Imperial College London

Software Engineering Practice

Introduction

Course Logistics

- Integrated with the MSc Group Projects
  - For details, see the group project web page and slides of introduction lecture
- Group Projects are mandatory
- Course runs during the first part of term 2

Assessment

- Two intermediary reports:
  - Specification for reports available on the group project page
  - Submitted on CATE (send a copy to your supervisor)
- Final report
- Final presentation and demo
- No exam
- Marking breakdown:
  - Report 1: 15%
  - Report 2: 15%
  - Final software product, final report and presentation: 70%

Imperial College London

Calendar: Project

End of January	Report 1 due
Beginning of March	Report 2 due
Middle of May	Final Report due
Middle of May	Final Presentations

(see CATE and project web page)

- 4 Month project
  - But don't forget your exams are in the middle

Report 1 - Content

- Project specification (“contract” between supervisor and you)
- Minimum requirements and possible extensions
- Requirements: prioritize, assess feasibility, completion times
- Specify version control system
- Identify development strategy
- 3 – 5 (max!) pages typed in LaTeX (template on CATE)

Imperial College London

Making Your Project A Success!

- Regular meetings with your supervisor
  - You should contact your supervisor this week and schedule a first meeting
- Regular meetings with your team
  - At least weekly, but probably more often

Course Syllabus

- Team communication
- Scheduling
- Introduction to Python
- Capturing requirements
- Agile methods (development strategies)
- Code management
- Software testing and profiling
- Technical writing and presentation

Software Engineering - Definition

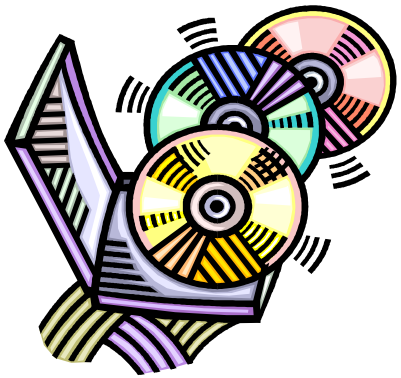
"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software." IEEE Standard Glossary of Software Engineering Terminology

Why is Software Engineering Hard?

- *Building Software vs. Building Bridges?*

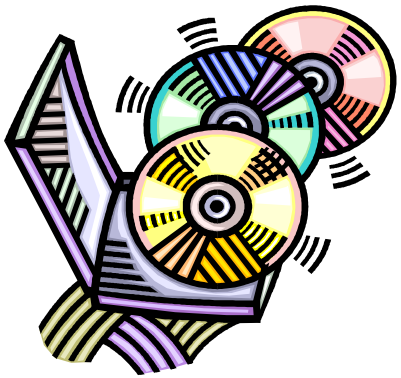
Why is Software Engineering Hard?

- High degree of novelty
  - No previous relevant experience
  - Hard to estimate costs, necessary resources, completion times



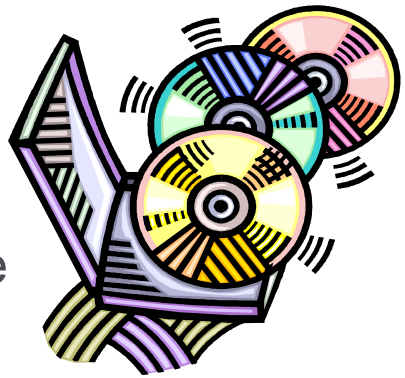
Why is Software Engineering Hard?

- High degree of change
  - Requirements often change mid-stream
    - Ability to change requirements seen as an inherent property of software
  - Big pressure to add new features



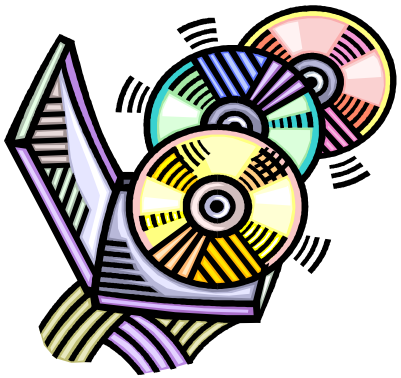
Why is Software Engineering Hard?

- High degree of complexity
  - Huge number of different pieces which interact with each other
  - Few constraints (e.g., gravity) = huge design space
  - Software expected to run in a variety of different environments (hardware, OS, etc.)



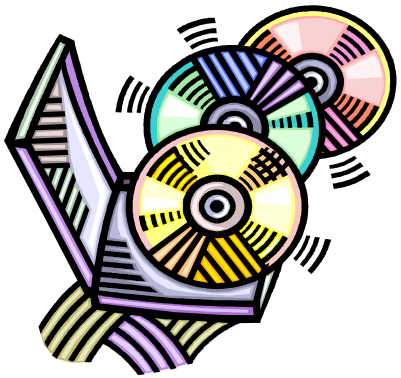
Why is Software Engineering Hard?

- Non-linear progress
  - Last 10% of work often takes 90% of the time



Why is Software Engineering Hard?

- Programming is hard
  - Often an art
  - Errors (bugs) play a disproportionate role



S/W Development: Design Activities

- Team organization
  - Management
  - Communication
- Scheduling
  - Setting effective deadlines
  - Dealing with slippage
- Code management
  - Version control systems
  - Documentation
- Process
  - Can vary depending on the project
  - Follow best-proven software industry practices