Enterprise versus Open Source Development Policy

Kostis Kapelonis

Athens Greece, December 2013

Question

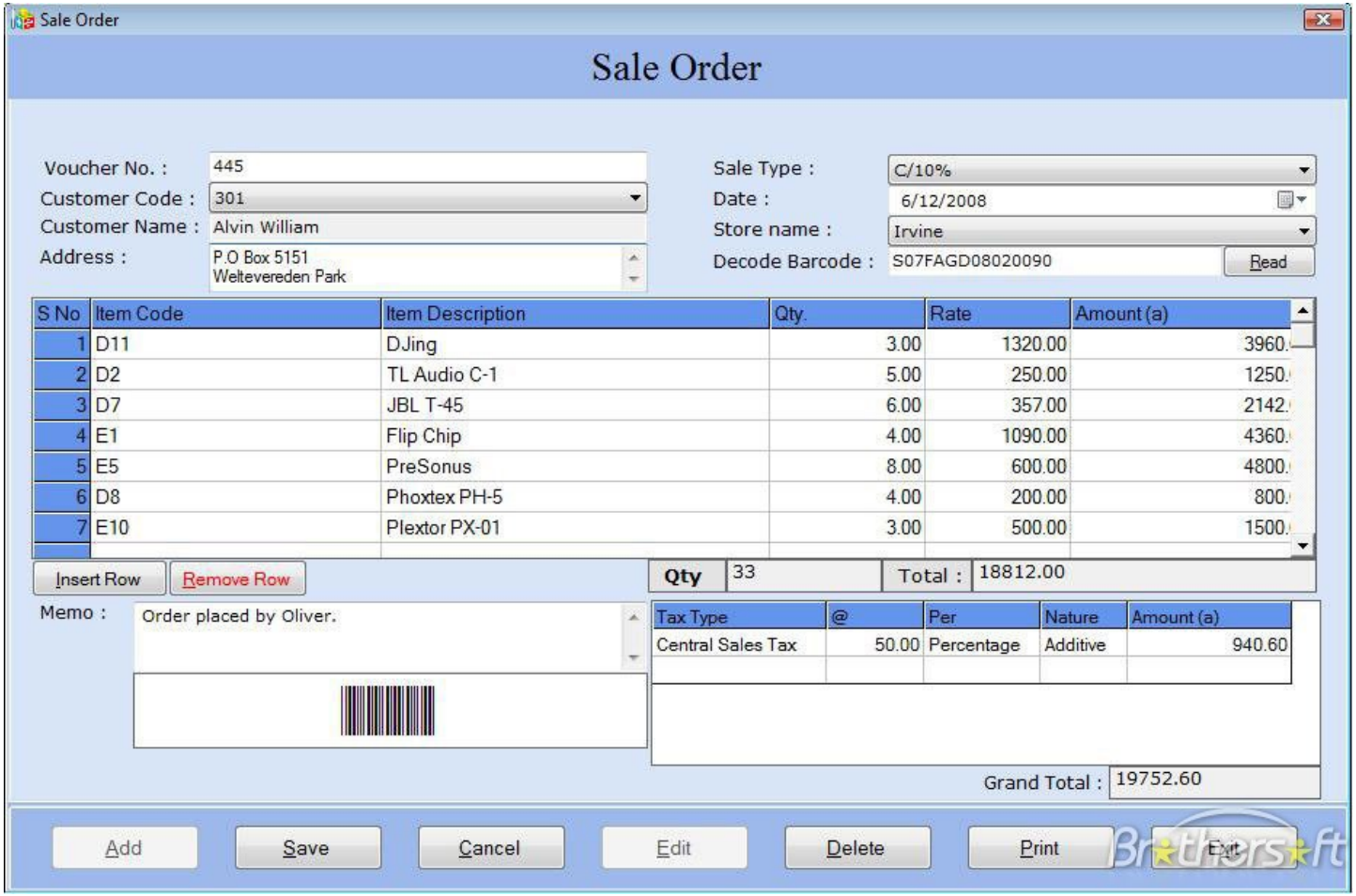
What is the best Methodology for developing software Applications?

Question

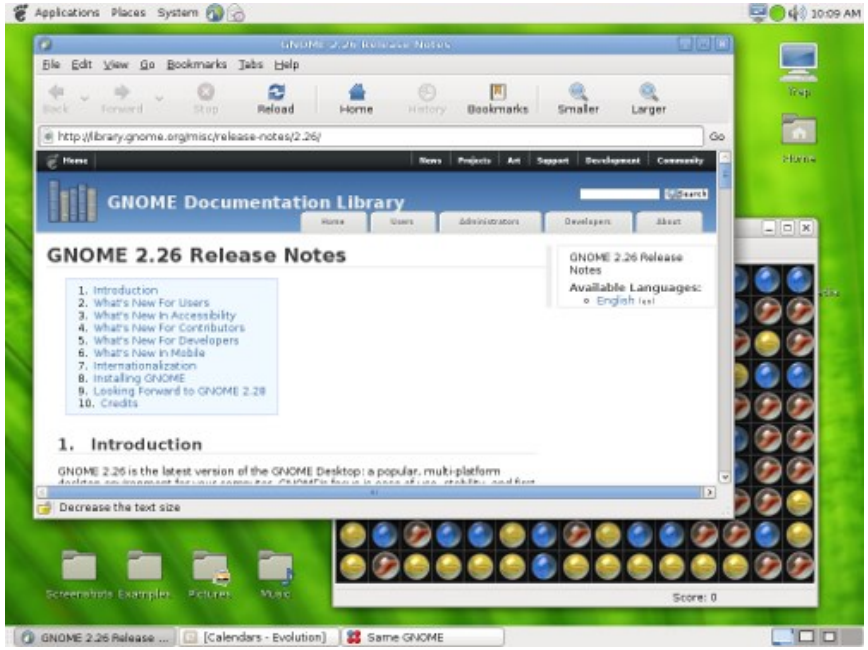
most appropriate

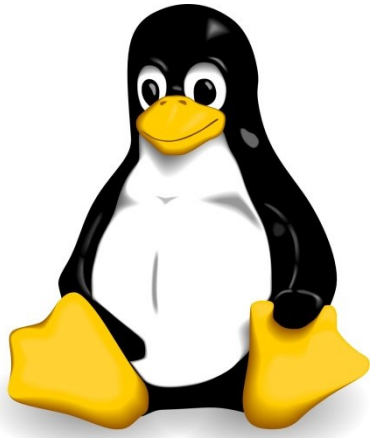
What is the best Methodology for developing software Applications?

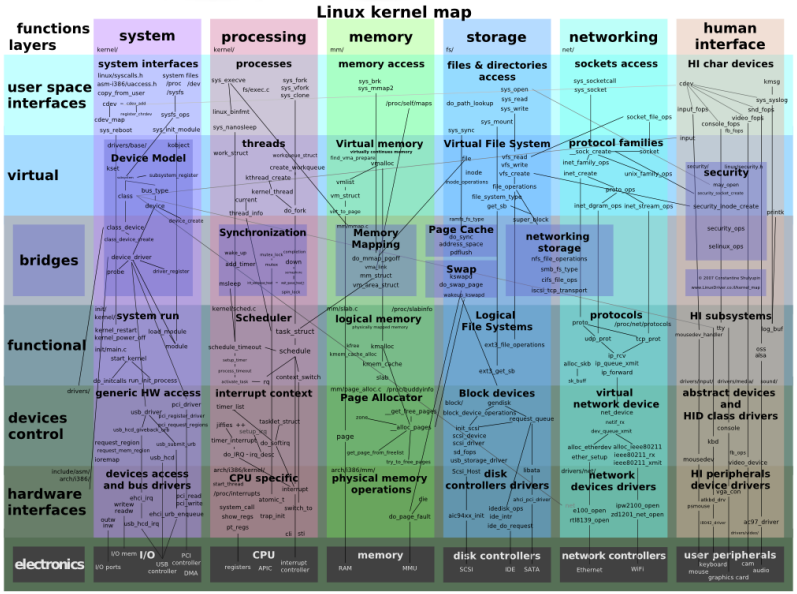
Enterprise Software



Open Source general purpose software

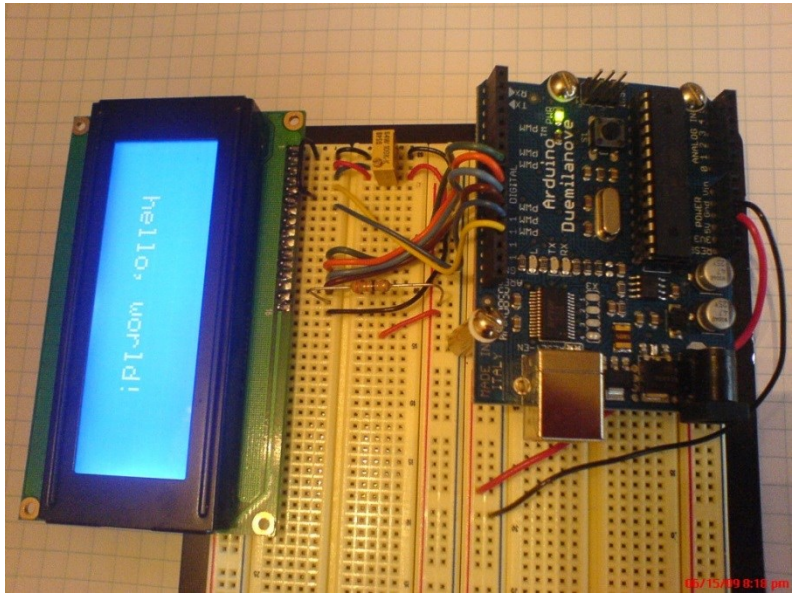




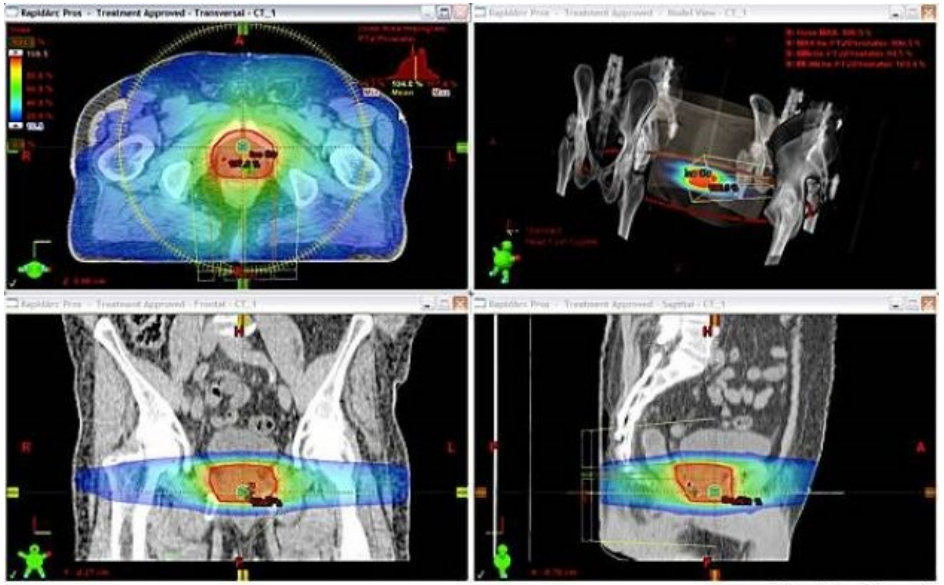


TRASYS
WE GET IT DONE

Embedded software



Medical Software



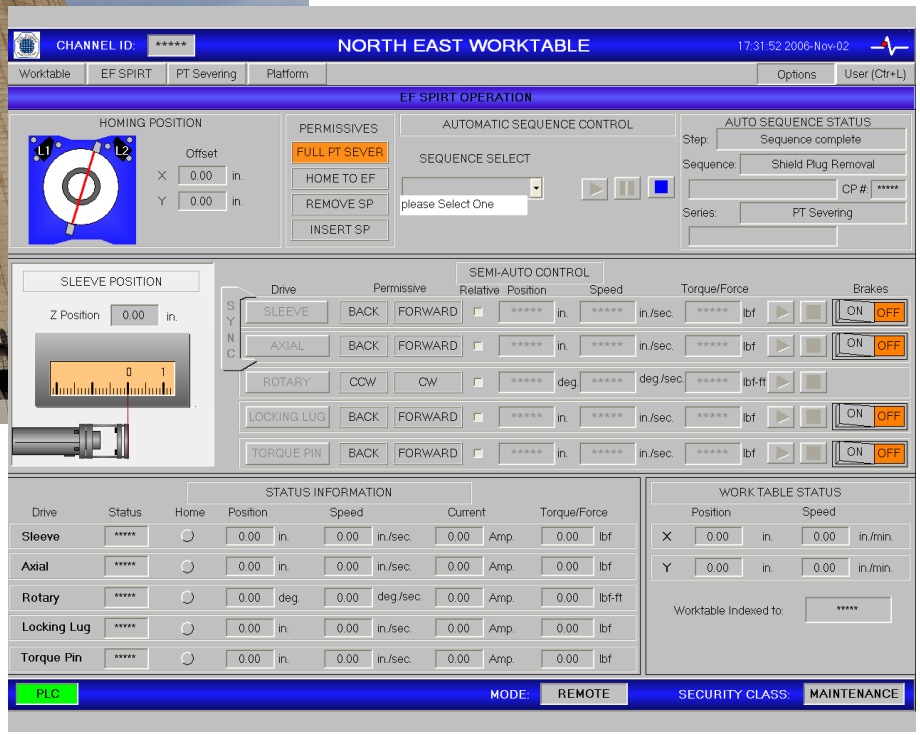
Liz Hafalia / The Chronicle

Military Software

QNX SOFTWARE SYSTEMS

VxWorks

Nuclear Reactor Software



TRASYS
WE GET IT DONE

The Team

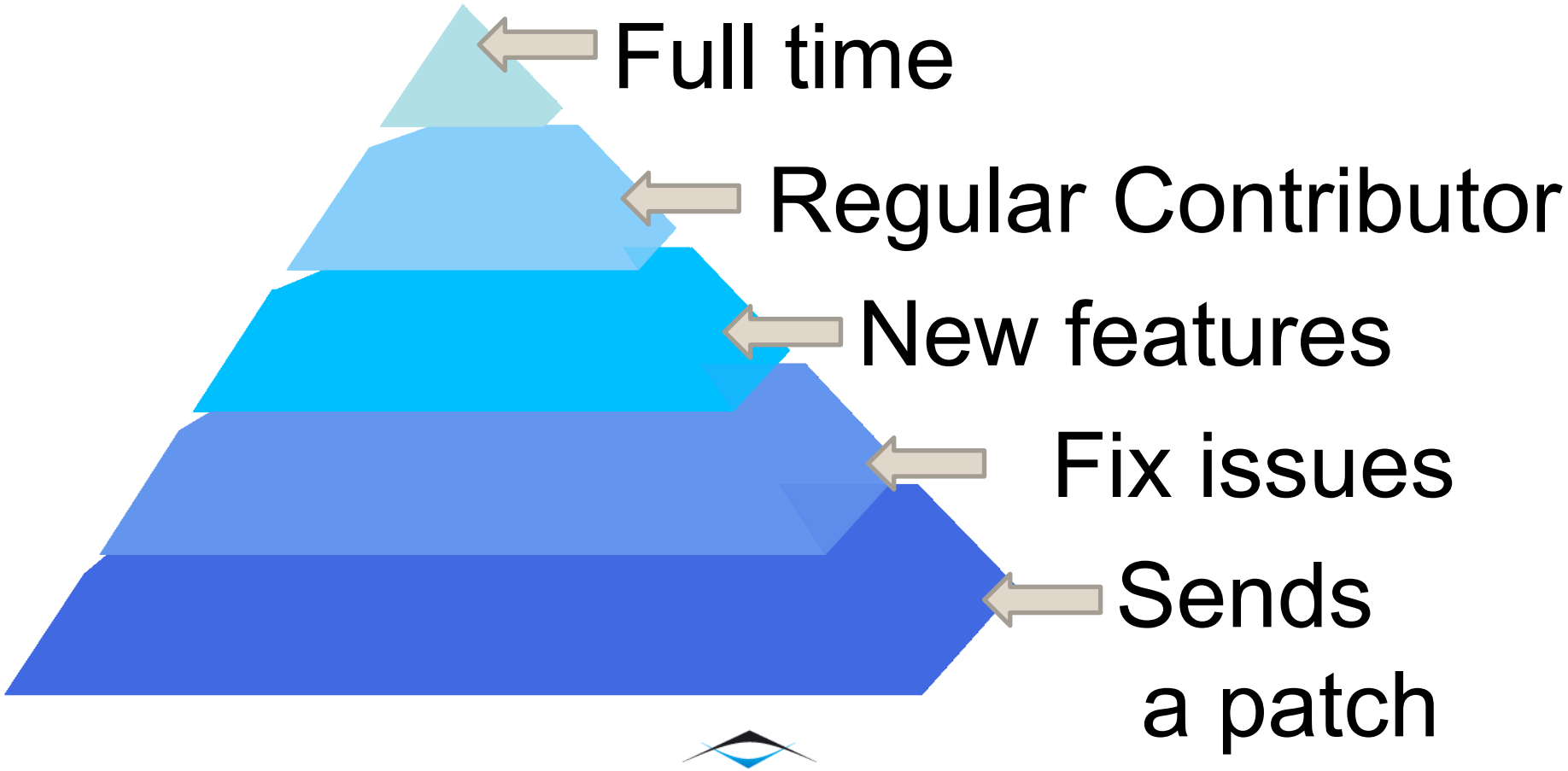
Roles and privilages

Open source roles (privilages)

- Client is everybody
- Developer could be anybody

Open source roles (involvement)

- Different level of involvement
- Author, Committer, Lieutenant, Contributor, Dictator



Enterprise Team

Unified Team

Features

Who approves what

Open source stakeholders

- There is no “hard” timetable
- There is no “hard” set of features

Features

“Client”

Features

“Developer”

Open source forks

MySQL vs MariaDB

Emacs vs XEmacs

Debian vs Ubuntu

WebKit vs KHtml

XFree vs X.org

LibreOffice vs OpenOffice

Xwiki vs FossWiki

Feature wars

Enterprise stakeholders

- Stakeholder is the client. Ultimate power on features
- Client proposes features, clients approves features (same entity)

Features

Client

Developer

Changesets

Choosing The base line

Enterprise stakeholders

- Opensource loves the latest and greatest version (constant change)
- Enterprise loves backwards (and forwards) compatibility

A Patch comes
in for an older version

ORACLE®

"Please verify on
latest version"

Regression
Testing

Other stuff

Different development mindset

Other differences

- Release early, release often
- It is done when it is done
- Show me the code
- If it compiles, ship it
- Users are lusers
- Cathedral versus Bazaar

Biggest difference

Trust on people

Enterprise stakeholders

- Opensource must deal with the crowd (also malicious commits)
- Enterprise has controlled teams

Open Source

Unknown

Enterprise

Handpicked

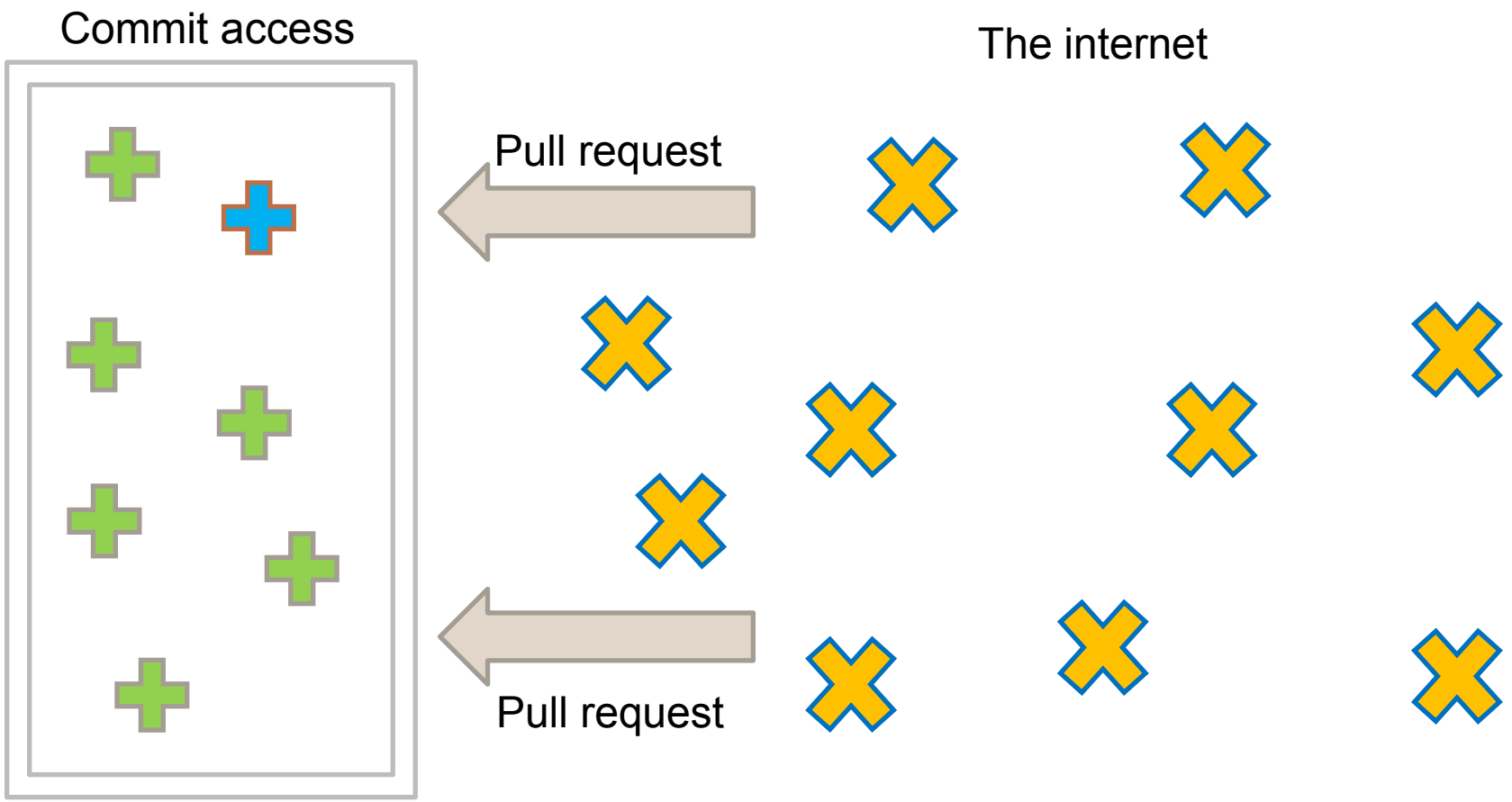
Border Control

Trust no one (until they get in)

Enterprise versus Open Source

Either you are trusted or not

Enterprise versus Open Source



One big barrier (In or out)

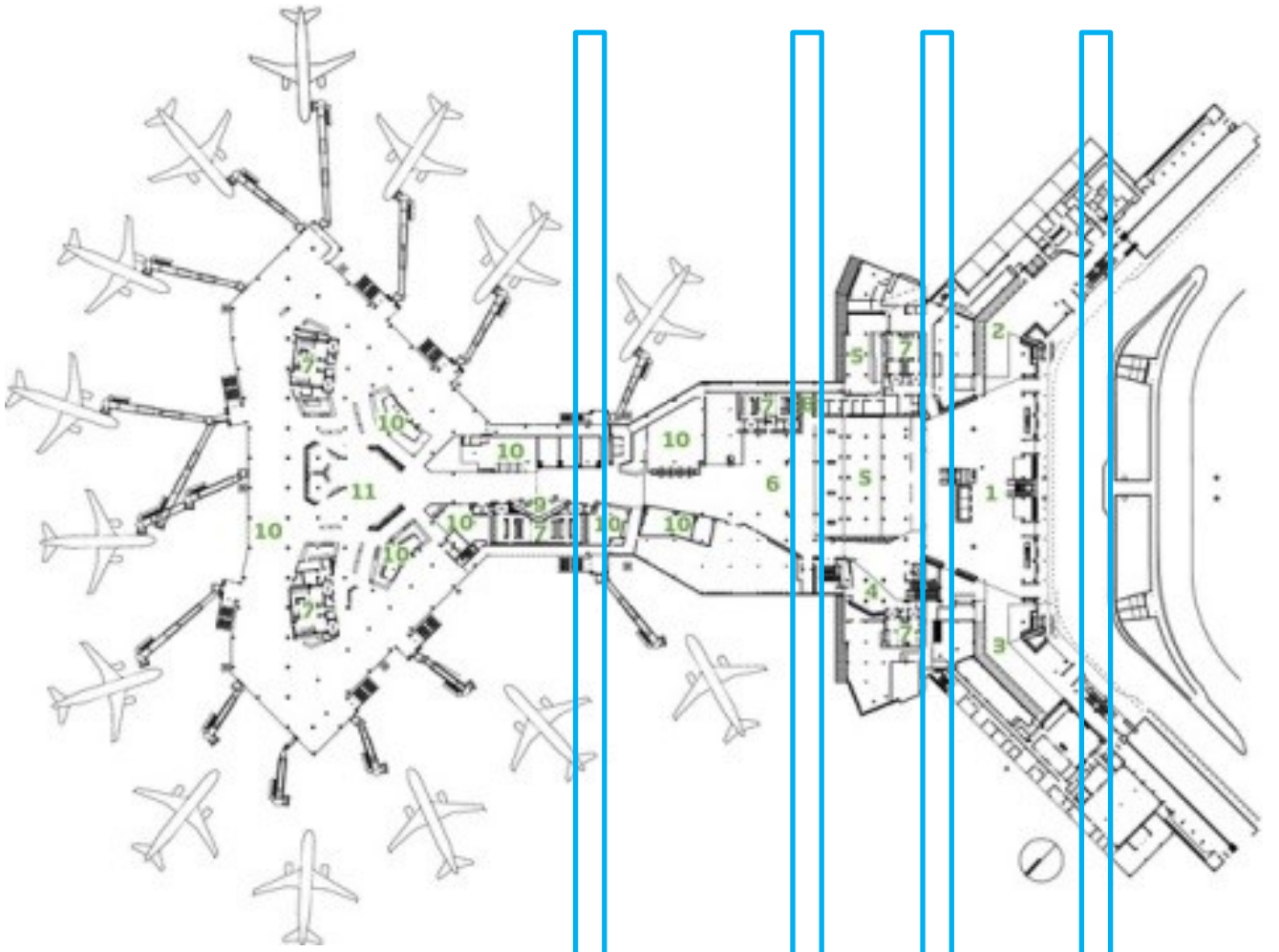
A different approach

Layers of trust of code
(not people)

Enterprise versus Open Source

The airport approach

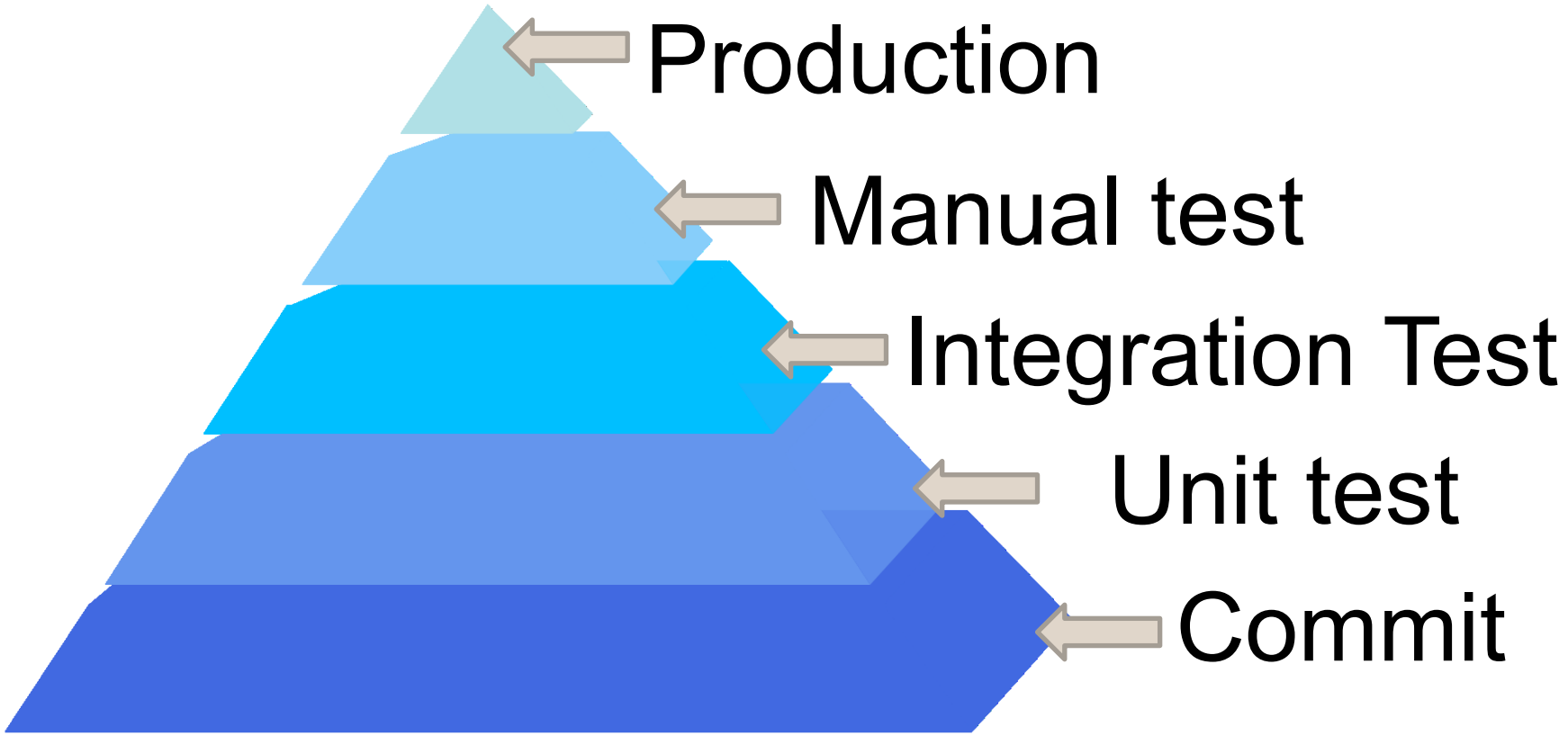
Enterprise versus Open Source



TRASYS
WE GET IT DONE

Code quality certainty

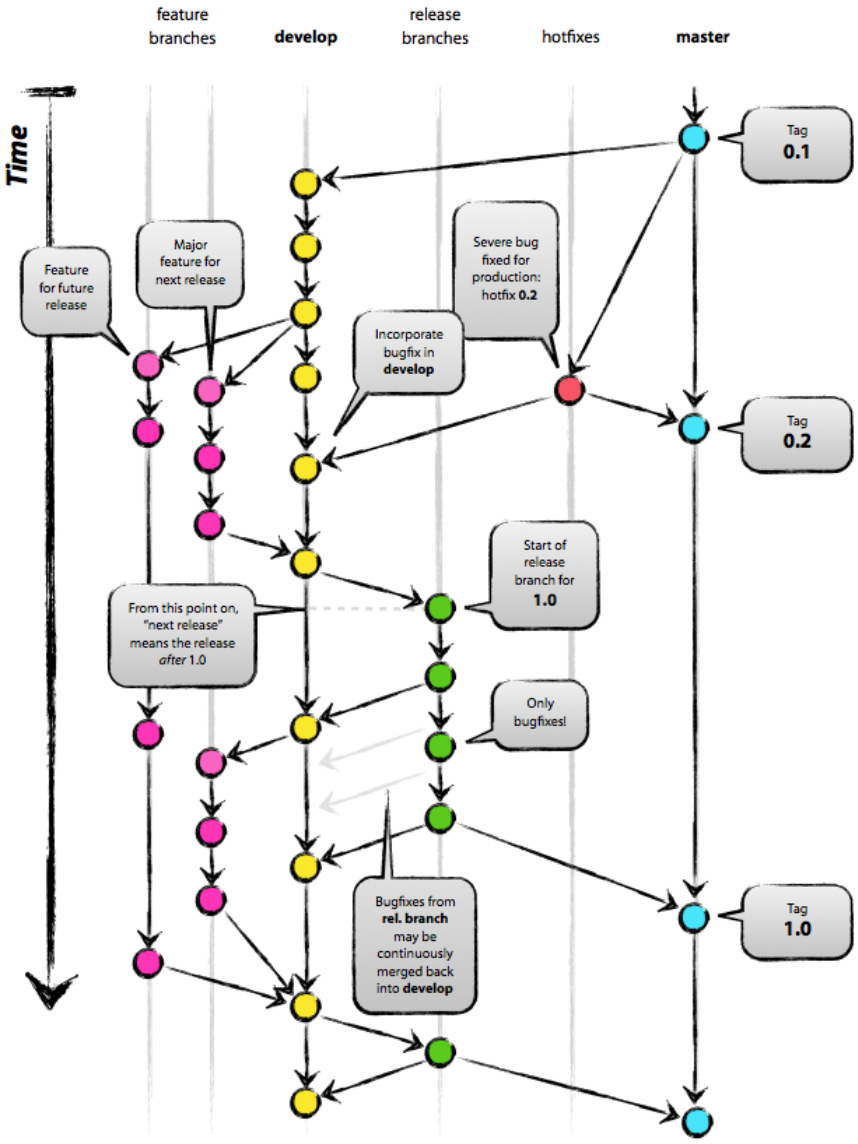
- Different level of code trust
- Code starts as unsafe and reaches production status



In action

Tools and support

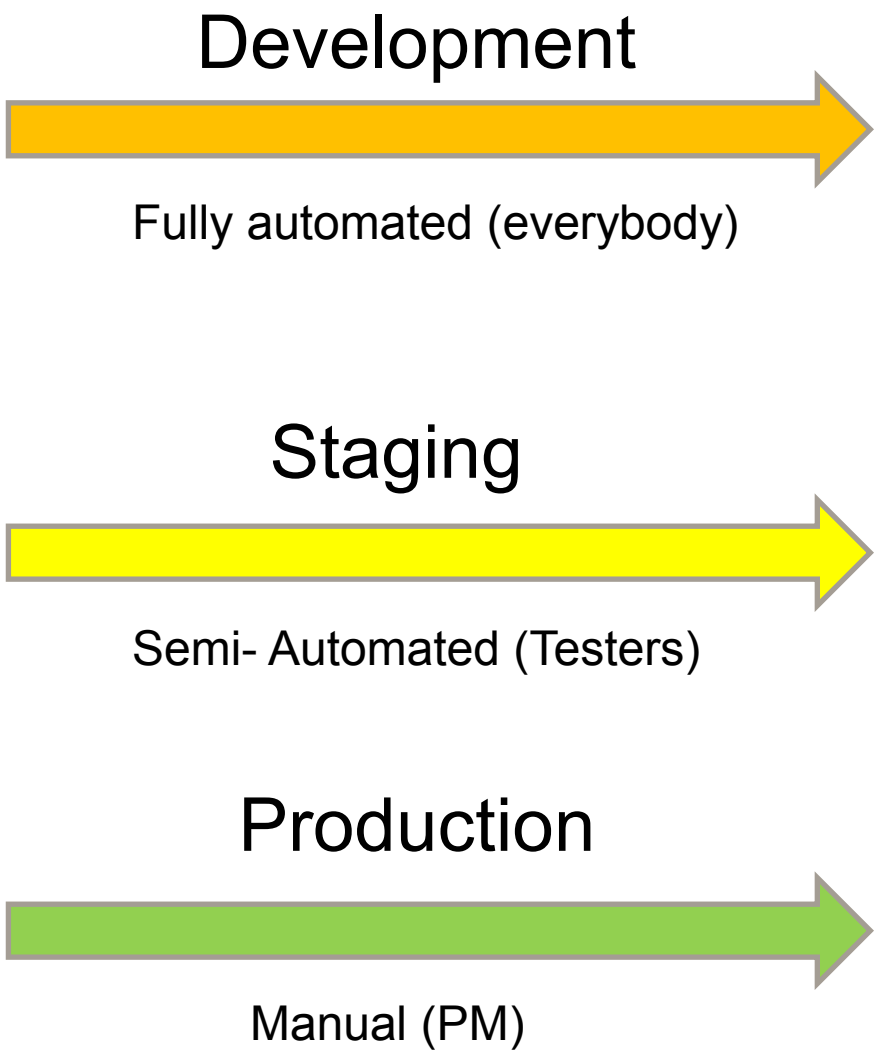
Branch model



A successful Git model

TRASYS
WE GET IT DONE

Layers of trust



All commits go on dev. Should compile.

Test environments use staging.

Tags come from Production

Build jobs

- Developer build (almost every commit) – unit tests
- Integration tests (every 30 minutes)
- Sonar build (once a day)
- Promotion job (for testers)
- Staging environments (multiple)
- Release jobs
- Completely automated for day usage

Build server

Developer responsibilities

- Commit on “dev” branch. Should run unit tests first locally
- Pull/Merge as needed freely (but only on dev branch)
- Multiple developers can work on same feature branch
- Monitor build status and fix broken builds
- All branches are remote (no local code)
- “Code that is not committed does not exist”
- Code review (before merging with “dev”)
- Commit messages have JIRA number
- Feature branches can be long or short
- For long lived – pull from “dev” daily

Developer

QA responsibilities

- Runs tests on “Frozen” staging
- Promote a build from dev to staging (build job)
- Approve staging to PM (so that it can be released)
- During releases there is no promotion (hot fixes go on staging)
- Regression testing with maintenance branch

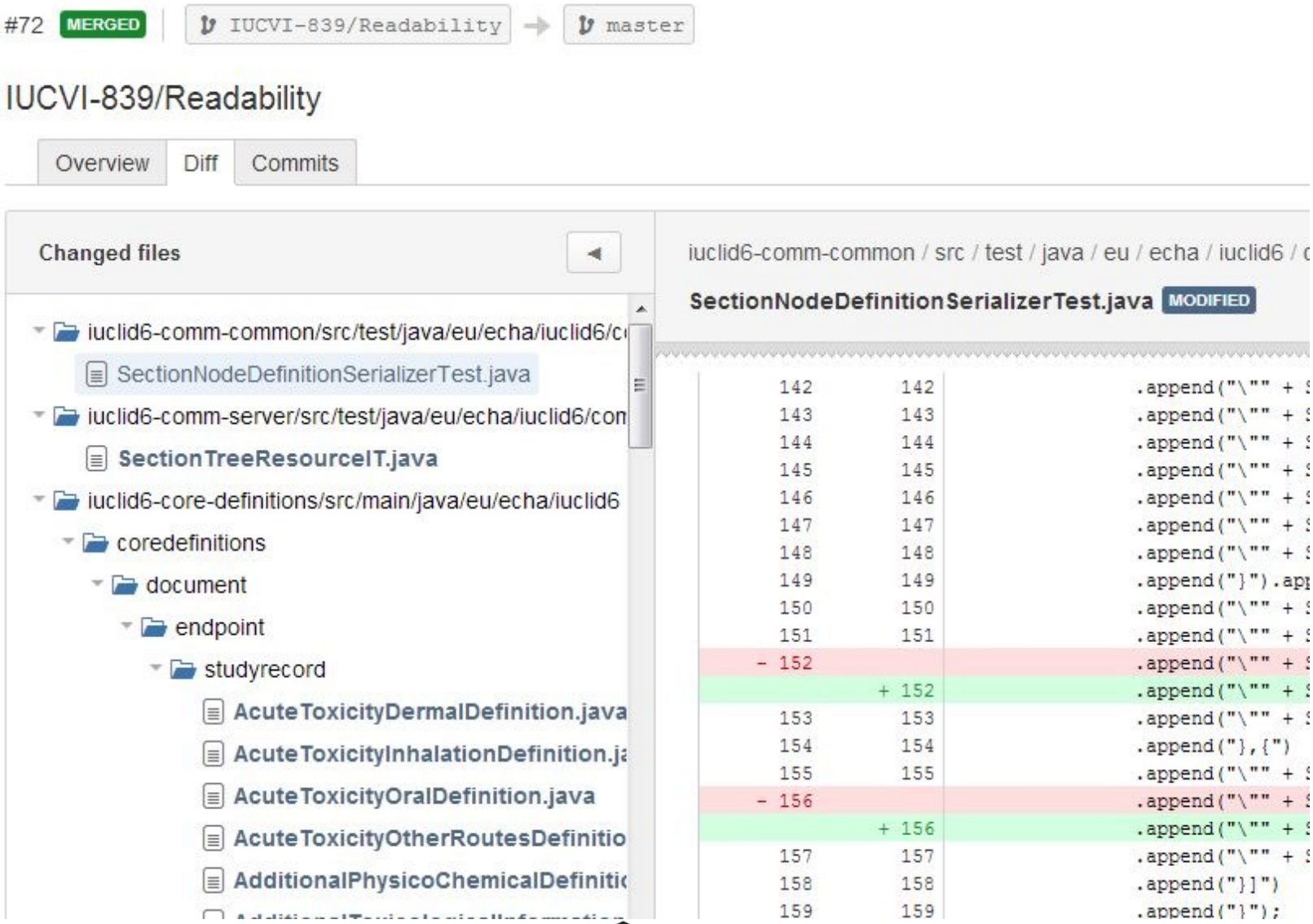
Tester

PM responsibilities

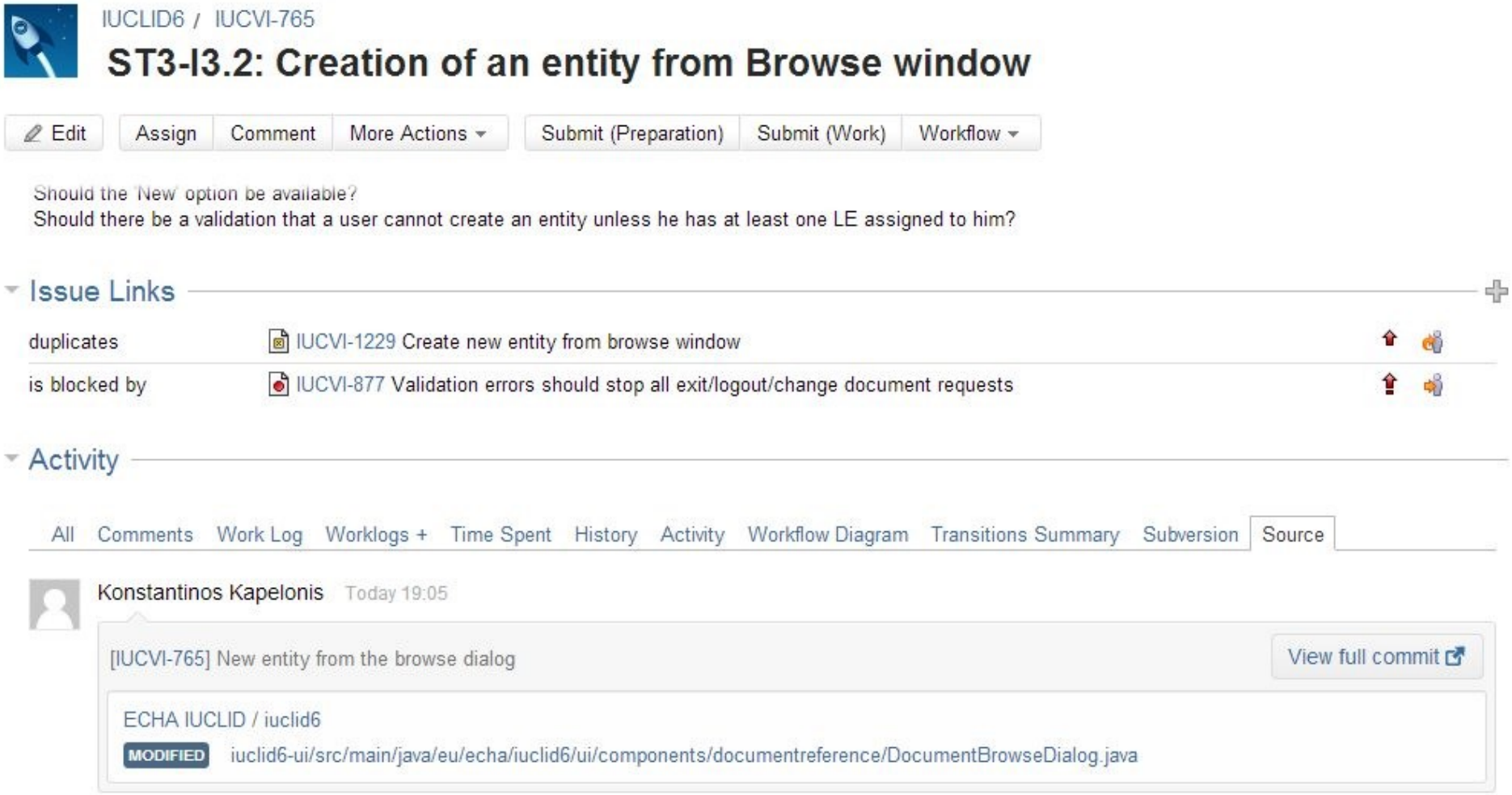
- Ranks features
- Approves features (by the client)
- Approves late release stages
- Approves Tags and releases from production branch

Project Manager

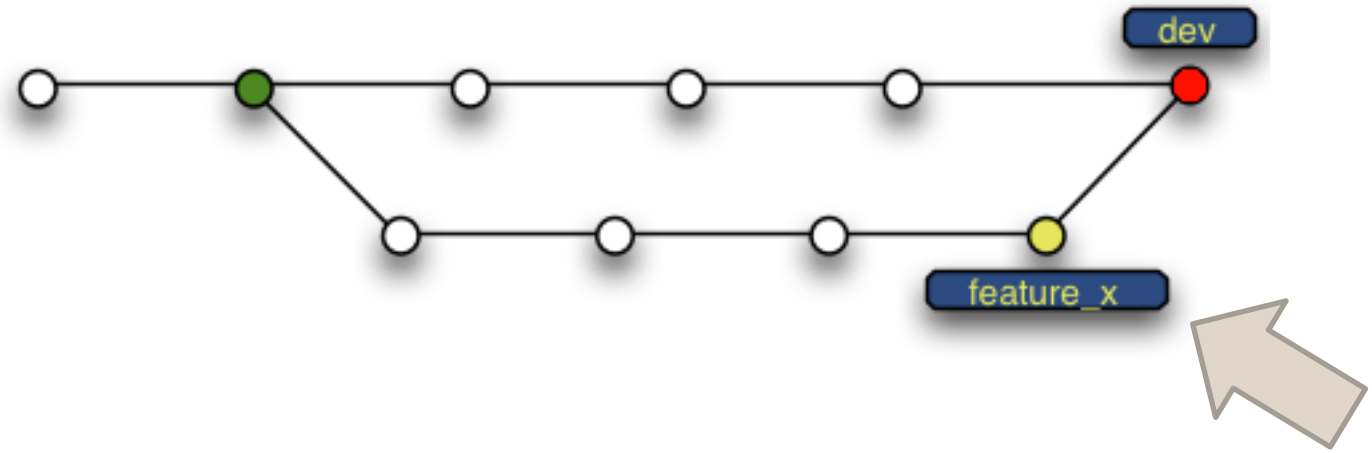
Code review of (merged) feature



JIRA - GIT


IUCLID6 / IUCVI-765
ST3-I3.2: Creation of an entity from Browse window
Edit
Assign
Comment
More Actions
Submit (Preparation)
Submit (Work)
Workflow
Should the 'New' option be available?
Should there be a validation that a user cannot create an entity unless he has at least one LE assigned to him?
Issue Links
duplicates
IUCVI-1229 Create new entity from browse window
is blocked by
IUCVI-877 Validation errors should stop all exit/logout/change document requests
Activity
All
Comments
Work Log
Worklogs +
Time Spent
History
Activity
Workflow Diagram
Transitions Summary
Subversion
Source
Konstantinos Kapelonis
Today 19:05
[IUCVI-765] New entity from the browse dialog
View full commit
ECHA IUCLID / iuclid6
MODIFIED
iuclid6-ui/src/main/java/eu/echa/iuclid6/ui/components/documentreference/DocumentBrowseDialog.java

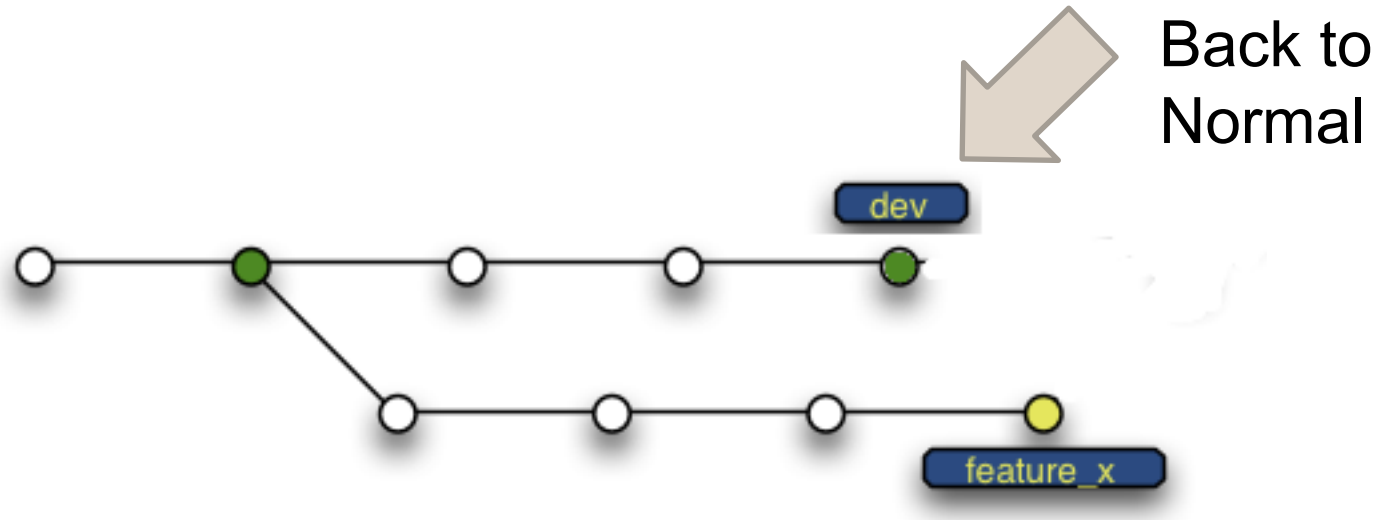
Bad commits



Crap

Oops – bad commit

Bad commits



Git reset to previous commit

Enterprise versus Open Source

Comparison

Simplicity in day to day operations

Open Source

Pull, push,
merge,
commit,
squash,
rebase

Enterprise

Pull, push,
merge,
commit

Simplicity in Build server setup

Open Source

Custom scripts, new jobs for pull requests

Enterprise

Usage out of the box

Simplicity in Build server load

Open Source

Jobs O (n)

(n = open PR)

Builds O(n^2)

(n = commits)

Enterprise

Jobs O (1)

Builds O(n)

(n= commits to dev)

How branches are treated

Open Source

1 Branch = 1 feature = 1contributor = 1 pull request

Enterprise

Only final result matters (all features integrated)

Stability of workspace

Open Source

Enterprise

Your branch may have been rebased. Checkout again!

Dude, wtf?

You can work on any branch

Co-operation among developers

Open Source

Enterprise

External regular contributors difficult (unless isolated work)

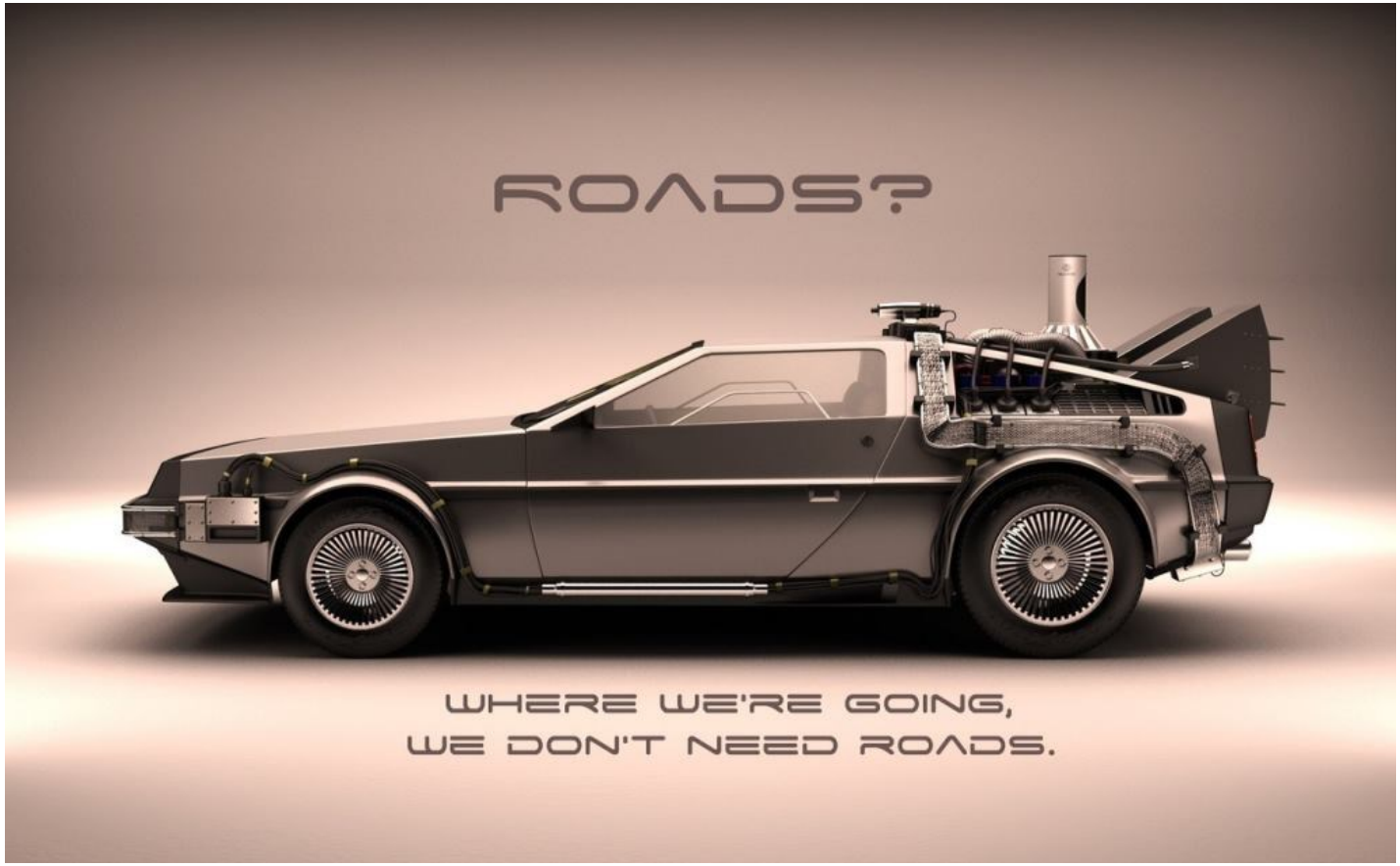
Multiple external developers on same branch

Conclusion

Use the open source approach for open source software

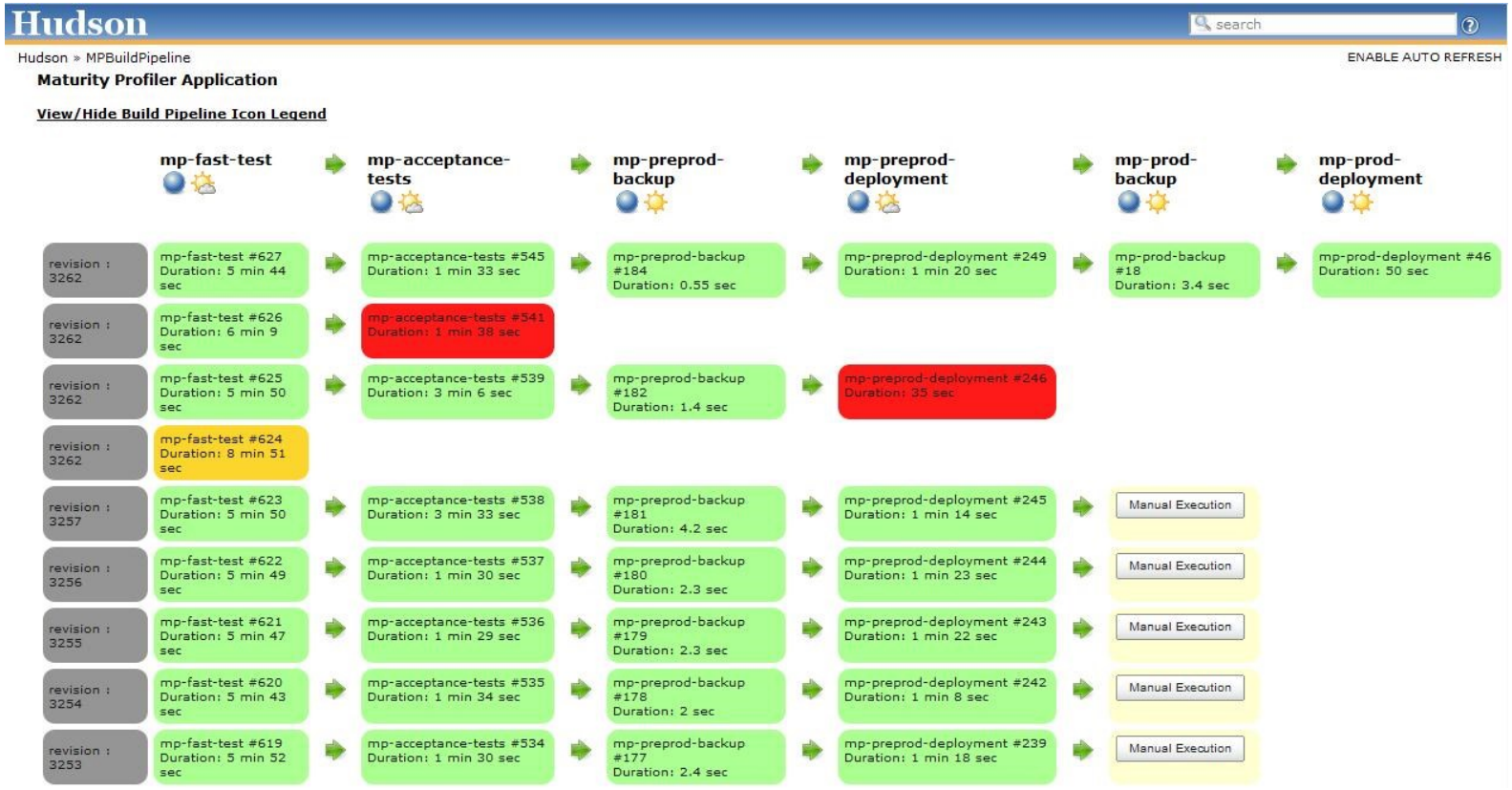
Use the Enterprise approach for Enterprise software

The future



Where we want to go

The future



Software Pipelines

TRASYS
WE GET IT DONE

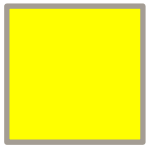
Thank you

Enterprise versus Open Source

Backup Slides

Enterprise versus Open Source

Valid commit



Invalid commit

“fix build” commit

Build (success)

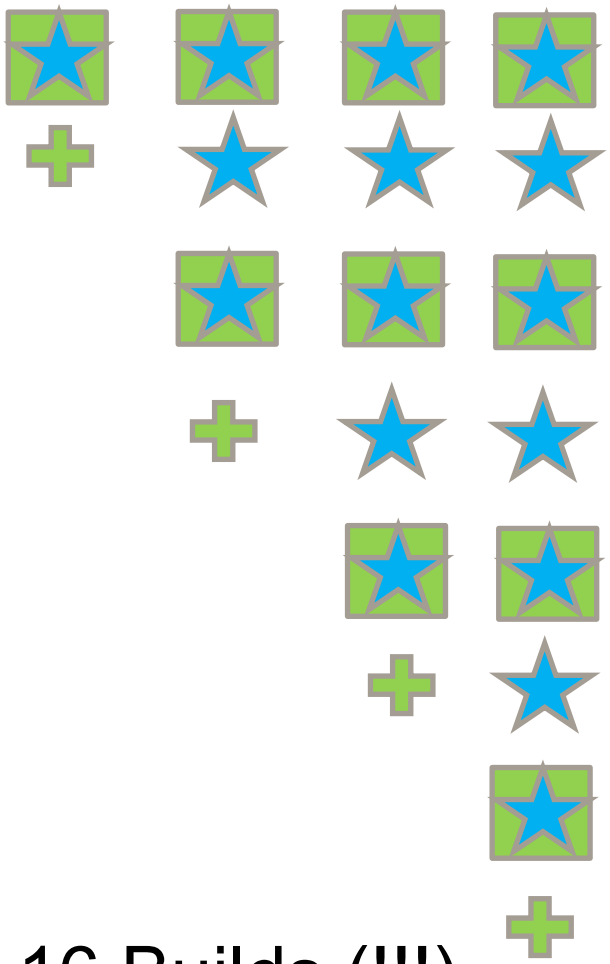
Build(fail)

Feature finished (ok)

Feature finished (not ok)

Merging 4 open pull requests

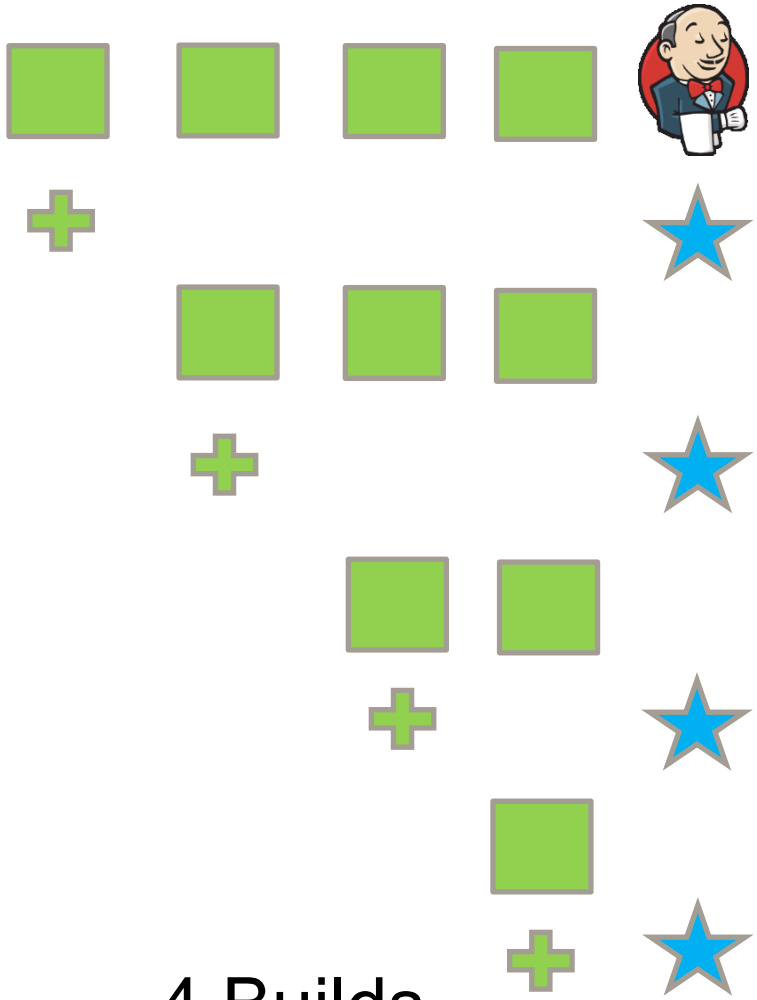
Open Source (4 PRs)



16 Builds (!!!)

Time

Enterprise (4 PRs)



4 Builds

TRASYS
WE GET IT DONE

Merging 4 open pull requests

4PRs =16 builds

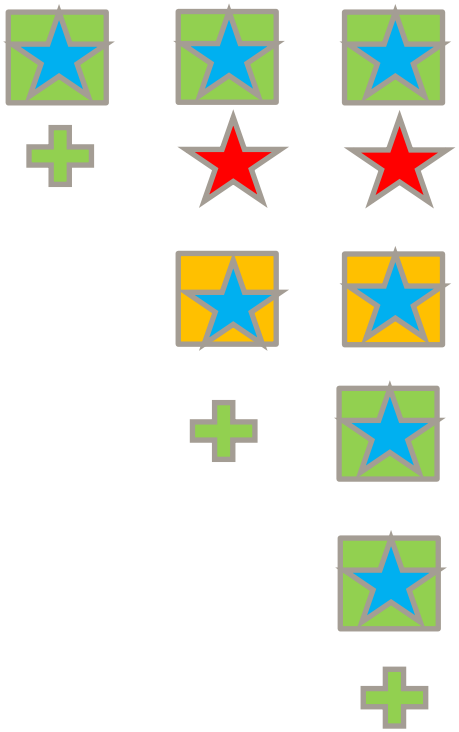
If 6 open pull requests = 36 Builds???

Another scenario

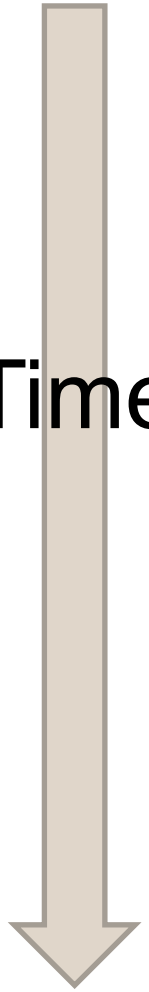
- 2 Regular committers on the same company
- 1 external contributor on different country
- 3 open pull requests (1 for each)
- 3 unrelated features that share code
- Scenario = the first committer changes the method signature of a module used by the other two.

Merging 3 open pull requests with errors

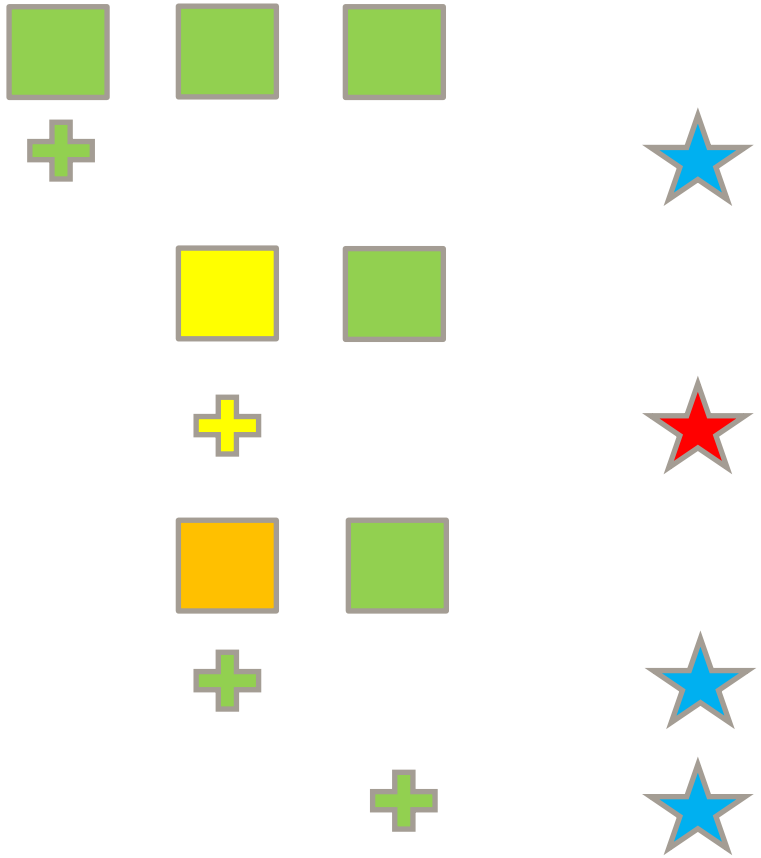
Open Source



Time



Enterprise



9 Builds, 2 fixes

4 Builds, 1 fix