

# Index

## A

### Acceptance tests

- automated, [97–99](#)
- communication and, [97](#)
- continuous integration and, [104–105](#)
- definition of, [94](#)
- developer's role in, [100–101](#)
- extra work and, [99](#)
- GUIs and, [103–105](#)
- negotiation and, [101–102](#)
- passive aggression and, [101–102](#)
- timing of, [99–100](#)
- unit tests and, [102–103](#)
- writers of, [99–100](#)

### Adversarial roles, [20–23](#)

### Affinity estimation, [140–141](#)

### Ambiguity, in requirements, [92–94](#)

### Apologies, [6](#)

### Apprentices, [183](#)

### Apprenticeship, [180–184](#)

### Arguments, in meetings, 120–121

### Arrogance, [16](#)

### Automated acceptance testing, [97–99](#)

### Automated quality assurance, [8](#)

### Avoidance, [125](#)

## B

### Blind alleys, [125–126](#)

### Bossavit, Laurent, [83](#)

*Bowling Game*, [83](#)  
Branching, [191](#)  
Bug counts, [197](#)  
Business goals, [154](#)

## C

Caffeine, [122](#)  
Certainty, [74](#)  
Code  
    control, [189–194](#)  
    owned, [157](#)  
    3 AM, [53–54](#)  
    worry, [54–55](#)  
*Coding Dojo*, [83–87](#)  
Collaboration, [14](#), [151–160](#)  
Collective ownership, [157–158](#)  
Commitment(s), [41–46](#)  
    control and, 44  
    discipline and, [47–50](#)  
    estimation and, [132](#)  
    expectations and, [45](#)  
    identifying, [43–44](#)  
    implied, 134–135  
    importance of, [132](#)  
    lack of, [42–43](#)  
    pressure and, [146](#)  
Communication  
    acceptance tests and, [97](#)  
    pressure and, [148](#)  
    of requirements, [89–94](#)  
Component tests  
    in testing strategy, [110–111](#)  
    tools for, [199–200](#)

Conflict, in meetings, 120–121  
Continuous build, [197–198](#)  
Continuous integration, [104–105](#)  
Continuous learning, [13](#)  
Control, commitment and, 44  
Courage, [75–76](#)  
Craftsmanship, [184](#)  
Creative input, [59–60](#), [123](#)  
Crisis discipline, [147](#)  
Cucumber, [200](#)  
Customer, identification with, [15](#)  
CVS, [191](#)  
Cycle time, in test-driven development, [72](#)

## D

Deadlines  
    false delivery and, [67](#)  
    hoping and, [65](#)  
    overtime and, [66](#)  
    rushing and, [65–66](#)  
Debugging, [60–63](#)  
Defect injection rate, [75](#)  
Demo meetings, 120  
Design, test-driven development and, [76–77](#)  
Design patterns, [12](#)  
Design principles, [12](#)  
Details, [201–203](#)  
Development. *see* test driven development (TDD)  
Disagreements, in meetings, 120–121  
Discipline  
    commitment and, [47–50](#)  
    crisis, [147](#)  
Disengagement, [64](#)

Documentation, [76](#)  
Domain, knowledge of, [15](#)  
“Done,” defining, [67](#), [94–97](#)  
“Do no harm” approach, [5–10](#)  
    to function, [5–8](#)  
    to structure, [8–10](#)  
Driving, [64](#)

## E

Eclipse, [195–196](#)  
Emacs, [195](#)  
Employer(s)  
    identification with, [15](#)  
    programmers vs., [153–156](#)  
Estimation  
    affinity, [140–141](#)  
    anxiety, [92](#)  
    commitment and, [132](#)  
    definition of, [132–133](#)  
    law of large numbers and, [141](#)  
    nominal, [136](#)  
    optimistic, [135–136](#)  
    PERT and, [135–138](#)  
    pessimistic, [136](#)  
    probability and, [133](#)  
    of tasks, [138–141](#)  
    trivariate, [141](#)  
Expectations, commitment and, [45](#)  
Experience, broadening, [87](#)

## F

Failure, degrees of, [174](#)  
False delivery, [67](#)

FitNesse, [199–200](#)  
Flexibility, [9](#)  
Flow zone, [56–58](#)  
Flying fingers, [139](#)  
Focus, [121–123](#)  
Function, in “do no harm” approach, [5–8](#)

## G

Gaillot, Emmanuel, [83](#)  
Gelled team, [162–164](#)  
Git, [191–194](#)  
Goals, [20–23](#), [118](#)  
Graphical user interfaces (GUIs), [103–105](#)  
Green Pepper, [200](#)  
Grenning, James, [139](#)  
GUIs, [103–105](#)

## H

Hard knocks, [179–180](#)  
Help, [67–70](#)  
    giving, [68](#)  
    mentoring and, [69–70](#)  
    pressure and, [148–149](#)  
    receiving, [68–69](#)  
“Hope,” [42](#)  
Hoping, deadlines and, [65](#)  
Humility, [16](#)

## I

IDE/editor, [194](#)  
Identification, with employer/customer, [15](#)  
Implied commitments, 134–135  
Input, creative, [59–60](#), [123](#)

- Integration, continuous, [104–105](#)
- Integration tests
  - in testing strategy, [111–112](#)
  - tools for, [200–201](#)
- IntelliJ, [195–196](#)
- Interns, [183](#)
- Interruptions, [57–58](#)
- Issue tracking, [196–197](#)
- Iteration planning meetings, [119](#)
- Iteration retrospective meetings, 120

## J

- JBehave, [200](#)
- Journeymen, [182–183](#)

## K

- Kata, [84–85](#)
- Knowledge
  - of domain, [15](#)
  - minimal, [12](#)
  - work ethic and, [11–13](#)

## L

- Lateness, [65–67](#)
- Law of large numbers, [141](#)
- Learning, work ethic and, [13](#)
- “Let’s,” [42](#)
- Lindstrom, Lowell, [140](#)
- Locking, [190](#)

## M

- Manual exploratory tests, in testing strategy, 112–113
- Masters, [182](#)

MDA, [201–203](#)

## Meetings

- agenda in, [118](#)

- arguments and disagreements in, 120–121

- declining, [117](#)

- demo, 120

- goals in, [118](#)

- iteration planning, [119](#)

- iteration retrospective, 120

- leaving, [118](#)

- stand-up, [119](#)

- time management and, [116–121](#)

Mentoring, [14–15](#), [69–70](#), [174–180](#)

Merciless refactoring, [9](#)

Messes, [126–127](#), [146](#)

Methods, [12](#)

Model Driven Architecture (MDA), [201–203](#)

Muscle focus, [123](#)

Music, [57](#)

## N

- “Need,” [42](#)

- Negotiation, acceptance tests and, [101–102](#)

- Nominal estimate, [136](#)

- Nonprofessional, [2](#)

## O

- Open source, [87](#)

- Optimistic estimate, [135–136](#)

- Optimistic locking, [190](#)

- Outcomes, best-possible, [20–23](#)

- Overtime, [66](#)

- Owned code, [157](#)

Ownership, collective, [157–158](#)

## P

Pacing, [63–64](#)

Pairing, [58](#), [148–149](#), [158](#)

Panic, [147–148](#)

Passion, [154](#)

Passive aggression, [28–30](#), [101–102](#)

People, programmers vs., [153–158](#)

Personal issues, [54–55](#)

PERT (Program Evaluation and Review Technique), [135–138](#)

Pessimistic estimate, [136](#)

Pessimistic locking, [190](#)

Physical activity, [123](#)

Planning Poker, [139–140](#)

Practice

- background on, [80–83](#)

- ethics, [87](#)

- experience and, [87](#)

- turnaround time and, [82–83](#)

- work ethic and, [13–14](#)

Precision, premature, in requirements, [91–92](#)

Preparedness, [52–55](#)

Pressure

- avoiding, [145–147](#)

- cleanliness and, [146](#)

- commitments and, [146](#)

- communication and, [148](#)

- handling, [147–149](#)

- help and, [148–149](#)

- messes and, [146](#)

- panic and, [147–148](#)

Priority inversion, [125](#)



Probability, [133](#)

Professionalism, [2](#)

Programmers

employers vs., [153–156](#)

people vs., [153–158](#)

programmers vs., [157](#)

Proposal, project, [31–32](#)

## Q

Quality assurance (QA)

automated, [8](#)

as bug catchers, [6](#)

as characterizers, [108–109](#)

ideal of, as finding no problems, [108–109](#)

problems found by, [6–7](#)

as specifiers, [108](#)

as team member, [108](#)

## R

Randori, [86–87](#)

Reading, as creative input, [59](#)

Recharging, [122–123](#)

Reputation, [5](#)

Requirements

communication of, [89–94](#)

estimation anxiety and, [92](#)

late ambiguity in, [92–94](#)

premature precision in, [91–92](#)

uncertainty and, [91–92](#)

Responsibility, [2–5](#)

apologies and, [6](#)

“do no harm” approach and, [5–10](#)

function and, [5–8](#)

- structure and, [8–10](#)
- work ethic and, [10–16](#)
- RobotFX, [200](#)
- Roles, adversarial, [20–23](#)
- Rushing, [34–35](#), [65–66](#)

## S

- Santana, Carlos, [83](#)
- “Should,” [42](#)
- Shower, [64](#)
- Simplicity, [34](#)
- Sleep, [122](#)
- Source code control, [189–194](#)
- Stakes, [23–24](#)
- Stand-up meetings, [119](#)
- Structure
  - in “do no harm” approach, [8–10](#)
  - flexibility and, [9](#)
  - importance of, [8](#)
- SVN, [191–194](#)
- System tests, in testing strategy, 112

## T

- Task estimation, [138–141](#)
- Teams and teamwork, [24–30](#)
  - gelled, [162–164](#)
  - management of, [164](#)
  - passive aggression and, [28–30](#)
  - preserving, [163](#)
  - project-initiated, [163–164](#)
  - project owner dilemma with, [164–165](#)
  - trying and, [26–28](#)
  - velocity of, [164](#)

## Test driven development (TDD)

- benefits of, [74–77](#)
- certainty and, [74](#)
- courage and, [75–76](#)
- cycle time in, [72](#)
- debut of, [71–72](#)
- defect injection rate and, [75](#)
- definition of, [7–8](#)
- design and, [76–77](#)
- documentation and, [76](#)
- interruptions and, [58](#)
- three laws of, [73–74](#)
- what it is not, [77–78](#)

## Testing

### acceptance

- automated, [97–99](#)
- communication and, [97](#)
- continuous integration and, [104–105](#)
- definition of, [94](#)
- developer's role in, [100–101](#)
- extra work and, [99](#)
- GUIs and, [103–105](#)
- negotiation and, [101–102](#)
- passive aggression and, [101–102](#)
- timing of, [99–100](#)
- unit tests and, [102–103](#)
- writers of, [99–100](#)

automation pyramid, [109–113](#)

### component

- in testing strategy, [110–111](#)
- tools for, [199–200](#)

importance of, [7–8](#)

### integration

- in testing strategy, [111–112](#)
  - tools for, [200–201](#)
- manual exploratory, 112–113
- structure and, [9](#)
- system, 112
- unit
  - acceptance tests and, [102–103](#)
  - in testing strategy, [110](#)
  - tools for, [198–199](#)
- TextMate, [196](#)
- Thomas, Dave, [84](#)
- 3 AM code, [53–54](#)
- Time, debugging, [63](#)
- Time management
  - avoidance and, [125](#)
  - blind alleys and, [125–126](#)
  - examples of, [116](#)
  - focus and, [121–123](#)
  - meetings and, [116–121](#)
  - messes and, [126–127](#)
  - priority inversion and, [125](#)
  - recharging and, [122–123](#)
  - “tomatoes” technique for, [124](#)
- Tiredness, [53–54](#)
- “Tomatoes” time management technique, [124](#)
- Tools, [189](#)
- Trivariate estimates, [141](#)
- Turnaround time, practice and, [82–83](#)

## U

- UML, [201](#)
- Uncertainty, requirements and, [91–92](#)
- Unconventional mentoring, [179](#). *see also* mentoring

## Unit tests

- acceptance tests and, [102–103](#)
- in testing strategy, [110](#)
- tools for, [198–199](#)

## V

- Vi, [194](#)

## W

- Walking away, [64](#)
- Wasa, [85–86](#)
- Wideband delphi, [138–141](#)
- “Wish,” [42](#)
- Work ethic, [10–16](#)
  - collaboration and, [14](#)
  - continuous learning and, [13](#)
  - knowledge and, [11–13](#)
  - mentoring and, [14–15](#)
  - practice and, [13–14](#)
- Worry code, [54–55](#)
- Writer’s block, [58–60](#)

## Y

- “Yes”
  - cost of, [30–34](#)
  - learning how to say, [46–50](#)