software development processes

Daniel Jackson

For now, all rights reserved. Daniel Jackson, 2013.
One of the planning documents for software research revealed --in a parenthetical remark only-- an unchallenged tacit assumption by referring to "the tradeoff between cost and quality". Now in all sorts of mechanical engineering it may make sense to talk about "the tradeoff between cost and quality", in software development this is absolute nonsense, because poor quality is the major contributor to the soaring costs of software development.

—Dijkstra, EWD690
The management question, therefore, is not whether to build a pilot system and throw it away. You will do that. The only question is whether to plan in advance to build a throwaway, or to promise to deliver the throwaway to customers. Seen this way, the answer is much clearer. Delivering that throwaway to customers buys time, but it does so only at the cost of agony for the user, distraction for the builders while they do the redesign, and a bad reputation for the product that the best redesign will find hard to live down.

Hence plan to throw one away; you will, anyhow.

—Fred Brooks, Mythical Man Month, 1975
the waterfall model, 1970

what Royce actually said

I believe in this concept, but the implementation described above is risky and invites failure.

Figure 3. Hopefully, the iterative interaction between the various phases is confined to successive steps.
what happens in practice

Figure 4. Unfortunately, for the process illustrated, the design iterations are never confined to the successive steps.
Royce’s fixes

program design comes first
› do some design between requirements and analysis phases

document the design
› how much? “my own view is quite a lot”

do it twice
› “If the computer program in question is being developed for the first time, arrange matters so that the version finally delivered to the customer for operational deployment is actually the second version insofar as critical design/operations areas are concerned”

plan, control and monitor testing
› with a separate testing team

involve the customer
› “in a formal way, committed... at earlier points before final delivery”
spiral model, 1986

Origins of iterative approaches

- Plan-do-study-act
  - Shewart, 1930s
  - Deming, 1940s

- Project Mercury
  - NASA, 1960s
  - Half-day iterations
  - Tests before each iteration
  - Became IBM Federal Systems Division

V model

tests developed in early phases, applied in later phases
extreme programming

Kent Beck, 1999
› take best practices to “extreme” levels
› developed during C3 project with Ron Jeffries

a sample of XP practices
› test first: acceptance and unit tests
› continuous integration
› pair programming
› repeated refactoring

Chrysler’s C3 payroll system
› started in 1996, cancelled in 2000
› implemented in Smalltalk
› running payroll took 1000 hours initially
› Chrysler said they abandoned XP after this
Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.
agile approaches

agile manifesto (2001)
› an articulation of common practices
› a reaction to traditional notions

rejected notions
› upfront design (“BDUF”)
› written documentation (“ceremonial”)
› planning for future modifications

key practices like XP
› continuous integration, test first, refactoring
› features added incrementally (“sprints” and “scrums”)