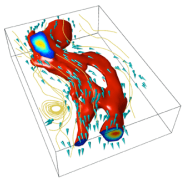
Software Design



Designer clothes?

Outline

- Announcements:
  - Homework I on web, due Wed., 5PM by e-mail
  - Starting Friday, Wed. and Fri. lectures will meet in ACCEL "Green" Room in Carpenter
- Updated Syllabus
- The Design process
- Importance of good design
- Design techniques

Updated Syllabus

1. Intro, Philosophy, Model problem
2. Design of algorithms and responsible coding
4. Editing, compiling: UNIX vs. IDE, intro to architectures
3. Formal & Informal Specification
5. Language issues: C, Fortran, Java, MATLAB
7. Debugging: UNIX db vs. IDE
8. Testing for correctness
6. Building with Make
9. Improving performance--profiling, tuning
11. Platform issues & how to spend your advisor's money
10. Software management, source code control
12. Trends for the future

General Development

- Development is the process by which things get made (e.g. engineering)
- The development process is different depending on the product
  - driven by cost, complexity, and reliability considerations

General Development

- Building a house
  - architects/engineers create detailed blueprints
  - general contractor organizes groups of workers for specific tasks
    - foundation
    - walls, windows & doors
    - electrical
    - plumbing
    - interior finishing
  - Good design is important because it is costly to rebuild (materials, time)
  - Good management is important to avoid having workers sitting idle

General Development

- Making a movie
  - screenwriter creates script
  - director plans shoots from the script
    - choose locations, organize personnel, timeline
  - movie is filmed
  - movie is edited and released
- Movies require lots of (expensive) people, so it is critical that time is used efficiently

Development Process I

1. Create a text file containing commands in some language
2. Pass the file to the compiler
3. Run the executable

Development Process II

1. Design: What will the program (or modification) do? How will it work?
2. Specification--formal statements of what code will do
3. Prototyping--a "proof-of-concept" version. Simple version written in an interpreted language (Matlab, Python)
4. Implementation: write the code
5. Build: Get it to compile and run
   a) Debug I: find and fix syntax errors
   b) Debug II: find and fix semantic errors (testing)
6. Improve performance through tuning or re-design

Development Process II (typical)

1. Start writing code, design=rewrite
2. Compile
3. Debug, debug, debug

Importance of Design in Scientific Software Development

Position	Base Salary	Hourly
undergrad	0-$3500 (summer)	$0-7.29/hr
grad	$15K/year	$8.33/hr
post. doc.	$32K/yr	$15.62/hr
assist. prof.	$60K/yr	$31.25/hr
Arnold	$30million for 24x80 hr weeks	$15,625/hr

- Despite our lowly status, we are paid for scientific results, not time spent hacking

Importance of Design in Scientific Software Development

- Even though our wages are low, good design is important for scientific programming
  - Reduces time spent debugging
  - Makes code easier to use (more people citing your work)
  - Makes code easier to extend (better luck next time)
  - Makes code easier to describe to colleagues

Definition of Design

- The design process will lead to a description of what your program will do and how it is organized.
- Some important questions to answer
  - How will you get data in and out of your program?
  - What tasks must your program perform?
  - How will data flow through your program?

Good Design

- An important characteristic of good design is modularity
  - Code should be divided into simple pieces (subroutines, method), each solving a specific task
  - Related pieces should be grouped together in a single file (module, class)
- Object oriented languages (Java, C++) are inherently modular

Design Techniques

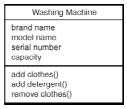
- Flow charts
  - Visual representation of your program
  - This should be at a high-level
- Universal Modeling Language (UML)
  - Industry-standard for design and management of object oriented development
  - Specifies several diagram types--each one takes a different view of a project

Stealing from UML

- Industrial UML-systems are overkill for most scientific problems, but we can borrow some useful views of our programs
  - Class Diagram--describe an object's fields (data) and methods (functions)
  - State Diagram--describe how an object's state (data) changes

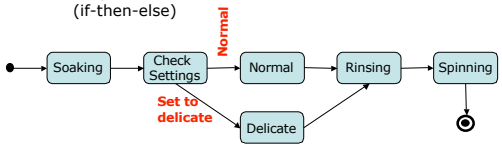
Stealing from UML

- Class Diagrams
  - Box with three regions: name, fields (data), methods (subroutines)



Stealing from UML

- State Diagrams
  - Start state (•), named states (ovals), end state (●)
  - Connect with arrows
  - Diagrams can branch when conditions are satisfied (if-then-else)



Iterative Refinement

- Iterative refinement is an important design technique
  - Takes a top-down view
  - Enforces modularity
- Iterative refinement is a 3 step process
  1. Describe--what will your program/subroutine do?
  2. Divide--what are the essential tasks?
  3. Repeat--subdivide tasks if possible.

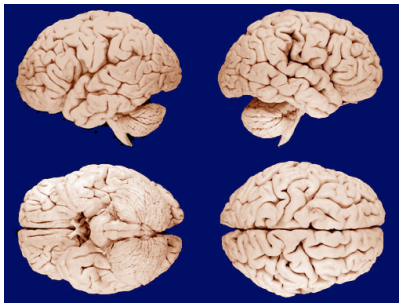
Iterative Refinement

I. Do Laundry
   A. Wash
   B. Dry
   C. Fold

Do Laundry

A. Wash
   1. Get clothes
   2. Place in washer
   3. Configure washer
   4. Start
B. Dry
   1. Move from washer to dryer
   2. Configure dryer
   3. Start
C. Fold
   1. Remove from dryer
   2. If (shirt) then
      a. Fold shirt
   else
      b. Fold pants

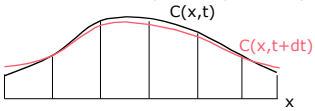
Main Point of Design



- Think before you code!

Numerical Solution

- We start with an initial distribution of C over the interval [0 1]
- Divide [0 1] into discrete points separated by dx



- C(x,t+dt) will depend on C(x), C(x-dx), & C(x+dx)
- I've placed a full derivation of the model problem on the web site which I will go through now

Designing RAD1

1. Get C0, u, k, dt,t, T, m, L (dx=L/(m-1)) from user
2. Build matrix A using k, dt, dx
3. Build RHS vector b using u, k, and reaction data
4. Solve A*C=b for C
5. t=t+dt
6. If(t<T)
   a) Copy C=C0
   b) Change u and k if needed
   c) Repeat 2-6

Else

   d) Output C and quit