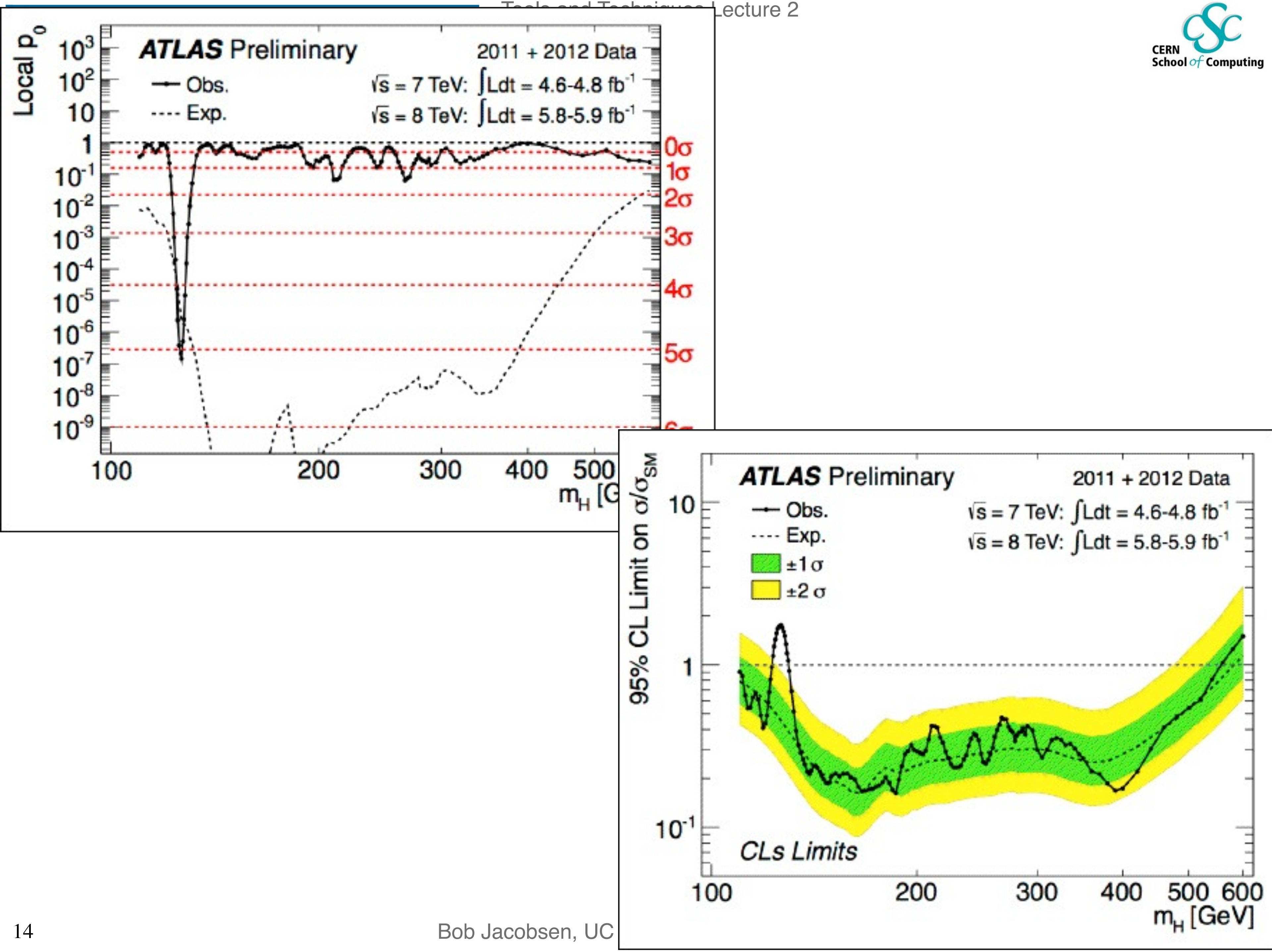
What do we do when we can't figure out the exact answer?



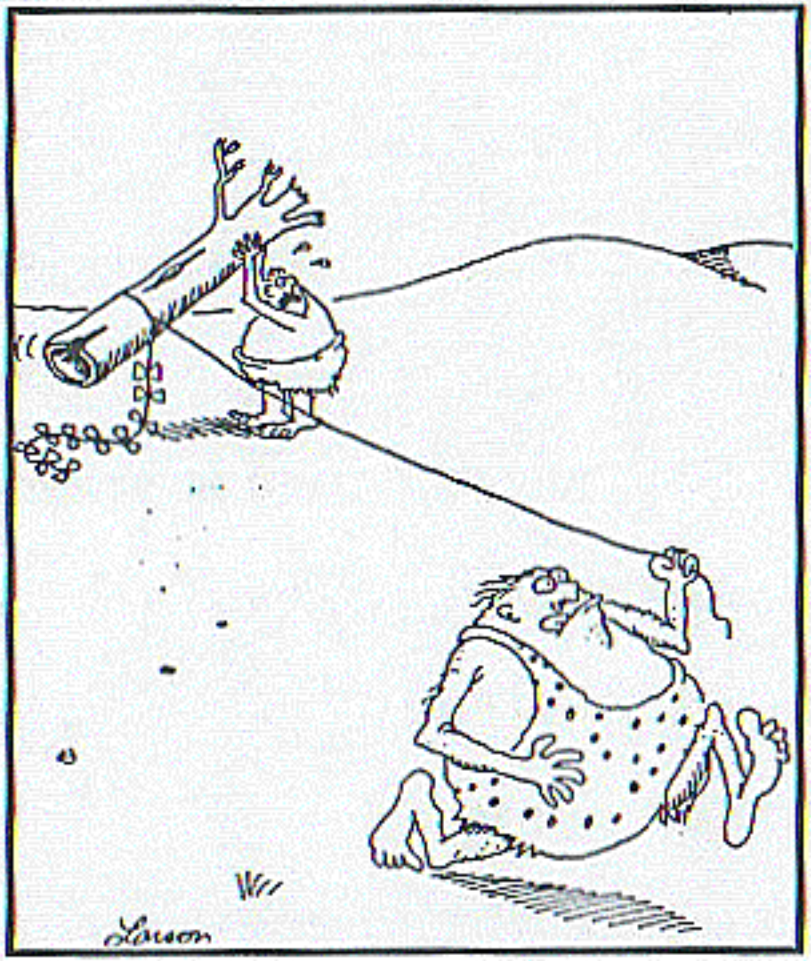

CMS Preliminary
$\sqrt{s}$ = 7 TeV, L = 5.1 fb^{-1}
$\sqrt{s}$ = 8 TeV, L = 5.3 fb^{-1}
S/B Weighted Data
S+B Fit
Bkg Fit Component
±1 σ
±2 σ
Weighted Events / (1.67 GeV)
0
200
400
600
800
1000
1200
1400
1600
1800
2000
120
140
$m_{\gamma\gamma}$ (GeV)


Selected diphoton sample
Data 2011 and 2012
Sig + Bkg inclusive fit (m_H = 126.5 GeV)
4th order polynomial
$\sqrt{s}$ = 7 TeV, $\int$ Ldt = 4.8 fb^{-1}
$\sqrt{s}$ = 8 TeV, $\int$ Ldt = 5.9 fb^{-1}
ATLAS Internal
Events / GeV
2400
2200
2000
1800
1600
1400
1200
1000
800
600
400
200
Data - Bkg
100
0
-100
100
110
120
130
140
150
160
$m_{\gamma\gamma}$ [GeV]


CMS Preliminary
√s = 7 TeV, L = 5.05 fb^{-1} ; √s = 8 TeV, L = 5.26 fb^{-1}
Events / 3 GeV
0
2
4
6
8
10
12
80
100
120
140
160
180
m_{4l} [GeV]
● Data
Z+X
Zγ*,ZZ
m_H=126 GeV


Local p_0
ATLAS Preliminary
2011 + 2012 Data
Obs.
Exp.
$\sqrt{s}$ = 7 TeV: $\int Ldt$ = 4.6-4.8 fb^{-1}
$\sqrt{s}$ = 8 TeV: $\int Ldt$ = 5.8-5.9 fb^{-1}
10^3
10^2
10
1
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
10^{-8}
10^{-9}
0σ
1σ
2σ
3σ
4σ
5σ
6σ
100
200
300
400
500
m_H [G
95% CL Limit on σ/σ_{SM}
ATLAS Preliminary
2011 + 2012 Data
Obs.
Exp.
±1 σ
±2 σ
$\sqrt{s}$ = 7 TeV: $\int Ldt$ = 4.6-4.8 fb^{-1}
$\sqrt{s}$ = 8 TeV: $\int Ldt$ = 5.8-5.9 fb^{-1}
10
1
10^{-1}
CLs Limits
100
200
300
400
500
600
m_H [GeV]

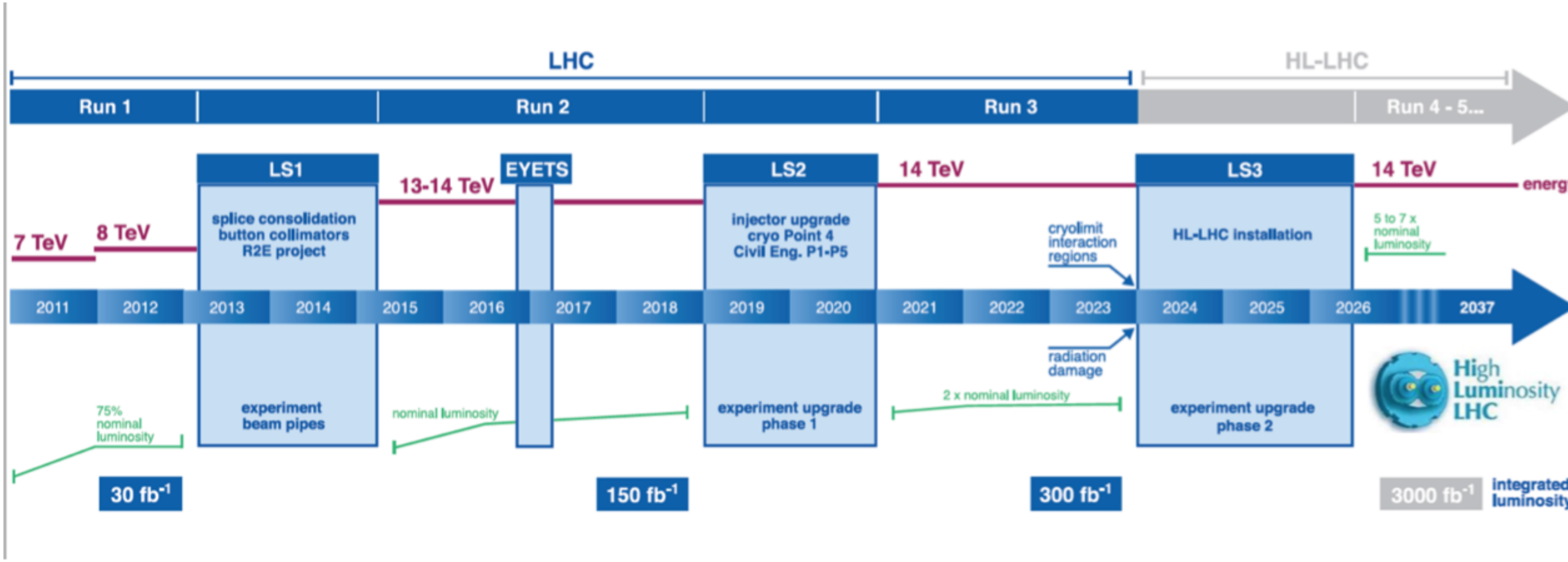
Big things are different from small things



"Okay, Bob! Go! Go!"

The life time of HEP software

Software is a long-term commitment



Many releases of the software are needed over its lifetime
to fix bugs, add new features, support new platforms etc

Can’t technology save us?

More

We’ve built a series of ever-larger tools to handle large code projects:

Git for controlling and versioning code

Tools for building “releases” of systems

Tools for “configuration management”

But we struggle against three forces:

•We’re always building bigger & more difficult systems

•We’re always building bigger & more difficult collaborations

•And we’re the same old people

Net effect: We’re always pushing the boundary of what we can do

Stupidity got us into this mess; why can’t it get us out? - Will Rogers

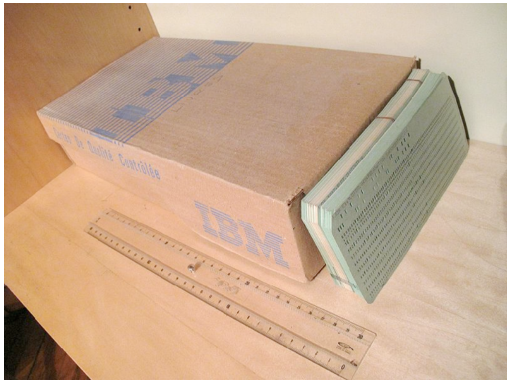
How we got here:

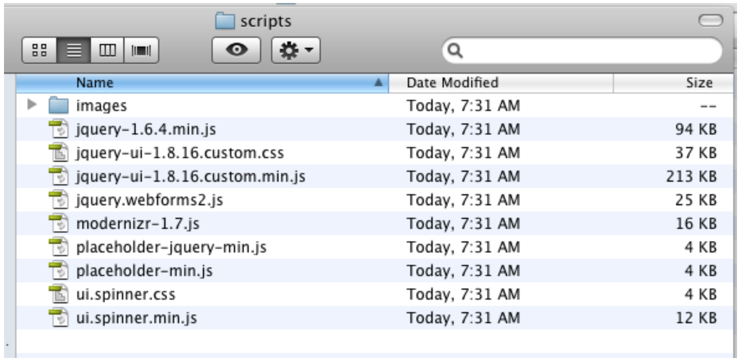
First, you just wrote a big program

But soon it was so big you wanted help

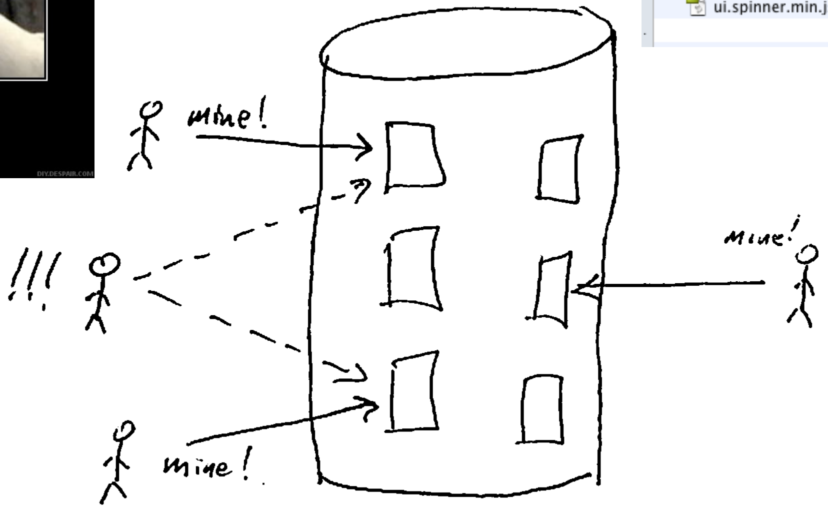
So you broke it into pieces/files/modules

But how do you share work on those?





Name	Date Modified	Size
images	Today, 7:31 AM	--
jquery-1.6.4.min.js	Today, 7:31 AM	94 KB
jquery-ui-1.8.16.custom.css	Today, 7:31 AM	37 KB
jquery-ui-1.8.16.custom.min.js	Today, 7:31 AM	213 KB
jquery.webforms2.js	Today, 7:31 AM	25 KB
modernizr-1.7.js	Today, 7:31 AM	16 KB
placeholder-jquery-min.js	Today, 7:31 AM	4 KB
placeholder-min.js	Today, 7:31 AM	4 KB
ui.spinner.css	Today, 7:31 AM	4 KB
ui.spinner.min.js	Today, 7:31 AM	12 KB



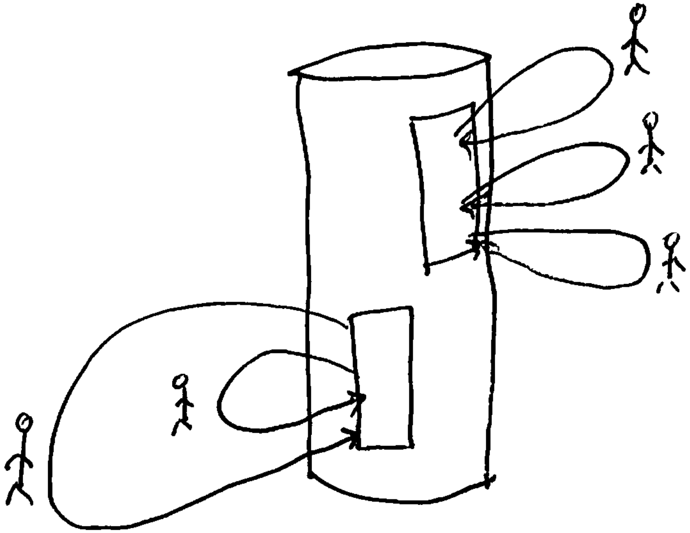
Version Control Systems (Hg, SVN, Git)

More

As systems & collaborations grow, efficiency goes down

"Version" idea: Track changes from one version to next

Anybody can get a specific set of source



Big advantage: checkout is not exclusive

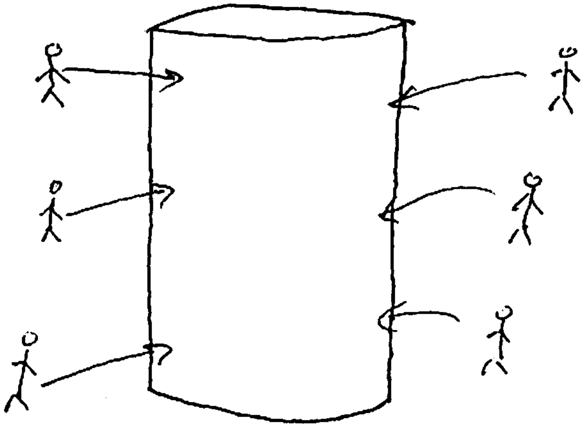
- More than one developer can have the same file checked out
- Developers can control their own use of the code for read, write
- Changes can come from multiple sources
- Tool handles (most) of the conflict resolution

Scaling is still an issue

More

Everybody is sharing a single repository

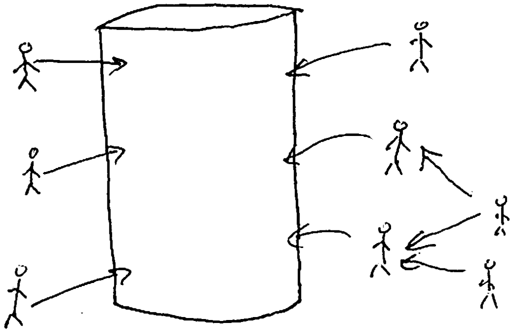
Every commit is immediately visible to everybody else



Development stands on shifting sand

Detailed records, but little understanding

Workarounds!



External record keeping tools

Package Coordinators

Issue with this arise at large & small level

At the level of developers and contributions, needed way to manage this

- Both tools and procedures
  - We’ll be discussing & exercising git as typical tool
  - Individual collaborations have their own ways of sharing info

More

At the collaboration leveled, need procedures to ensure it all works

- “Nightly builds”
  - Now common in HEP - Gives early feedback on consistency problems
- “Continuous Integration”, including automated testing
  - Only works when people actually integrate early and often
- Reduces problems, but integration is still a lot of work

When Boeing wanted to design the 747, they had two choices:

1. Hire “SuperEngineer”, who could do it alone
2. Hire 7,200 engineers and organize them to cooperate

Which did they choose?

Why?

What can we learn from this?

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

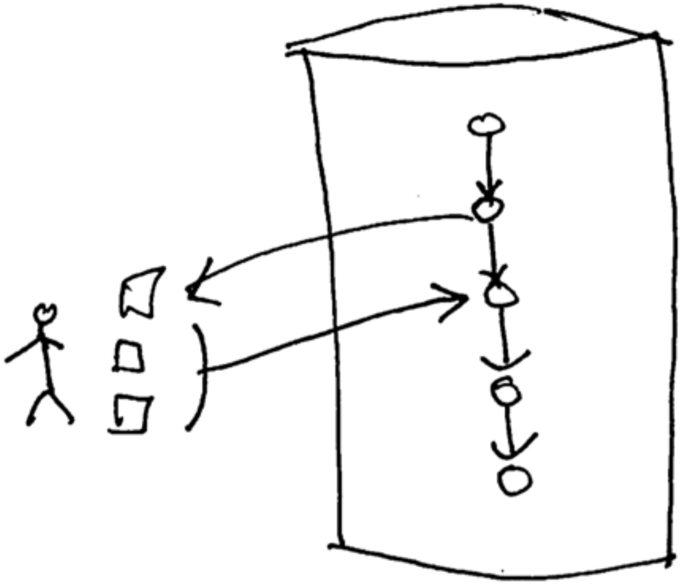
At first, Git looks like a simple file system…

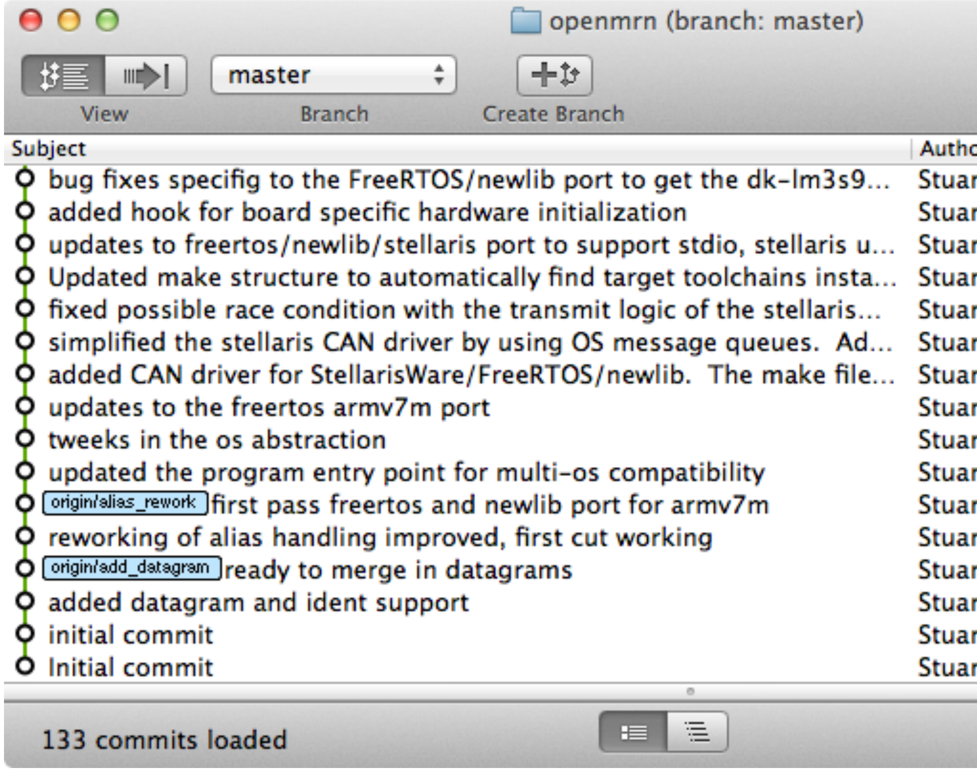
CERN School of Computing

More

You bring out a copy, work on it, and commit

Git repository contains all that history





openmrn (branch: master)

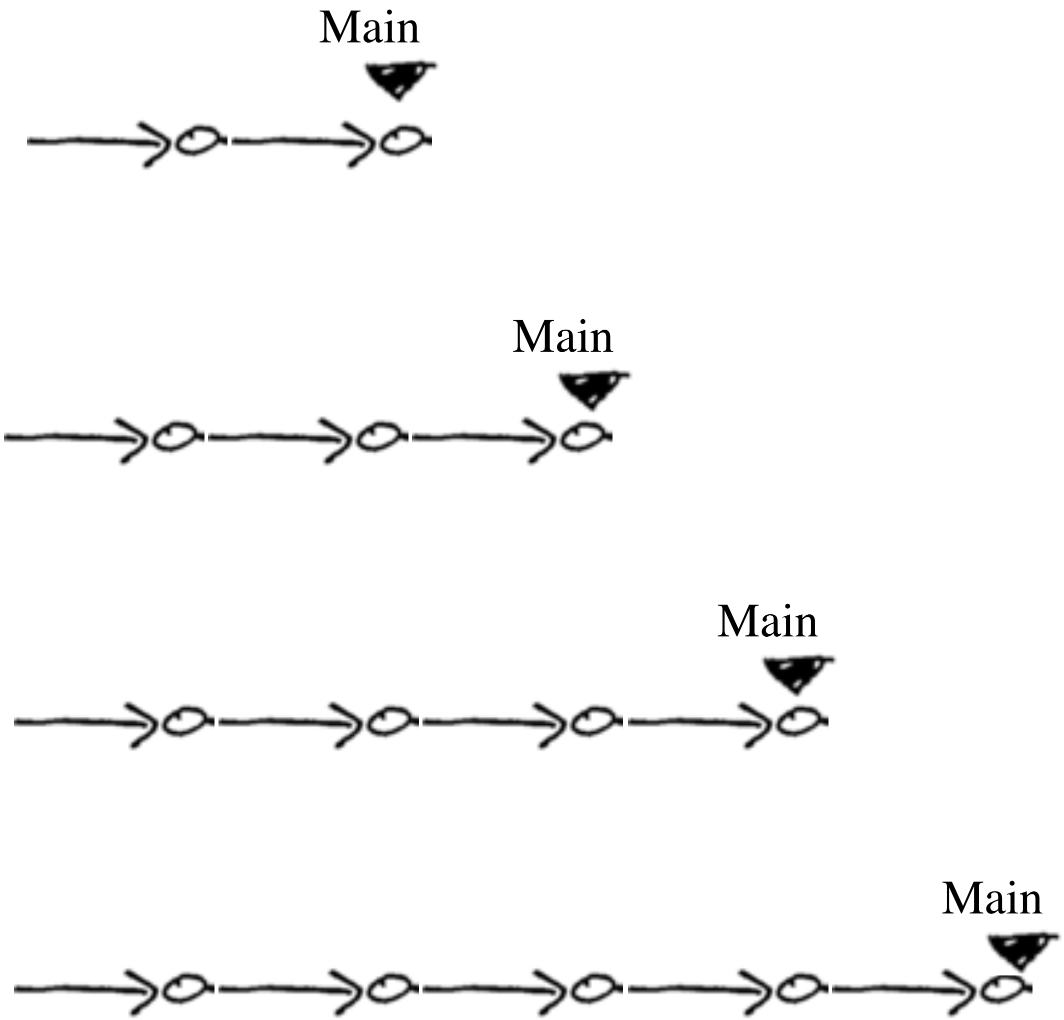
Subject	Autho
bug fixes specifig to the FreeRTOS/newlib port to get the dk-lm3s9…	Stuar
added hook for board specific hardware initialization	Stuar
updates to freertos/newlib/stellaris port to support stdio, stellaris u…	Stuar
Updated make structure to automatically find target toolchains insta…	Stuar
fixed possible race condition with the transmit logic of the stellaris…	Stuar
simplified the stellaris CAN driver by using OS message queues. Ad…	Stuar
added CAN driver for StellarisWare/FreeRTOS/newlib. The make file…	Stuar
updates to the freertos armv7m port	Stuar
tweeks in the os abstraction	Stuar
updated the program entry point for multi-os compatibility	Stuar
origin/alias_rework first pass freertos and newlib port for armv7m	Stuar
reworking of alias handling improved, first cut working	Stuar
origin/add_datagram ready to merge in datagrams	Stuar
added datagram and ident support	Stuar
initial commit	Stuar
Initial commit	Stuar

133 commits loaded

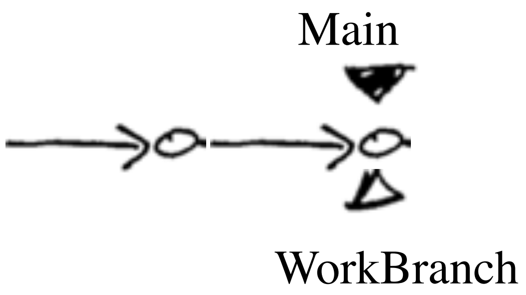
“Scratchpad” idea lets you control what you commit: Shaping the story

More

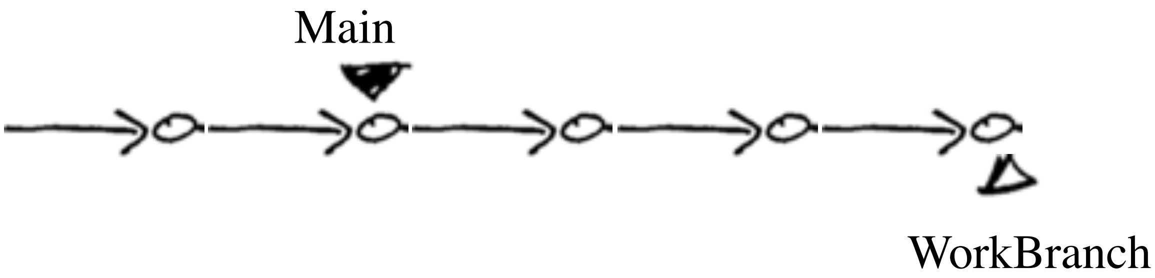
Committing to the Main Branch



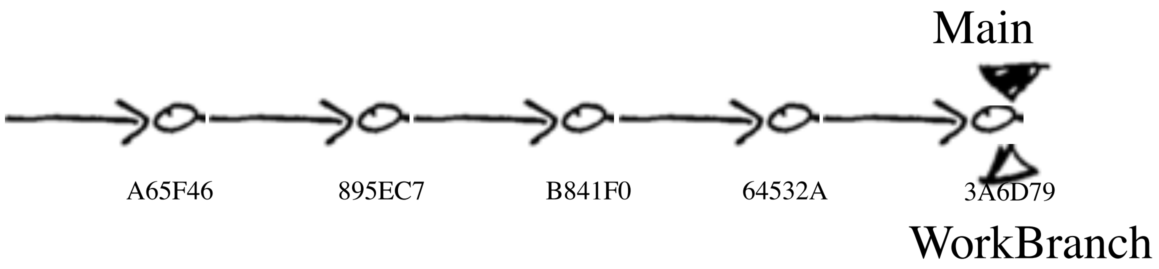
Committing on a Branch and Merging to Main


Main

WorkBranch


Main

WorkBranch

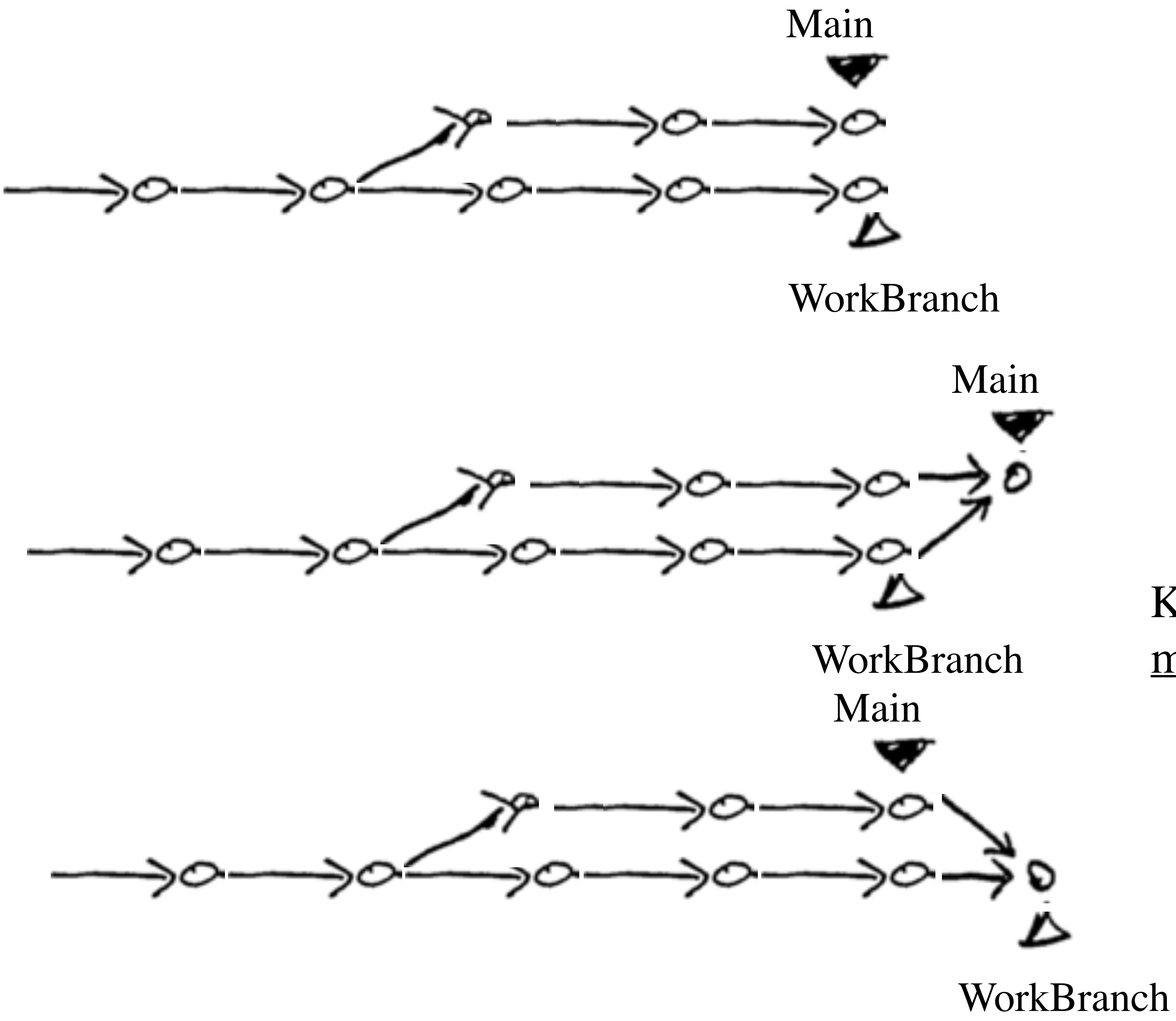

Main

A65F46 895EC7 B841F0 64532A 3A6D79

WorkBranch

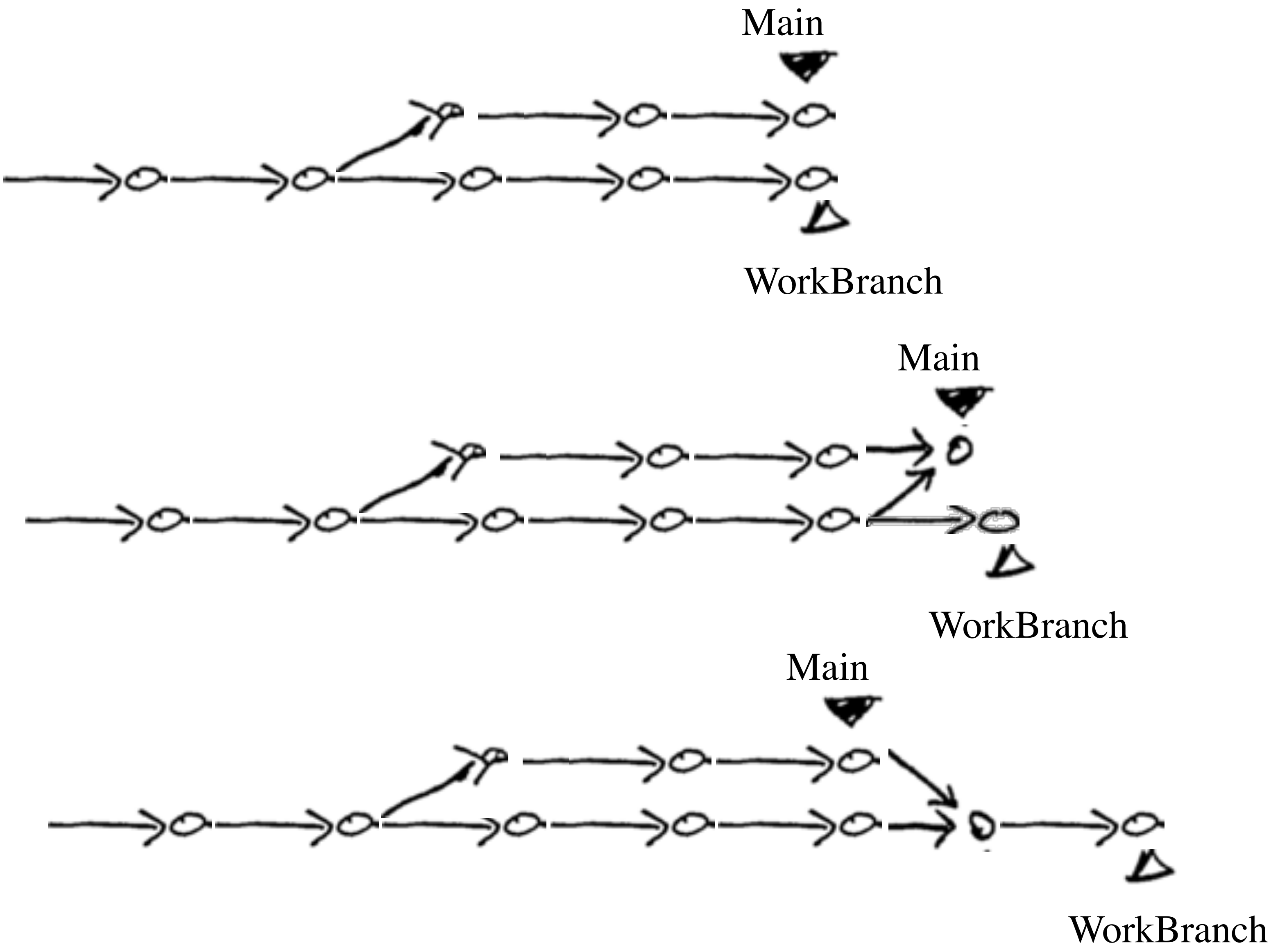
“Fast forward” form of merge

Committing on a Branch and Merging to Main



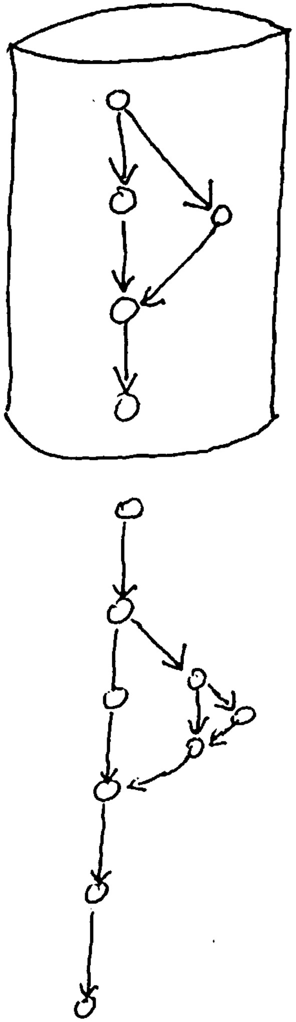
Key concept:
merge commits

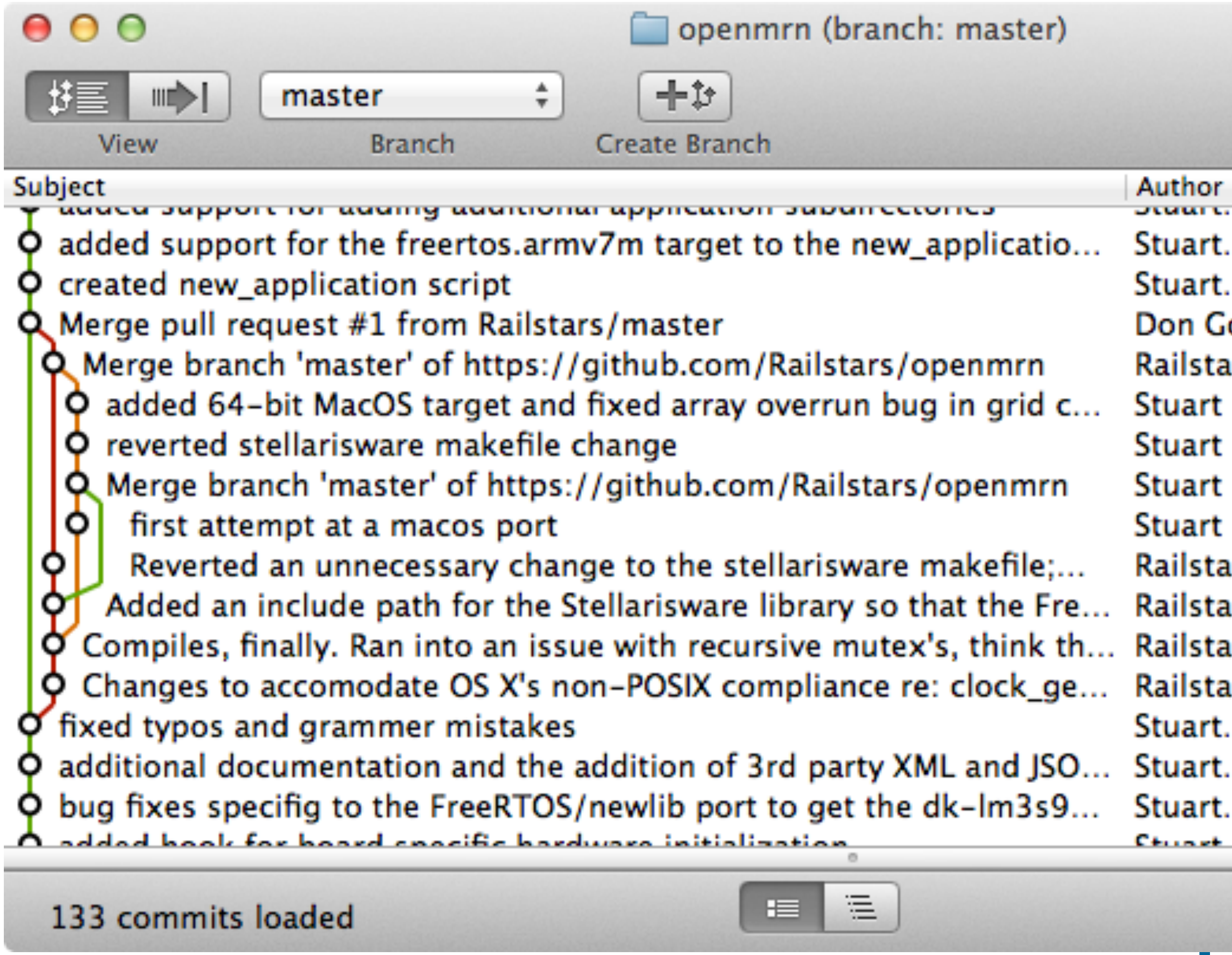
Committing on a Branch and Merging to Main



Merging

Because Git focused on commits, not on single file versions, you get powerful merging





openmrn (branch: master)

View

Branch: master

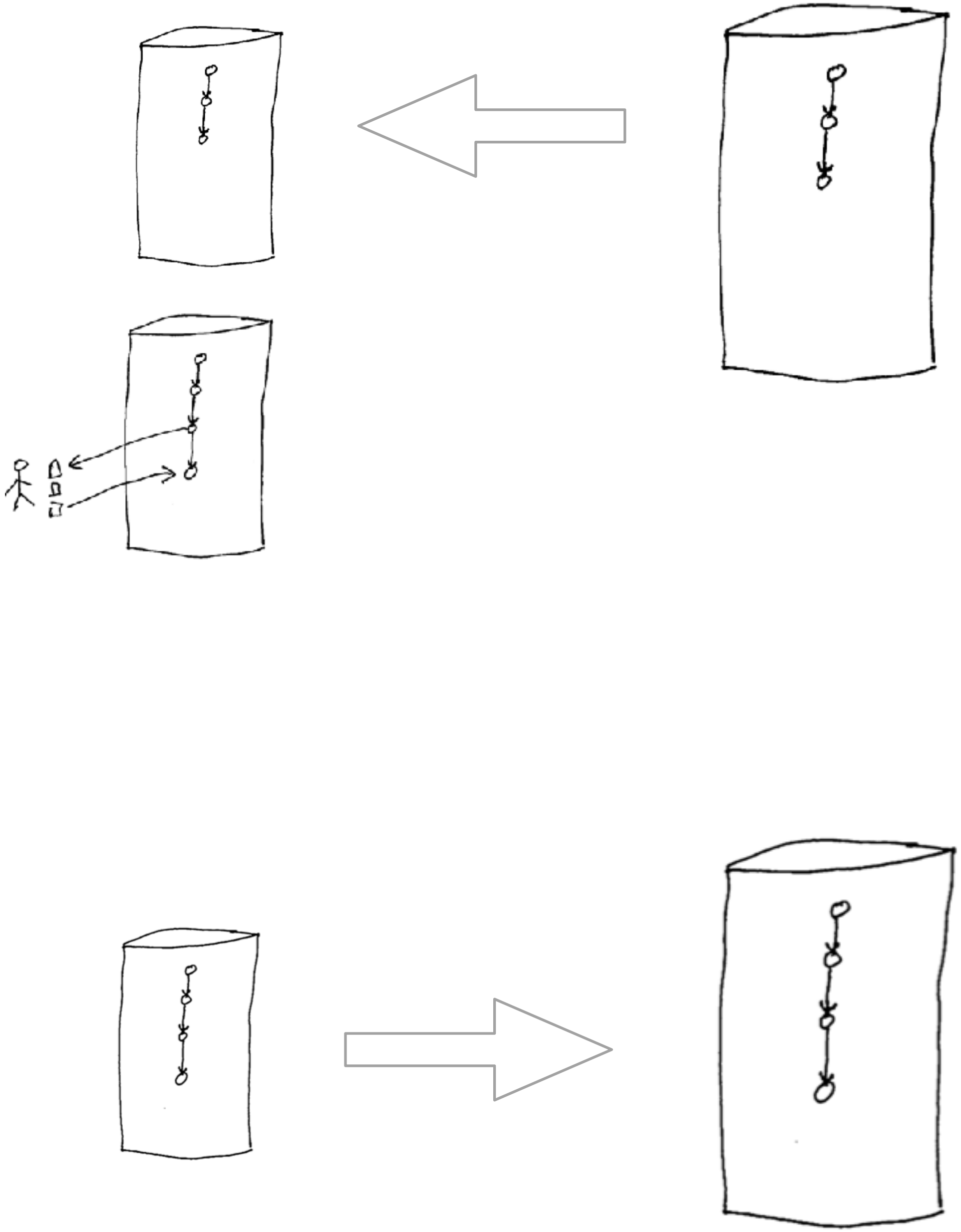
Create Branch

Subject	Author
added support for adding additional application subdirectories	Stuart.
added support for the freertos.armv7m target to the new_applicatio...	Stuart.
created new_application script	Stuart.
Merge pull request #1 from Railstars/master	Don G
Merge branch 'master' of https://github.com/Railstars/openmrn	Railsta
added 64-bit MacOS target and fixed array overrun bug in grid c...	Stuart
reverted stellarisware makefile change	Stuart
Merge branch 'master' of https://github.com/Railstars/openmrn	Stuart
first attempt at a macos port	Stuart
Reverted an unnecessary change to the stellarisware makefile;...	Railsta
Added an include path for the Stellarisware library so that the Fre...	Railsta
Compiles, finally. Ran into an issue with recursive mutex's, think th...	Railsta
Changes to accomodate OS X's non-POSIX compliance re: clock_ge...	Railsta
fixed typos and grammer mistakes	Stuart.
additional documentation and the addition of 3rd party XML and JSO...	Stuart.
bug fixes specifig to the FreeRTOS/newlib port to get the dk-lm3s9...	Stuart.
added hook for board specific hardware initialization	Stuart

133 commits loaded

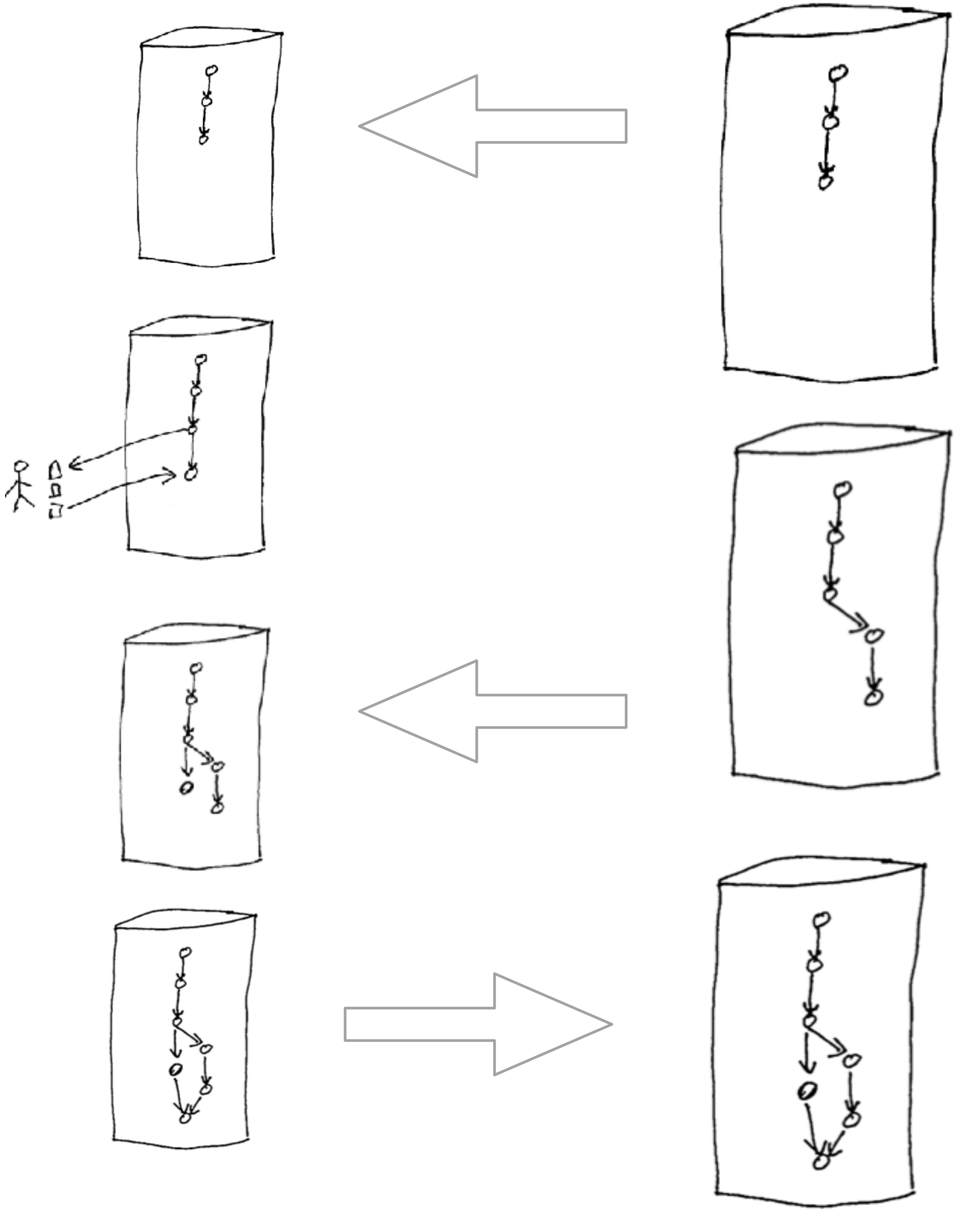
Multiple repositories with easy transfer of commits between

More



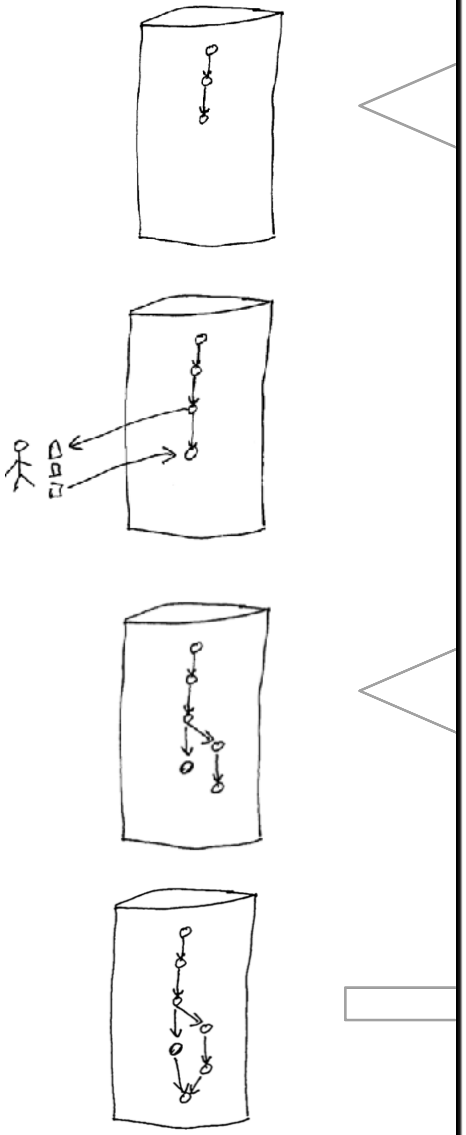
More than just mirroring

More



More than just mirroring

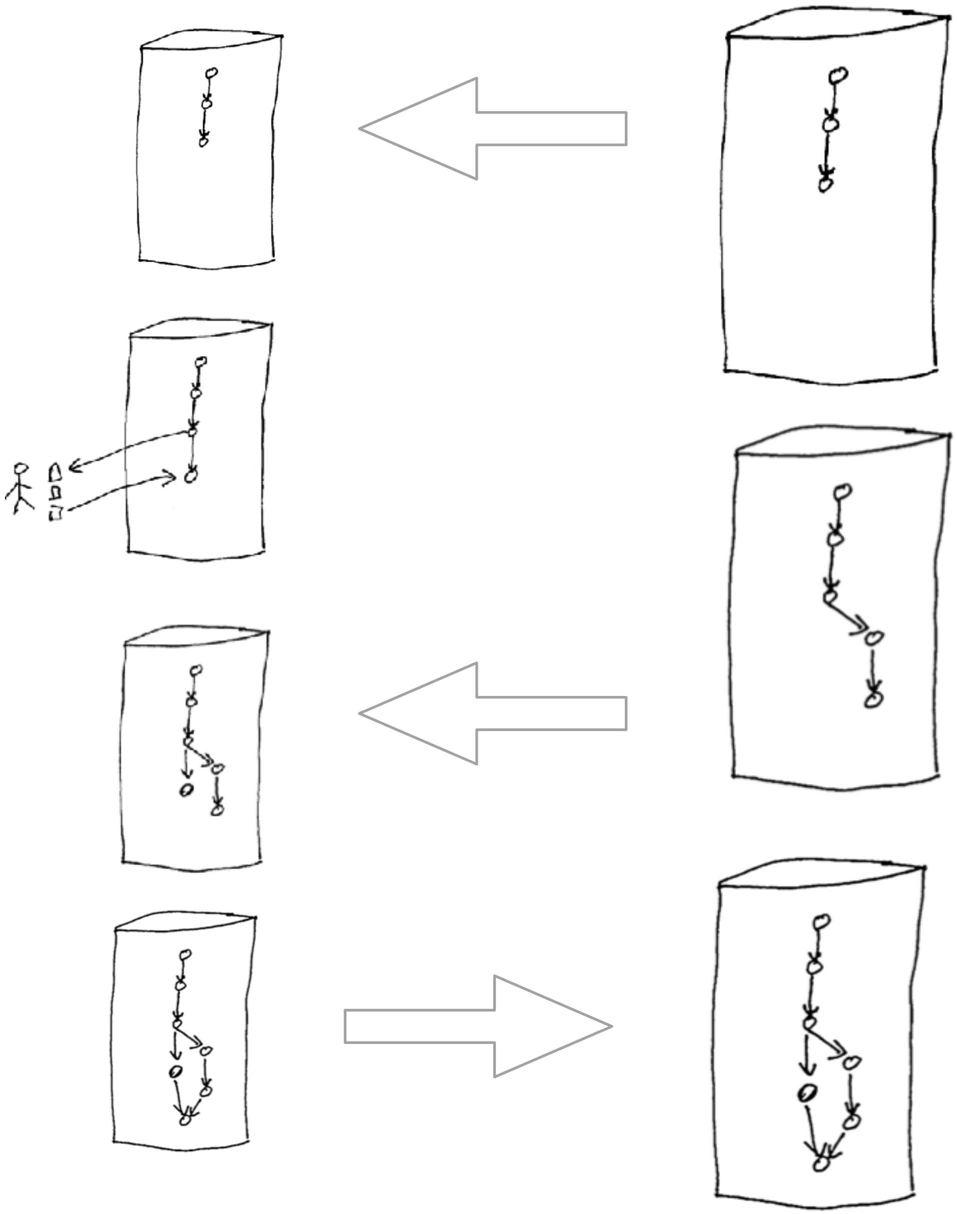
More



THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

More than just mirroring

More



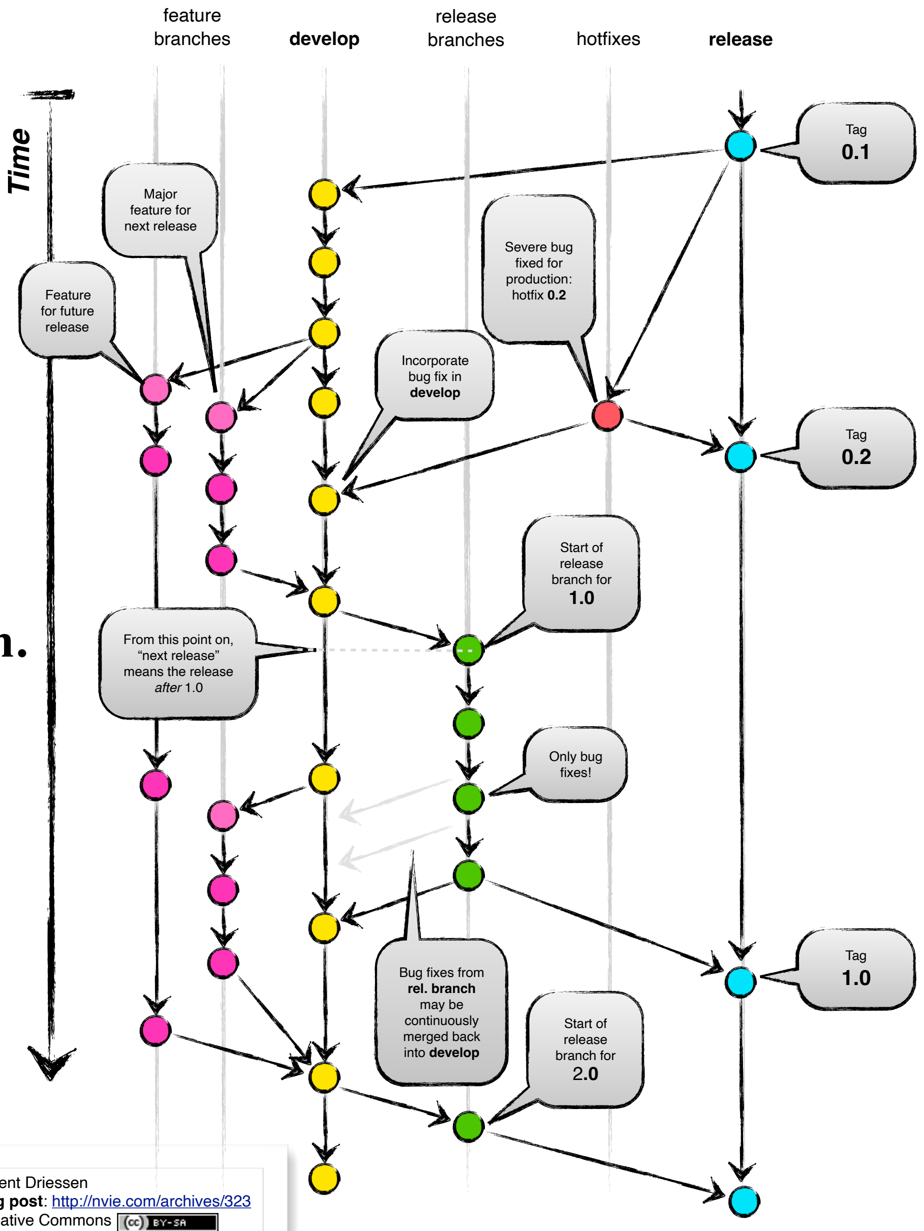
Branches are key

- Develop on a separate branch
- Future Big Feature on branch
- And another one for || work
- Pays off for bug fix!
- Git merge to get fix across
- Feature done, merges in
- New branch holds release
-   and its inevitable fixes
-   until merge and release main.
- Meanwhile, work proceeds
- And the process repeats

Keys: cheap branches,

reliable merges

Gives understandable story?


feature branches
develop
release branches
hotfixes
release
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix 0.2
Incorporate bug fix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release *after* 1.0
Only bug fixes!
Tag 1.0
Bug fixes from rel. branch may be continuously merged back into develop
Start of release branch for 2.0

Author: Vincent Driessen
Original blog post: http://nvie.com/archives/323
License: Creative Commons (cc) BY-SA

Using all that history:

My feature broke between 0.1 and 1.0

Which commit broke it?

“git bisect” works through the graph

Was it in 0.2? No?

Was it in merge before the release branch? Yes

….

I found a bug in a specific commit SHA

Which releases does it affect?

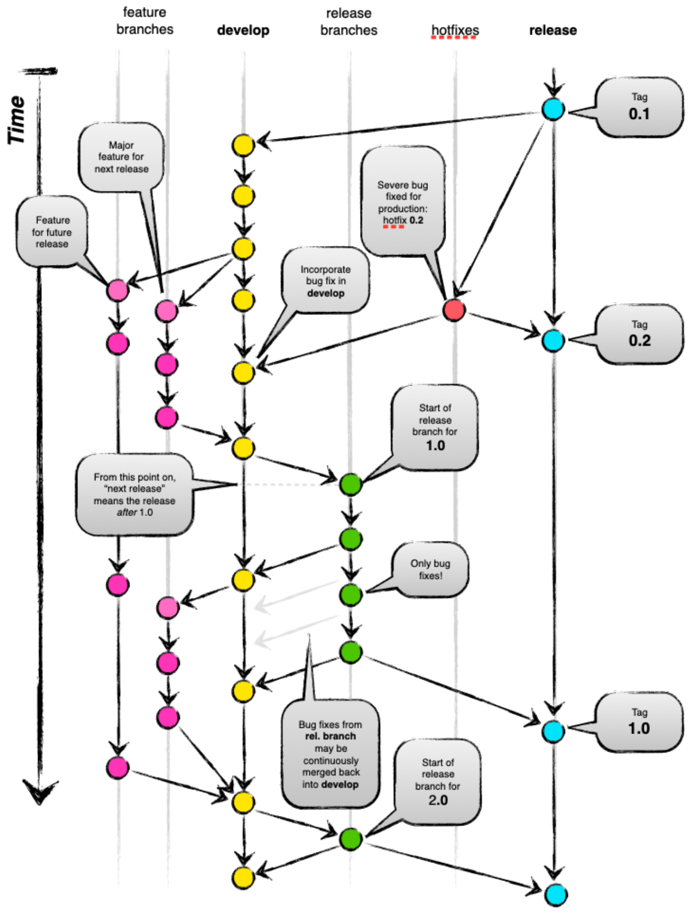
What’s not affected?

“git diff tag1.0...SHA” to see if included

“git log” and “git revlog” explore history

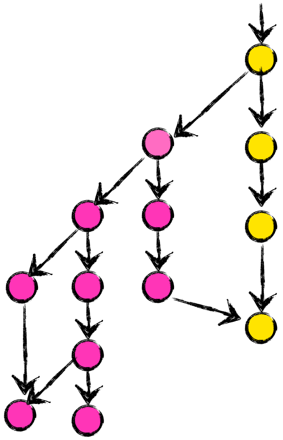
Graphical representations can help a lot

gitk, gitg tools


feature branches
develop
release branches
hotfixes
release
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix 0.2
Incorporate bug fix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, “next release” means the release after 1.0
Only bug fixes!
Tag 1.0
Bug fixes from rel. branch may be continuously merged back into develop
Start of release branch for 2.0

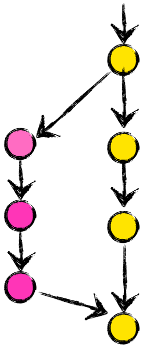
Complex! Linear history in repository would resolve these much easier

Git Rebase: An Editor for the Story

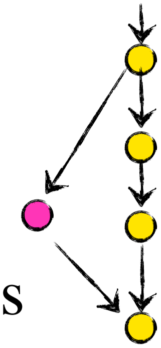


Finished difficult development task, after several dead ends, lots of little bits of progress & dead ends

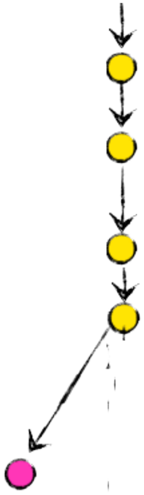
More



Deleting only gets you so far



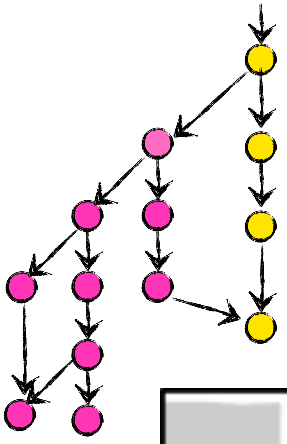
“Squashing” commits



“Rebase” operation

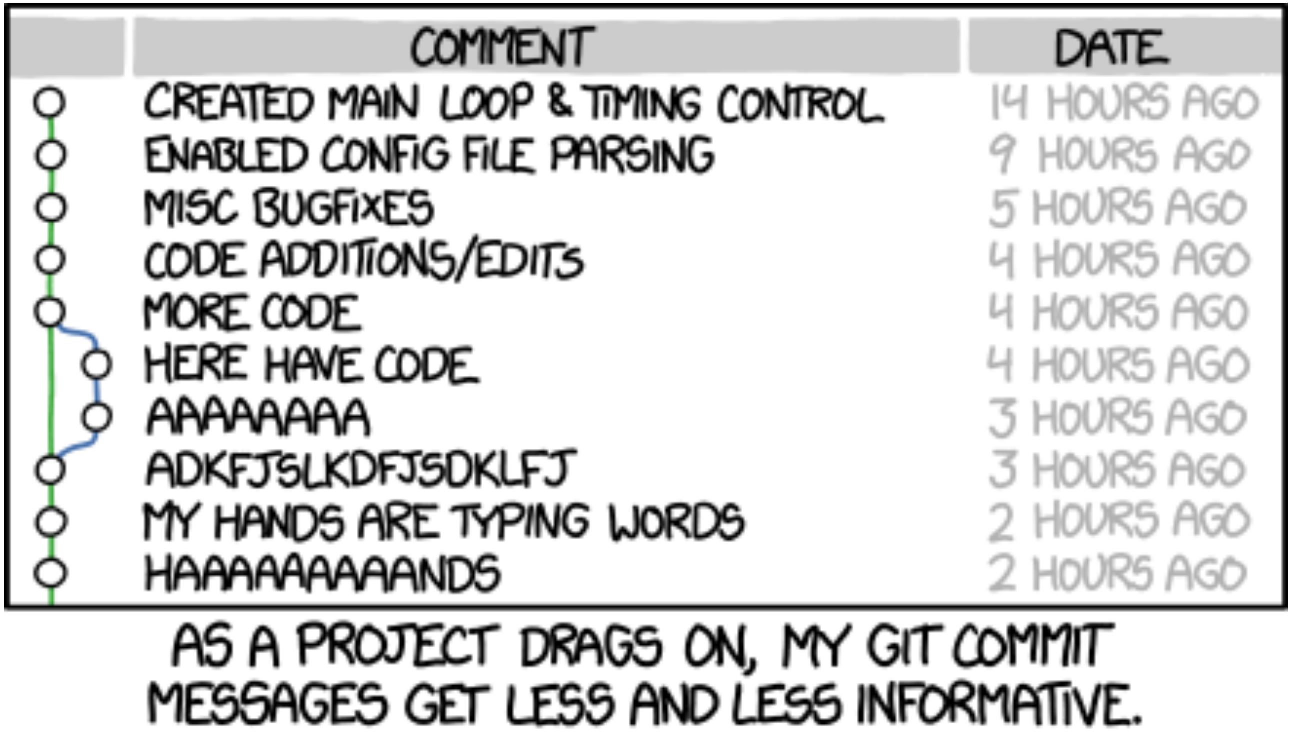
Fast-forward merge

Git Rebase: An Editor for the Story



Finished difficult development task, after several dead ends, lots of little bits of progress & dead ends

More



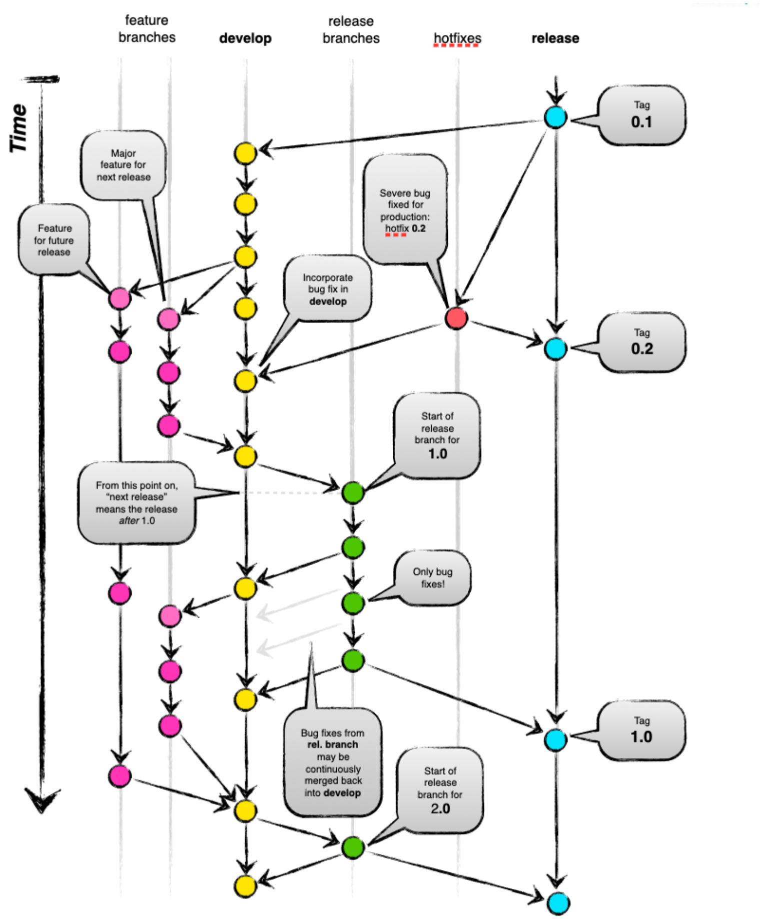
	COMMENT	DATE
○	CREATED MAIN LOOP & TIMING CONTROL	14 HOURS AGO
○	ENABLED CONFIG FILE PARSING	9 HOURS AGO
○	MISC BUGFIXES	5 HOURS AGO
○	CODE ADDITIONS/EDITS	4 HOURS AGO
○	MORE CODE	4 HOURS AGO
○	HERE HAVE CODE	4 HOURS AGO
○	AAAAAAAA	3 HOURS AGO
○	ADKFJSLKDFJSDKLFJ	3 HOURS AGO
○	MY HANDS ARE TYPING WORDS	2 HOURS AGO
○	HAAAAAAAANDS	2 HOURS AGO

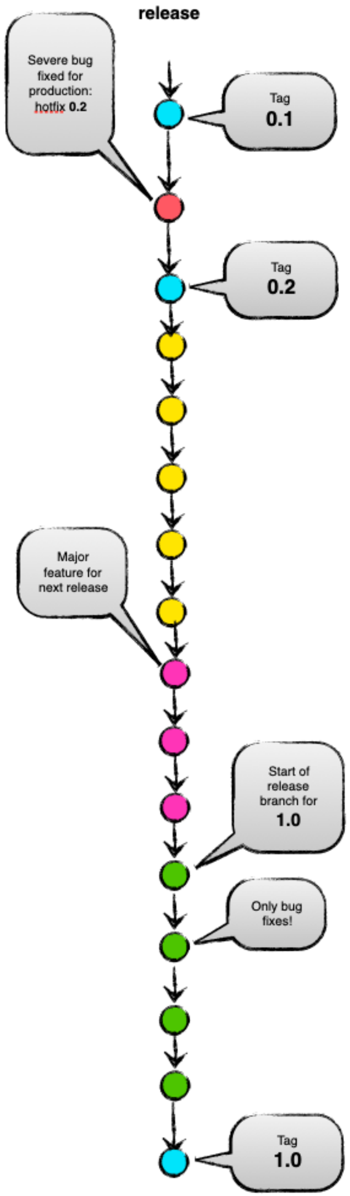
AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

CERN School of Computing

Linear history:

Using rebase and fast-forward merges:

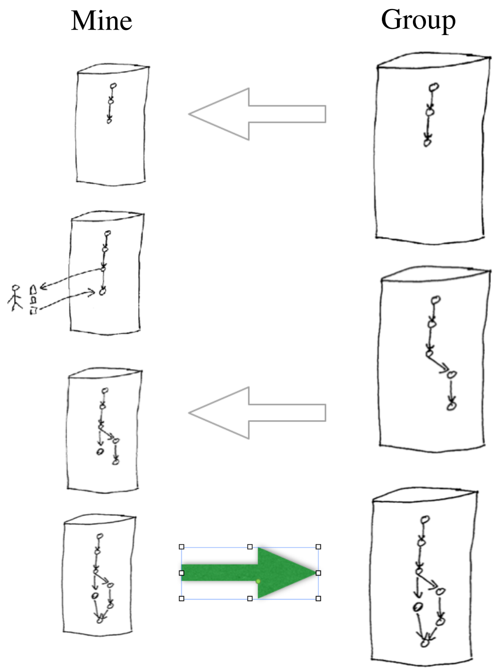

feature branches
develop
release branches
hotfixes
release
Time
Tag 0.1
Major feature for next release
Feature for future release
Severe bug fixed for production: hotfix 0.2
Incorporate bug fix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bug fixes!
Bug fixes from rel. branch may be continuously merged back into develop
Tag 1.0
Start of release branch for 2.0


release
Severe bug fixed for production: hotfix 0.2
Tag 0.1
Tag 0.2
Major feature for next release
Start of release branch for 1.0
Only bug fixes!
Tag 1.0

You want me to trust how many people?

How do you give 6,000 people access to a central repository?

Use a distributed repository and “pull requests”



Git-based developers have a full local repository

Commits have full context

“Push” moves all that to target

A “pull request” sends all that to somebody at the target, who can accept or not

When accepted, the merge is completed & both repositories in sync

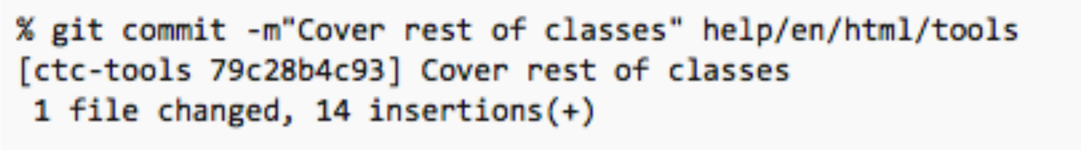
(Pull requests rarely rejected outright - usually it’s “fix these things and resend”)

Strong tools exist to make pull requests easy: CI test results, etc automated

More

Life Cycle of a Pull Request

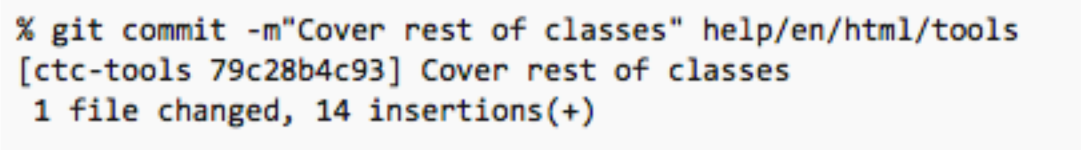
Bob is working on his laptop, and commits another change locally:



```
% git commit -m"Cover rest of classes" help/en/html/tools
[ctc-tools 79c28b4c93] Cover rest of classes
 1 file changed, 14 insertions(+)
```

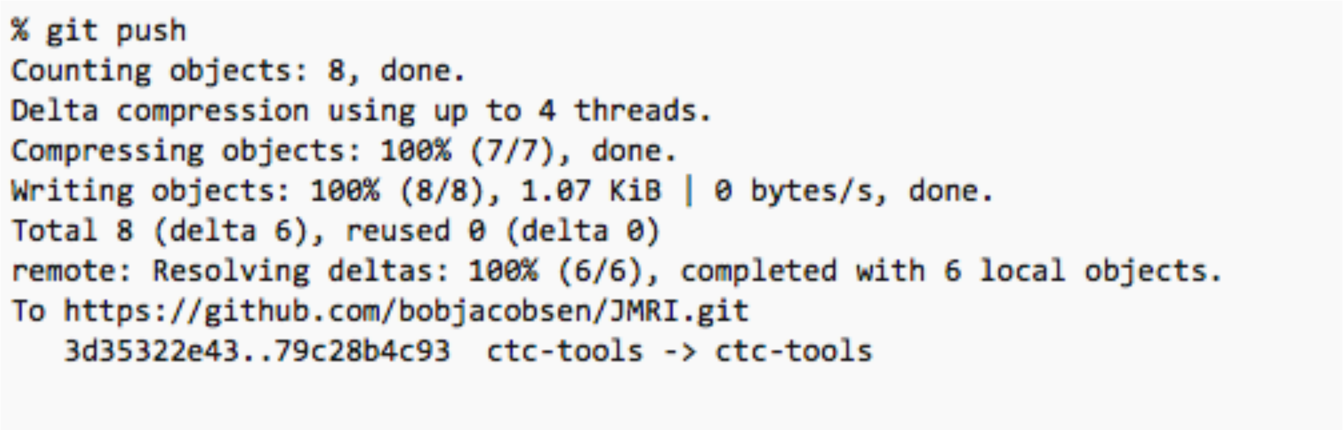
Life Cycle of a Pull Request

Bob is working on his laptop, and commits another change locally:



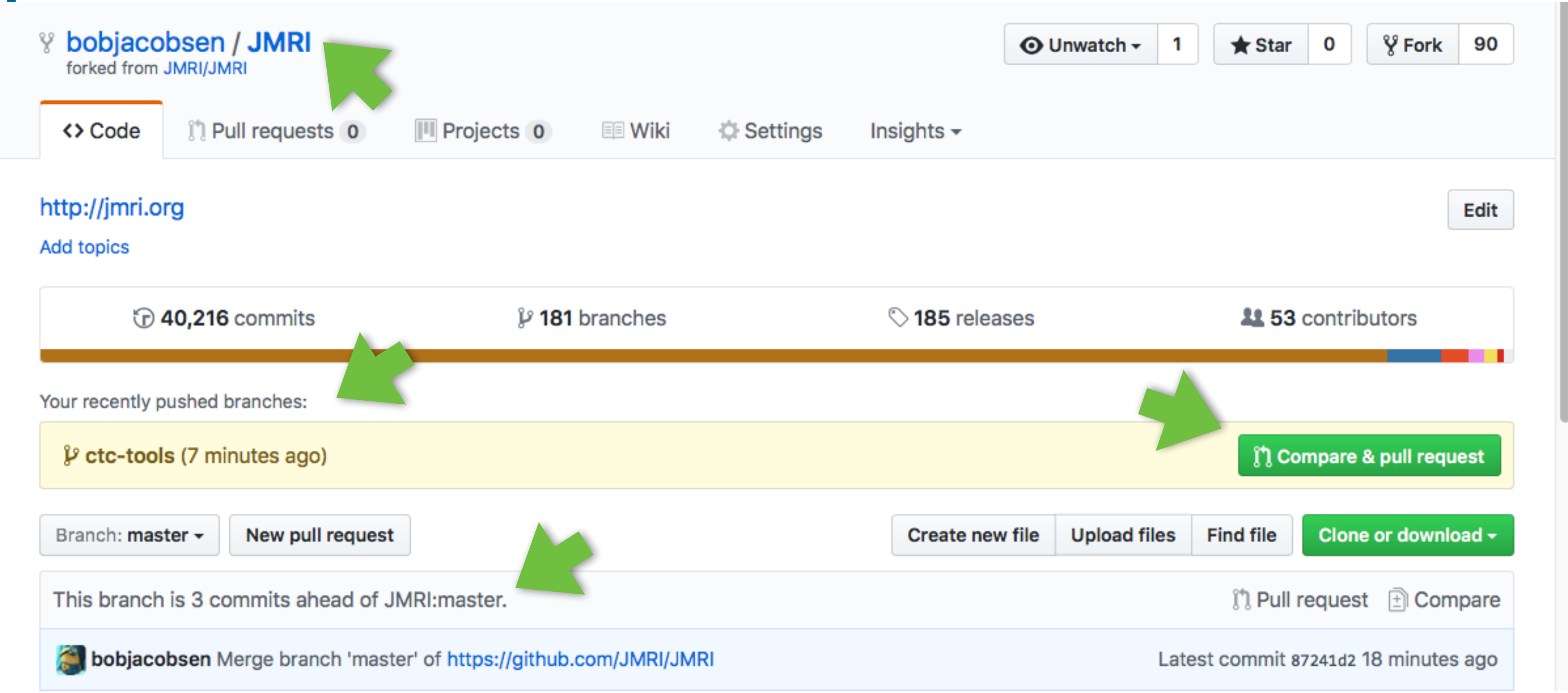
```
% git commit -m"Cover rest of classes" help/en/html/tools
[ctc-tools 79c28b4c93] Cover rest of classes
 1 file changed, 14 insertions(+)
```

He’s ready for that work to be reviewed, and wants to move it to a repository that’s always online:



```
% git push
Counting objects: 8, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (7/7), done.
Writing objects: 100% (8/8), 1.07 KiB | 0 bytes/s, done.
Total 8 (delta 6), reused 0 (delta 0)
remote: Resolving deltas: 100% (6/6), completed with 6 local objects.
To https://github.com/bobjacobsen/JMRI.git
   3d35322e43..79c28b4c93  ctc-tools -> ctc-tools
```

Life Cycle of a Pull Request



bobjacobsen / JMRI
forked from JMRI/JMRI

http://jmri.org

Add topics

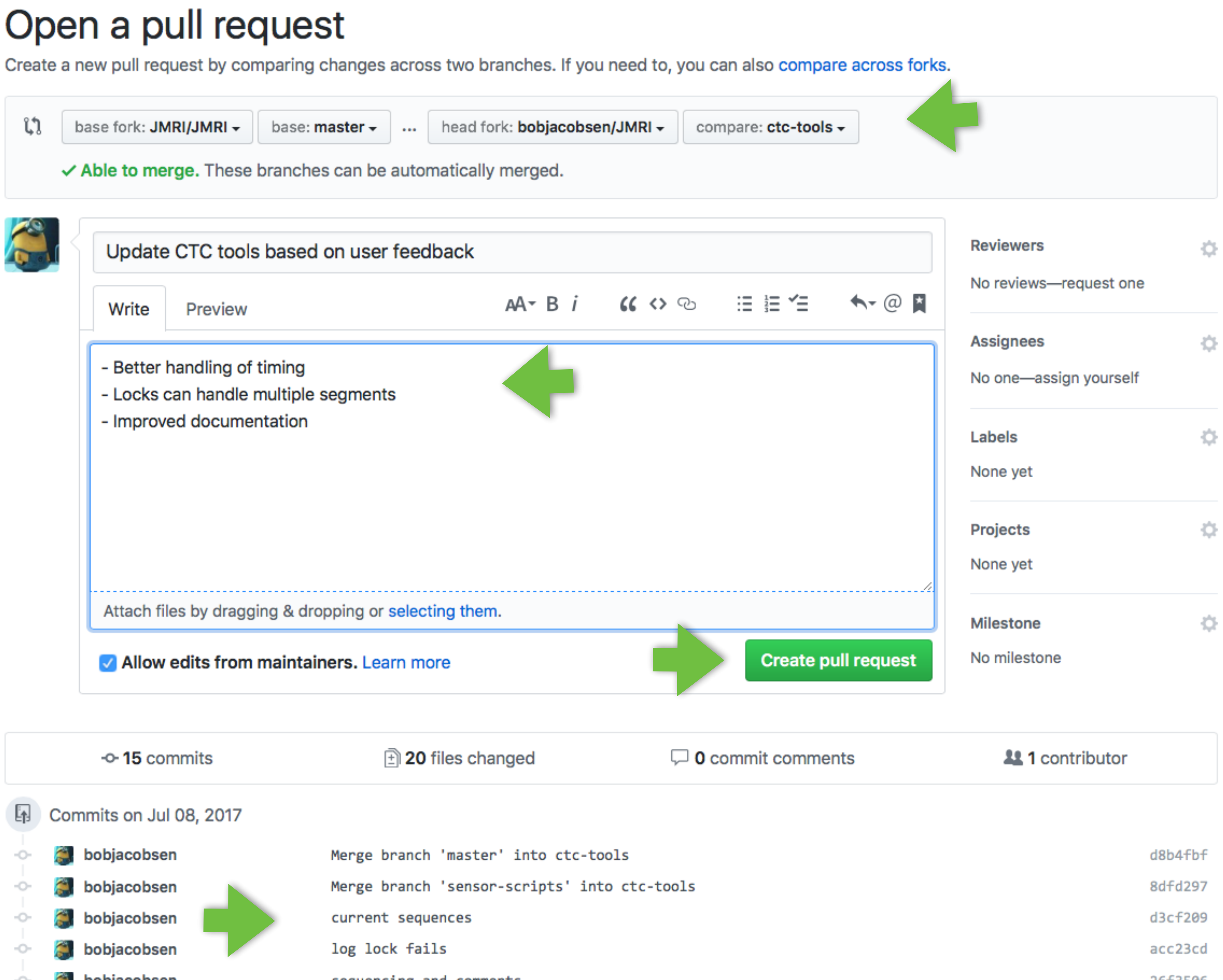
40,216 commits | 181 branches | 185 releases | 53 contributors

Your recently pushed branches:

ctc-tools (7 minutes ago)

This branch is 3 commits ahead of JMRI:master.

bobjacobsen Merge branch 'master' of https://github.com/JMRI/JMRI Latest commit 87241d2 18 minutes ago



Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks.

base fork: JMRI/JMRI ▾ | base: master ▾ ... head fork: bobjacobsen/JMRI ▾ | compare: ctc-tools ▾

✓ Able to merge. These branches can be automatically merged.

Update CTC tools based on user feedback

Write | Preview

- Better handling of timing
- Locks can handle multiple segments
- Improved documentation

Attach files by dragging & dropping or selecting them.

☑ Allow edits from maintainers. Learn more

Create pull request

Reviewers

No reviews—request one

Assignees

No one—assign yourself

Labels

None yet

Projects

None yet

Milestone

No milestone

15 commits | 20 files changed | 0 commit comments | 1 contributor

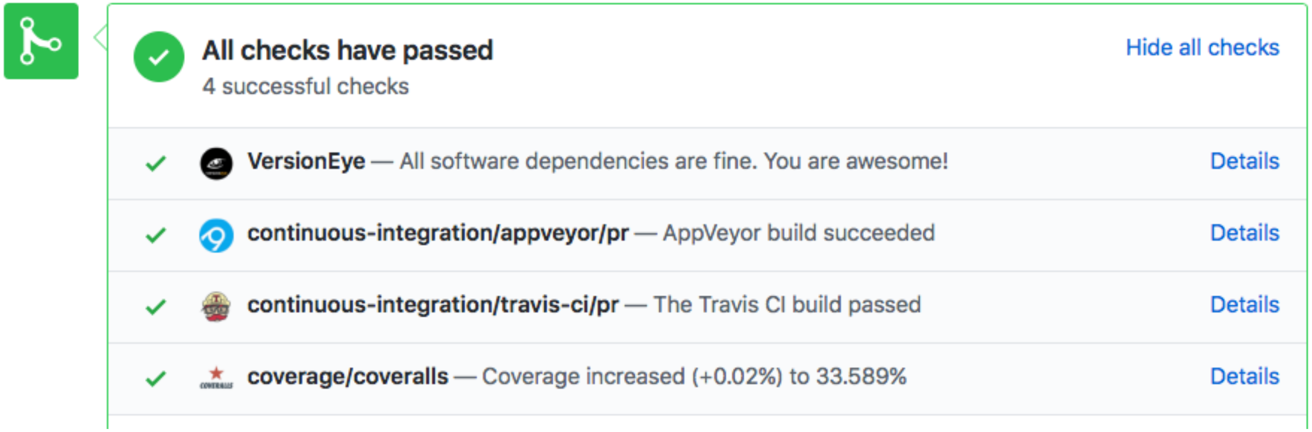
Commits on Jul 08, 2017

bobjacobsen	Merge branch 'master' into ctc-tools	d8b4fbf
bobjacobsen	Merge branch 'sensor-scripts' into ctc-tools	8dfd297
bobjacobsen	current sequences	d3cf209
bobjacobsen	log lock fails	acc23cd
bobjacobsen	sequencing and comments	26f2596

Life Cycle of a Pull Request

Once created:

Continuous integration tests are run

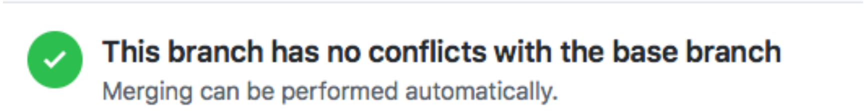


All checks have passed
4 successful checks
Hide all checks

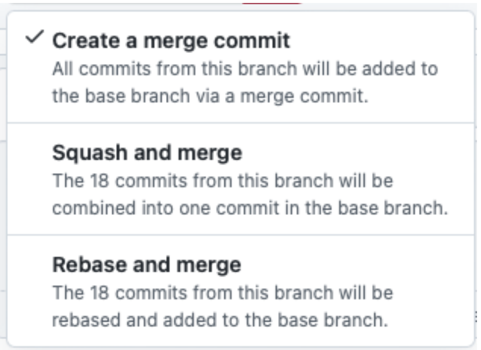
✓ VersionEye — All software dependencies are fine. You are awesome! Details
✓ continuous-integration/appveyor/pr — AppVeyor build succeeded Details
✓ continuous-integration/travis-ci/pr — The Travis CI build passed Details
✓ coverage/coveralls — Coverage increased (+0.02%) to 33.589% Details

Reviews happen

Merge checks are done



✓ This branch has no conflicts with the base branch
Merging can be performed automatically.

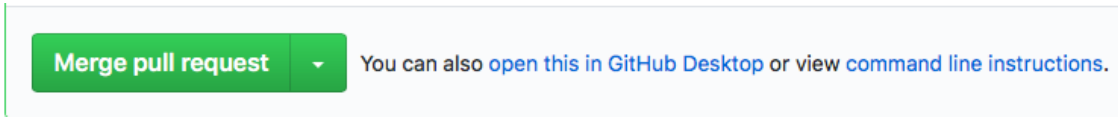


✓ Create a merge commit
All commits from this branch will be added to the base branch via a merge commit.

Squash and merge
The 18 commits from this branch will be combined into one commit in the base branch.

Rebase and merge
The 18 commits from this branch will be rebased and added to the base branch.

And finally, somebody with authorization can click this:



You can also open this in GitHub Desktop or view command line instructions.

to complete the merge onto the desired branch in the main repository.

Three choices for merging PRs:

Merge Commit

- **Contains the entire development history from the merged branch**
- **Usually a merge commit, sometimes fast-forward**
- **For many, this is the default approach**

Squash and Merge

- **Merges entire change as a single commit**
- **Usually a merge commit, sometimes fast-forward**
- **Contains the entire change in a single commit**
  - **Optionally, a more comprehensive, holistic comment**

Rebase

- **Puts a single commit on the end**
- **Always a fast-forward commit**
- **Contains the entire change in a single commit**
  - **Optionally, a more comprehensive, holistic comment**

How do you use this all?

Individually:

Use it to work independently

- Both of others, and of yourself!

Collaborate on intermediate results

- Clean branches easy to share: “Try bobj/FixIssue10343”

Shape your work result to make it understandable

- Comments, squashing, comments, rebasing as tools

Integrate early and often!

- Pull “main” and make sure work is still OK

For a collaboration project:

Help people work at the scales they need to

- Individually, in small groups, large groups, …

Control how code is added/updated

- Shaping contents of common development, releases

Make the contents understandable

- Tags, known branching / linear history

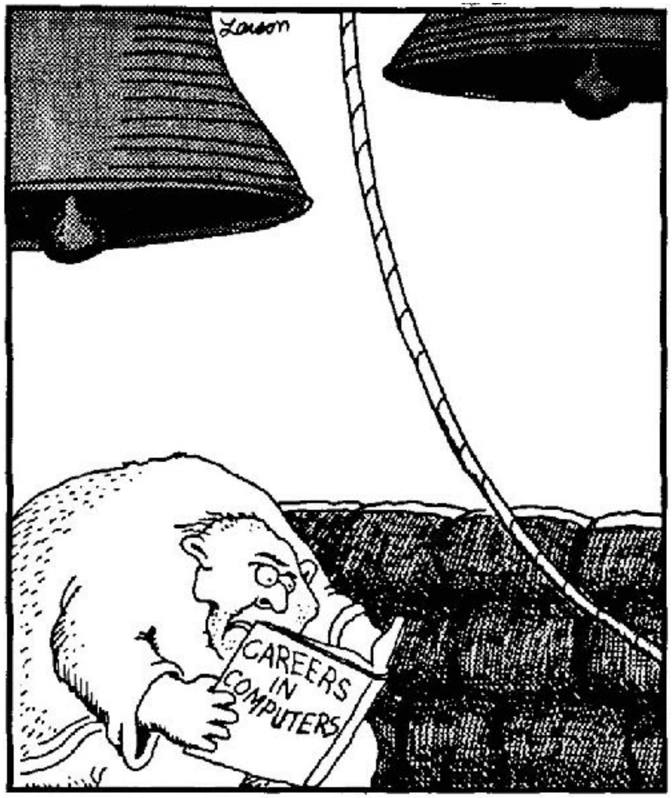
Series summary

Software engineering is the art of building complex computer systems

It’s ideas and techniques spring from our need to handle size & complexity

As you do your own work & develop your own skills, consider:

- How your effort effects or contributes to things 10X, 100X, 1000X larger
- How you’ll do things different/better when it’s your problem



Questions? jacobsen@berkeley.edu

Exercises

Test Frameworks

Performance Profiling

Memory Issues

Code Management

More

Instructions to get started on Indico (Tools & Techniques E1)

https://indico.cern.ch/event/1254984/contributions/5272132/

You’ll work in pairs. Try to find somebody with complementary skills!

Learn about each topic, spend more time on the ones that interest you.

Speed is not the issue: no reward for first done, no complaint about last.

Think about what you’re doing: There are larger lessons to be found!