Joel Test, McConnell Survival Test

CSE 4322, Fall 2013
Christoph Csallner
University of Texas at Arlington (UTA)

These slides contain textbook material by:

→ The Joel Test: 12 Steps to Better Code
http://www.joelonsoftware.com/articles/fog0000000043.html

→ Steve McConnell: “Software Project Survival Guide”
http://www.construx.com/survivalguide/

Read it, it is worthwhile:

http://www.joelonsoftware.com/articles/fog0000000043.html

THE JOEL TEST: 12 STEPS TO BETTER CODE

The Joel Test

- Joel Spolsky
  - Worked as a program manager on Microsoft's Excel team
  - Now has his own software company in New York: http://www.fogcreek.com/
- "I've come up with my own, highly irresponsible, sloppy test to rate the quality of a software team. The great part about it is that it takes about 3 minutes."
- Practical entry to software quality
  - "you *really shouldn't* use it to make sure that your nuclear power plant software is safe"

The Joel Test

- 12 yes/no questions
  - “Give your team 1 point for each "yes" answer.”
- 10 or lower is bad
  - “A score of 12 is perfect, 11 is tolerable, but 10 or lower and you've got serious problems. The truth is that most software organizations are running with a score of 2 or 3, and they need serious help, because companies like Microsoft run at 12 full-time.”

Use Joel Test to assess your project

- We will slightly adapt the test to our setup
- Self-score your process
- Document your score details in every deliverable
  - Written and presentations

Joel Test

- Do you use source control?
  - Subversion, CVS, etc.
  - Built-in redundancy
    - Makes it hard to lose important code, in case a disk dies
- Can you make a build in one step?
  - From source code from repository
  - Full build from scratch, all versions
  - Full automation → Reproducible, minimize human errors
- Do you make daily builds?
  - Notice accidental breaking changes early
  - Do not block other developers from working

Joel Test (cont'd)

- Do you have a bug database?
  - Have to keep track of bugs formally
  - Keeping them in your head will not work
  - Even for a team of one
  - Minimum data for each bug:
    - complete steps to reproduce the bug
    - expected behavior
    - observed (buggy) behavior
    - who it's assigned to
    - whether it has been fixed or not
  - Simple five-column table may be enough (Google Docs)

Joel Test (cont'd)

- Do you fix bugs before writing new code?
  - The longer you delay fixing, the more expensive it will be
  - Time for fixing a bug is harder to estimate than time to implement new code

  →The more bugs left to be fixed, the more uncertain the schedule
- Do you have an up-to-date schedule?
  - Other business tasks depend on code being ready
  - Have to communicate, update schedule
  - Forces focus on high-priority features

Joel Test (cont'd)

- Do you have a spec?
  - Do not jump directly to code
  - Changes are cheaper in documents than in code
  - Rule: "No code without spec"
- Do programmers have quiet working conditions?
  - Best work done in full concentration ("in the zone")
  - Programmers, writers, scientists, basketball players, ..
  - Takes 15 minutes to get into the zone
  - Interruptions, noise, etc. quickly kick you out of the zone

  →Find a **very** quiet work environment, try not to interrupt fellow software engineers

  →Developers at Microsoft do not have cubicles..

Joel Test (cont'd)

- Do you use the best tools money can buy?
  - Prevent programmers from waiting (being interrupted)
- Do you have testers?
  - Testers are typically cheaper than programmers
  - Microsoft/Google employ a lot of testers
  - Programmers still have to unit-test their code
- Do new candidates write code during their interview?
  - Would you hire a basketball player without knowing how he plays?
- Do you do hallway usability testing?
  - Make 5 people use your product, listen to their feedback

This course: Joel-10

- Do you use source control?
- Can you make a build in one step?
- Do you make daily builds?
  - Do you create a full build on each day you work on the project?
- Do you have a bug database?
- Do you fix bugs before writing new code?
- Do you have an up-to-date schedule?
- Do you have a spec?
- Do programmers have quiet working conditions?

Joel-10 (cont'd)

- Do you use the best tools money can buy?
  - In this project we do not have any budget
  - Still, are you using the best free tools available?
- Do you have testers?
  - May not be applicable, depending on your process
- Do new candidates write code during their interview?
  - Does not apply to our project
- Do you do hallway usability testing?
  - Ask your friends, roommates, fellow students

Steve McConnell

http://www.construx.com/Thought_Leadership/Books/Survival_Guide/Resources_by_Subject/Survival_Test/

SURVIVAL TEST QUESTIONS

Survival Test

- From Steve McConnell's book "Software Project Survival Guide"
  - Test is online at: http://www.construx.com/Thought_Leadership/Books/Survival_Guide/Resources_by_Subject/Survival_Test/
- Questions similar to the Joel Test, but more comprehensive
- Both tests check for common best practices
  - Many of these best practices covered by Capability Maturity Model (CMM)

Survival Test Mechanics

- 33 yes/no questions
  - Yes → 3 points
  - Probably → 2 points
  - Kind of but not really → 1
  - No → 0
- Early project stage: Answer based on project plan
- Later project state: Answer based on what is happening in the project
- Use the Survival Test to score your project

Requirements (1)

- Does the project have a clear, unambiguous vision statement or mission statement?
- Do all team members believe the vision is realistic?
- Does the project have a business case that details the business benefit and how the benefit will be measured?
- Does the project have a user interface prototype that realistically and vividly demonstrates the functionality that the actual system will have?

Requirements (2)

- Does the project have a detailed, written specification of what the software is supposed to do?
- Did the project team interview people who will actually use the software (end users) early in the project and continue to involve them throughout the project?

Planning (1)

- Does the project have a detailed, written Software Development Plan?
- Does the project's task list include creation of an installation program, conversion of data from previous versions of the system, integration with third-party software, meetings with the customer, and other "minor" tasks?
- Were the schedule and budget estimates officially updated at the end of the most recently completed phase?
- Does the project have detailed, written architecture and design documents?

Planning (2)

- Does the project have a detailed, written Quality Assurance Plan that requires design and code reviews in addition to system testing?
- Does the project have a detailed Staged Delivery Plan for the software, which describes the stages in which the software will be implemented and delivered?
- Does the project's plan include time for holidays, vacation days, sick days, and ongoing training, and are resources allocated at less than 100 percent?

Planning (3)

- Was the project plan, including the schedule, approved by the development team, the quality assurance team, and the technical writing team—in other words, the people responsible for doing the work?

Project Control (1)

- Has a single key executive who has decision-making authority been made responsible for the project, and does the project have that person’s active support?
- Does the project manager’s workload allow him or her to devote an adequate amount of time to the project?
- Does the project have well-defined, detailed milestones ("binary milestones") that are considered to be either 100 percent done or 100 percent not done?

Project Control (2)

- Can a project stakeholder easily find out which of these binary milestones have been completed?
- Does the project have a feedback channel by which project members can anonymously report problems to their own managers and upper managers?
- Does the project have a written plan for controlling changes to the software's specification?
- Does the project have a Change Control Board that has final authority to accept or reject proposed changes?

Project Control (3)

- Are planning materials and status information for the project—including effort and schedule estimates, task assignments, and progress compared to the plan thus far—available to every team member?
- Is all source code placed under automated revision control?
- Does the project environment include the basic tools needed to complete the project, including defect tracking software, source code control, and project management software?

Risk Management

- Does the project plan articulate a list of current risks to the project? Has the list been updated recently?
- Does the project have a project risk officer who is responsible for identifying emerging risks to the project?
- If the project uses subcontractors, does it have a plan for managing each subcontract organization and a single person in charge of each one? (Give the project full score if it doesn't use subcontractors.)

Personnel

- Does the project team have all the technical expertise needed to complete the project?
- Does the project team have expertise with the business environment in which the software will operate?
- Does the project have a technical leader capable of leading the project successfully?
- Are there enough people to do all the work required?
- Does everyone work well together?
- Is each person committed to the project?

Total

- Preliminary score. Add up the points next to each answer.
- Size multiplier. Write in 1.5 if the project team has 3 or fewer full-time– equivalent people including developers, quality assurance personnel, and first-level management. Write in 1.25 if it has 4 to 6 full-time–equivalent people. Otherwise, write in 1.0.
- Final score. Multiply the preliminary score by the size multiplier.

Scoring Guidelines

- 90+ Outstanding
- 80–89 Excellent: High probability to deliver software close to schedule, budget, and quality targets
- 60–79 Good: Good chance to meet either schedule or budget target, but probably not both
- 40–59 Fair: Typical. High stress and shaky team dynamics. Less functionality than desired at greater cost and with longer schedule
- < 40 At Risk: Will project finish?