Course goals

- Learn how to scratch-build “small” (5-15 KLOC) and “medium” (15-100 KLOC) programs
  - programming → engineering
  - individual, not team
Why is SW hard?

- Little effort / difficulty in development is:
  - coding
  - algorithms
- Much is management
- Technical is reqs, design, validation, maintenance
Software Engineering

• Attempt to:
  – Adapt other fields' tech
  – Manage “accidental” difficulties
  – Ensure SW “goodness”
  – Acquire mantle of legitimacy
Engineers

- Solve problems using tech
- Optimize under constraints
- Build tech infrastructure
- Evaluate tech (theirs and others')
Engineering principles

- Control of Complexity
- Separation of Concerns
- Module Decomposition
- Problem Analysis
- Record-Keeping
- Satisficing
- Future-proofing / Reuse
Separation of concerns

- Problems are too large to solve all at once
- Split the solution across
  - domains: mgmt, mktg, eng
  - areas: coding, tech areas
  - phases: see lifecycle
Module decomposition

- Make things out of pieces (cf Niven, *The Mote In God's Eye*)
- Hierarchical $7 \pm 2$
- Narrow interfaces
- No more than necessary
Problem analysis

- Domain understanding
- Sufficient detail (may mean mathematical reasoning)
- What's the real problem? i.e., what should a success look like?
Record keeping

- “Documentation”, e.g.
  - Notebooks
  - Manuals
  - Internal docs

- Not always textual!
  - Makefiles
  - Revision history
Satisficing

- Pick an appropriate level of generality
  - Must solve problem (duh)
  - Should provide for future
  - Too much is a disaster!

- Failure will impact
  - goodness
  - schedules
Reuse

- Reuse of engineering ≠ reuse of “stuff”
- Engineers reuse
  - ideas
  - plans / methods
  - tools
- Separates eng from dev
SE principles

- Code is not stuff
- Computing is inter-domain
- Computation / DP
- Tool-driven development
- SE has advantages
Code is not stuff

- Code is magic
  - trivially replicable
  - trivially distributable
  - utterly reliable
  - reflective

- Easy to confuse code with engineering, with stuff
Computing is inter-domain

- Every interesting problem involves non-computing
- *(Thus, we mostly do non-interesting computing)*
- Engineering + art + science + domain knowledge + communication + ...
Computation vs Data Processing

- Two basic kinds of problem
  - compute something
  - slosh data around
- Most problems are one or the other, not both
- Separation of concerns!
- This course: DP
Tool-Driven Development

- SE is not done on paper
- Integration of tools, process, dev is vital
- Open source leads the way
  - build tools
  - source code management
  - collaboration
Why other engineers are jealous of us

- **Backups**: we are never more than a few minutes from the best place we've ever been
- **Replicability**: never fighting physics
- **Irresponsibility**: expectations are low
Next Steps

- Lifecycle
- “Management”
- Project