“The Software Engineering process is the total set of Software Engineering activities needed to transform requirements into software.”

Watts S. Humphrey. Software Engineering Institute, CMU. http://portal.acm.org/citation.cfm?id=75122
Activities in the process?

- Implementation
- ...
Activities in the Process

- Requirements Gathering / Specification
- Analysis & Design
- Implementation
- Environment Setup
- Training
- Testing
- Deployment
- Maintenance
- Project Management
Software Processes

- Waterfall
- ...
Software Processes

- Waterfall (Royce)
- V Model (German Ministry of Defense)
- Prototyping
- Operational Specification
- Transformational (automated software synthesis)
- Phased Development: Increment and Iteration
- Spiral Model (Boehm)
- The Rational Unified Process (RUP)
- Extreme Programming (XP)
Waterfall
- Requirement Analysis
- System Design
- Program Design
- Coding
- Unit & Integration Testing
- System Testing
- Acceptance Testing
- Operation & Maintenance
Prototyping

Build Prototype

Gathering Requirements

Customer Feedback / Test

Final Product
The Rational Unified Process (RUP)
RUP: Observations

- Waterfall-like sequence of

- Not pure waterfall
  - Phased Development (iterative)
  - Overlap (concurrency) between activities

- Testing
  - Regression (test not only newly developed, but also previously developed code)
  - Testing starts before design and coding
Agile Manifesto

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

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

- The best architectures, requirements, and designs emerge from self-organizing teams.

- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.
Extreme Programming (XP)

www.extremeprogramming.org
- User Stories are written by the customers as things that the system needs to do for them.
  - They drive the creation of acceptance tests.
- The project is divided into Iterations.
- Code the Unit Test First
  - Write the simplest code that succeed the test
- Make frequent small releases
- Pair Programming
  - Two programmers, 1 desk
- Refactor Mercilessly
  - Because less code is easier to maintain
Collective Code Ownership

Extreme Programming

Collective Code Ownership

Next Task or Failed Acceptance Test → Pair Up → Create a Unit Test → CRC Cards → Simple Design or Complex Problem

Move People Around → Change Pair → We Need Help

Pair Programming → Failed Unit Test → Passed Unit Test → Simple Code or Complex Code

Refactor Mercilessly

Continuous Integration → New Unit Tests → New Functionality

Acceptance Test Passed

100% Unit Tests Passed → Run Failed Acceptance Test
Pair Programming Workstation

Pair Programming

www.charm.net/ jriley/pairall.html