The Software Life Cycle

Software Engineering
Andreas Zeller • Saarland University
SIMPLY EXPLAINED

SOMETHING

DEVELOPMENT PROCESS

GREAT SOFTWARE
A Software Crisis
Code and Fix
(1950–)

1. Build first version
2. Modify until client is satisfied
3. Operate
4. 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
- **Planning**: estimating, scheduling, tracking
- **Modeling**: analysis, design
- **Construction**: code, test
- **Deployment**: delivery, support, feedback
6.6 Map Series Tool

**Use Case Description**

<table>
<thead>
<tr>
<th>Summary</th>
<th>User generates one or more maps from a series of maps for a given boundary feature (compartment, landscape etc).</th>
</tr>
</thead>
<tbody>
<tr>
<td>Actors</td>
<td>EIMS User</td>
</tr>
<tr>
<td>Pre-Conditions</td>
<td>User requires one or more maps sheets from a series, for a boundary feature.</td>
</tr>
<tr>
<td>Post-Conditions</td>
<td>Map or series of maps is generated and printed</td>
</tr>
<tr>
<td>Priority</td>
<td>Required</td>
</tr>
</tbody>
</table>

**Scenario**

1. User starts the tool.

   System displays a list of map series that the user can select from. Default map series will be 'Landscape 1:7920'. Can be set at any scale.

2. User selects map series on form.

   System then determines if any boundary features are selected.
   **A. Features Selected:**
   i. If features are selected, it asks the user if they want to generate a map series for the selected feature. Only one feature can be used at a time.
   **B. No Features Selected:**
   i. If no features are selected, or user opts to select the feature manually, the system prompts the user to select the district and compartment of interest from pull downs. It then zooms to that location, generates the map sheet boundaries, draws them with the map sheet names.

3. User can select individual sheets on screen, or select to print just an index map, or the entire series.

   System starts generating and printing maps based on the selected sheets.

4. User collects maps from printer

**Notes**

**Deployment**

Tool in ArcMap and in ArcGIS Server
Waterfall Model
(1968)

- Communication
  - project initiation

- Planning
  - estimating
  - scheduling
  - tracking

- Modeling
  - analysis
  - design

- Construction
  - code
  - test

- Deployment
  - delivery
  - support
  - feedback
## Planning

### Planning
- estimating
- scheduling
- tracking

### Gantt Chart

<table>
<thead>
<tr>
<th>ID</th>
<th>Task Name</th>
<th>1998</th>
<th>1999</th>
<th>2000</th>
<th>2001</th>
<th>2002</th>
<th>2003</th>
<th>2004</th>
<th>2005</th>
<th>2006</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Detectors</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
</tr>
<tr>
<td>2</td>
<td>R&amp;D and prototypes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Construction</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Installation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>DAQ system</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Architecture/Protocols</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Technology choices</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Hardware Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Production/Purchasing</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Software Development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>System Installation/Integration/test</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>LHCb startup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Control system (DCS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>UR (sub-detectors, hall infrastructure)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>Architecture/Evaluations/R&amp;D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>Interface technology recommendations</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>Interim development/Test beams/Prod,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>Final Technology/Product choice</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>Acquisition</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>Common infrastructure development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>Subdetector systems development</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>System Installation/Integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>LHCb startup</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Waterfall Model
(1968)

Communications
- project initiation

Planning
- estimating
- scheduling
- tracking

Modeling
- analysis
- design

Construction
- code
- test

Deployment
- delivery
- support
- feedback
Waterfall Model (1968)

Modeling
analysis
design
Waterfall Model
(1968)

- Communication
  - project initiation

- Planning
  - estimating
  - scheduling
  - tracking

- Modeling
  - analysis
  - design

- Construction
  - code
  - test

- Deployment
  - delivery
  - support
  - feedback
Waterfall Model

Construction

code
test
Waterfall Model
(1968)

- **Communication**
  - project initiation

- **Planning**
  - estimating
  - scheduling
  - tracking

- **Modeling**
  - analysis
  - design

- **Construction**
  - code
  - test

- **Deployment**
  - delivery
  - support
  - feedback
Deployment

QUESTIONNAIRE

Very often
Often
Sometimes
Rarely

Deployment
delivery
support
feedback
Waterfall Model
(1968)

- Communication
  - project initiation
- 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?
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?
That will be easy. We already spent 6 months to draw the maps.
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
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

- Coding
- Unit test
- Component test
- System test
- Field
Incremental Model

Increment #1
- Communication
- Planning
- Modeling
- Construction
- Deployment

Increment #2
- Communication
- Planning
- Modeling
- Construction
- Deployment

Increment #3
- Communication
- Planning
- Modeling
- Construction
- Deployment

Features

Time
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.
Prototypes

Top Layer (GUI)

Bottom Layer
Horizontal Prototype

Page Setup

Margins, Paper Size, Paper Source, Layout

Paper Size:
- Letter (8.5 x 11 in)
- Width: 8.5
- Height: 11
- Orientation:
  - Portrait
  - Landscape

Default...
OK
Cancel

Preview
Prototypes

Top Layer (GUI)

Bottom Layer
Vertical Prototype

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)

Communication
Planning
Modeling
Construction
Test
Deployment + Feedback
**Spiral Model**

- 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
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
Production

- Software is deployed
- Problems are monitored
Re-Iteration

- Feedback results in new iteration for next
Unified Process

- Draws on best features of conventional process models
- Emphasizes software architecture and design
- Integrates with UML modeling techniques (more on this later)
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
Agile?

Communications
  project initiation

Planning
  estimating
  scheduling
  tracking

Modeling
  analysis
  design

Construction
  code
  test

Deployment
  delivery
  support
  feedback
Agile Processes

Credits: Prof. Bodik
# Agile vs. Plan-driven

<table>
<thead>
<tr>
<th>Agile</th>
<th>Plan-driven</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Low criticality</td>
<td>• High criticality</td>
</tr>
<tr>
<td>• Senior developers</td>
<td>• Junior developers</td>
</tr>
<tr>
<td>• Requirements change very often</td>
<td>• Requirements don't change too often</td>
</tr>
<tr>
<td>• Small number of developers</td>
<td>• Large number of developers</td>
</tr>
<tr>
<td>• Culture that thrives on chaos</td>
<td>• Culture that demands order</td>
</tr>
</tbody>
</table>
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–)
• In XP, planning takes place by means of stories

• Each story captures essential behavior
Extreme Programming

Planning → Design

Design → Coding

Coding ← Test

Test ← Planning

Software Increment
Extreme Programming

- 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)
Extreme Programming

- Planning
- Design
- Test
- Coding
- Software Increment

Flow diagram showing the cycle of planning, design, coding, and testing in Extreme Programming.
• Each story becomes a unit test that serves as specification
• The program is continuously refactored to have the design match
To ensure continuous review, XP mandates pair
Extreme Programming

- Planning
- Design
- Test
- Coding
- Software Increment

Flow:
- Planning → Design
- Test → Coding
- Coding → Test
- Software Increment → Planning
Testing

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

- The resulting prototypes result in new stories.
Extreme Programming

- Planning
- Design
- Software Increment
- Coding
- Testing
- Planning
Spot the Difference

Extreme Programming
- Planning
- Design
- Test
- Coding
- Software Increment

Code and Fix
- Build first version
- Modify until client is satisfied
- Operate
- Retirement

(1950–)
Scrum
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.
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

**Scrum**: 15 minute daily meeting. Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?

**Sprint Backlog**: Feature(s) assigned to sprint

**Backlog items expanded by team**

**New functionality is demonstrated at end of sprint**

**Product Backlog**: Prioritized product features desired by the customer

**30 days**

**every 24 hours**
Scrum

**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

Bottom Layer

2. Top Layer

3. May-Haves
SIMPLY EXPLAINED

DEVELOPMENT CYCLE:
1. Think
2. Copy & Paste
3. Google
4. Run
Summary