CE/CS/SE 3354
Software Engineering

Introduction

What is software engineering

Software engineering is the establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines

[by Prof. Fritz Bauer at the 1968 NATO conference on software technology, in Garmisch, Germany]

- In short, software engineering is about developing quality software in a productive way

Software Engineering vs. Civil Engineering -- Similarities

- Size matters
- Teamwork with careful planning
- Leverage components
- Penalties for failures
- Sharing terms: building, architecture, components, …

Software Engineering vs. Civil Engineering -- Differences

- Much Harder to predict the behavior of the software
  - Physics laws guide civil engineering, no such laws for software
  - Software systems are more complex (incomputable)
  - Complex features so that user behaviors are unpredictable

    Consider a bridge vs. a notepad program (edit, find/replace, open, save, …)
- Need to consider the evolution of software
  - Software is easier to change
  - But it is still expensive to change

The Facts

- Only 32% of software projects are considered successful (full featured, on time, on budget)
- Software failures cost the US economy $59.5 billion dollars every year [NIST 2002 Report]
- Blame can be partly passed to:
  - The engineer
  - The manager
  - The customers

Engineer's fault

- Let's just write the code, so that we will be done sooner
  - Writing code sooner may cause it take longer to finish
  - 80% of effort is spent after the first delivery of code

- I have to finish it to assess its quality
  - Design reviews help to find severe design defects
  - Good coding style leads to fewer bugs
  - Static checker and unit testing help to find bugs earlier

Engineer's fault (Cont'd)

- There is no time for software engineering
  - It will take you more time without software engineering
  - Misunderstood requirements (may need to redo the whole thing)
  - Comprehensive design / code changes for feature changes

  Tons of Bug fixes

When do Engineers do Software Engineering

- Consider the following cases:
  - Write a text format changer for one-time usage

    (nothing)
  - Write a personal utility library

    (+design for potential change, +testing)
  - Write a notepad program to share online

    (+requirement collection, + usage documentation)
  - Collaborate on a small project with several people

    (+modeling, +API doc, +comments, +version control)
  - Work on a large project in a large company

    (+design documentation, +coding style, +code review, +static checker, +other regulations)

Manager's fault

- We add more programmers if we are late
  - Adding manpower to a late software project makes it later [Brooks' law, The mythical man-month, 1975]
- We can outsource it
  - If you do not manage it well inside, you cannot do it well outside
  - Much more communications, more risk for requirement misunderstanding
  - Impairs long-term maintenance

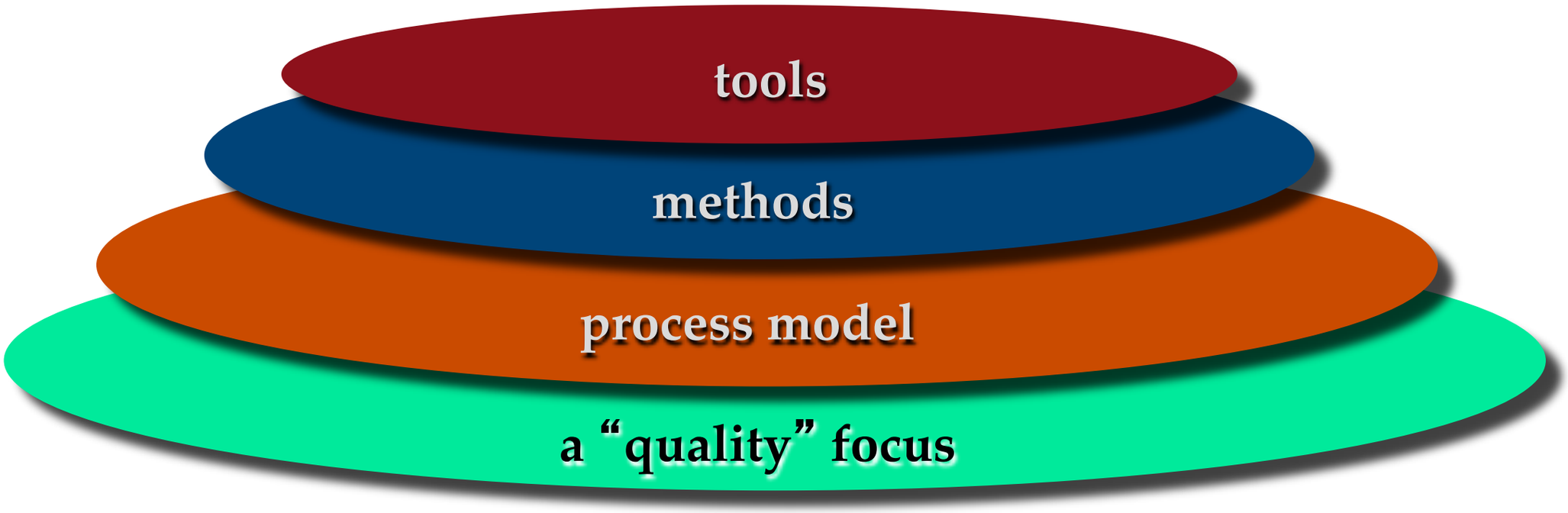
Customer's fault

- We do not need to be involved in the project
  - Customers should be involved all the time to provide requirements (requirements are always changing)

- Anyway, we can change the software later
  - Yes, but the cost goes exponentially as time goes by!

Discussion:
Why learn software engineering?

Software Engineering: A Layered View



- Tools: provide automated or semi-automated supports for the process and the methods
- Methods: provide "how to's" for building software
- Process: provides a framework for software development