Agile in an Hour

Joe Bergin & Fred Grossman
Seidenberg School
Computer Science and Information Systems
Pace University

PACE UNIVERSITY
IVAN G. SEIDENBERG
SCHOOL OF COMPUTER SCIENCE
AND INFORMATION SYSTEMS

Introduction

- Agile Software Development is a high discipline and very iterative development method
- It avoids early commitment and early infrastructure development to achieve:
  - Low cost of change and
  - Easy retargeting of a project

Why Projects Fail

- Trying to over-control the dimensions
  - Features
  - Cost
  - Schedule
  - Quality
- In reality you can only control 3 of these

Failure

- Cubicles
- Lying
- Coercion
  (“... by Friday”, or “You can’t have that”)
- Late testing, too little testing
- Working from inadequate documentation
- High cost of change, inflexible design
- Parts OK, but don’t fit together

Agile Manifesto

- We Value:
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

Agile Sweet Spot

- Reasonable Size
- Uncertainty on Features…
- Change Likely
- Standard process likely to fail

- If you *can't* plan, then build on a tight feedback loop

Many Ways

- Scrum -- overall management process
- XP -- day to day practices
- Crystal -- scaling, flexibility
- …
- **Being** agile, not just **doing** agile.

Agile Synonyms

XP	Scrum
Iteration	Sprint
Customer	Product Owner
Coach	Scrum Master
Big Boss (sheltering mgr)	Scrum Master
Tracker	Scrum Master
Project Stories	Product Backlog
Iteration Stories	Sprint Backlog
Stand up meeting	Daily Scrum Meeting
Planning Game	Sprint Planning Meeting

Values in XP

- Courage
- Communication
- Simplicity
- Feedback
- Respect

Key Ideas

- Practices are synergistic & support each other
- Communication Distance is expensive
- Schedules never slip (time-box)
- Balance between rights & responsibilities
- Set of practices is humane

What It Gives You

- Rights AND Responsibilities
- Humane work environment
- Skills that are valuable
- Pride of workmanship

What It Requires

- Discipline
- Commitment
- Honesty
- Courage

What is missing?

- Upfront requirements gathering and sign-off -- hence no need to commit early
- Upfront design documents -- hence easy to retarget
- Early costs amortized over life of project -- hence lower cost of change
- Intimidation: schedule, cost, or value

Agile Roles

- Customer, Product Owner, Stakeholder
- Developer
- Tester -- all developers do this
- Coach -- responsible for process and guidance
  - ScrumMaster (super coach plus downfield blocker)
- Others (tracker, documentation, …

Roles: Customer/ Product Owner

- Write short “story cards” describing features
- Answer questions throughout to add specificity to the stories (just in time requirements)
- Write/specify acceptance tests to verify stories
- Make *all* business decisions: function, priority, feature value, acceptance
- Obtains consensus/consent among stakeholders to guide development

A Good Customer

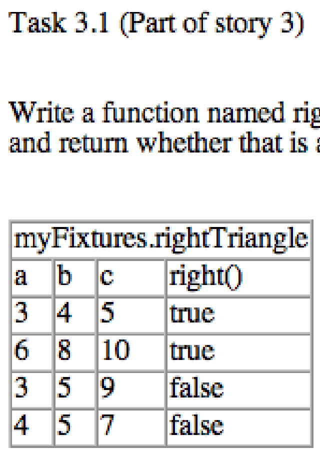
- Understands the domain well by working in that domain and also by understanding how it works (not always the same thing)
- Can understand, with development’s help, how software can provide business value in the domain
- Is determined to deliver value regularly and is not afraid to deliver too little rather than nothing
- Can make decisions about what’s needed now and what’s needed later
- Is willing to accept ultimate responsibility for the success or failure of the project. [Beck & Fowler]

Sample Story

Triangles Story 3 2

- The system will correctly classify triangles: right triangles, equilateral, etc.

Sample Acceptance Test



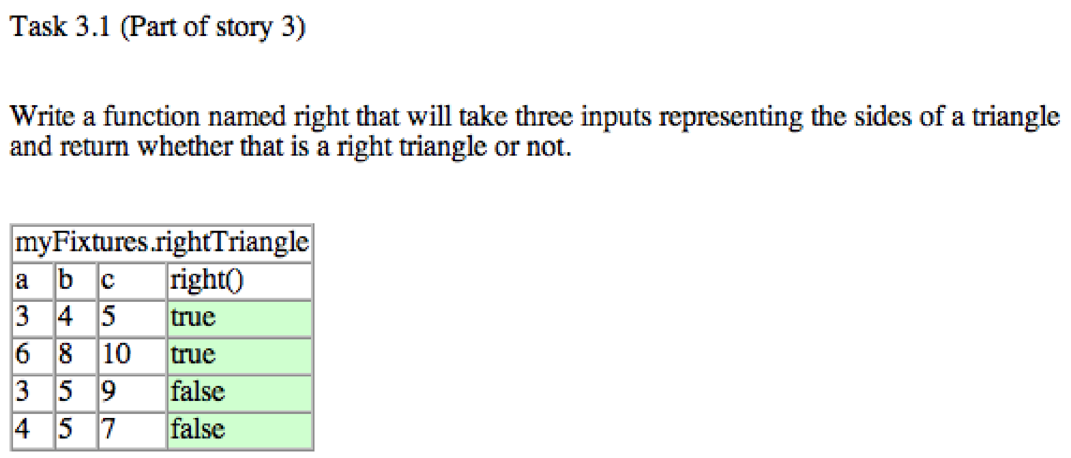
Task 3.1 (Part of story 3)

Write a function named right that will take three inputs representing the sides of a triangle and return whether that is a right triangle or not.

myFixtures.rightTriangle			
a	b	c	right()
3	4	5	true
6	8	10	true
3	5	9	false
4	5	7	false

These are created in Excel or HTML, but are executable

After Execution



Task 3.1 (Part of story 3)

Write a function named right that will take three inputs representing the sides of a triangle and return whether that is a right triangle or not.

myFixtures.rightTriangle			
a	b	c	right()
3	4	5	true
6	8	10	true
3	5	9	false
4	5	7	false

Failed tests show up in red.

Roles: Developer

- Estimate stories
- Break stories into tasks
- Build tasks -- with customer feedback
- Write unit tests (all tests always succeed)
- Do continuous integration

Roles: Other

- Tracker (keep everyone aware of progress)
- Coach (conscience of the team)
- Big-Boss (management and shelter)
- Tester (write/run unit tests...)
- Consultant (extra knowledge as needed)

Concept: Pigs v Chickens

Contract

- For best effort and full communication, NOT for deliverables on a given date
- Customer/Product Owner may terminate project at any time
- Short release cycles (4-6 weeks) ensure constant delivery of customer value
- Schedule never slips, though features may be dropped from an iteration (1-2 weeks)

Controlling Cost

- Build the high value features first -- controlled by customer
- Make expensive decisions as late as possible
- When the cost and value curves cross - quit!

Staying Happy

- Customer steers like a bicycle
- If something is not “right” then write a new story and prioritize it like any other (no guilt, no blame)
- Developers build only the stories in the current iteration and always do the simplest thing that could possibly work
- Stories are fine-grained to enable short iterations

Practices

- XP has a dozen or so key (daily) practices. The most important overall are
  - Onsite Customer
  - Whole Team

Practices-- Customer

- The most important practices for the customer are:
  - Onsite customer - available customer
  - Planning Game
  - Customer Written Acceptance Tests

Onsite Customer

- Customer is needed on site because
  - Developers should not make business decisions but
    - no upfront requirements
    - no upfront design documents
- A story is a contract to talk in the future
  - actual requirement is captured just-in-time

Whole Team

- In addition to the customer, the “whole team” includes all personnel with key skills needed to develop the system
  - Software developers
  - Designers - architects - analysts
  - Information architects
  - Others as appropriate - testers, documentation specialists …
- BUT it favors generalists over specialists
- AND it is SELF-ORGANIZING

Key Ideas

- Everyone has responsibility for the project
  - Not just for their little piece
- Just in Time - Just Enough
  - Lack of anticipation and scaffolding
- Strict time-boxing of iterations

Flow

- The customers write stories and prioritize them
- The other members task out the stories and estimate them
- Members with appropriate skills estimate and perform tasks
- Tasks support the stories

Planning Game 1

- This is a periodic task (every 1-2 weeks) in which the customer chooses the high value features (stories) for the next release or iteration
- Based on cost estimates from the developers
- Estimates are not a contract, so re-steering is required throughout the iteration.

Planning Game 2

- Customer writes stories
- Developers estimate stories
- Customer prioritizes stories
- Developers give the “velocity”
- Customer chooses stories up to velocity

Planning Game 3

- Developers/Customer discuss stories
- Developers divide stories up into tasks
- Individual developer with appropriate skills chooses a task and estimates it
- If sum of task times > velocity then back to planning, otherwise build, test, & integrate

Build Phase

- Tracker keeps track of everyone’s progress
- If all tasks/stories can’t be completed on time some are dropped. Customer chooses which
- At end of each task, all tests pass. Customer verifies - accepts or rejects
- If the customer still isn’t happy, write a new story - no time wasted on assigning blame

Build Phase (cont.)

- If developers finish early, go back to customer for more work. Customer chooses
- Developers give a new "mini velocity"
- Next iteration velocity is adjusted based on what we ***complete*** this iteration

Concept: Done = built, thoroughly tested, integrated, documented, accepted

Practices--Developer

- Standup Meeting
- Sustainable Pace - energized work
- Test Driven Development
    - No code without a failing test
- Small Releases - 2 or 3 iterations
- Collective Code Ownership
- Coding Standard

Practices--Developer

- Pair Programming
- Constant Refactoring
- Continuous Integration
- Simple Design
- Metaphor
- Retrospectives

New Practices

- The above practices may not all be appropriate as stated for an integrated team
- Practices are built on principles to give benefits
- Need to discover and implement appropriate practices for THIS team on THIS project to achieve desired goals (so, hold Retrospectives)

Distributed Agile

- Minimize Communication Distance
  - Provide situational awareness
- Acceptance Tests
- Overlapping sub-teams
- Everyone takes responsibility
  - No one succeeds unless the team does
  - Everyone succeeds if the team does
    - “not my job” is not an option

Areas of Change (?)

- Collective Ownership (vs skills)
- Development Standard (each area)
- Test Driven Development (automated)
- Pair (mob?) Practices