Agile Programming

Principles

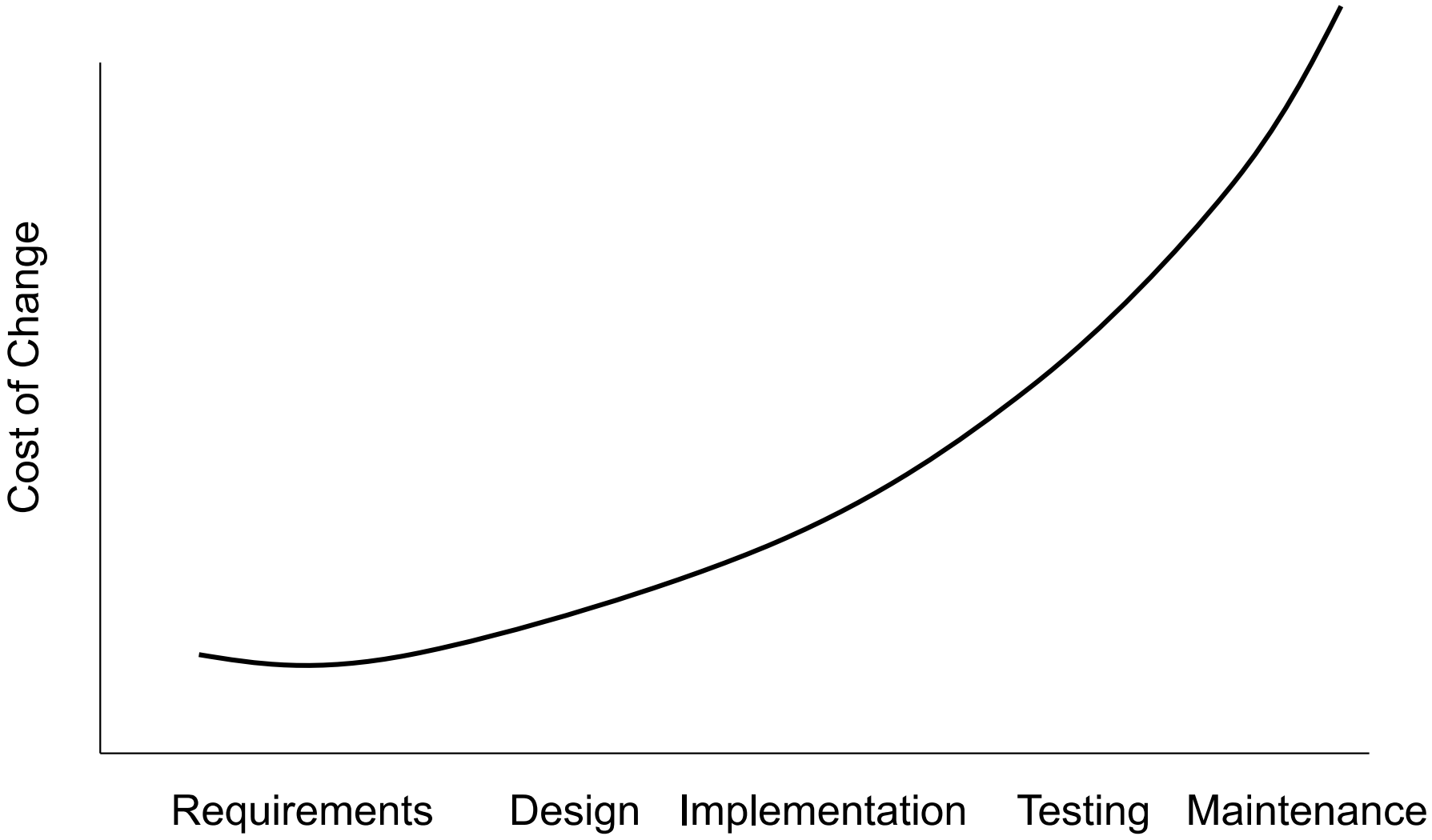
What's wrong with software today?

- Software development is risky and difficult to manage
- Customers are often dissatisfied with the development process
- Programmers are also dissatisfied

One Alternative: Agile Development Methodologies

- Variants
  - XP
  - Agile Programming
  - SCRUM
  - Lean Software Development
- Alternative to “heavy-weight” software development models
  (which tend to avoid change and customers)
  - "Extreme Programming turns the conventional software process sideways. Rather than planning, analyzing, and designing for the far-flung future, XP programmers do all of these activities a little at a time throughout development.”
    -- IEEE Computer , October 1999

Traditional Processes are ‘Heavy’



Boehm's Curve

- To accomplish this:
  - We need lots of up front planning, resulting in "heavy" methodologies
  - Every bug caught early saves money, since models are easier to modify than code
  - Large investments are made in up front analysis and design models, because the of the cost of late error discovery
  - This leads to a waterfall mentality with BDUF (Big Design Up Front)
- Proponents of Agile argue that logic is based on development in the 1970's and 1980's

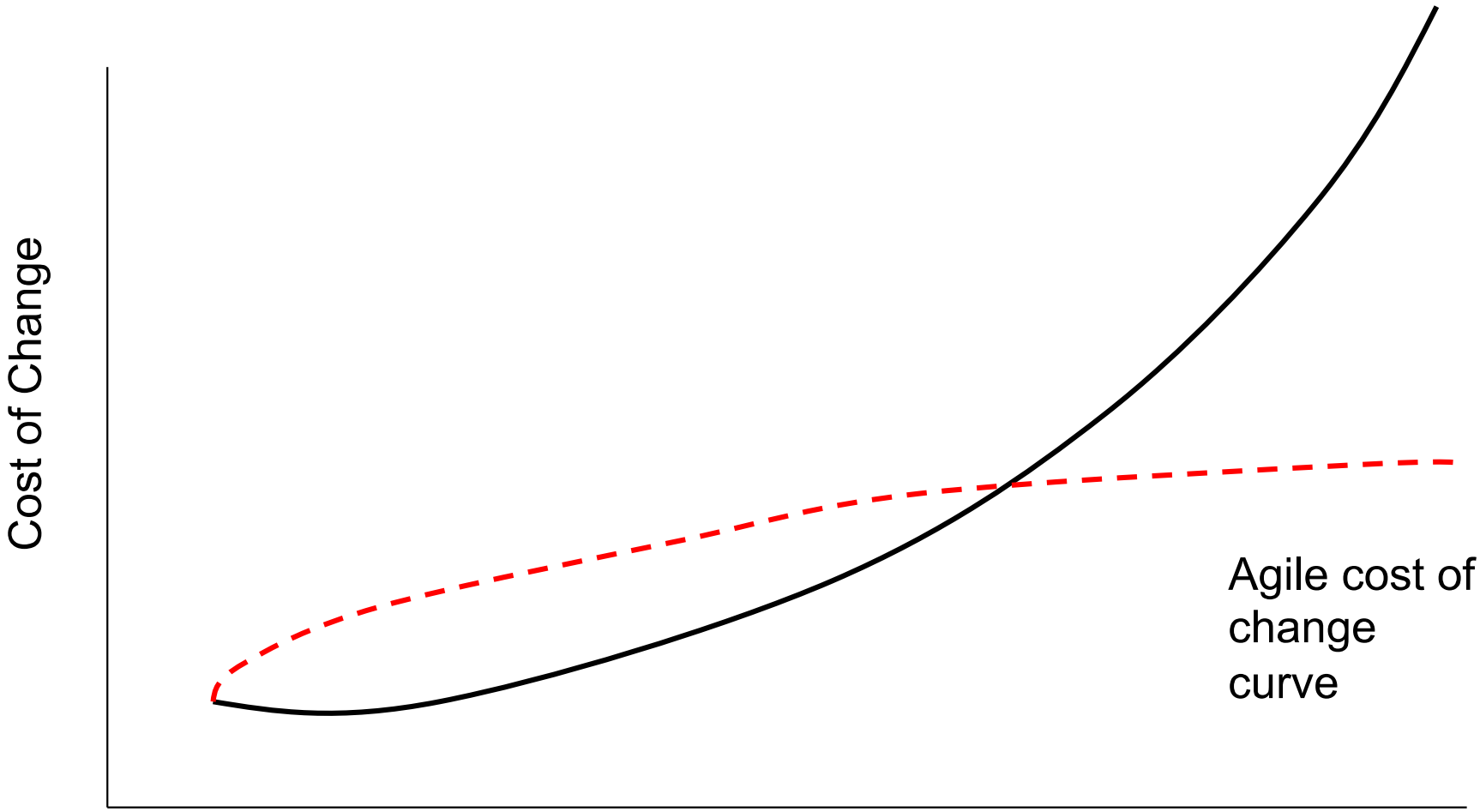
What's Changed?

- Computing power has increased astronomically
- New tools have dramatically reduced the compile/test cycle
- Used properly, OO languages make software much easier to change
- The cost curve is significantly flattened, i.e. costs don't increase dramatically with time
- Up front modeling becomes a liability – some speculative work will certainly be wrong, especially in a business environment

Why Agile Helps

- Agile Programming is a “light” process that creates and then exploits a flattened cost curve
- Agile is People-oriented rather than process oriented, explicitly trying to work with human nature rather than against it
- Agile Practices flatten the cost of change curve.

Cost of Change Curve



Embrace change

- In traditional software life cycle models, the cost of changing a program rises exponentially over time
- A key assumption of Agile Programming is that the cost of changing a program can be hold mostly constant over time
- Hence Agile Programming is a lightweight process:
  - Instead of lots of documentation nailing down what customer wants up front, Agile emphasizes plenty of feedback
  - Embrace change: iterate often, design and redesign, code and test frequently, keep the customer involved
  - Deliver software to the customer in short (2 week) iterations
  - Eliminate defects early, thus reducing costs

Why does Agile Help?

- “Software development is too hard to spend time on things that don't matter. So, what really matters? Listening, Testing, Coding, and Designing.” - Kent Beck, “father” of Extreme Programming
- Promotes incremental development with minimal up-front design
- Results in a “pay as you go” process, rather than a high up-front investment
- Delivers highest business value first
- Provides the option to cut and run through frequent releases that are thoroughly tested

The Agile Manifesto

- 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.

- http://agilemanifesto.org/

Twelve Agile Principles

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.

Twelve Agile Principles

5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Twelve Agile Principles

9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.