Individuals and interactions over processes and tools

working software over comprehensive documentation

customer collaboration over contract negotiation

responding to change over following a plan

highest priority is to satisfy the customer through early and continuous delivery of valuable software.

welcome changing requirements, even late in the development. Agile process harness change for the customer's competitive advantage.

Deliver working software frequently from a couple of weeks to a couple of months, with a preference to the shorter timescale.

business people and developers must work together daily throughout the project.

build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

working software is the primary measure of progress.

agile processes promote sustainable development. The sponsors, developers, and the users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity   the art of maximizing the amount of work not done   is essential

at regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

XP—Kent Beck

a reaction against the culture(process, plans, diagrams) that was predominant in software engineering circles at the time.
Drawing attention to the fact that what really matters in the end is the programs, and of course, the programming and the programmers.
XP made a major contribution by rehabilitating, the work of programmers, and putting programs and programming at the center of software development.

Lean —Mary Poppendieck

An attempt to apply to software a number of ideas and principles which have proved their value in other engineering fields, in particular, in the car industry.(把其他领域有价值的原则应用于软件)

Crystal— Alistair Cockburn

a combination of agile ideas and more traditional ideas.Crystal is actually the name for a set of methods characterized by various degrees of importance of process in management.
combine the best of agile with a best of more process oriented approaches.

Scrum—Schwaber&Sutherland

Less technical not so much software-focused than XP

A management method which emphasizes ideas such as the importance of self-organizing teams as opposed to teams that are closely managed by a manager, and the importance of a specific kind of short release iterations known as Sprints.
It is important to note that in practice nowadays when we talk about Agile, we really talk about Scrum.

agile principles

organizational
1. put the customer at the center
2. accept change
3. let the team self-organize
4. maintain a sustainable pace
5. produce minimal software
    1. produce minimal functionality
    2. produce only the product requested
    3. develop only code and tests

6.develop iteratively
    1.produce frequent working iterations
    2.freeze requirements during iterations

7.treat tests as a key resource:
    1.do not start any new development until all tests pass
    2.test first

8.express requirements through scenarios

problems with the waterfall

late appearance of actual code
lack of support for requirements change — and more generally for extendibility and reusability
lack of support for the maintenance activity
division of labor hampering total quality management
impedance mismatches
highly synchronous model

traditional view: managers tell workers to do their job
agile view: managers listen to developers, explain possible actions, provide suggestions for improvements.

the leader is there to:
1. encourage progress
2. help catch errors
3. remove impediments
4. provide support and help in difficult situations
5. make sure that skepticism does not ruin the team’s spirit

XP avoids “crunch time” of traditional projects thanks to short release cycles
To help achieve these goals:
1.frequent code-merge
2.always maintain executable, test-covered, high-quality code
3.constant refactoring, helping keep fresh and alert minds
4.collaborative style
5.constant testing

Minimalism:
1. minimal functionality
2. product only
3. only code and tests

Lean view:

seven wastes:
1.extra/unused features(overproduction)
2.partially developed work not released(Inventory)
3.intermediate/unused artifacts(extra processing)
4.seeking information(Motion)
5.escaped defects not caught by tests/reviews(Defects)
6.waiting(including customer waiting)
7.handoffs(transportation)

produce frequent working iterations
freeze requirements during iterations

do not start any new development until all tests pass
test first

express requirements through scenarios

Iterativeness:
1.frequent working iterations
2.freeze requirements during iteration
(closed-window rule: during an iteration, no one may add functionality)

user story is simply something a user wants

Stories are more than just text written on an index card but for our purposes here, just think of user story as a bit of text saying something like

put the customer at the center

accept change
let the team self-organize
maintain a sustainable pace
produce minimal software

develop iteratively   produce frequent working iterations   freeze requirements during iteration

do not start any new development until all tests pass
test first

express requirements through scenarios

eliminate waste
unnecessary code
unnecessary functionality
delay in process
unclear requirements
insufficient testing
avoidable process repetition
bureaucracy
slow internal communication

partially done coding
waiting for other activities, team, processes
defects, lower quality
managerial overhead

Amplify learning

1. to prevent accumulation of defects, run tests as soon as the code is written(及时测试)
2. instead of adding documentation or planning, try different ideas by writing and testing code and building
3. present screens to end-users and get their input
4. enforce short iteration cycles, each including refactoring and integration testing
5. set up feedback sessions with customers

delay decisions as much as possible until they can be made based on facts, not assumptions, and customers better understand their needs.

the more complex a system, the more capacity for change should be built in

use iterative approach to adapt to changes and correct mistakes, which might be very costly if discovered after system release

planning remains, but concentrates on the different options and adapting to the current situation, as well as clarify confusing situations by establishing patterns for rapid action

focus on individual task to ensure progress:
control flow of progress
deal with interruptions:
    two-hour period without interruption
    assign developer to project for at least two days before switching

focus on direction of project
    define goals clearly
    prioritize goals

deliver as fast as possible

it is not the biggest that survives, but the fastest
The sooner the end product is delivered, the sooner feedback can be received, and incorporated into the next iteration
For software, the Just-in-Time production ideology means presenting the needed result and letting the team organize itself to obtain it in a specific iteration

Another key idea from Toyota is set-based design. If a new brake system is needed, three teams my design solutions to the problem

If a solution is deemed unreasonable, it is cut

at period end, the surviving designs are compared and one chosen, perhaps with modifications based on learning from the others — an example of deferring commitment until the last possible moment.

Optional role:
guides team
mentor team
lead by example
teaches when necessary
may teach by doing
may offer ideas to solve thorny problems
may serve as intermediary with management

Daily Meeting
“stand-up meeting”
15 minutes
defining commitments
uncovering impediments
set the day’s work, in the broader context of the project
resolution will take place outside of meeting

“are there any impediments in your way”
“What did you do yesterday”
“What will you do today”

planning meeting

at beginning of every sprint

goal: define work for sprint

outcome: Sprint Backlog, with time estimate for every task

8-hour time limit

retrospective
all team members reflect on past sprint
make continuous process improvements

what went well?
What could be improved?

3-hour time limit
Normally does not involve customers

Review work:
    completed
    not completed
4-hour time limit

pair programming
goals:
1.make thinking process explicit
2.keep each other on task
3.brainstorm refinements to system
4.clarify ideas
5.take initiative when other stuck, lowering frustration
6.hold each other accountable to team practices

shared code
 most non-trivial features extend across many layers in the application
code ownership creates unnecessary dependencies between team members and delays
what counts is implemented features, not personal responsibility
avoid blame game avoid specialization
minimize risk(team members leaving)

把优化工作放到最后，这样才能分析出什么需要来优化

developers work in small steps, validating each before moving to the next three parts:
1.start by creating the simplest design that could possibly work
2.incrementally add to it as the needs of the software evolve
3.continuously improve design by reflecting on its strengths and weaknesses

A metaphor is meant to be agreed upon by all members of a project as a means of simply explaining the purpose of the project and thus guide the structure of the architecture.

Refactoring
“Disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior”

only one pair integrates code at a time(to avoid conflicts)
Collective code ownership
development proceeds in parallel

continuous integration(一天好几次)
the combination of frequent releases with relentless testing

XP teams practice small releases in two important ways:
1.release running, tested software, delivering business value chosen by the customer, every iteration
2.release to end users frequently as well.

ten-minute build
compile source code
run tests
configure registry settings
initialize database schemas
set up web servers
launch processes
build installers
deploy

weekly cycle
1.review progress
2.让顾客 选择一个一周工作量的user stories
3.把这个user story分成任务

CORE idea of XP: do not write code without associated unit tests

code that does not pass tests is waste
在所有test通过之前不要进行到下一步

write tests before code
the test replaces the specification

Code the unit test first

1. find out what you have to do
2. write a unit test for the desired new capacity. Pick the smallest increment of new capability you can think of.
3. run the unit test. If it succeeds, you are done.（回到步骤一，如果项目完成了就回家歇着了）
4. fix the immediate problem: 修复中间不正常的method

Test-Driven Development
1. add a test
2. run all tests and see if the new one fails
3. write some code
4. run the automated tests and see them succeed
5. refactor code

expected benefits:
1.catch bugs early
2.write more tests
3.drive the design of the program
4.replace specifications by tests
5.use debugger less
6.more modular code
7.better coverage
8.improve overall productivity

XP: a bug is not an error in logic, it is a test you forgot to write

Root-Cause Analysis

When finding a defect, do not just fix it but analyze cause and make sure to correct that cause, not just the symptom

Acceptance tests are black box system tests. Each acceptance test represents some expected result from the system.

Automate them so they can be run often

publish acceptance test score is to the team

each day after daily scrum
clusters of teams discuss areas of overlap and integration
A designated person from each team attends
Agenda as daily scrum, plus the following four questions:
1.what has your team done since we last met?
2.what will your team do before we meet again?
3.Is anything slowing your team down?
4.Are you about to put something in another team’s way?

整个团队：
1.business representative
2.programmers
3.may include testers
4.may include analysts, helping to define requirements
5.often includes a coach
6.may include a manager

Planning Poker
1.present individual stories for estimation
2.discuss
3.deck has successive numbers(quasi-Fibonacci)
4.each participant chooses estimate from his deck
5.keep estimates private until everyone has chosen a card
6.reveal estimates
7.repeat until consensus

Open Workspace
1.organized around pairing stations
2.with whiteboard space
3.locating people according to conversations they should overhear
4.with room for personal effects
5.with a place for private conversations

Expected benefits: improve communication, resolve problems quickly with the benefits of face-to-face interaction

Osmotic Communication

Sprint: a specific kind of short release iterations known as Sprints.

the closed-window Rule:
during an iteration, no one may add functionality (the sprint is cancelled)

The product backlog is an ordered list of features that are needed as part of the end product and it is the single source of requirements for any changes to be made to the product.

the product backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. Product Backlog items have the attributes of a description, order, estimate, and value. These items are normally termed as User Stories.
A product Backlog is an evolving artifact. The earliest version of it may contain only the initially known and best understood requirements. The product backlog gets developed as the product, and the environment in which it will be used, progress. The Product Backlog constantly changes to incorporate what is required to make it effective. As long as a product exists, its product Backlog also exists.

scrum artifacts
1.product backlog
2.sprint backlog
3.product increment

the sprint backlog is the set of product backlog items selected for the sprint, plus a plan for delivering the product increment and realizing the sprint goal.

The sprint backlog is a forecast by the team about what functionality will be made available in the next increment and the work needed to deliver that functionality as a working product increment and realizing the Sprint Goal

Increment
The increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints.

user story:
a short description in everyday or business language, of a feature told from the perspective of the person who desires the new capability.
defines an atom of functionality
A task can be seen as a subunit of user stories. They can be seen as units of work that are contained within user stories. For a user story to be counted as completed, all of its tasks must be in turn also completed.

1. description of requirement or need
2. conversations during backlog grooming and iteration planning added as details
3. tests that confirm the story's satisfactory completion

story point:
an arbitrary measure used by Scrum teams. This is used to measure the effort required to implement a story.

good quality of user stories

INVEST:
I: independent 是否可以自己stand-alone
N: Negotiable 是否可以changed or removed without impact to everything else?
V: Valuable Does this story have value to the end user?
E: Estimable Can you estimate the size of the story?
S: Small Is it small enough?
T: Testable Can this story be tested and verified?

user story:
simple
written by customer
incomplete, possibly inaccurate
does not handle exceptional cases
starting point for additional discussions with customer

use case:
more complex
written by developer in cooperation with customer
attempts to be complete, accurate
should handle all possible cases
intended to answer any developer questions about customer
requirements without further interaction with customer

Product Backlog:

maintained throughout project

property of product owner

open and editable by anyone

contains backlog items:broad descriptions of all potential features, prioritized by business value

include estimates of business value

include estimates of development effort, set by team

Velocity:
- measure of progress in a project:
  - number of items delivered

Burndown Chart:
publicly displayed chart, updated everyday, showing, for the sprint backlog
1.remaining work
2.progress

Bullpen:
Single, open room

Task/story board
1.transparency
2.collaboration
3.prioritization
4.focus
5.self-organization
6.empiricism
7.”humility”
8.morale

pair programming
1.make thinking process explicit
2.keep each other on task
3.brainstorm refinements to system
4.clarify ideas
5.take initiative when other stuck, lowering frustration
6.hold each other accountable to team practices

Shared code
1.most non-trivial features extend across many layers in the application
2.code ownership creates unnecessary dependencies between team members and delays
3.what counts is implemented features, not personal responsibility
4.avoid blame game
5.avoid specialization
6.minimize risk(team members leaving)