Agile Software Development Process

Prof. Jonathan Lee (李允中)
CSIE Department
National Taiwan University

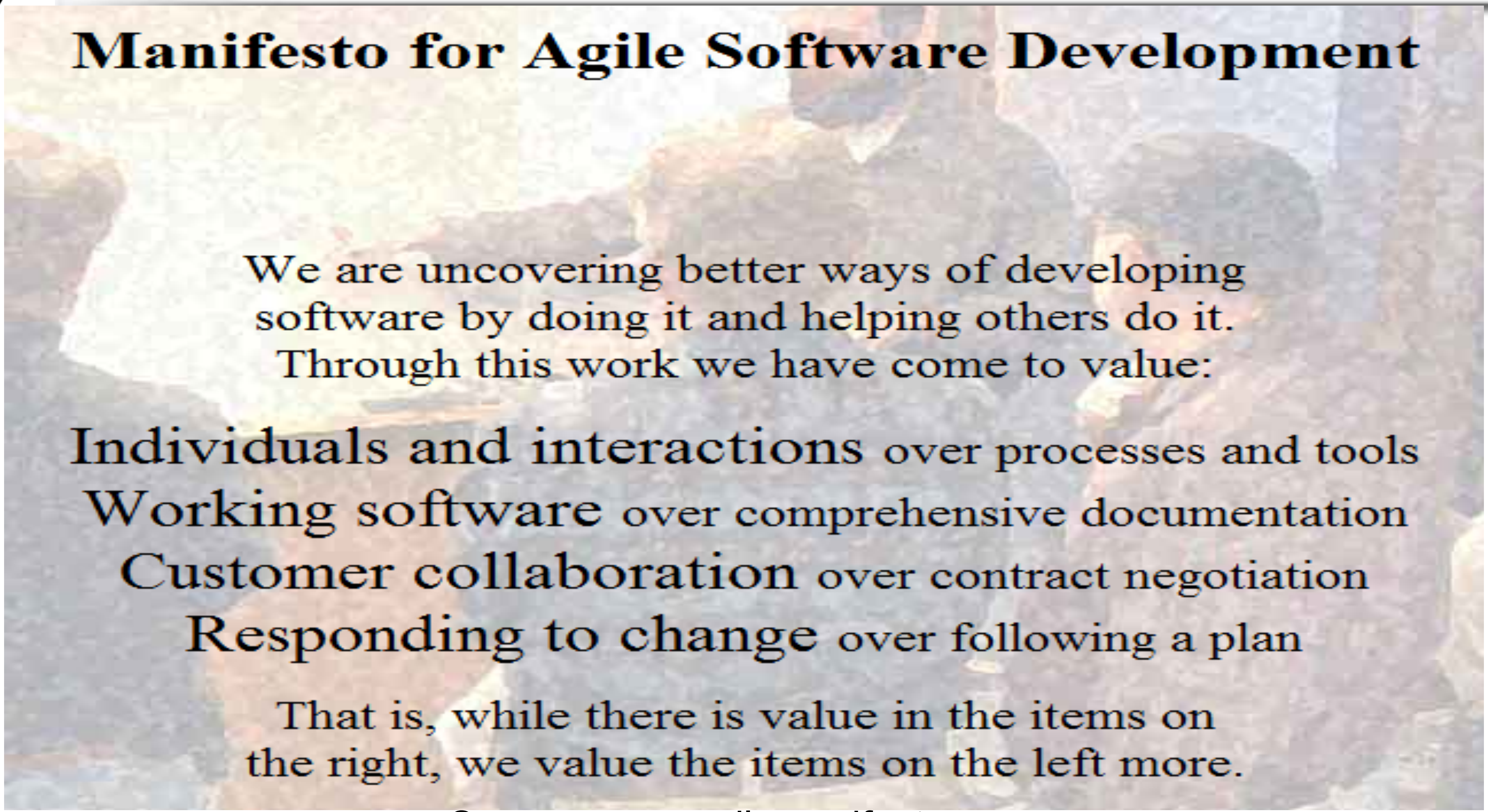
Homework (Due 26/12/2017)

- How to come up with the test cases before coding in order to implement Test-Driven Development?
- Explain why estimation plays the most important role in project management.
- Design and present your own software process for each team.

Traditional SW Development

- Acquire all the requirements before starting to design
- Freeze the requirements before starting to develop
- Resist changes: changes will lengthen schedule
- Build a change control process to ensure that proposed changes are examined carefully and no change is made without intensive scrutiny

The Agile Manifesto



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to 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

That is, while there is value in the items on the right, we value the items on the left more.

Source: www.agilemanifesto.org

Agile Methods

- SCRUM
- eXtreme Programming
- and many others …..

SCRUM$_1$

- Hirotaka Takeuchi and Ikujiro Nonaka (1986, Harvard Business Review) described a commercial product development approach based on case studies from **manufacturing firms** in the automotive, photocopier and printer industries.
- It is called the holistic or **rugby** approach, as the whole process is performed by one cross-functional team (scrum) across multiple overlapping phases, where the team "tries to go the distance as a unit, passing the ball back and forth”.

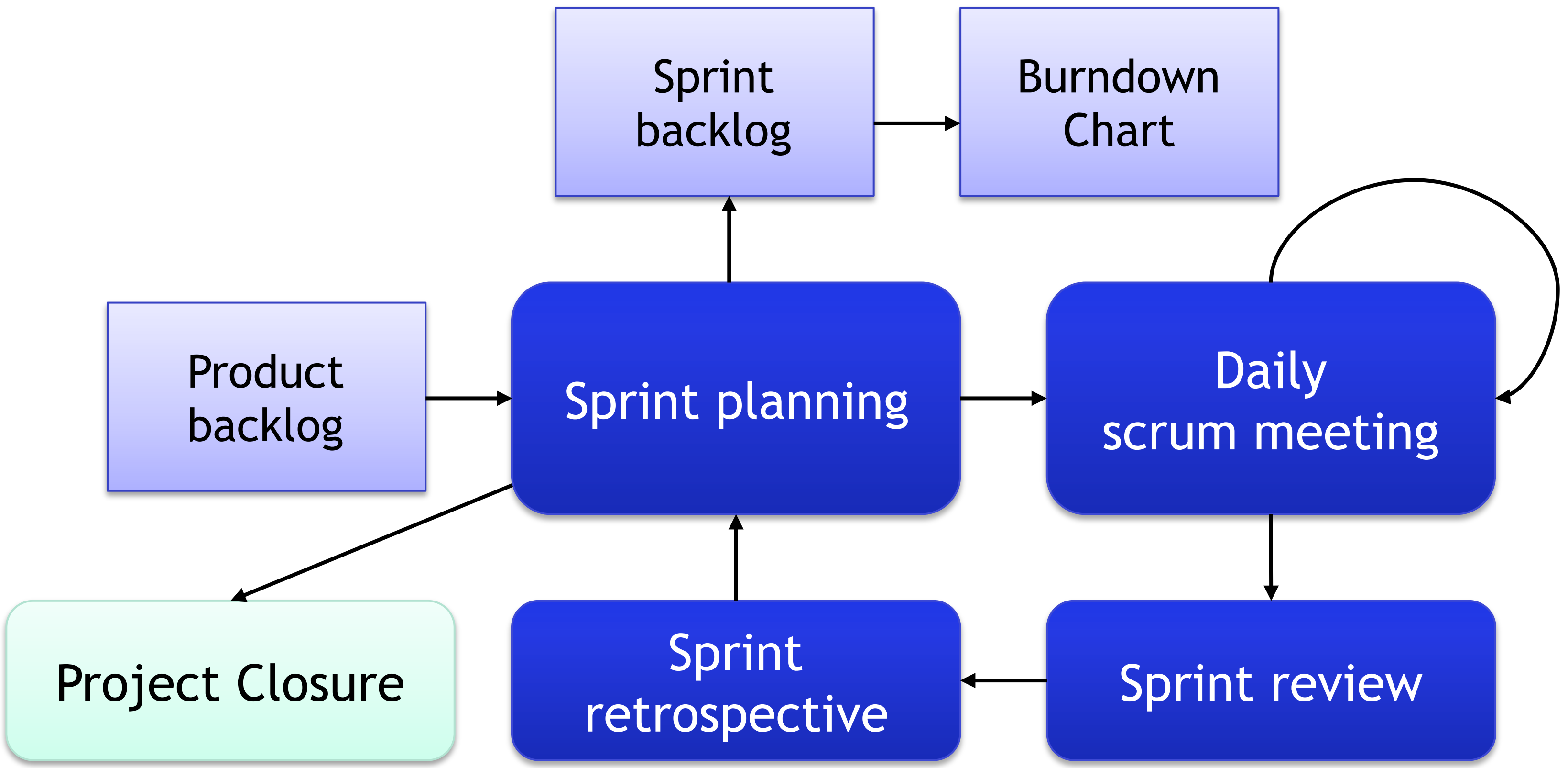
SCRUM$_2$

- SCRUM allows us to rapidly and repeatedly inspect actual working software (every two weeks to one month, estimate plays a key role).
- The business sets the priorities. Teams self-organize to determine the best way to deliver the highest priority features (Customers prioritize requirements).
- Every two weeks to a month anyone can see real working software and decide to release it as is or continue to enhance it for another sprint (iteration).

Sprint

- Scrum projects make progress in a series of “sprints” or iterations.
- Several sprints to constitute a milestone.
- Typical duration is 2–4 weeks or a calendar month at most.
- A constant duration leads to a better rhythm.
- Product is designed, coded, and tested during the sprint.
- Sprint goal: A short statement of what the work will be focused on during the sprint.

SCRUM Process: Cross-Functional-Team-Driven



SCRUM Framework

(People)
Roles (SCRUM team)
- Product owner
- ScrumMaster
- Development Team

(Process)
Core Processes
- Sprint planning
- Daily scrum meeting
- Sprint review
- Sprint retrospective

(Product)
Artifacts
- Product backlog
- Sprint backlog
- Burndown charts

Product Owner

- **Define the features of the product**
- Decide on release date and content
- Be responsible for the profitability of the product (ROI)
- Prioritize features according to market value
- Adjust features and priority every iteration, as needed
- Accept or reject work results

ScrumMaster (Leader)

- Represent **management** to the project
- Remove impediments
- Ensure that the team is fully functional and productive
- Enable close cooperation across all roles and functions
- Shield the team from external interferences

Development Team

- Typically 5-9 (7 ± 2) people
- Cross-functional: Programmers, testers, user experience designers, and etc.
- Teams are self-organizing
- Membership should change only between sprints

Sprint Planning

- Team selects items from the product backlog they can commit to completing
- Sprint backlog is created
  - Tasks are identified and each is estimated (1-16 hours)
  - Collaboratively, not done alone by the ScrumMaster
- High-level design is considered

Daily Scrum Meeting

- Parameters
  - Daily
  - 15-minutes
  - **Stand-up meeting**
- Not for problem solving, but for being aware of the current status: Ask three questions,
  - What did you do yesterday?
  - What will you do today?
  - Is anything in your way?
- Helps avoid other unnecessary meetings

Sprint Review

- ❑ Held when the sprint ends.
- ❑ Team presents **what it accomplished during the sprint**
- ❑ Typically takes the form of a **demonstration** of new features
- ❑ Whole team participates

Sprint Retrospective

- Held after the sprint review meeting
- The team considers three things:
  - what went well
  - what didn’t
  - what improvements could be made in the next sprint
- Typically 15–30 minutes

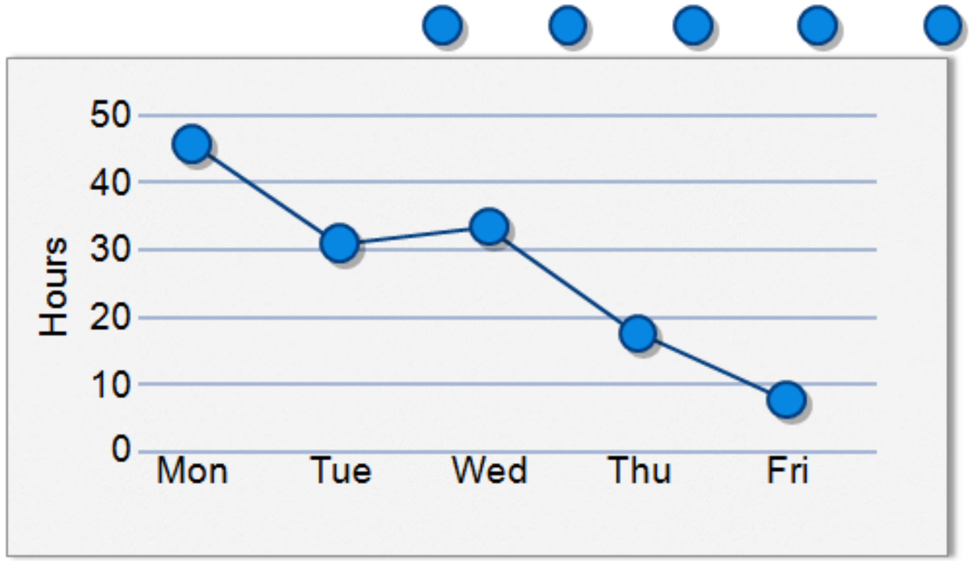
Sprint Burndown Chart

- The sprint burn down chart is a publicly displayed chart showing remaining work in the sprint backlog.

Sprint Backlog

Tasks	Mon	Tues	Wed	Thur	Fri
Code the user interface	8	4	8		
Code the middle tier	16	12	10	7	
Test the middle tier	8	16	16	11	8
Write online help	12				

Sprint Burndown Chart for the sprint



Product Backlog

- The **requirements**
- A list of all desired work on the project
- Ideally expressed such that each item has value to the users or customers of the product
- Prioritized by the product owner
- Reprioritized at the start of each sprint

Sprint Backlog

- The list of tasks a scrum team needs to complete during a sprint.
- An output of a sprint planning meeting.
- Turn a selected set of product backlog into a deliverable of increment of functionality.
- Each task in a sprint backlog has a time-based (hourly or daily) estimate.

Managing Sprint Backlog

- ❑ Estimated work remaining is **updated daily** so as to reflect on the Burndown Chart
- ❑ Any team member can add, delete or change the sprint backlog
- ❑ If work is unclear, define a sprint backlog item with a larger amount of time and break it down later
- ❑ Update work remaining as more becomes known

Scalability

- Typical individual team is 7 ± 2 people
  - Scalability comes from Scrum of Scrums (or Hierarchical Scrum)
  - Each Scrum team identifies one person who attends the Scrum of Scrums meeting to coordinate the work of multiple Scrum teams
- Factors in scaling
  - Type of application
  - Team size
  - Team dispersion
  - Project duration
- Scrum has been reported to be applied on multiple 500+ person projects through hierarchical scrum

Meeting Scheduler Product Backlog

ID	Backlog Item	Notes	How to test	Estimate (md)	Priority
1	身為一個使用者，我可以觀看所有會議之列表。	無。	使用者登入後點選"瀏覽全部會議"功能，系統列出所有已發起的會議列表。	6	20
2	身為一個使用者，我可以瀏覽會議行事曆，行事曆中包含我發起之會議或受邀請之會議。	行事曆預設為目前月份。	使用者登入後點選"會議行事曆"功能後，系統列出在行事曆中列出所有我發起的會議或受邀請的會議。	4	40
3	身為一個使用者，我可以在系統上發起一項會議。	使用者可以選擇此會議之類型、以及選擇欲調查之時間/地點範圍，決定討論項目以及負責人，以及選擇所欲邀請之參與人。	使用者登入後點選"發起會議"功能，填入所有應填資料後，此會議資料可透過"瀏覽全部會議"功能看到。	3	50

Meeting Scheduler Product Backlog

4	身為一個使用者，我可以觀看我發起之會議或受邀請之會議的詳細資料。	會議詳細資料包含會議名稱、會議之類型、會議發起人、邀請之與會者、確認參加之與會者、規劃之時間/地點範圍、排程後之時間/地點範圍。	使用者登入後點選"會議行事曆"功能後，再點選其中一個會議，可看到此會議之詳細資料。	3	10
5	身為一個使用者，我可以參與其他使用者發起的會議。	使用者只能參加受邀請之會議。 使用者可填寫可參加會議之時間/地點。	使用者登入後點選"參與會議"功能，點選其中一個受邀請之會議，並輸入數個可參與會議時間。此使用者及填入之時間將可透過"會議詳細資料"功能看到。	4	10
6	身為一個使用者，我可以對我發起的會議進行排程。	此會議排程，會針對參與者之參與的時間/地點意願來進行排程 會議排程給予會議一個會議排程等級用於解決時間/地點衝突時的優先權。	使用者登入後點選"會議行事曆"功能後，再點選其中一個會議，選擇"會議排程功能"，系統將會自動進行時程與地點之安排。	5	10

Meeting Scheduler Project Sprint Backlog

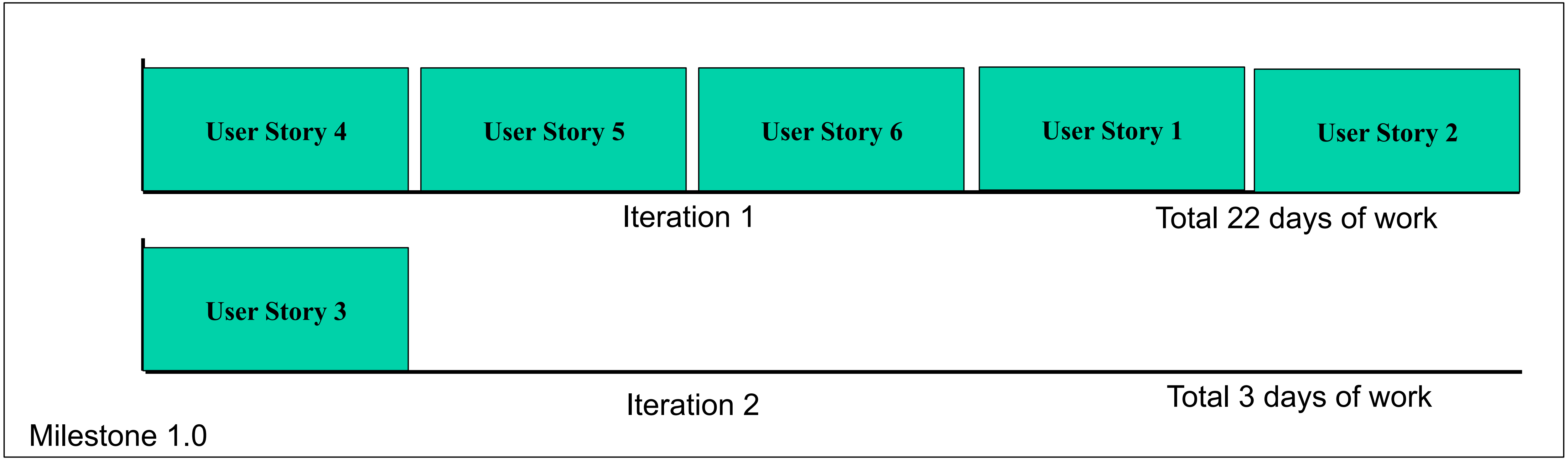
ID	Backlog Item	Task ID	Task	Estimate (hr)	Responsi-bility
1	身為一個使用者，我可以觀看所有會議之列表。。	1-1	DB schema design and creation	16	陳石佳
		1-2	Code the Meeting Info DAO	12	陳石佳
		1-3	Code the List GUI	12	陳石佳
		1-4	Write test fixtures	8	陳石佳
2	身為一個使用者，我可以瀏覽會議行事曆，行事曆中包含我發起之會議或受邀請之會議。	2-1	Code the Calendar UI	16	洪東昇
		2-2	Link DAO and the Calendar UI	8	洪東昇
		2-3	Write test fixtures	8	洪東昇
3	身為一個使用者，我可以在系統上發起一項會議。	3-1	Code the Initiation GUI	8	鄭聖翰
		3-2	Link DAO and Initiation UI	8	鄭聖翰
		3-3	Write test fixtures	8	鄭聖翰
4	身為一個使用者，我可以觀看我發起之會議或受邀請之會議的詳細資料。	4-1	Code the GUI	8	丘偉廷
		4-2	Link DAO and Browing UI	8	丘偉廷
		4-3	Write test fixtures	8	丘偉廷

Meeting Scheduler Project Sprint Backlog

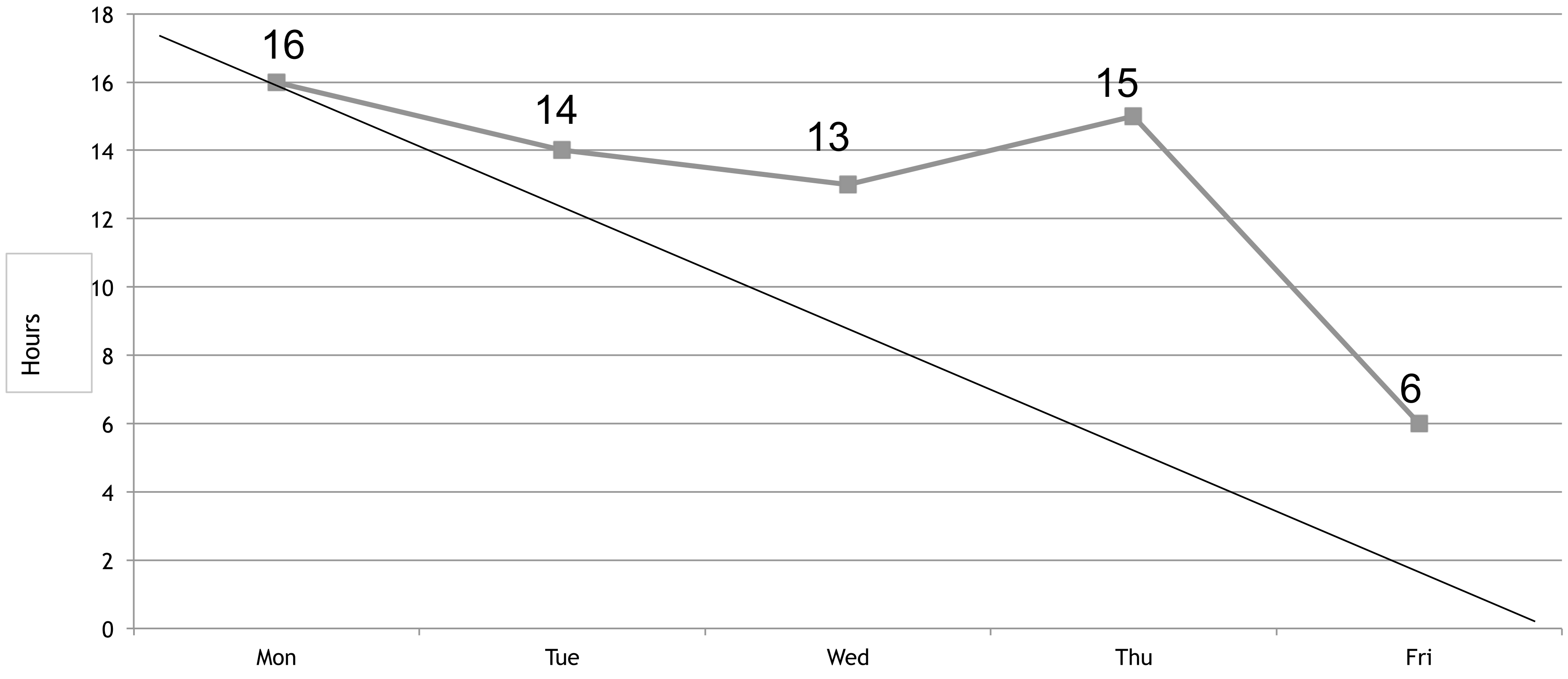
ID	Backlog Item	Task ID	Task	Estimate (hr)	Responsi-bility
5	身為一個使用者，我可以參與其他使用者發起的會議。	5-1	DB schema design and creation	8	陳石佳
		5-2	Code the Meeting Info DAO	12	陳石佳
		5-3	Code the List GUI	12	陳石佳
6	身為一個使用者，我可以對我發起的會議進行排程。	6-1	Code the Calendar UI	18	洪東昇
		6-2	Link DAO and the Calendar UI	12	洪東昇
		6-3	Write test fixtures	10	洪東昇

Total: 200 hrs

Estimation of Meeting Scheduler



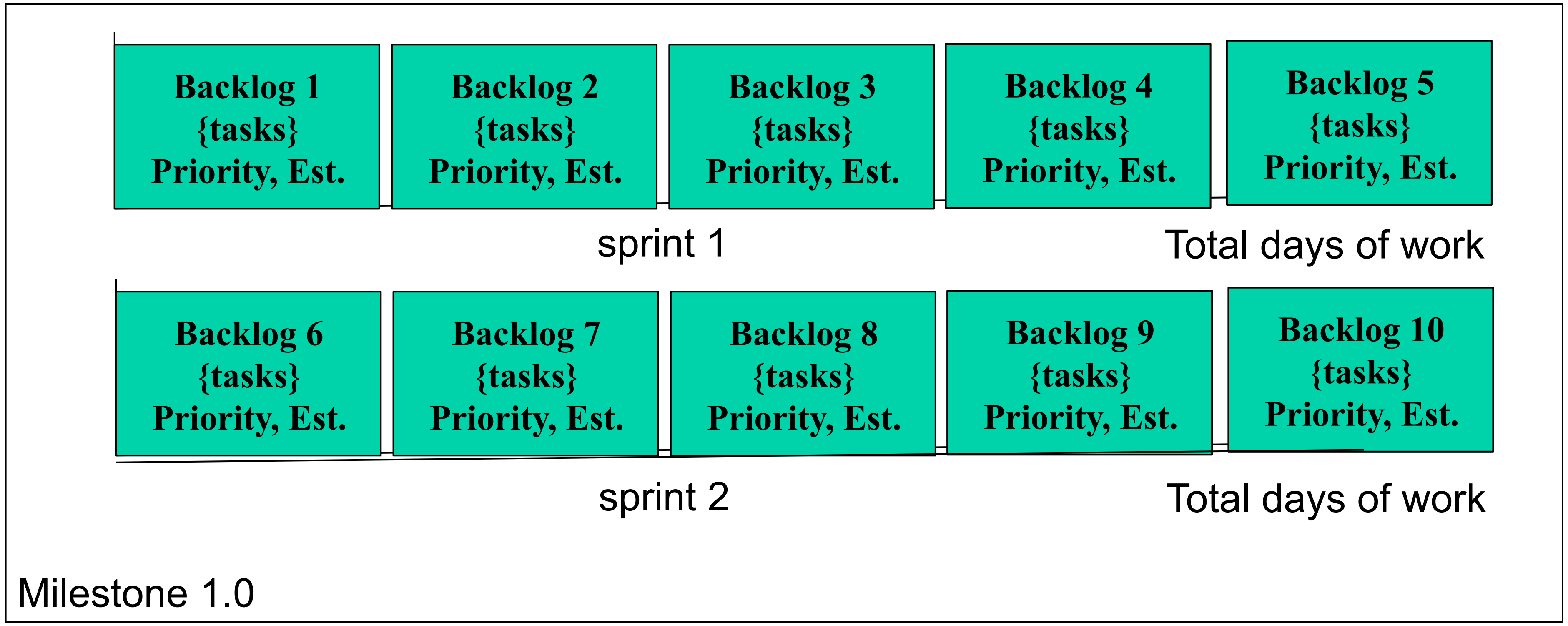
Meeting Scheduler Project Burndown Chart



Total: 200 hrs

Hours Spent: 64 (16+14+13+15+6)
Remaining Hours: 136

SCRUM in Estimation


Backlog 1 {tasks} Priority, Est.
Backlog 2 {tasks} Priority, Est.
Backlog 3 {tasks} Priority, Est.
Backlog 4 {tasks} Priority, Est.
Backlog 5 {tasks} Priority, Est.
sprint 1
Total days of work
Backlog 6 {tasks} Priority, Est.
Backlog 7 {tasks} Priority, Est.
Backlog 8 {tasks} Priority, Est.
Backlog 9 {tasks} Priority, Est.
Backlog 10 {tasks} Priority, Est.
sprint 2
Total days of work
Milestone 1.0

Next Milestone (More iterations)

Velocity: the Reality Bite

- 30 days of calendar month
- 20 to 22 days of working days (weekend, vocation, and things come up along the day)
- Velocity is a percentage: given X number of days, how much of that time is productive work.
  - During the work time, a percentage of which will be taken by holidays, software installation, paperwork, phone calls, and other non-development tasks.
  - Real work days: 20 * 0.7 (velocity) = 14 days.
  - If you have 5 people in your team, then the velocity is: 5 * 14 = 70 days
  - Float: 5 * 20 – 70 = 30 days

SCRUM Summary

- For the whole process: Requirements → Product backlog → Estimate sprint cycles → Milestones

- For each sprint: Sprint backlog (tasks with estimates) → Burndown chart (monitoring the performance) → Release

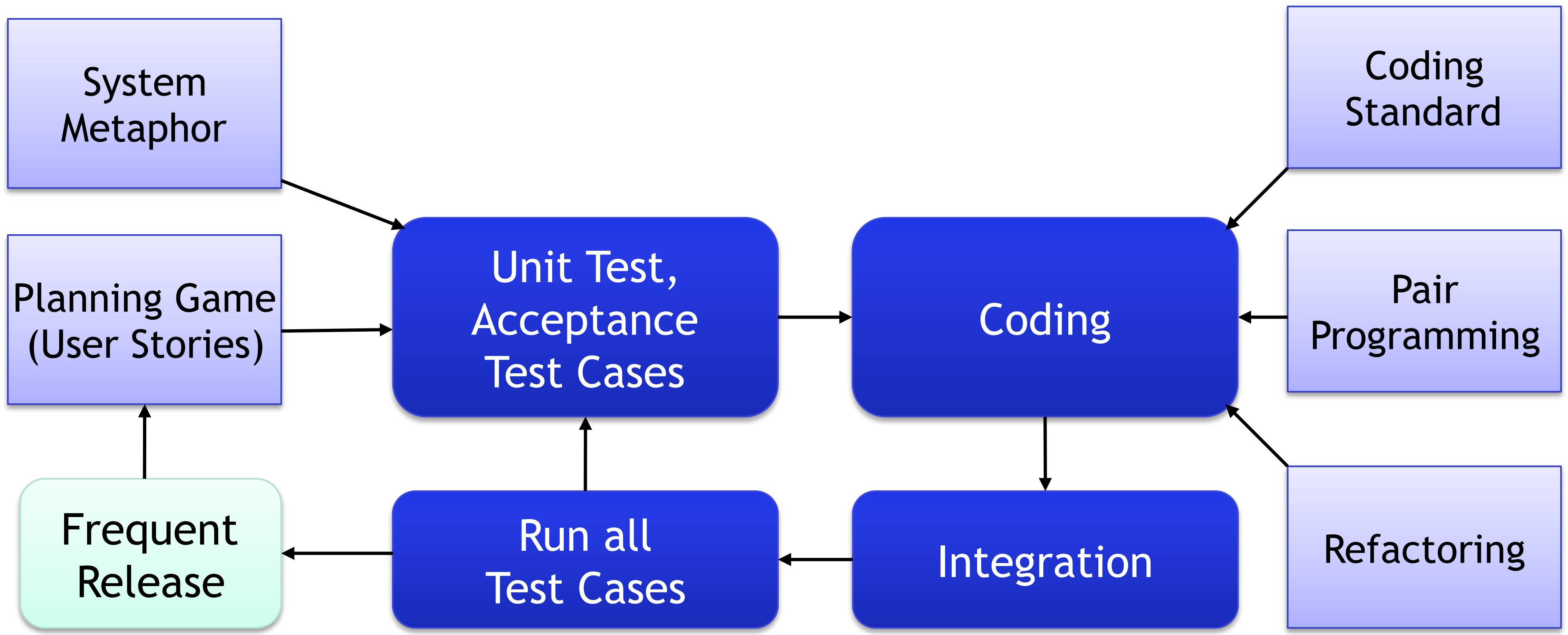
eXtreme Programming (XP)

- Recognize the fact that:
  - All requirements will not be known at the beginning, and requirements will change
- Use tools to accommodate change as a natural process
  - Tools for build, unit testing, version control, issue tracking, and etc.
- Do the simplest thing that could possibly work and refactor mercilessly
  - Follow design principles

The 12 Practices

- XP is based on the extreme application of 12 practices that support each other:
  - Incremental Planning (Planning game)
  - Frequent release (Small release)
  - Test-driven development
  - Refactoring
  - Pair programming
  - Continuous integration
  - Coding standards
  - System metaphor
  - Collective code ownership
  - Simple design with iteration
  - Forty-hour week
  - On-site customer

XP Process: Change-Driven



System Metaphor

- The system metaphor provides a broad view of the project's goal.
- It defines the overall theme to which developers and clients can relate.
- Common concept of what the system is like.
- The system is built around one (or more) system metaphor from which classes, methods, variables and basic responsibilities are derived.
- Metaphor Example:
  - Buffered Text Displayer → Garbage Truck

Incremental Planning

- Requirements → User story
  - Requirements are written **by the customer** on small index cards
  - User stories are written in business language and describe things that the system needs to do
  - A user story is usually around three lines long, and are accompanied by an estimate of person-day and and a priority.
- Commitment
  - Customer and developer decide which user stories constitute the release.

User Story Examples

Story: Handle overdraft

When a transaction causes a customer’s account to go into overdraft, transfer money from the overdraft protection account, if any.

Priority: 20 Estimate: 6

Story: Compute balance

For each account, compute the balance by adding up all the deposits, and subtracting all the deductions.

Priority: 30 Estimate: 4

Test-Driven Development$_1$

- Tests play the most important and central role in XP.
- **Tests are written before the code is developed.**
  - forces concentration on the interface.
  - accelerates development.
  - test serves as a safety net **for coding and refactoring.**
- **All** tests are automated (test suites, testing framework).

Test-Driven Development$_2$

- Two kinds of test:
  - Acceptance tests (functional tests).
    - Clients provide test cases for their stories.
    - Developers transform these in automatic tests.
  - Unit tests.
    - Developers write tests for their classes (before implementing the classes).
    - All unit tests must run 100% successfully all the time.
- **Write test before your code!**

Refactoring

- Process of improving code structure while preserving its behaviour by following design principles.
- Change it even if it is not broken.
- The aim of refactoring is to:
  - make the design simpler
  - make the code **more understandable**
  - **remove duplicate code**
  - **improve the tolerance of code to change**
- Tests guarantee that refactoring didn't break anything that worked.

Pair Programming

- Two programmers sit together in front of a workstation.
  - one enters code.
  - one reviews the code and thinks.
- Pairs change continuously (few times in a day).
  - every programmer knows all the aspects of the system.
  - a programmer can be easily replaced in the middle of the project.
- Costs 10-15% more than stand-alone programming.
- Code is simpler with less defects.

Continuous Integration

- Continuous integration wraps version control, compilation, and testing into a single repeatable process.
- **Daily integration** at least
- **XP feedback cycle: Develop unit test, Code, Integrate, Run all units tests and acceptance tests, Release.**
- A working tested system is always available
- Need *version control system*, *issue tracking*, *automatic testing*, *and build tools.*
- DevOps

Coding Standards

- Coding standards make pair progamming and collective code ownership easier.
- Common name choosing scheme
- Common code formatting

Simple Design with Iteration

- Do the simplest thing that could possible work.
  - Create the best design you can.
  - Improve through testing and designing in an iterative manner.
  - Do not spend time implementing potential future functionality (requirements will change).
- **Put in what you need when you need it**

On-Site Customer

- User stories are not detailed, so there are always questions to ask the customer.
- The customer must always be available.
  - to resolve ambiguities
  - set priorities
  - provide test cases
- Customer is considered part of the team.

Collective Code Ownership

- The code does not belong to any programmer.
- Any programmer **can (should)** change any of the code at any time to:
  - make it simpler
  - make it better
- Encourage the entire team to work more closely together.
- Everybody tries to produce a high-quality system.
  - code gets cleaner
  - system gets better all the time
  - everybody is familiar with most of the system

Forty-Hour Week

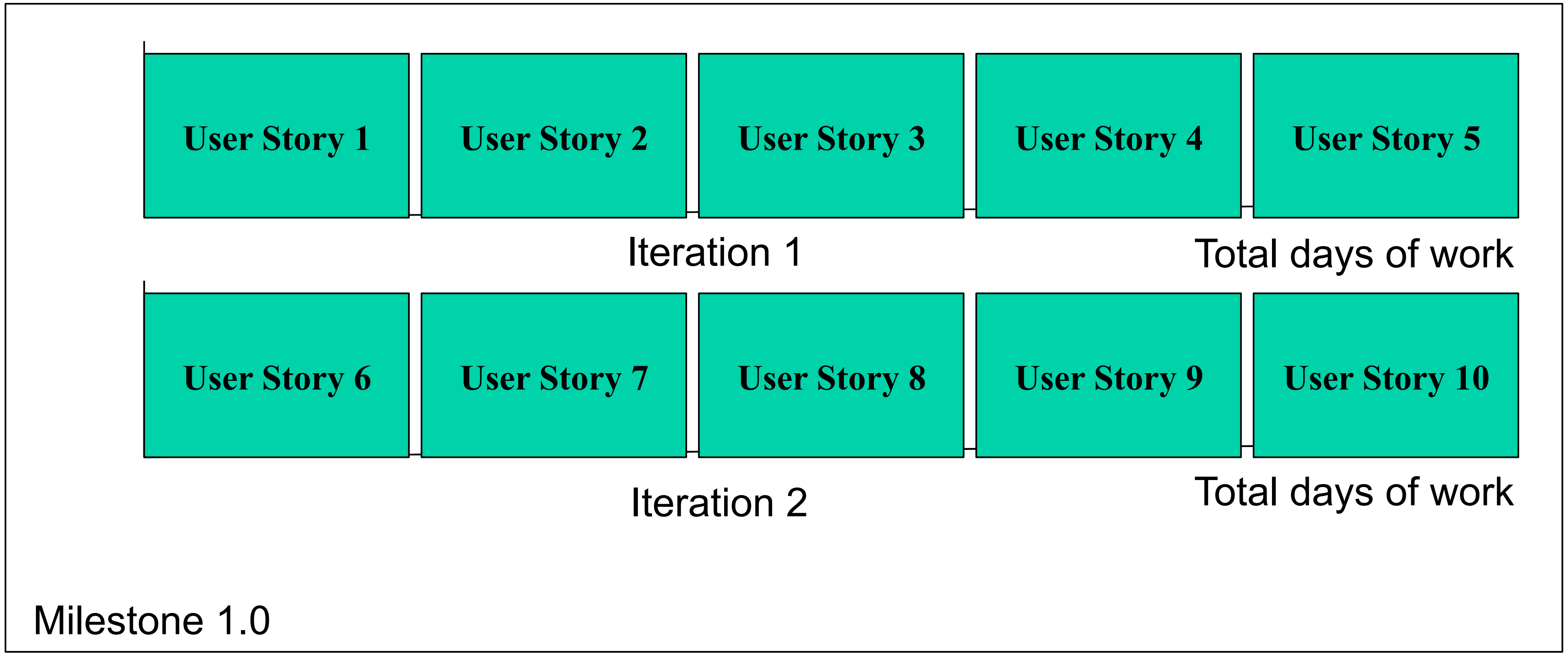
- "Overtime is defined as time in the office when you don't want to be there" Ron Jeffries.
- Programmers should not work more than one week of overtime.
- **If more is needed then something is wrong with the schedule.**
- Keep people happy and balanced.

XP Summary

- Requirements → Prioritize user stories (on-site customers) → estimate iteration cycles → Milestones → Release

- Each iteration → User stories → Test cases → Coding (with Pair programming and coding standard) → Continuous integration → Refactoring → Release (Y/N)

XP in Estimation



Next Milestone (More iterations)

References

- http://www.agilemanifesto.org
- Hirotaka Takeuchi and Ikujiro Nonaka, The New New Product Development Game, Harvard Business Review, Jan. 1986.
- ScrumKen Schwaber and Mike Beedle, Agile Software Development with Scrum, 2001.
- Kent Beck and Cynthia Andres, “Extreme Programming Explained: Embrace Change”, 1999.
- 李允中, *軟體工程*, 台灣軟體工程學會, 2013.