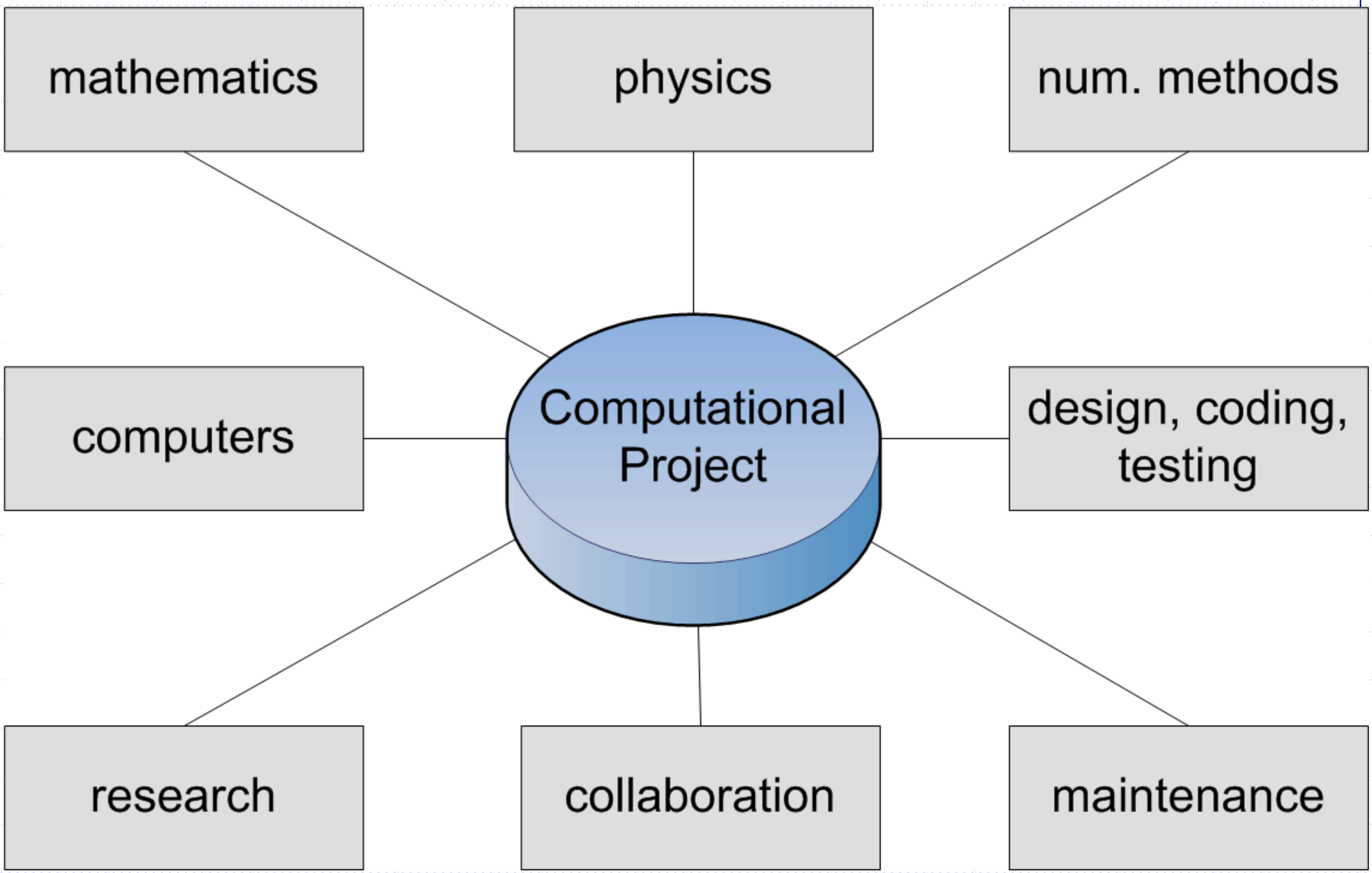
Computational Projects


mathematics
physics
num. methods
computers
Computational Project
design, coding, testing
research
collaboration
maintenance

Art and Science

Computational Physics is an art
(requires imagination and creativity)
and science
(uses specific methods and techniques)

Milestones

1. Problem definition
2. Problem analysis
3. Equations and data
4. Computational project (detailed design)
5. Numerical model(s) & libraries
6. Program coding (writing a computer code)
7. Get the code running (data flows, bugs)
8. Testing
9. Calculations and analysis of results
10. Program maintenance

Steps 1-2: Problem Solving Skill

- The most valuable quality of physics majors on job market
- Experience
- Learning

Interesting books:
How to Solve It : A New Aspect of Mathematical Method (Princeton Science Library) by G. Polya

The Art and Craft of Problem Solving by Paul Zeitz

The Thinker's Toolkit: 14 Powerful Techniques for Problem Solving by Morgan D. Jones

Techniques from The Thinker's Toolkit: by Morgan D. Jones

1. Problem restatement
2. PROs-CONs-FIXes
3. Divergent Thinking
4. Sorting, Chronologies and Timelines
5. Causal Flow Diagramming
6. Matrices
7. Decision/Event Trees
8. Weighted Ranking
9. Hypothesis Testing
10. Devil's Advocacy
11. Probability Tree
12. Utility Tree
13. Utility Matrix
14. Advanced Utility Analysis.

Techniques from http://www.mindtools.com/pages/main/newMN_TMC.htm

1. Appreciation - Extracting All Most Information From Facts
2. Drill-Down - Breaking Problems Down into Manageable Parts
3. Cause & Effect Diagrams - Identifying Likely Causes of Problems
4. Systems Diagrams - Understanding How Factors Affect Each Other
5. SWOT - Analyzing Your Strengths, Weaknesses, Opportunities & Threats
6. Cash Flow Forecasting With Spreadsheets - Analyzing Whether an Idea is Financially Viable
7. Risk Analysis
8. Porter's Five Forces - Understanding the Balance of Power in a Situation
9. PEST Analysis - Understanding "Big Picture" Forces of Change
10. Value Chain Analysis - Achieving Excellence in the Things That Matter
11. USP Analysis

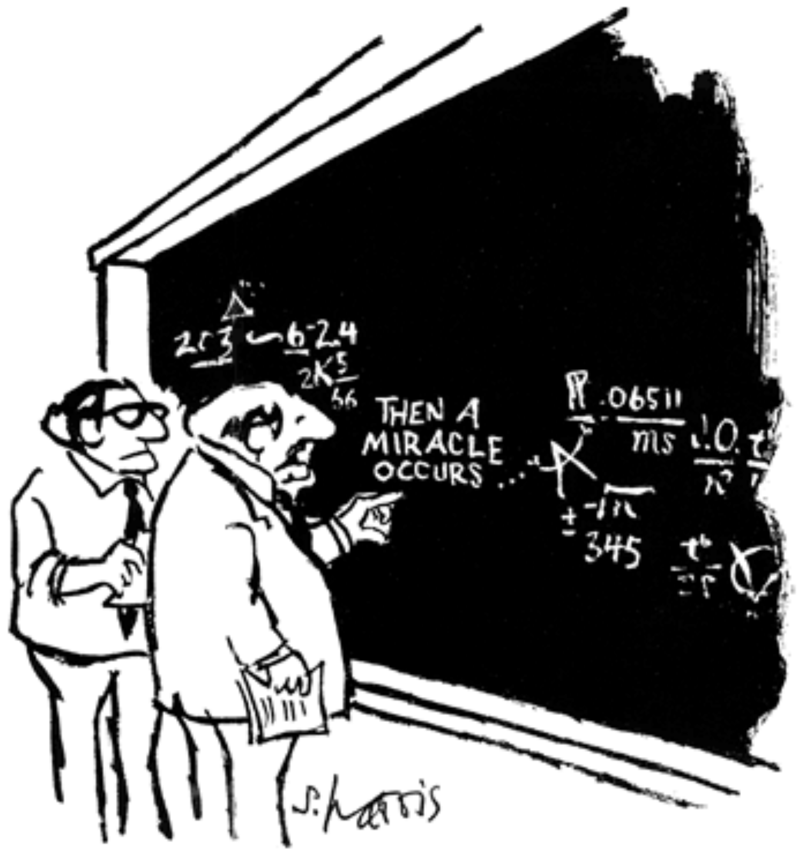
The Expert Mind

Scientific American, August 2006

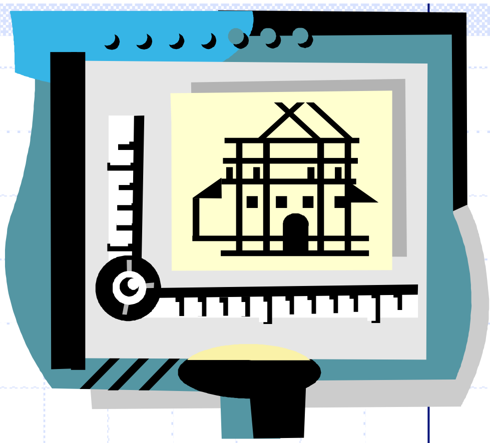
The 10-year rule states that it takes approximately a decade of heavy labor to MASTER ANY FIELD

The preponderance of psychological evidence indicates that experts are made, not born.

Step 3: Equations.



"I think you should be more explicit here in step two."



Step 4: Design.

- Top-down design
  (or hierarchical approach)
  Break a problem into a set of subtasks (called modules) until you are at the subroutine level
- Object-oriented design
  A problem-solving methodology that produces a solution to a problem in terms of self-contained entities called objects

What is better for Physics?

Step 4: Design (more)

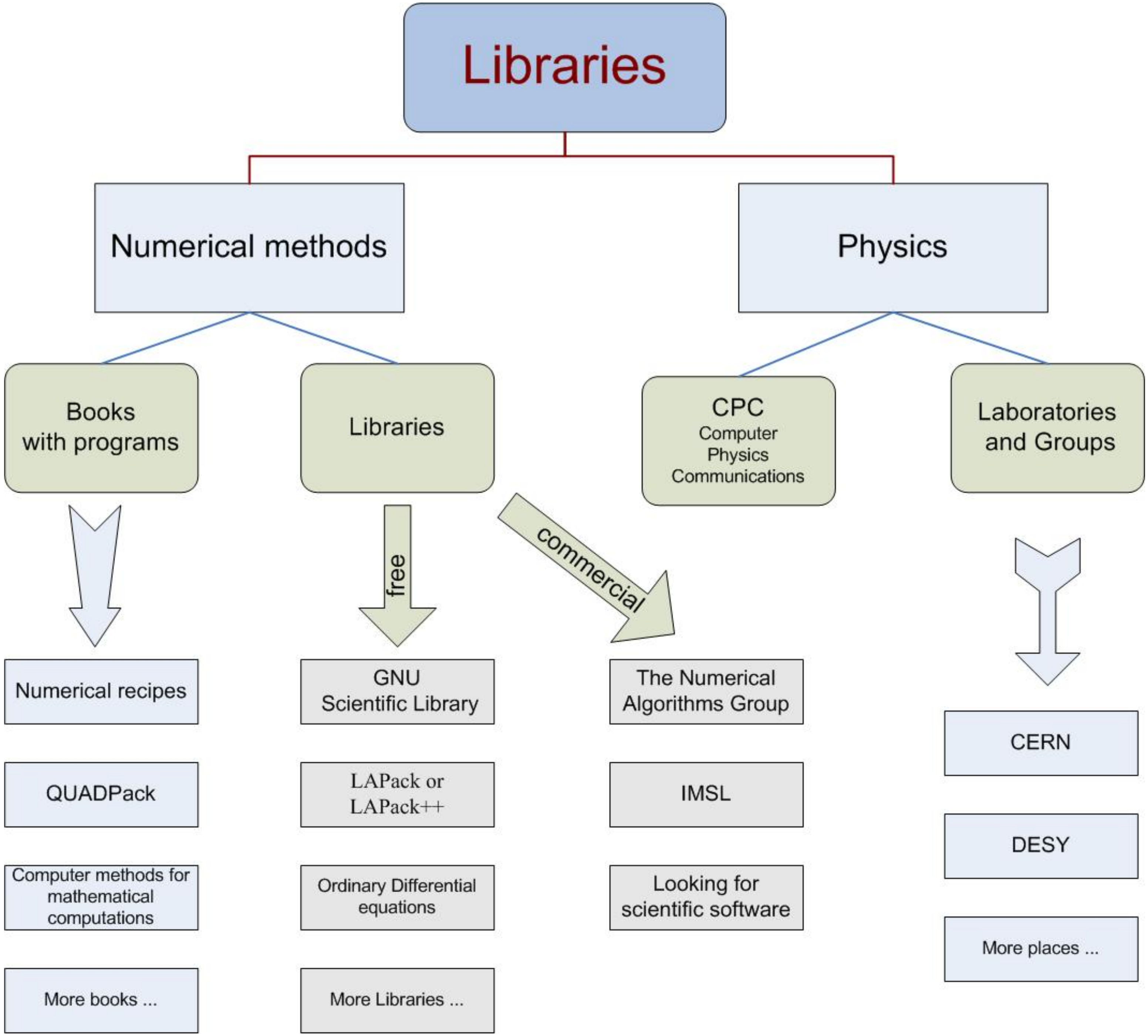
- Take into account available hardware and software and time.
- Arrange major tasks in order in which they need to be accomplished
- Draw diagrams
- For each module (subroutine)
  - well-defined input and output
  - reasonably independent from other modules
- Clear logic and data structure
- Easy to use and modify
- Protect your program from invalid inputs

Step 5: Models and libraries

- You should never reinvent the wheel.
- Computational Physics Libraries
- Numerical Libraries and Depositories

www.odu.edu/~agodunov/computing/lib_net.html

However, do not use routines as black boxes without understanding them.


Libraries
Numerical methods
Physics
Books
with programs
Libraries
CPC
Computer
Physics
Communications
Laboratories
and Groups
free
commercial
Numerical recipes
QUADPack
Computer methods for mathematical computations
More books ...
GNU
Scientific Library
LAPack or
LAPack++
Ordinary Differential equations
More Libraries ...
The Numerical
Algorithms Group
IMSL
Looking for
scientific software
CERN
DESY
More places ...

Step 6. Writing a computer code

- Stick to the design
- Use pseudocode first
- Select programming language
- Style: DOs and DON 'Ts

Step 6. Style (Good habits vs. bad habits)

- DO
  - keep it clear and simple
  - separating logic groups (structured programming)
  - internal comments and external documentation
  - linear programming
- DON’T
  - complex logic
  - tricky technique
  - computer dependent
  - avoid implementations for specific computers, i.e. steer clear of interactive or graphics-related routines.

Step 6. more about style

Each program unit should be well documented

- Opening documentation (what program does, when the program was written and modified, list of changes, …)
- Comments to explain key program sections
- Meaningful identifiers
- Labels for all output data

Step 6. And … more

A program should be readable

- Use spaces
- Use blank lines between sections
- Use alignment and indentation to stress relations between various sections of the program unit
- Labels for all output data
- Do not use “magic numbers” that appear without explanation

Programming hints

- Always keep an updated working version of your program; make modifications on a copy
- Use the standard version of the program language (easy to move to a different computer)
- Use descriptive names for variables and subroutines
- Declare ALL variables
- Do not optimize the program until you have right results

Step 7. Bugs, bugs, bugs, …

A computer bug is an error, flaw, mistake, failure, or fault in a computer program that prevents it from working correctly or produces an incorrect result.

Software horror stories

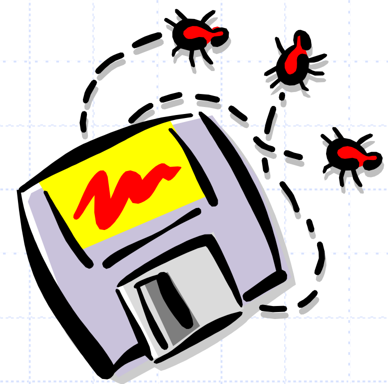
- NASA Mariner 1 went off-course during launch, due to a missing 'bar' in its FORTRAN software (July 22, 1962)
- NASA Mars Rover freezes due to too many open files in flash memory (January 21, 2004).
- The Mars Orbiter crashed in September 1999 because of a “silly mistake”: wrong units in a program
- The year 2000 problem, popularly known as the "Y2K bug"
- The MIM-104 Patriot bug - rounding error, which resulted in the deaths of 28 solders (February 25, 1991).
- August 1991 – Sleipner A oil rig collapse (large elements in the Finite Element method for solving PDE)

Ariane 5 Flight 501 the most expensive computer bugs in history (June 4, 1996)

- The Ariane 5 software reused the specifications from the Ariane 4, but the Ariane 5's flight path was considerably different and beyond the range for which the reused code had been designed.
- Because of the different flight path, a data conversion from a 64-bit floating point to 16-bit signed integer value caused a hardware exception (an arithmetic overflow).
- This led to a cascade of problems, culminating in destruction of the entire flight.
- self-destruction - 40 seconds after takeoff



Common types of computer bugs



- Divide by zero
- Infinite loops
- Arithmetic overflow or underflow
- Exceeding array bounds
- Using an uninitialized variable!
- Accessing memory not owned (Access violation)
- Deadlock
- Off by one error - a loop iterates one too many or one too few times
- Loss of precision in type conversion

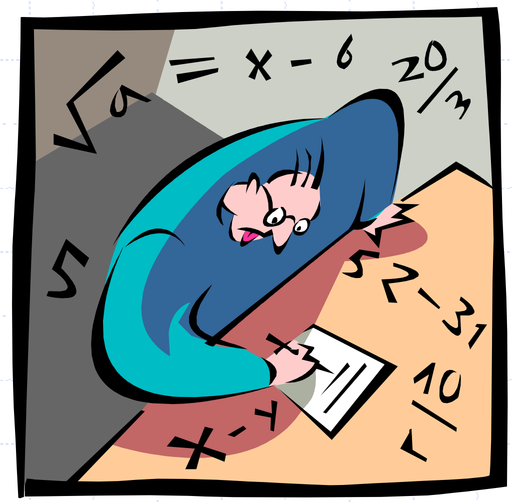
How to find it?

- Working like a detective: who did what?
- Check global logic
- Check modules
  - Local logic
  - Data
  - Operators
  - Arrays
- Debugging

Step 8. Verification & validation

Programs MUST NOT be used for research or applications until they have been validated!

- Plan ahead
- Analytical solutions
- Other calculations
- Experiment
- Trends (does it make sense?)
- Special cases



The few existing studies of error levels in scientific computer codes indicate that the defect rate is about seven faults per 1000 lines of Fortran.2 That's consistent with fault rates for other complex codes in areas as diverse as computer operating systems and real-time switching.

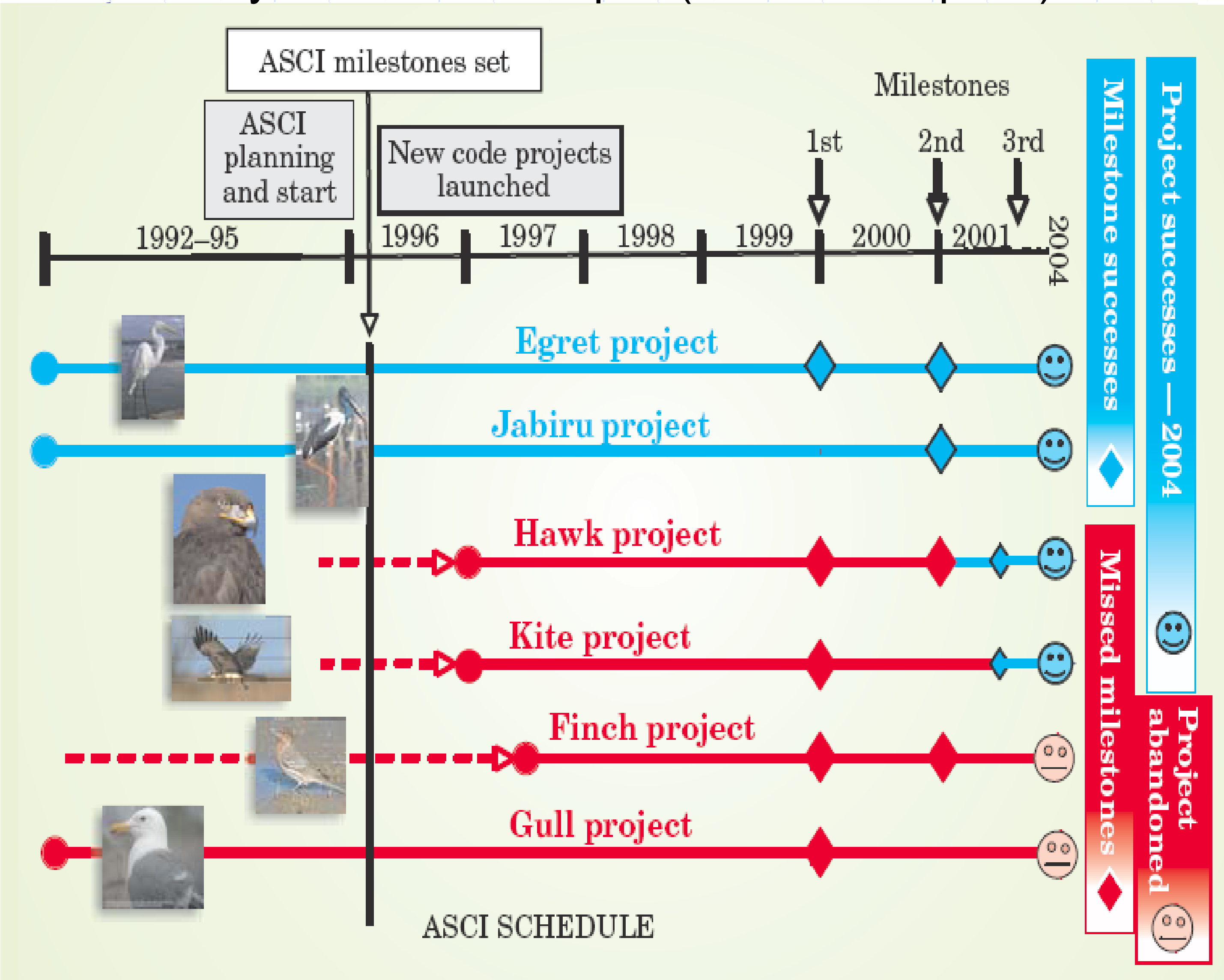
Even if a code has few faults, its models and equations could be inadequate or wrong. As theorist Robert Laughlin puts it, "One generally can't get the right answer with the wrong equations."

It's also possible that the physical data used in the code are inaccurate or have inadequate resolution.
(garbage in – garbage out)

Or perhaps someone who uses the code doesn't know how to set up and run the problem properly or how to interpret the results.

In 1996 Department OE launched Accelerated Strategic Computing Initiative (ASCI) in 1996 at the Livermore, Los Alamos, and Sandia national labs.

Aim - to develop computer infrastructure and codes that would serve to certify the reliability of the US stockpile (nuclear weapons) in the absence of testing.



it takes about eight years for a staff of 15–30 to develop a massively parallel three-dimensional weapons simulation.

Five common verification techniques

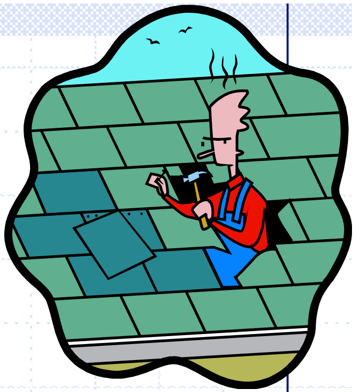
One must first verify and validate each component, and then do the same for progressively larger ensembles of interacting components until the entire code has been verified and validated.

- Comparing code results to a related problem with an exact answer
- Establishing that the convergence rate of the truncation error with changing grid spacing is consistent with expectations
- Comparing calculated with expected results for a problem specially manufactured to test the code
- Monitoring conserved quantities and parameters, preservation of symmetry properties, and other easily predictable outcomes
- Benchmarking—that is, comparing results with those from existing codes that can calculate similar problems.

Steps 9: Calculations and analysis

- Plan ahead what to calculate
- Keep records
  - Date
  - Version
  - Changes to the code
- Graphics
- Analyze results
- … and be ready to revise you project from any step

Step 10: Program maintenance



Real programs are often used for years.

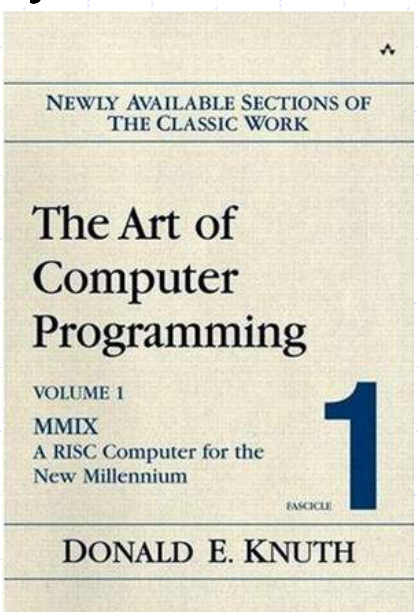
- Large projects may have obscure bugs that were not detected during testing
- Program may require to improve performance
- Program may need new features
- …

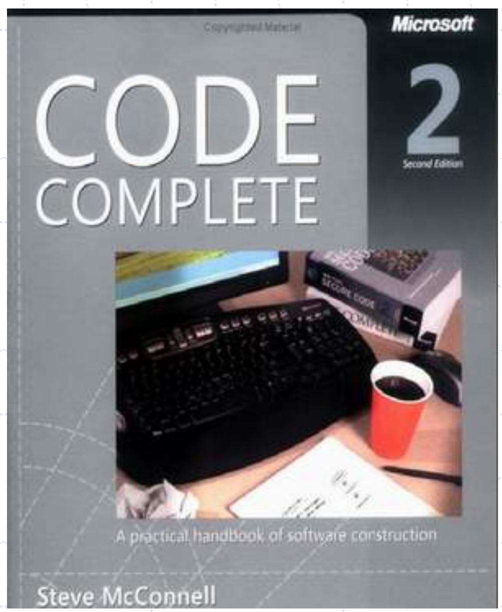
Average distribution of efforts

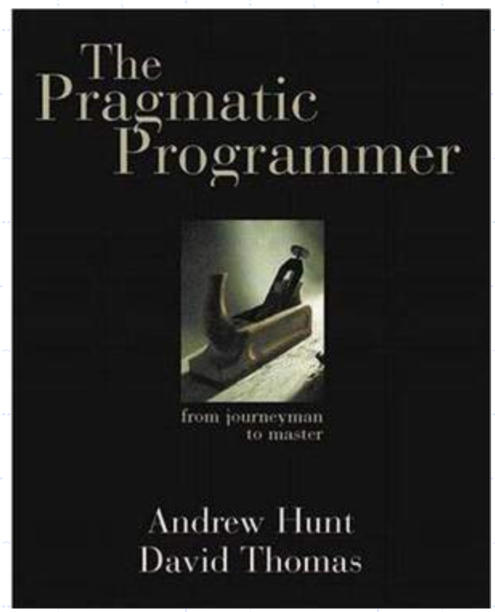
Computational project (design)	20%
Numerical model(s) & libraries	5%
Pseudo code	15%
Coding (using some language)	10%
Get the code running (data flows, bugs)	15%
Testing	30%
Documentation	5%

Books

- The art of scientific programming by Donald Knuth
- Code Complete, (Second Edition)
  by Steve McConnell
- The Pragmatic Programmer: From Journeyman to Master
  by Andrew Hunt, David Thomas







A poor project v.s. a good project