The Software Development Pendulum

Those who cannot remember the past are condemned to repeat it.

George Santayana
1960’s
The Software Crisis

- Airline Reservation Systems
  - Earliest “Real Time” Systems
  - Grandiose marketing vision
    - Software complexity overwhelmed projects
    - Hardware response time fell short
- IBM TSS/360 Operating System
  - Announced in 1965 – canceled in 1971
- Multics Operating System (MIT, GE, Bell Labs)
  - Led to Unix when Bell Labs pulled out (1969)
- Large Software Projects
  - Late, over budget, full of bugs
  - [This hasn’t changed….]
- 1968: NATO Conference on Software Engineering
1960’s
Apollo

“I believe that this nation should commit itself to achieving the goal, before this decade is out, of landing a man on the moon and returning him safely to the Earth.”

John F. Kennedy – 1961

“That's one small step for man, one giant leap for mankind.”

Neil Armstrong – 1969
1960’s

Apollo Guidance Computer

- Instructions wired into computer
- Two digit commands

- Everything was an experiment
- Extensive preparation for failures and recovery
1970’s Software Engineering

Structure
- Structured Analysis and Design
- Top-Down Programming
- Provably Correct Code
- Sequential Life Cycle

Foundational Ideas
- “Go-to Considered Harmful” – Dijkstra 1968
- Waterfall [doesn’t work] – Royce 1970
- Information Hiding – Parnas 1971
- “Mythical Man Month” – Brooks 1975

Large Software Projects
- Still late, over budget, and full of bugs
  [Despite Software Engineering....]
- But small projects improve
1970’s

ARPANET

- DARPA (Defense Advanced Research Projects Agency)
  - Charter: ‘high-risk, high-gain’ research
- Intergalactic Network – J.C.R. Licklider
  - A decade of protocol experimentation
- E-mail is born
  - A decade of header wars
- The ’70’s also brought us
  - Microprocessors
  - Smalltalk & C
  - The mouse
  - The Apple Computer
  - WordStar
  - VisiCalc
1980’s
User Programming

- 4th Generation Languages
  - Report Generators
- Personal Computers
  - Word Processors
  - Spreadsheets
- Sequential life-cycle loses favor
  - “Life Cycle Concept Considered Harmful”
    Daniel McCracken & Michael Jackson 1982
  - Prototyping Replaces Specifying
  - Spiral Life-Cycle
    Barry Boehm 1988

- Large Software Projects
  - Still late, over budget, and full of bugs
    [No surprise…]
  - Most projects are small and do well
1980’s

Personal Computers

- 1982: Personal Computer is Time’s “Man of the Year”
  - Affordable computers enter the home
  - Endless possibilities ahead
- Software products gain stature
  - Microsoft fortunes rise as IBM fades
  - Surge of software start-up companies
- Computers are used for control
  - Microprocessors control hardware
  - Computers schedule and track activities
- ARPANET grows up
  - TCP/IP & DNS established
  - E-mail & ftp are standardized
  - Spread outside the US
  - MILNET split off, NSF takes over
  - Commercial use not allowed
1990’s Rigorous Process

- High Profile Software Failures
  - Therac-25 (1985-1987)
  - Denver Baggage Handling System (1993)
  - Long list of expensive, canceled projects

- CMM
  - Levels of Process Maturity
  - “Do It Right the First Time.”
  - Focused on custom development

- Y2K Consumes Corporate IT

- Large Software Projects
  - Still late, over budget, and full of bugs
  - But Software Products fare better
    - Early release of beta software
    - Frequent upgrades to add features
1990’s

Internet

  - Hypertext invented at CERN
  - to deal with constant change

- Internet is born (1990)
  - ARPANET shut down
  - NSF permits commercial use

- Mosaic – The first Browser (1992)
  - Developed at NCSA in Champaign Urbana
  - Modified the CERN hypertext approach

- The Internet Explodes
  - The dot-com boom
  - Rapid software development
2000’s Consumer Technologies

- Technology moves from home to office
  - Discussion Lists, Blogs, Social Networks
  - IM, VoIP (Skype, GoogleTalk)
  - Search, Desktop Search, Streaming Video
  - Handheld Devices, Memory Sticks
  - Mashups (Combining Multiple Sites)
  - Hosted Solutions (Sales, Supply Chain)

- Agile Software Development
  - Individuals and interactions over processes and tools
  - Working software over comprehensive documentation
  - Customer collaboration over contract negotiation
  - Responding to change over following a plan

- Large Software Projects
  - Still late, over budget, and full of bugs
    [Why is this not a surprise?]...
  - But they have fallen out of favor
Breakthrough Innovations in Software Development

- Apollo Guidance Computer
  - Vision, leadership, team skill, & dedication
- ARPANET
  - Robustness: distributed, independent agents
- E-mail
  - Standards evolve through broad-based discussion
- PC Software
  - Great software focuses on customers & evolves over time
- Internet Browser
  - Designed to accommodate constant change
- Search / Auction / Shopping / News / Blogs / VoIP
  - Easy to use, simple, and free!
Why Agile?

- Brilliant ideas emerge over time.
  - They need the time and space to develop.
  - They come from interactions of engaged people.

- Sequential processes don’t work.
  - Great development requires learning cycles.
  - Predictability is compatible with learning.

- Big software projects will always be:
  - Late, over budget, and full of bugs.
  - Small projects are much safer.

- Incremental delivery gives superior financial results.
  - Lower investment, earlier breakeven, more profit.
  - Potential for competitive advantage.
A Financial Model

Cost

Investment

Payback

Profit

Self-Funding

Breakeven

Time

Software by Number by Mark Denne and Jane Cleland-Huang
Staged Releases

<table>
<thead>
<tr>
<th>Time</th>
<th>Cost</th>
<th>Total Profit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Release 1 Profit</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Release 2 Profit</td>
</tr>
</tbody>
</table>
Increased Profit

Software by Number by Mark Denne and Jane Cleland-Huang
Lean Software Development

- **Just-in-Time Flow**
  - Deliver feature sets – rapidly – in small increments
  - Delay decisions to the Last Responsible Moment
  - Pay down technical debt, including regression deficit

- **Stop-the-Line Quality**
  - Mistake-proof with test harnesses
  - Synchronize early, synchronize often
  - Establish architectural vision and development standards

- **Respect for People**
  - Create a self-directing work environment
  - Engage everyone on the team
  - Provide leadership

- **Global Optimization**
  - Focus on the Whole Product
  - Measure UP
Thank You!

More Information:  www.poppendieck.com