The Software Life Cycle

SIMPLY EXPLAINED

A Software Crisis

Denver International Airport (DIA)
Construction started in 1989 • 53 sq miles
• Planned: 1.7 bio USD costs, opening 1993
Code and Fix (1950–)

Build first version

Modify until client is satisfied

Operate

Retirement

Code and Fix: Issues

- No process steps – no specs, docs, tests…
- No separation of concerns – no teamwork
- No way to deal with complexity

Code and Fix
Waterfall Model

(1968)

Communication
- project initiation
- requirements gathering

Planning
- estimating
- scheduling
- tracking

Modeling
- analysis
- design

Construction
- code
- test

Deployment
- delivery
- support
- feedback

Communication

6.6 Map Source Tool

List Case Description

- Input: A list of map names stored in a list called `mapNames`.
- The map names are continuous, e.g., `mapNames = ['Map1', 'Map2', 'Map3']`, or list of maps from a server.
- Elements from `mapNames` are stored on sheets.

Steps:

1. Display the first map name in the `mapNames` list.
2. Show the next map name in the `mapNames` list.
3. Show the next map name in the `mapNames` list.
4. Show the next map name in the `mapNames` list.
5. Show the next map name in the `mapNames` list.

Inputs:
- `mapNames`: List of map names.

Outputs:
- `mapNames`: Updated list of map names after processing.
Waterfall Model
(1968)

- Communication: project initiation, requirements
- Planning: estimating, scheduling, tracking
- Modeling: analysis, design
- Construction: code, test
- Deployment: delivery, support, feedback

Waterfall Model

- Construction: code, test

Waterfall Model
(1968)
Deployment

**Questionnaire**
- Very often
- Often
- Sometimes
- Rarely

**Deployment**
- Delivery
- Support
- Feedback

**Waterfall Model**
(1968)

**Communication**
- Project initiation
- Requirements gathering

**Planning**
- Estimating
- Scheduling
- Tracking

**Modeling**
- Analysis
- Design

**Construction**
- Code
- Test

**Deployment**
- Delivery
- Support
- Feedback

**Simply Explained**

We’re planning a expedition to a part of the jungle where no men have ever been.

WOW! That sounds exciting! And how will you find your way when you are there?

[http://geekandpoke.typepad.com/geekandpoke/2012/05/simply-explained-wtf.html](http://geekandpoke.typepad.com/geekandpoke/2012/05/simply-explained-wtf.html)
Waterfall Model (1968)

- Real projects rarely follow a sequential flow
- Hard to state all requirements explicitly
- No maintenance or evolution involved
- Customer must have patience
- Any blunder can be disastrous

http://geekandpoke.typepad.com/geekandpoke/2012/05/simply-explained-wtf.html
Boehm’s first law

Errors are most frequent during requirements and design activities and are the more expensive the later they are removed.

Problem Cost

- Relative cost of problem per phase

![Problem Cost Graph]

Incremental Model

This and other laws are found in Endres/Rombach: Handbook of Software and Systems Engineering. Evidence: Several studies before 1974.
Incremental Model

- Each linear sequence produces a particular "increment" to the software
- First increment typically core product; more features added by later increments
- Allows flexible allocation of resources

Prototyping

Communication → Quick Plan → Quick Design → Prototype Construction → Deployment and Feedback

Prototypes

Top Layer (GUI)

Bottom Layer
Prototypes

- A *horizontal prototype* tests a particular *layer* (typically the GUI) of the system
- A *vertical prototype* tests a particular *functionality* across all layers
- Resist pressure to turn a prototype into a final result!

Spiral Model

(1988)

- System is developed in series of evolutionary releases
- Milestones for each iteration of the spiral
- Process does not end with delivery
- Reflects iterative nature of development
Unified Process
(1999)

Inception

• Encompasses communication with user + planning
• Results in a set of use cases
• Architecture is just a tentative outline

Elaboration

• Refines and expands preliminary use cases
• Provides architecture and initial design model
Construction

• Builds (or acquires) software components according to architecture
• Completes design model
• Includes implementation, unit tests, acceptance tests

Transition

• Software given to end users for beta testing
• Feedback reports defects and changes
• Support information written

Production

• Software is deployed
• Problems are monitored
**Re-Iteration**

- Feedback results in new iteration for next release

**Unified Process**

- Draws on best features of conventional process models
- Emphasizes software architecture and design
- Integrates with UML modeling techniques (more on this later)
If a traditional process is like a battleship, protected against everything that might happen…

an agile process is like a speedboat, being able to change direction very quickly

Manifesto for Agile Software Development (2001)

- Individuals and activities over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.
What is Agile Development?

- Agility = ability to react to changing situations quickly, appropriately, and effectively.
- notice changes early
- initiate action promptly
- create a feasible and effective alternative plan quickly
- reorient work and resources quickly and effectively

Incremental Model
Agile Processes

Credits: Prof. Bodik

Agile vs. Plan-driven

Agile
• Low criticality
• Senior developers
• Requirements change very often
• Small number of developers
• Culture that thrives on chaos

Plan-driven
• High criticality
• Junior developers
• Requirements don’t change too often
• Large number of developers
• Culture that demands order

What is an Agile Process?

• Difficult to predict which requirements will persist or change in the future.
• For many types of software, design and development are interleaved.
• Analysis, design, construction, and testing are not as predictable.
So, how to tackle unpredictability?

make the process adaptable...

Extreme Programming
(1999–)

Planning

Design

Coding

Test

Software Increment

Planning

• In XP, planning takes place by means of stories
• Each story captures essential behavior

Software Increment
- Design is made on the fly, using the KISS (keep it simple) principle
- Virtually no notation besides CRC cards (object sketches) and spike solutions (prototypes)
Coding

- Each story becomes a unit test that serves as specification
- The program is continuously refactored to have the design match the stories

Coding

- To ensure continuous review, XP mandates pair programming

Extreme Programming
Testing

Unit tests
- detect errors
- find missing functionality
- measure progress

Extreme Programming

Planning

Design

Test

Software Increment

- The resulting prototypes result in new stories

Extreme Programming is fast – with multiple deliverables per day!
Scrum

• An iterative and incremental agile software development method for managing software projects and product or application development.

• Small working teams to maximize communication, minimize overhead and maximize knowledge sharing.

• Adaptable to technical and business changes.

• Yields frequent software increments that can be inspected.

So, aren’t agile techniques just “code and fix” in disguise? Why not? (Hint: Think about explicit requirements, and explicit quality assurance)

Scrum = iterative and incremental agile software development method for managing software projects and product or application development. In rugby, a scrum refers to the manner of restarting the game after a minor infraction.
Scrum

- Development work and the people who perform it are partitioned into clean, low coupling partitions.
- Constant testing and documentation is performed.
- Ability to declare project “done” whenever required.

**Scrum**

Demos: Demonstrate software increment to the customer for evaluation.

Backlog: A prioritized list project requirements or features that provide business value.

Sprints: Consists of work units that are required to achieve a defined backlog into a predefined time-box (usually 30 days).

Scrum Meetings: Short 15 mins. meetings held daily by the scrum team. The Scrum master leads the meeting.

Demos: Demonstrate software increment to the customer for evaluation.
Your Sprints

Top Layer (GUI)

1. Core Use Case

2. Top Layer

3. May-Haves

Bottom Layer

http://geekandpoke.typepad.com/geekandpoke/2012/05/development-cycle.html

Summary