macadamian
software engineering

(Mostly) Painless Code Reviews

OSQA

October - 2005

Presenting:
Francis Beaudet
Chief Architect, Macadamian Technologies, Inc.
francis@macadamian.com

macadamian
software engineering

Overview

- The ROI of code reviews
- If Fagan-style review meetings are so good for you then…
- A less painful method of code review
- Single committer 101
- Streamlining peer reviews with tools
- The next step, metrics

© 1997-2005
Macadamian Technologies, Inc.

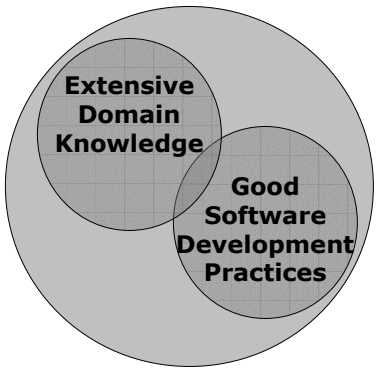
macadamian
software engineering

ROI of Code Reviews

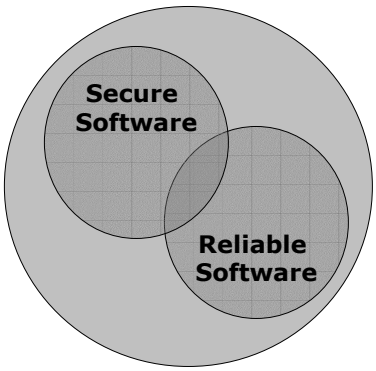
- Catch more defects earlier in the cycle
- Cost of fixing bugs later in the cycle
  - Non-incremental cost of discovering a defect at each phase (inspection > milestone > field)
- The bus number
- Security

macadamian
software engineering

Quality



Quality Product



Quality Software

macadamian
software engineering

Code Review Meetings (Eat Your Broccoli)

- How many companies have a policy of group code reviews?
- How many don't know why?
- If code review meetings are so good for us,
  - Why are so few of us doing them?
  - Why do they fall off in crunch?

macadamian
software engineering

Marathon code reviews

- Schedule constraints usually mean marathon sessions
- Marathon review sessions mean:
  - Defects are caught later
  - Some are missed due to fatigue and boredom
  - Redundant work
  - Not everyone is speaking freely, while some are speaking too freely

macadamian
software engineering

Less Painful Reviews

- Peer reviews:
  - Catch more bugs earlier in the cycle
  - Help junior developers learn from their more experienced peers
  - Help developers avoid common mistakes
  - Communication — everyone is aware of everyone else's status
  - Produce code that is easier to maintain

macadamian
software engineering

Tips — Implementing Reviews

- Teams are uncomfortable with change — this might take a while
- Have the team create the review list
  - That way, they are documenting the unofficial standards they are already using, not new standards from on high
- Code review works best when integrated into the cycle as a step that can't be skipped (more later)

macadamian
software engineering

Best Practices

- Don't always use the same reviewer
- Find a champion
  - Every team has an influential developer you need to get onside
- Proof is in the pudding
  - Team will need to see that it actually improves code quality
- Review small chunks
- Review only clean code

macadamian
software engineering

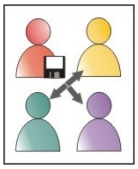
Single Committer 101

- Confession:
  - Not entirely our idea
- Adapted from open source
  - Our team was heavily involved in an open source project
  - Level of organization and high code quality was a shock
  - Adapted for use in commercial/IT dev

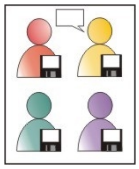
macadamian
software engineering

How It Works

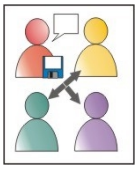
Developer writes code



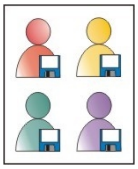
Code (diff) sent to team on mailing list



Team comments on quality, defects



Developer resolves issues and sends code back to list



Team members comment

If no issues, committer adds code to SCCS

macadamian
software engineering

Our Ingredients

- Egoless programming
- Feedback loop
- Teach someone how to fish …
- Keenness of the individuals

macadamian
software engineering

Comparison of time spent

•Preparation for meeting •Inspection meeting	•1hr x 3 reviewers •1hr x 4 attendees	•3.00 •4.00 7.00p/h
•Review of code •Reviewer-developer communication •Review of fixes	•1hr x 1 reviewer •.25hr x 2 •.25hr x 1 reviewer	•1.00 •.50 •.25 1.75p/h
•Review of code •Communication •Review of fixes	•1hr x 3 reviewers •.25hr x 2 people x 3 •.25hr x 3	•3.00 •1.50 •.75 5.25p/h

macadamian
software engineering

Benefits of Single Committer

- Scales from small teams to large distributed projects
- Everyone in the loop
  - Mailing lists/Discussion groups
  - Increase your bus number
- Everything in source tree always reviewed
  - Source code control ≠ backup system

macadamian
software engineering

Tools

- There are a variety of tools
  - You can integrate any existing automated testing tool into this process – takes out the grunt work of unit testing or enforcing coding conventions
- Tools
  - Impose some process
    - Is the process good for your organization?
  - Standardize and automate data gathering
    - Less lost data an data entry
  - Corporate standard

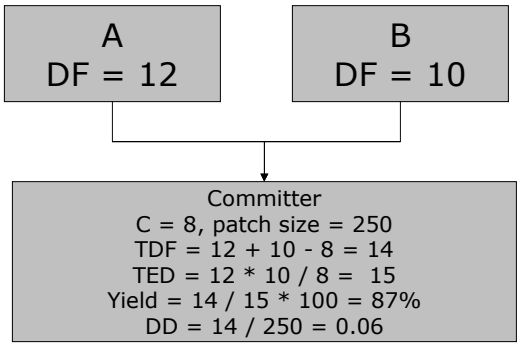
macadamian
software engineering

The next step, metrics

- **Total Defects Found = A + B - C**, where A and B are the number found by reviewer A and B respectively and C is the number found by both A and B.
- **Total Estimated Defects = AB/C**
- **Yield = Total Defects Found/Total Estimated Defects * 100%**
- **Defect Density = Total Defects Found/Size**

macadamian
software engineering

Example



- DF = Defects Found
- TDF = Total Defects Found
- TED = Total Estimated Defects
- DD = Defect Density

macadamian
software engineering

Re-review?

- The committer compiles data points from both review sheet, calls for a second review based on:
  - Yield too low
  - Majors found/Total found too low
  - Unusual defect distribution
  - High defect density
- Project dependent, but must be known

macadamian
software engineering

Resources

- Code Review Checklist:
  - http://www.macadamian.com/codereview.htm
- MSDN security checklist:
  - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnnetsec/html/THCMCh21.asp
- Single committer development primer
  - http://www.macadamian.com/column/singleCommitter.html
- Download the CodeReview Add-In for VS .NET
  - http://www.macadamian.com/products/codereview/download.html
- Smart Bear Software
  - http://www.codehistorian.com/

macadamian
software engineering

Reference for Defect Estimation

- Let the number of defects found by one reviewer be the tagged population (A).
- Assume an even likelihood of finding all defects (even distribution)
- Count the number of defects found by the second reviewer (B)
- Count the number of defects found by the second reviewer that were also found by the first (C the common defects).
- Calculate the proportion of common defects in the second reviewers defects (T=C/B).
- We assume that T is representative of the proportion of common defects in the total estimated number of defects (TED), so
  - T * TED=A, or for our purposes TED = A/T = (A*B)/C.

macadamian
software engineering

Q&A

- Questions?