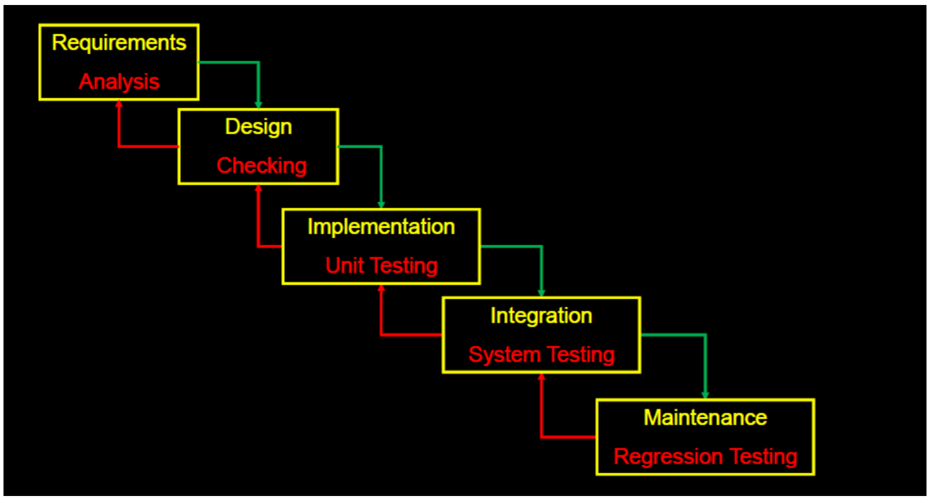
Lecture3

Waterfall模型

- requirements–what software should do
- Design – structure code into modules; architecture
- implementation – hack code
- 下面两个经常被合并称为verification
  - Integration – put modules together
  - Testing – check if code works
- Maintenance – keep making changes



Extreme programming (xp)

- 发明者KENT BECK
- 不同于rigid waterfall process
  - replace it with a collaborative and iterative design process
- Main ideas
  - 不要写太多文档
    - working code and tests are the main written product
  - 逐一实现功能(features)
  - release code frequently
  - work closely with the customer
  - 与团队成员大量沟通

some key practices

- **Planning game** for requirements
- **Test-driven development** for design and testing
- **Refactoring** for design
- **Pair programming** for development
- **continuous integration** for integration

xp is an iterative process

- iteration -> (1-3 weeks/cycle)
- 每次迭代的开始都计划一场迭代会议
- iteration is going to implement set of user stories
- 将工作分成一天天的小任务
- each day, work in pairs.

pair programming (since 1950s)

Pair programming is a simple, straightforward concept. Two programmers work **side-by-side** at one computer, continuously collaborating on the **same** design, algorithm, code, and test. It allows two people to produce a **higher quality** of code than that produced by the summation of their solitary efforts.

- **driver**: types or writes （码代码的）
- **navigator**: observer (looking for tactical & strategic defects) （在旁边看的）
- periodically (30分钟) **switch** roles of driver and navigator
- pair coding, design, debugging, testing

Empirical study

- higher quality code (15% fewer defects)
- complete half the time (58% of elapsed time)
- happier programmers
  - enjoy work more (92%)
  - confident in the product (96%)

预期收益

- 高产品质量
- improved cycle time
- 提升编程人员的满足感
- 增强学习
- pair rotation
    - ease staff training and transition
    - knowledge management/reduced product risk
    - enhanced team building

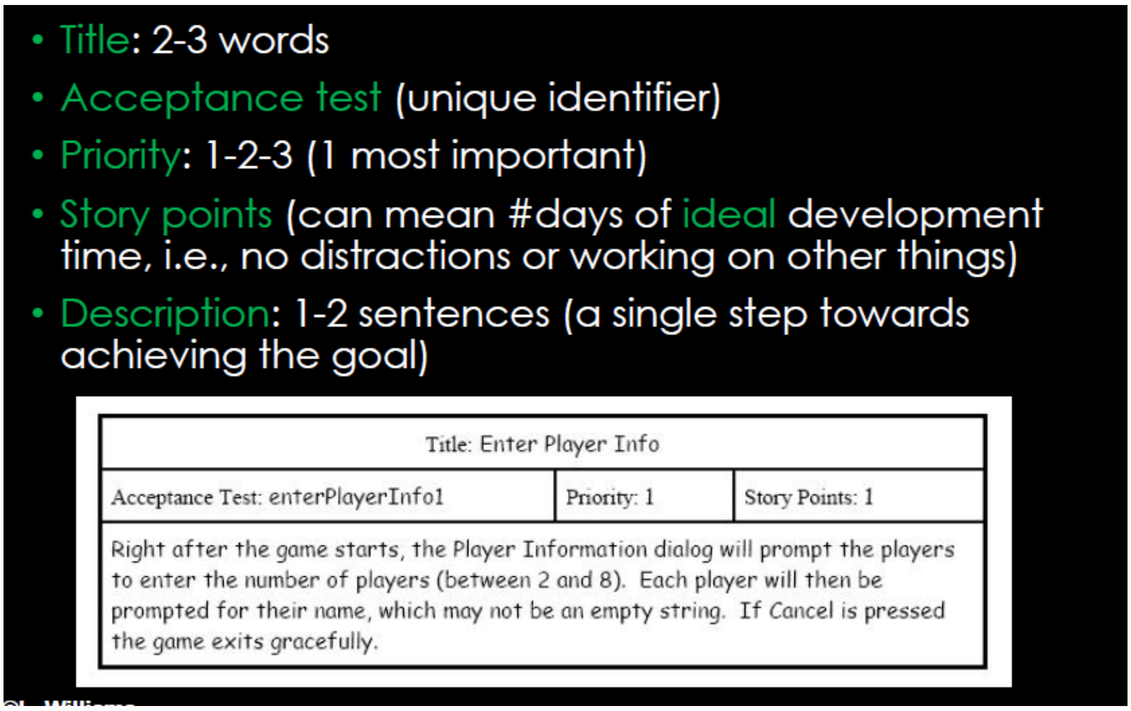
如何起作用的

- pair pressure
- pair negotiation
- pair courage
- pair reviews
- pair debugging
- pair learning

User stories

- represents a feature customers want in the software
- the smallest amount of information (a step) necessary to allow the customer to define (and steer) a path through the system
- written by **customers**, not developers
- written on index cards
    - no more than one step on each card

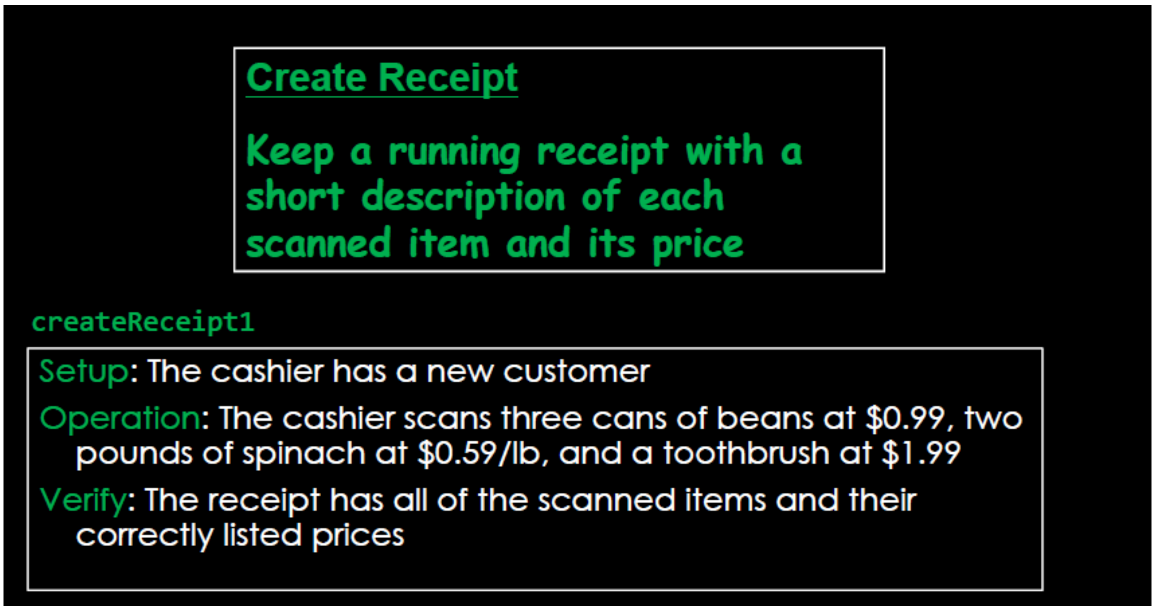
format



- Title: 2-3 words
- Acceptance test (unique identifier)
- Priority: 1-2-3 (1 most important)
- Story points (can mean #days of ideal development time, i.e., no distractions or working on other things)
- Description: 1-2 sentences (a single step towards achieving the goal)

Title: Enter Player Info		
Acceptance Test: enterPlayerInfo1	Priority: 1	Story Points: 1
Right after the game starts, the Player Information dialog will prompt the players to enter the number of players (between 2 and 8). Each player will then be prompted for their name, which may not be an empty string. If Cancel is pressed the game exits gracefully.		

acceptance test example



Create Receipt

Keep a running receipt with a short description of each scanned item and its price

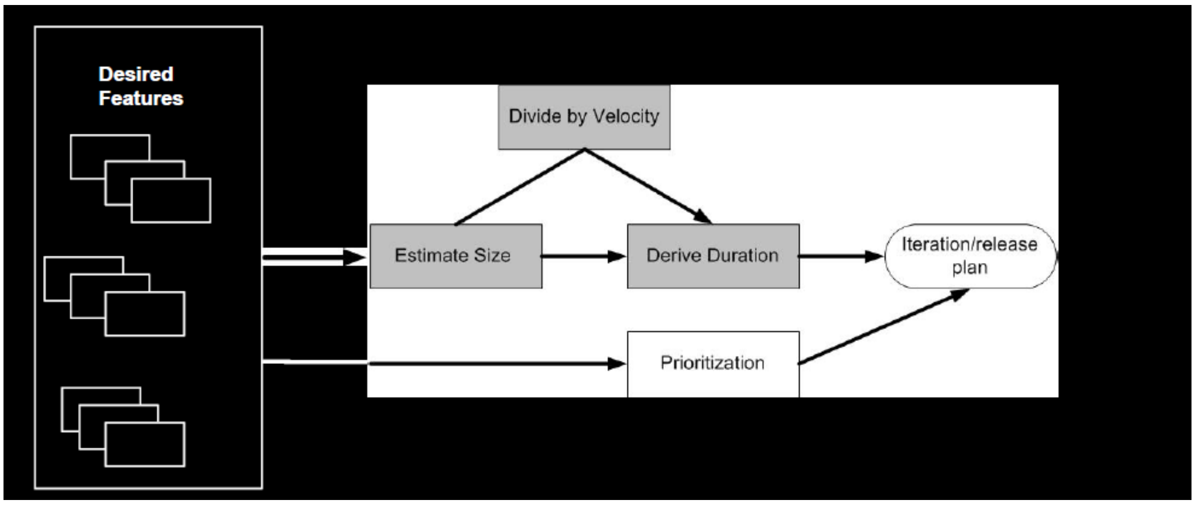
createReceipt1

Setup: The cashier has a new customer

Operation: The cashier scans three cans of beans at $0.99, two pounds of spinach at $0.59/lb, and a toothbrush at $1.99

Verify: The receipt has all of the scanned items and their correctly listed prices

plan



概念

- story point: unit of measure for expressing the overall size of a user story, feature, or other piece of work. The **raw value of a story point is unimportant**. What matters are the **relative values**.
  - Related to how hard it is and how much of it there is
  - **Not** related to amount of time or the number of people
  - Unitless, but numerically-meaningful
- ideal time 理想时间
- elapsed time 实际时间
- velocity: measure of a team's rate of progress

优先级

- high/medium/low

deriving an estimate

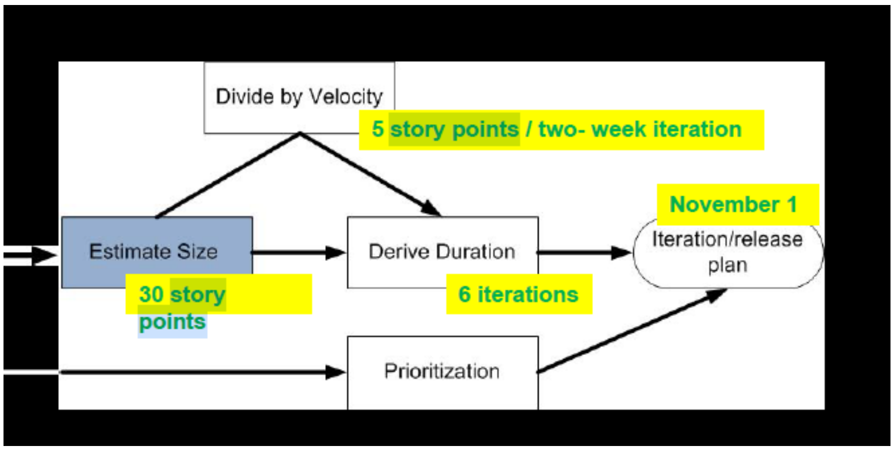
- expert opinion
- analogy （类比于其他类似的user stories）
- disaggregation （分成小部分去做，合起来的时候需要查看是否正常工作）
- planning poker (结合以上三种方式)

velocity

- Velocity is calculated by summing the number of **story points** assigned to each user story that the team **completed** during the operation.
- 假设下一迭代阶段的速度是之前的平均速度

prioritization

- driven by customer, in conjunction with developer
- Choose features to fill up velocity of iteration, based on:
    - Desirability of feature to a broad base of customers/users
    - Desirability of feature to a small number of important customers/users



planning game

- Customer writes user stories
- Programmers estimate time to do each story
- If story is too big, customer splits it
- Customer chooses stories to match project velocity
- Project velocity is amount of work done in the previous iteration(s)

planning

- Programmers only worry about **one iteration at a time**
- Customer can plan as many iterations as desired, but can change future iterations

simplicity

- Add one feature (user story) at a time
- Don’t worry about future stories
- Make program as simple as possible

unit tests and refactoring

- because code is as simple as it can be, adding a new feature tends to make it less simple
- to recover simplicity, you must refactor the code
- to refactor safely, you should have a rigorous set of unit tests

working software

- 所有的软件都有自动化的（单元）测试
- **all tests pass**, all the time
  - never check in broken code
- how to work on a task
  - 获取最新版本的代码，通过所有测试用例
  - 先写测试，失败
  - 写代码是测试用例通过
  - 重构
  - check

  （test-first programming/test-driven development）