Reaktor

Concurrent Development: Agile Workflows for Version Control

Lasse Koskela

50 minutes

1. common workflows
2. agile concerns
3. critical evaluation
4. food for thought
5. run for lunch

common workflows

Project X

Reaktor

I just love it when we get to use the best tools there are. Subversion kicks ass like Chuck Norris.

Developer X

Hey, what’s this? Linus has written his own version control system from scratch? Cool. Installing git-svn...

Developer Y

*It’s Sprint #35 and we’re still breaking the build almost every day, especially when Mark gets his drink on, and we end up panicking on the release day over the f***ups, wondering whether all features are ready for show time.*

Developer Z

trunk development

private branches

feature branches

Agile Workflows?

- ✓ working software as measure of progress
  - ➡ continuous integration
- ✓ attention to technical excellence
  - ➡ refactoring
- ✓ early and continuous delivery
  - ➡ one-piece flow
- ✓ team reflecting on its behavior
  - ➡ highlighting problems

Working software is the primary measure of progress.

Agile Manifesto

continuous integration

Working software as measure of progress

Trunk development is almost synonymous with continuous integration... All teams see the true progress all the time. (Except when somebody breaks the build for everyone.)

Teams live in the fantasy world of their feature branch until they integrate *and* everybody else integrates. (Continuous integration is only possible in the trunk and that's not continuous integration.)

Continuous attention to technical excellence and good design enhances agility.

Agile Manifesto

refactoring

Attention to technical excellence

Small changes are simple. Preference to wait until after hours for doing broader refactorings. (Due to fear of breaking stuff.)

Simple refactorings are trouble-free within the feature branch. Broader refactorings are scary. (Because I don't know how far others have deviated from the trunk.)

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Agile Manifesto

one-piece flow

Early and continuous delivery

We can deploy the latest and greatest while the trunk is green or quickly fix what's broken.
(Assuming we're good at this software stuff.)

We can deploy what we had a few hours ago or quickly integrate that one feature you really wanted.
(Assuming we're good at this software stuff.)

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Manifesto

highlighting problems

Reflecting on our way of working

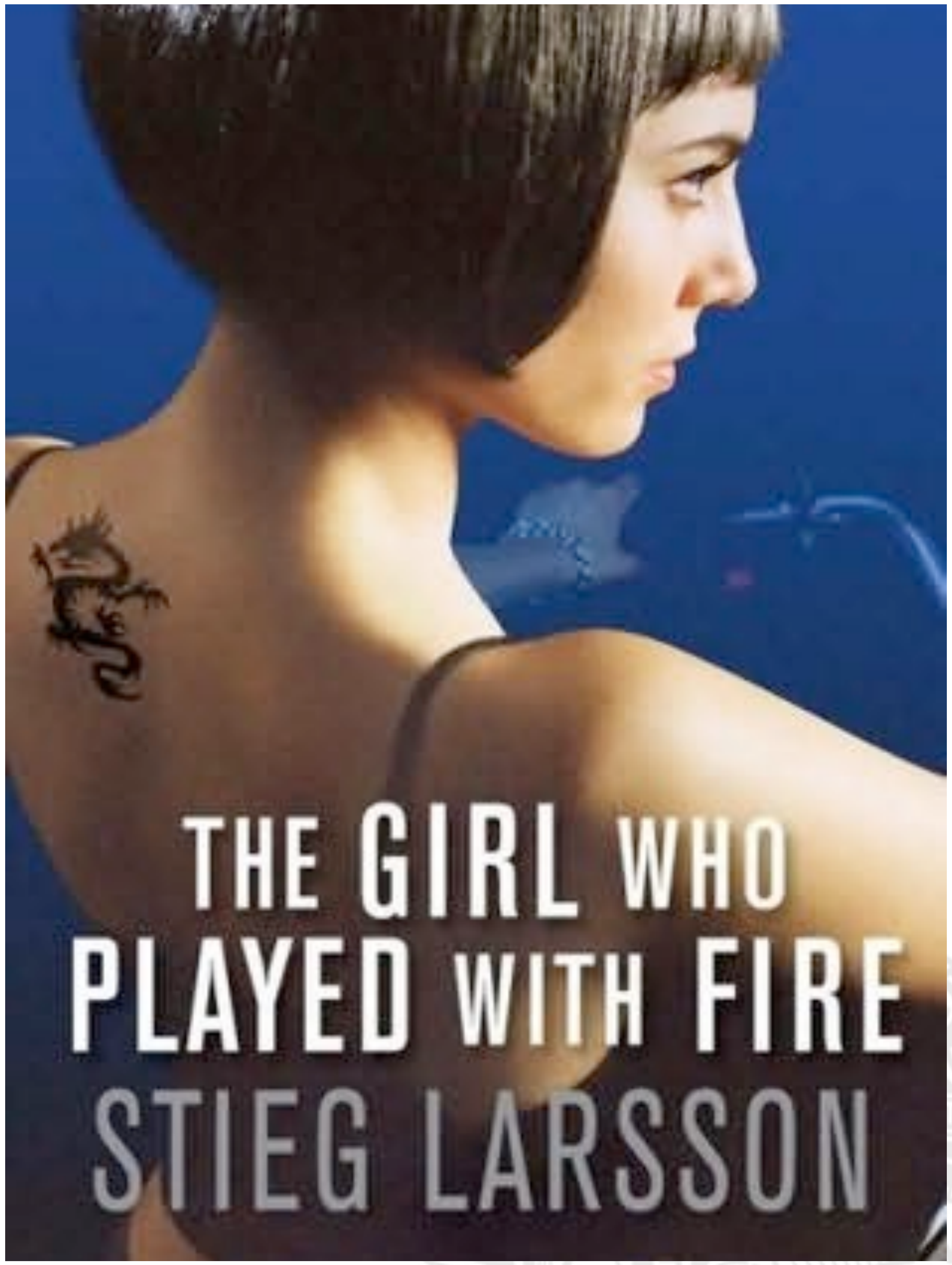
Problems are visible right away. Pressure to fix the build. Pressure to find a way to collaborate on the same branch (trunk), same files, and same feature. Focus is on finding ways to collaborate.

Option of deciding what goes into a release. Attention drawn to questions like enabling and disabling incomplete features. Focus is on managing product development.

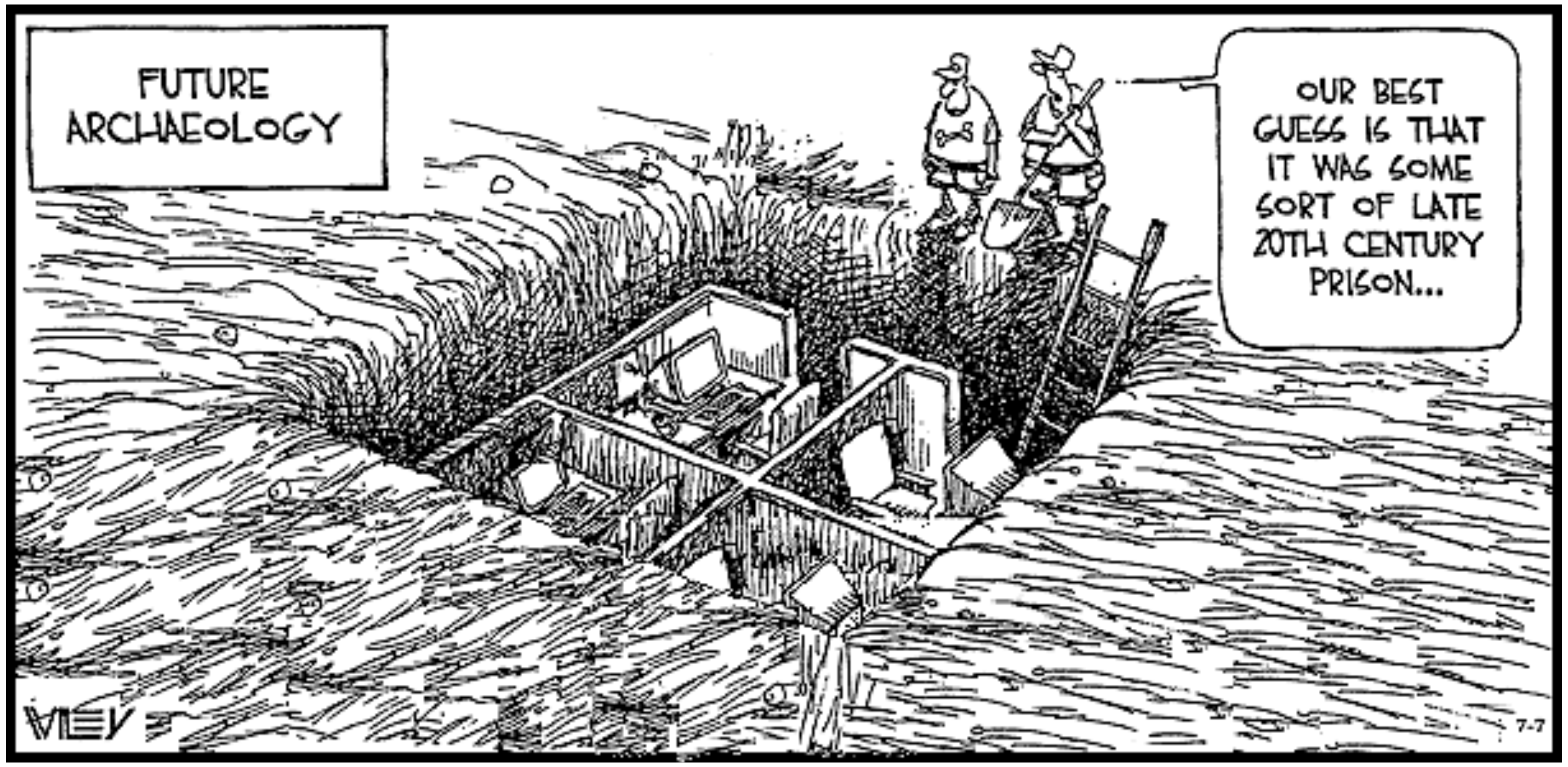
Food for thought

(stuff that might matter to you)

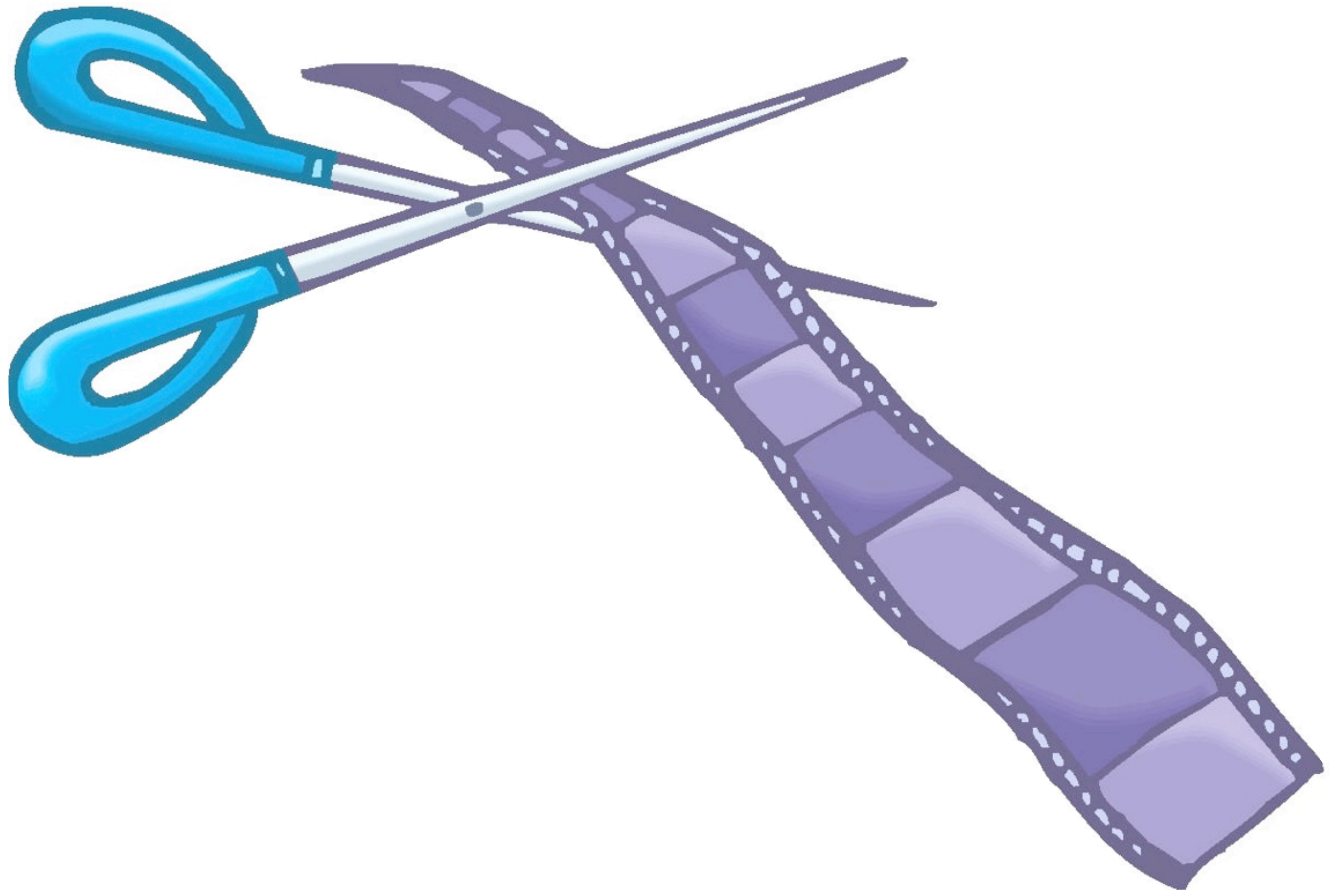
(There.)



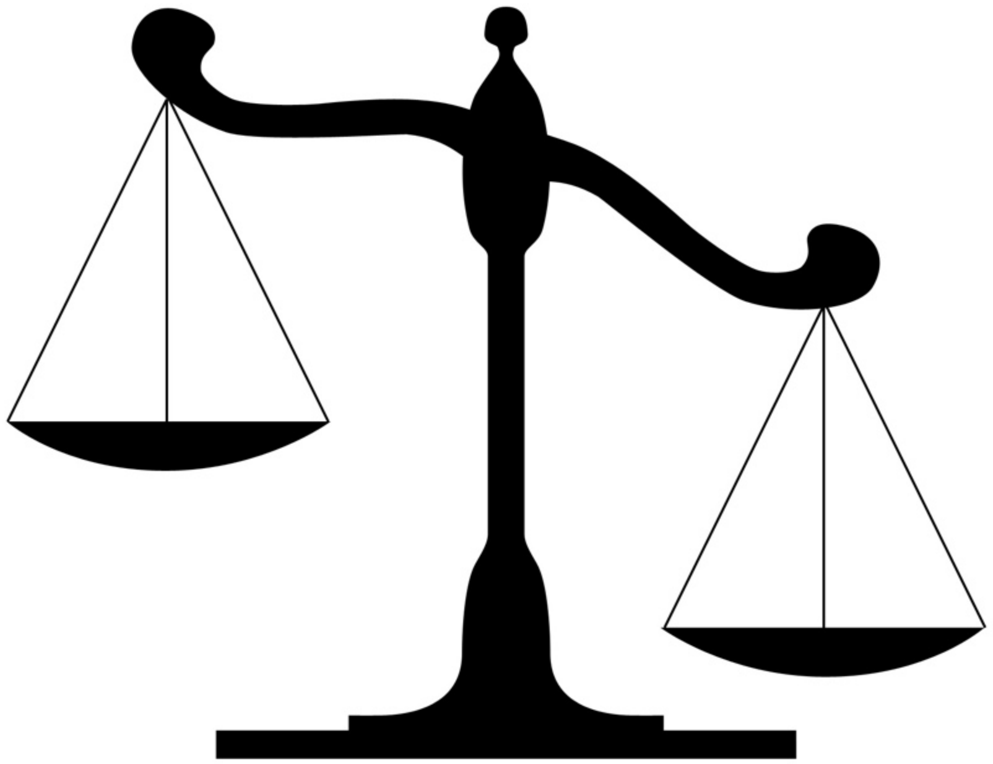
Code archaeology


FUTURE ARCHAEOLOGY
OUR BEST GUESS IS THAT IT WAS SOME SORT OF LATE 20TH CENTURY PRISON...

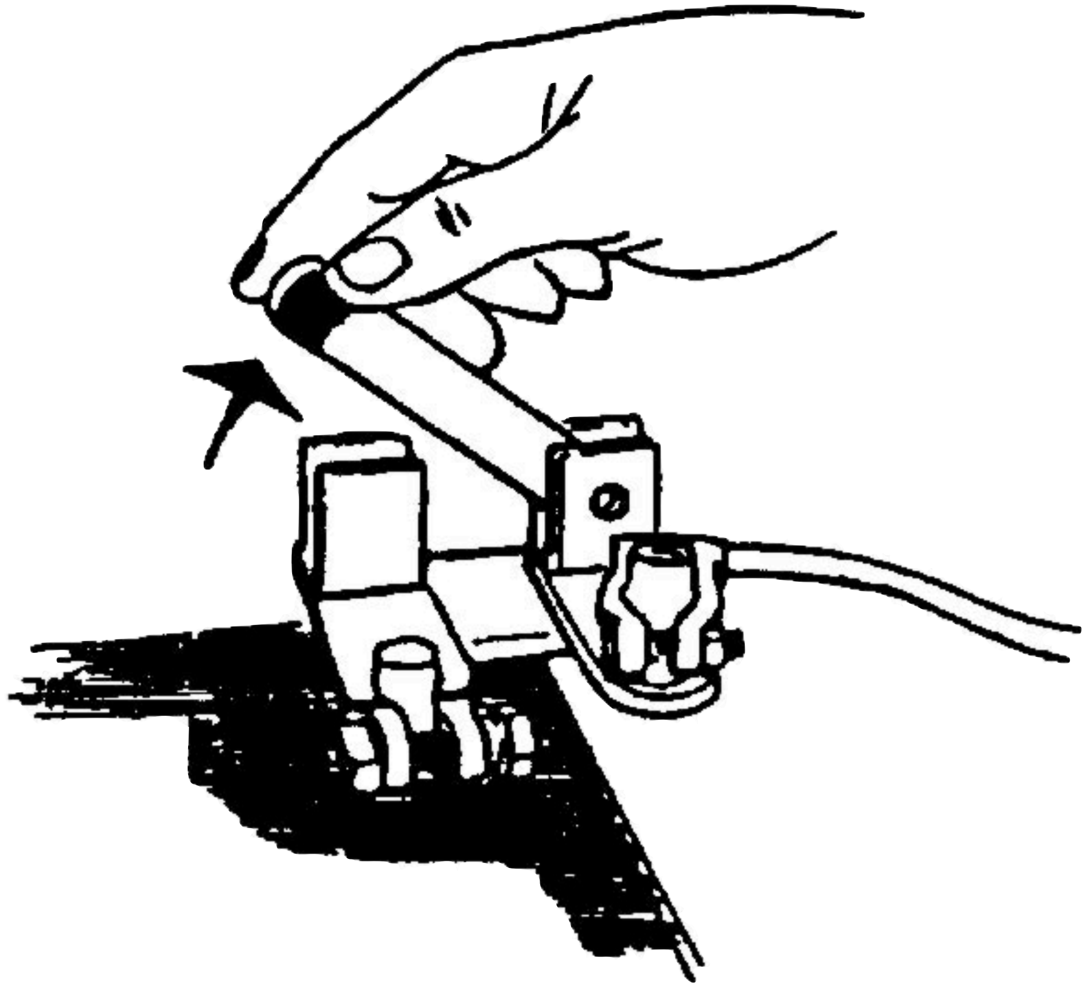
What goes into a release?



Balancing proximity and stability



Branching by Abstraction



Reaktor

questions?

thoughts?

thank you.

lasse.koskela@reaktor.fi