What is Software?

- Software
  - A set of items or objects that form a "configuration" that includes
    - Programs
    - Documents
    - Data

Software contd.

- Engineered
- Doesn’t wear out
- Is complex
- A differentiator
- Like an ‘aging factory’

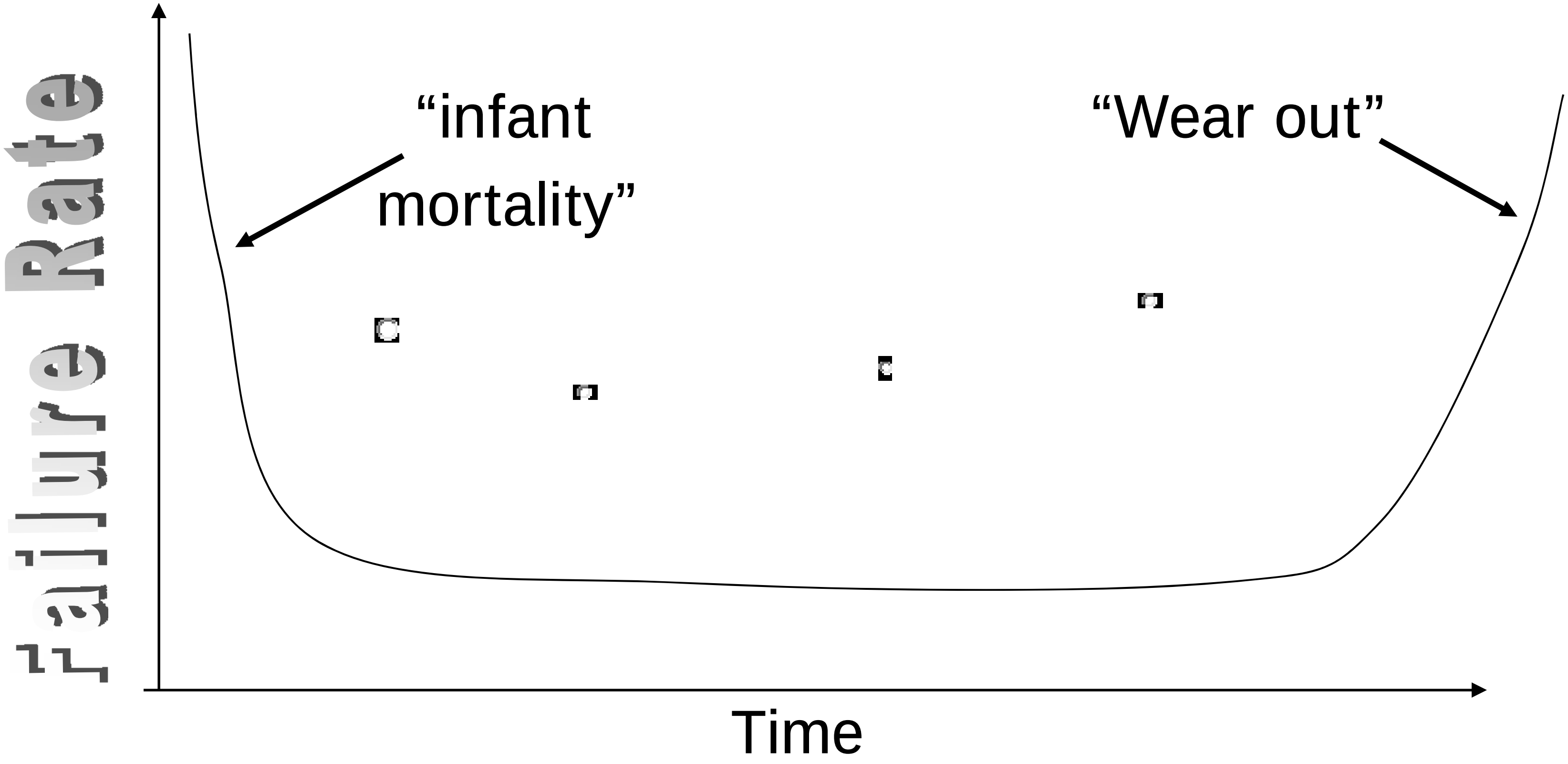
Software

- Programs – underlying source code, O.S. etc
- Documents – describe the programs, its use, limitations, installation, etc
- Data – Generation of data, type of data (range, output, storage, etc)
- Data as s.e. – drives application programs
- Data used to test application
    - Must be created, managed, documented

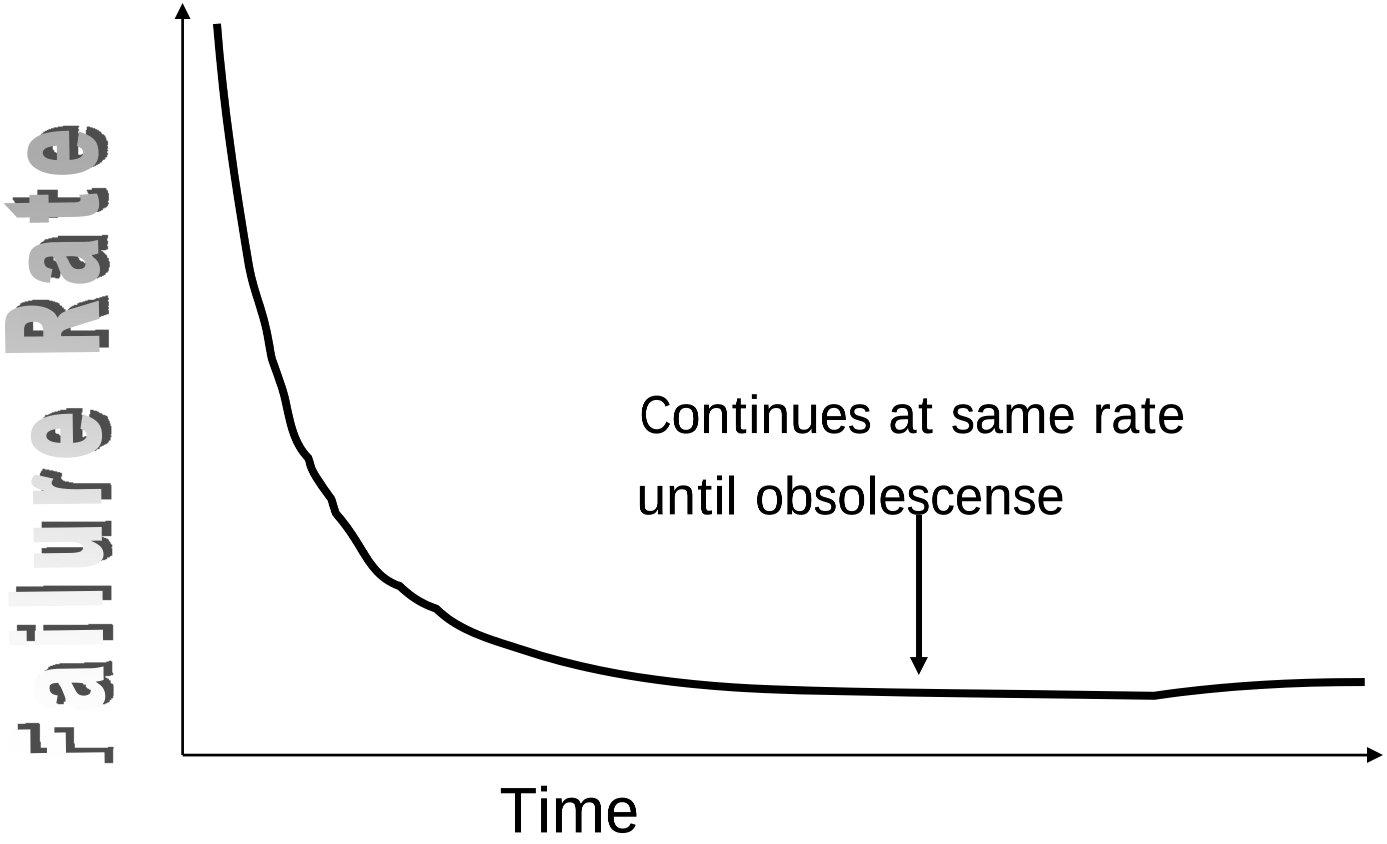
Software Failure Rate

- A function of time for hardware
  - “bathtub” curve
  - Hardware exhibits high failure rate early in its life
  - Corrected defects - failure rate drops
    - over time failure rate increases again → Cumulative effects so, wears out
  - Software does not suffer from the same maladies

Bathtub Curve

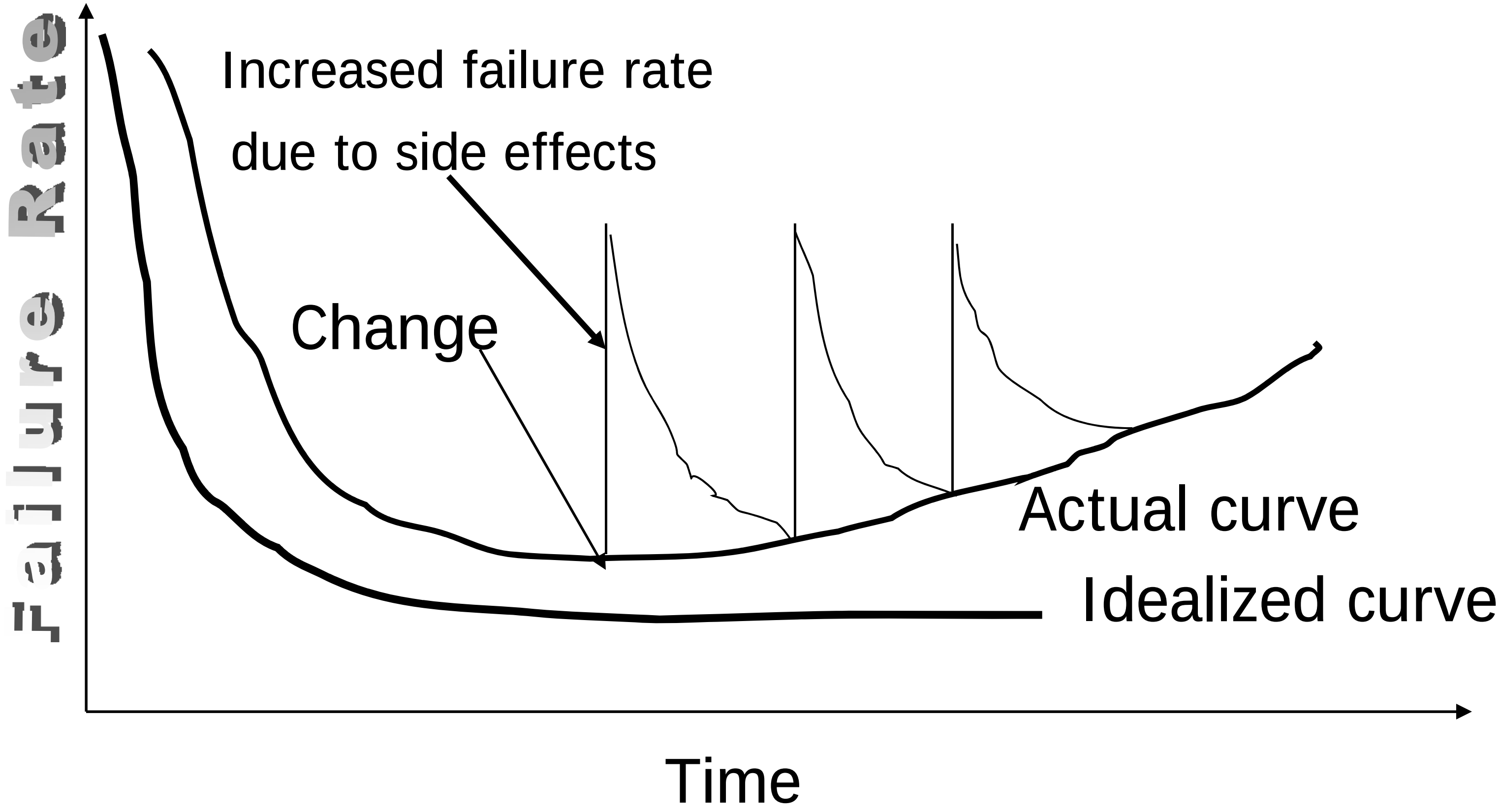


Idealized Failure Curve

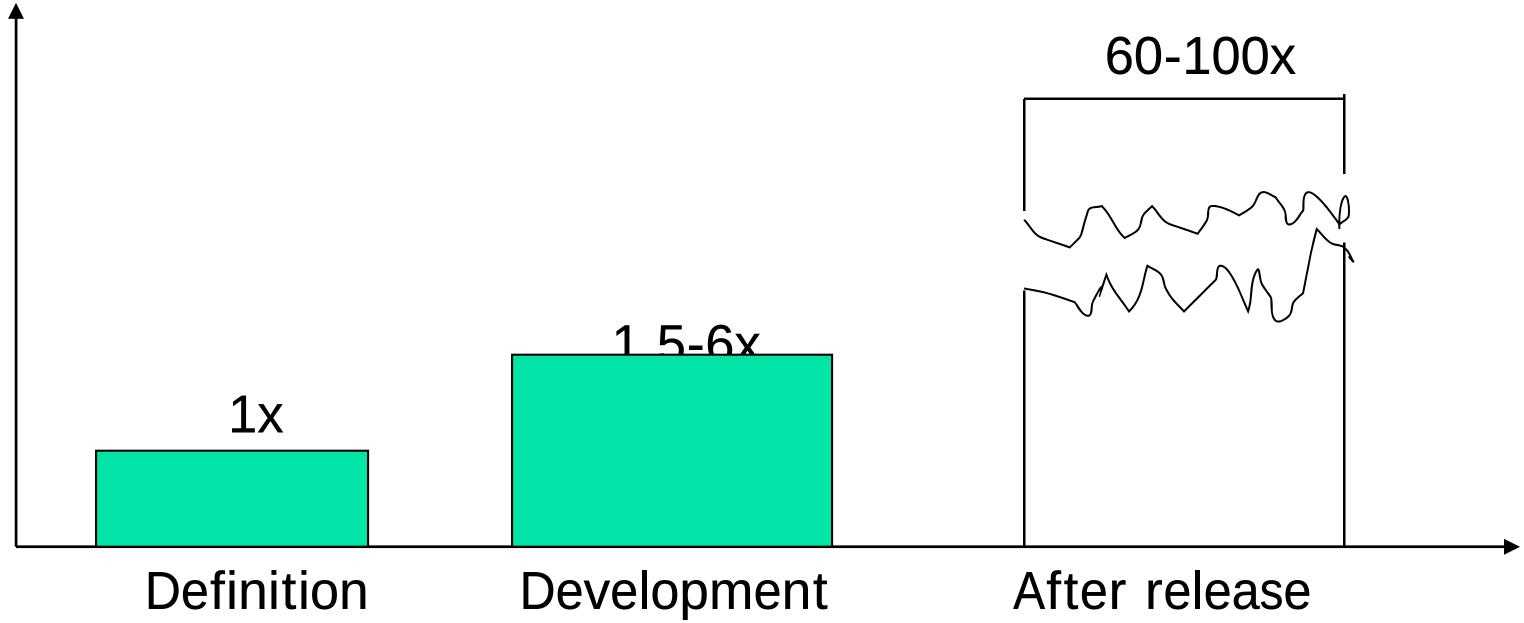


Continues at same rate until obsolescense

Actual Failure Rate



The Cost of Change



- Software costs are concentrated in engineering
- Hardware costs are concentrated in manufacturing
- Our focus for software should thus be in engineering

State of the Art of S.E.

- Software development projects have low productivity
- Software products are often full of faults and do not meet users’ needs

Nature of Software Development

- IS literature filled with examples of
  - Failed projects
  - Projects exceeding deadlines and budgets
  - Projects with faulty solutions
  - unmaintainable systems
- Standish Group report (1998):
  - 3 out of 4 software project failed in one or more of the above areas

Extent of the Problem

- Study on numerous government software projects
  - 2 percent worked upon delivery
  - 3 percent worked after some correction
  - Over 45 percent were used but either extensively reworked or abandoned
  - The remaining 30 percent of the projects were paid for but never delivered
- How bad is this????

A Better Perspective

- Findings of 9 DOD software development contracts totaling $6.8m:
  - Delivered but never successfully used, $3.3m
  - Paid for but not delivered, $1.95m
  - Delivered and used, but had to be extensively reworked or abandoned, $1.3m
  - Used as delivered, $119,000

Reasons for State of Affairs

- Managers in charge of projects who are graduates of top universities -never heard of S.E. let alone s.e. principles
- Development team members who have never heard of software engineering
- Paramount questions:
  - What causes software projects to fail?
  - What're the symptoms of project problems and what's the treatment?

Project Failure

- Answer lies in understanding the nature of software development
- Brooks (1987) identified the essence and accidents of s.e.
  - Difficulties inherent in software itself
- Consequence of the inherent complexity, conformity, changeability, invisibility

Essential Difficulties

- Essential difficulties of software define a software development invariant
- Invariant
  - Software - a product of a creative act of development (a craft or an art)
  - Software - not a result of a repetitive act of manufacturing

Categories of Accidental Difficulties

1. Stakeholders
2. Process
3. Modeling language and tools

Software Development Invariant

- Software is not manufactured
- Algorithms, code libraries, classes, software components, etc are incomplete solutions when developing software systems
- The Challenge:
  - Putting together pieces of the problem into a coherent enterprise system that meets the needs of complex processes

Software Characteristics

- High quality dependent on good design
- Hardware manufacturing phase can introduce quality problems that's non-existent in software
- Both activities depends on people
- Increased output achieved in hardware manufacturing by adding people
  - Not true for software – adding more people increase the need for communication

Software Poses Challenges

- How do we ensure the quality of the software that we produce?
- How do we meet growing demand and still maintain budget control?
- How do we upgrade an aging “software plant”?
- How do we avoid disastrous time delays?
- How do we successfully institute new software technologies?

Solution to Poor State of S.E.

- Better education about
  - the problems of s.e. and
  - the best tools and techniques to solve them
- Not adequately addressed by academic and industrial courses in programming and s.e.

Observation (Frakes et al 1991)

- Widespread lack of knowledge about s.e. problems and bad practices
- Managers with no background in s.e. responsible for technical work in major software projects
- Employees
  - with little s.e. experience responsible for difficult technical tasks, portions of software systems
  - with inadequate technical training and guidance

Observation contd.

- Graduates of computer science programs at major universities who have never heard of s.e., let alone the tools and techniques for producing high-quality software products.