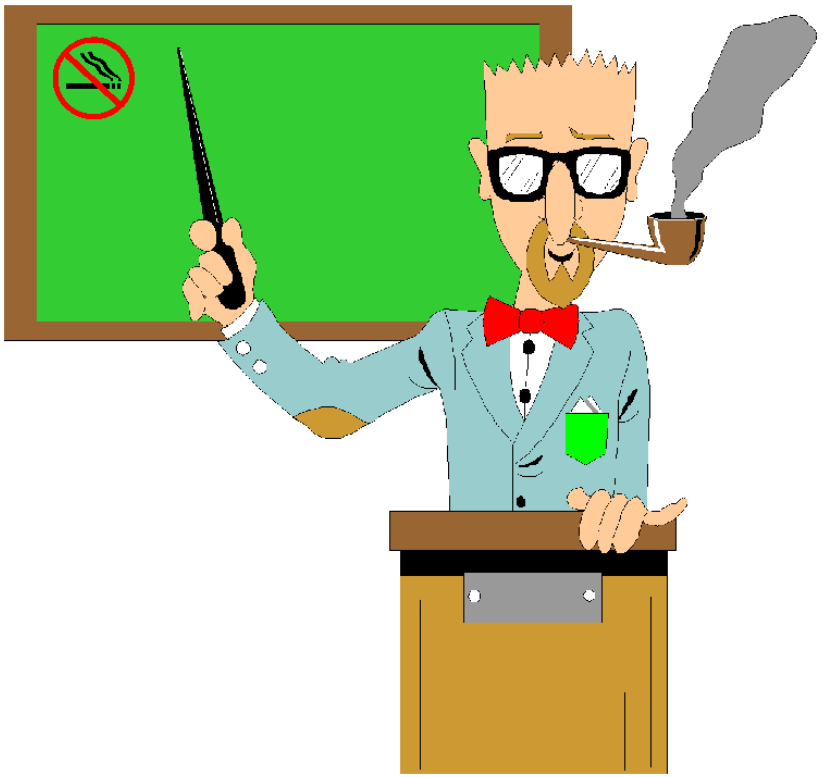
TEN GOLDEN RULES FOR TEACHING COMPUTER SCIENCE

Andrew S. Tanenbaum
Dept. of Computer Science
Vrije Universiteit
Amsterdam, The Netherlands
http://www.cs.vu.nl/~ast/

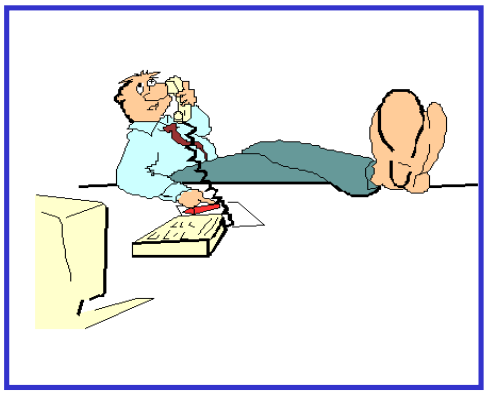


GOLDEN RULE #1:

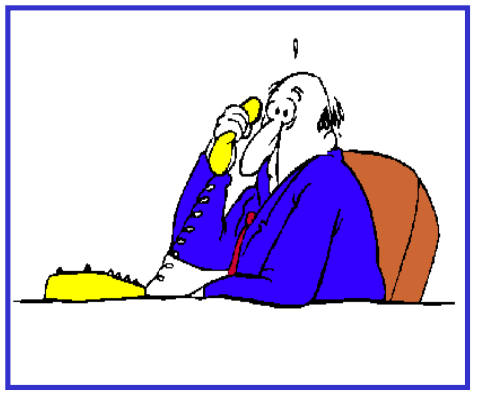
THINK LONG TERM

CURRENT STUDENTS MAY WORK UNTIL 2040

2000

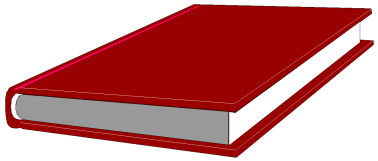


2020



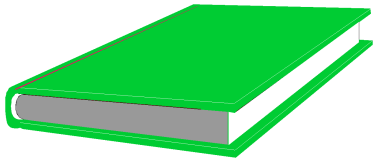
2040

COMPARISON OF TWO ELDERLY OS BOOKS



Per Brinch Hansen

1. Overview of Oper. Sys.
2. Sequential Processes
3. Concurrent Processes
4. Process Management
5. Store Management
6. Scheduling Algorithms
7. Resource Protection
8. A Case Study: RC4000

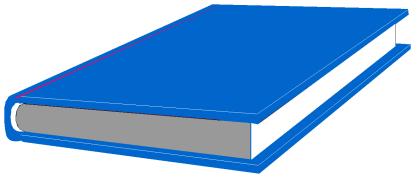


William S. Davis

1-4. Introductory Material
5. Single Program Systems
6. Multiprogramming
7. Job Control on the 360
8. The JOB and EXEC cards
9. The DD card
10. Function of an Op. Sys.
11. Principles of the 360
12. IBM 360 Disk Oper. Sys.
13. System 360 MFT

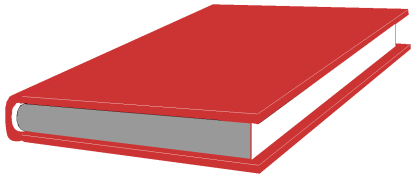
...

COMPARISON OF 2 OLDER ARCHITECTURE BOOKS



John P. Hayes

1. The evolution of computers
2. Design Methodology (register level)
3. Processor Design (instructions, arithmetic)
4. Control Design (sequencing, microcode)
5. Memory Organization (virtual memory, caching)
6. System Organization (I/O, communication)



Ivan Flores

1. Introduction
2. The Channel Controller
3. Interrupts
4. System 360 Interrupts
5. The PDP-8
6. SDS-92, SCC 650
7. IBM 1401
8. Honeywell 200
9. System 360
10. Spectra 70
11. Univac 9000

THE YEAR 2000 PROBLEM

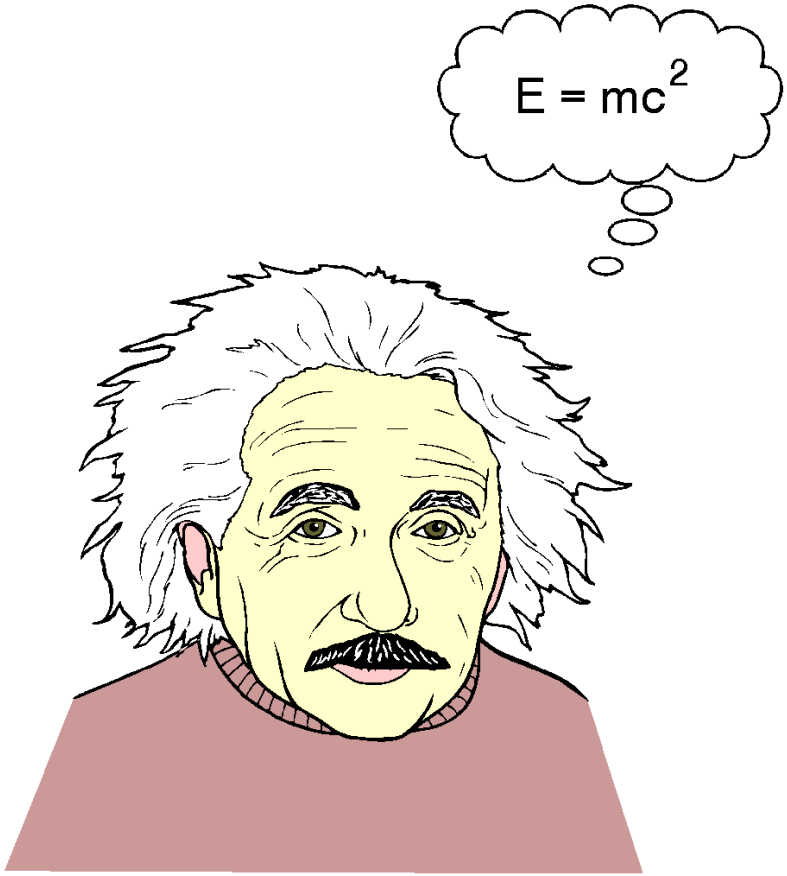
Jane P. Smith, born in 00

Jane P. Smith, born in 00

- In the 1970s and 1980s, COBOL programmers used two decimal digits to represent the year
- Get it right this time, or the mess in 9999 will be unbelievable (8000 years of old COBOL to fix)

GOLDEN RULE #2:

EMPHASIZE PRINCIPLES, NOT FACTS



SOME EXAMPLE PRINCIPLES

- Iteration vs. recursion
- Compilation vs. interpretation
- Caching
- Use of hints
- Hashing
- Locality in space and time
- Delayed binding

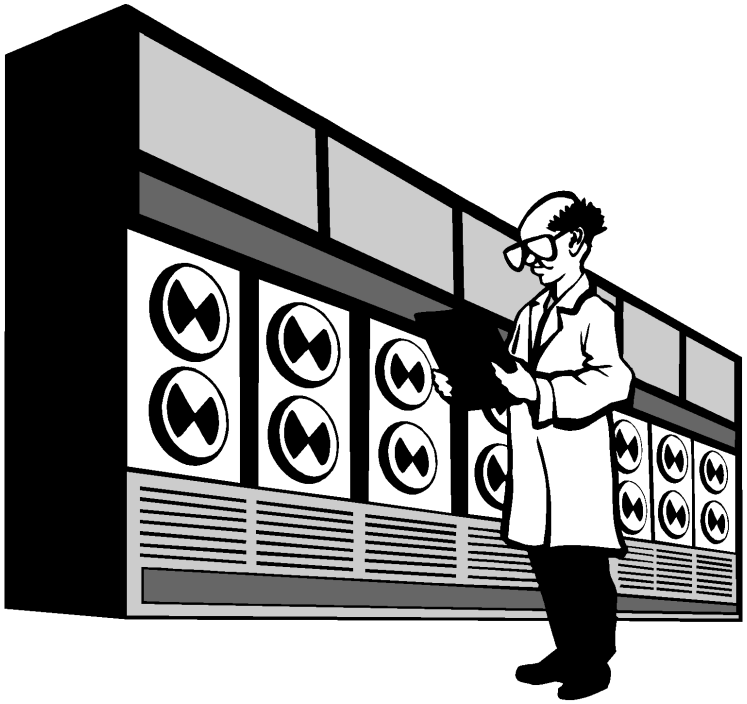
BUT: Illustrate each principle with at least *two* examples

PRINCIPLES FOR A FEW SELECTED COURSES

Course	Some principles
Architecture	Data paths, memory hierarchies, buses
Compilers	Grammars, parsing, code generation
Networks	Layering, protocols, routing
Operating systems	IPC, memory management, file system
Programming langs	Paradigms, data types, syntax, semantics

GOLDEN RULE #3

EXPECT PARADIGM SHIFTS



1965

1995

EXAMPLES OF PARADIGM SHIFTS

- Assembly language to high-level languages
- Batch systems to timesharing to personal computers
- Spaghetti programming to structured programming
- Imperative programming to object-based programming
- Text-based interfaces to icon-based interfaces
- Isolated machines to LANs
- Local computing to the Internet
- Computers for computing vs. communication

HOW DO YOU DEAL WITH CONSTANT CHANGE?

Teach the students to

- Be critical

- Learn on their own

- Constantly examine their own assumptions

GOLDEN RULE #4:

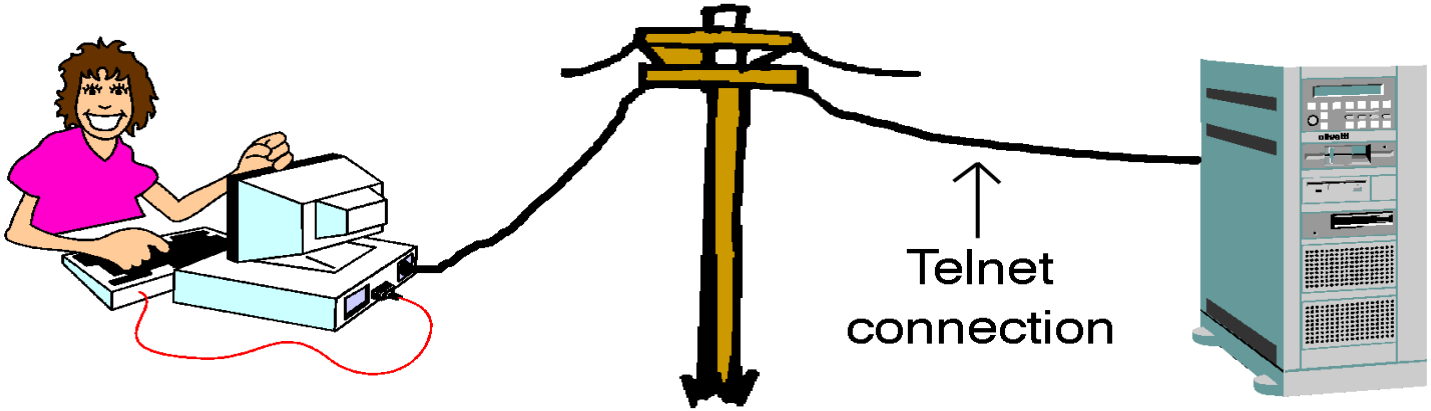
EXPLAIN HOW THINGS WORK INSIDE

EXAMPLE: THE WORLD WIDE WEB

To many students, operation of the web is a mystery.

Try typing:

telnet www.cs.*univ*.edu



to establish a TCP connection to the server. Then type:

GET *filename* HTTP/1.0

Presto! A web page appears as an HTML file. No more mystery.

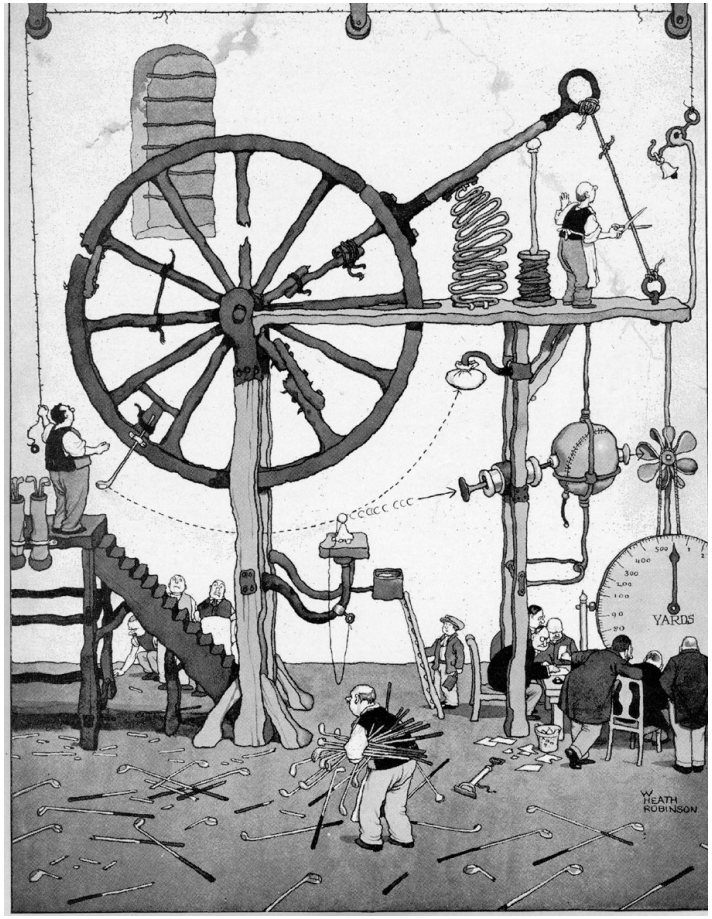
SIMULATORS GIVE STUDENTS INSIGHT

Examples where simulators can be useful

- CPU cache behavior as a function of size and strategy
- Performance of network protocols when packets get lost
- Paging algorithms in a virtual memory system

GOLDEN RULE #5:

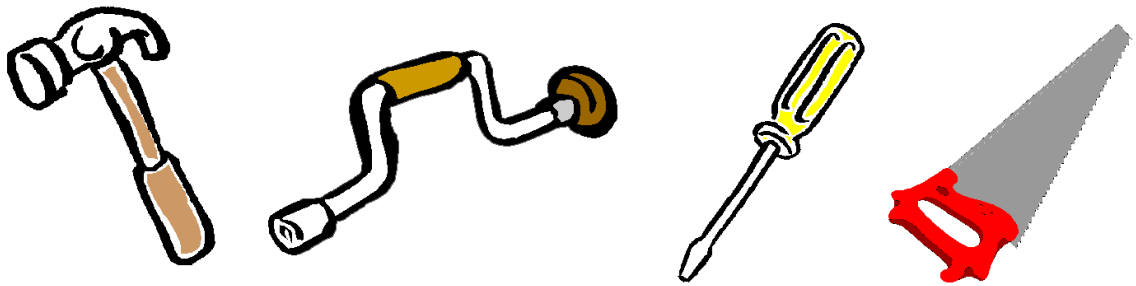
SHOW STUDENTS HOW TO MASTER COMPLEXITY



PREVENT IT IN THE FIRST PLACE

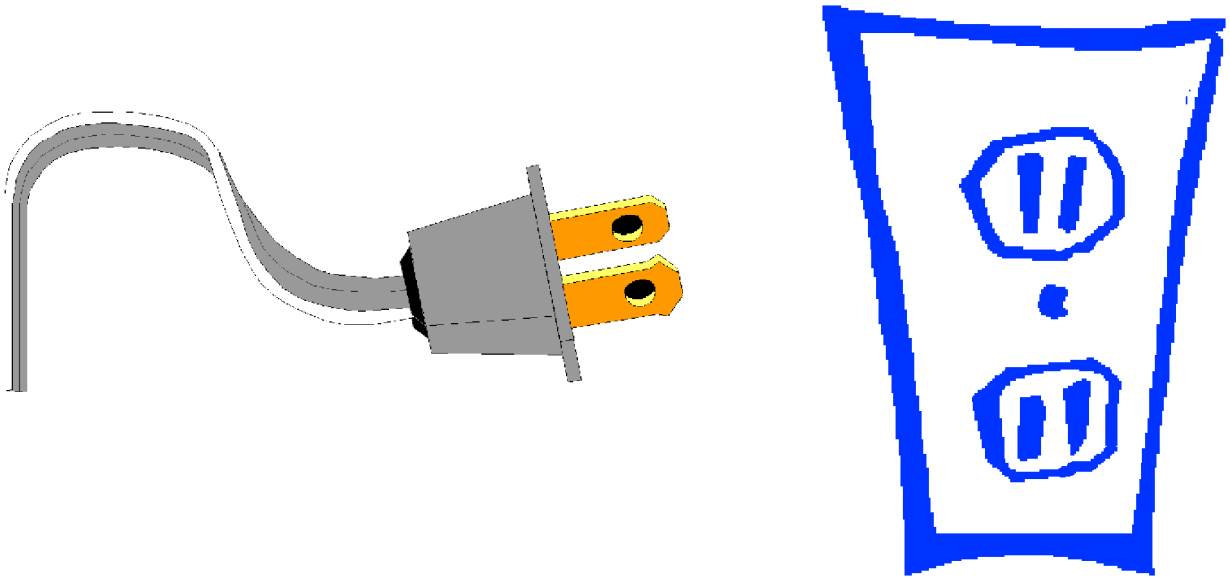
KISS: KEEP IT SIMPLE, STUPID

TOOLS FOR MASTERING COMPLEXITY



- Abstraction
- Information hiding (ADTs, modules, objects)
- Hierarchies and layering
- Separation of data from their description

INTERFACE DESIGN IS THE KEY ELEMENT



- Interfaces are contracts between implementers and users
- They should be designed very carefully to be stable in time
- Interfaces should be minimal but complete

ON PERFECTION

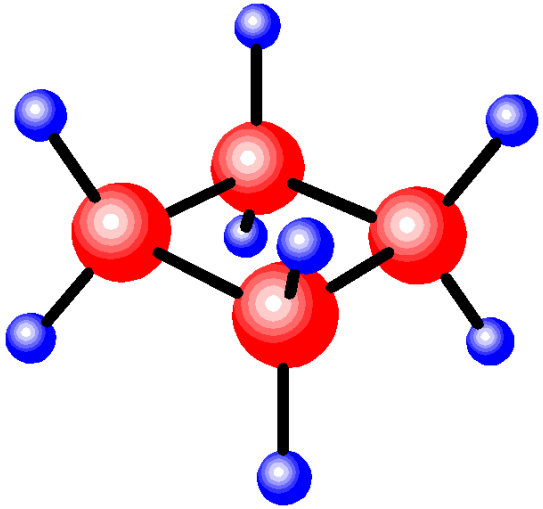
Perfection is reached not when there is no longer anything to add, but when there is no longer anything to take away

- Antoine de Saint Exupery

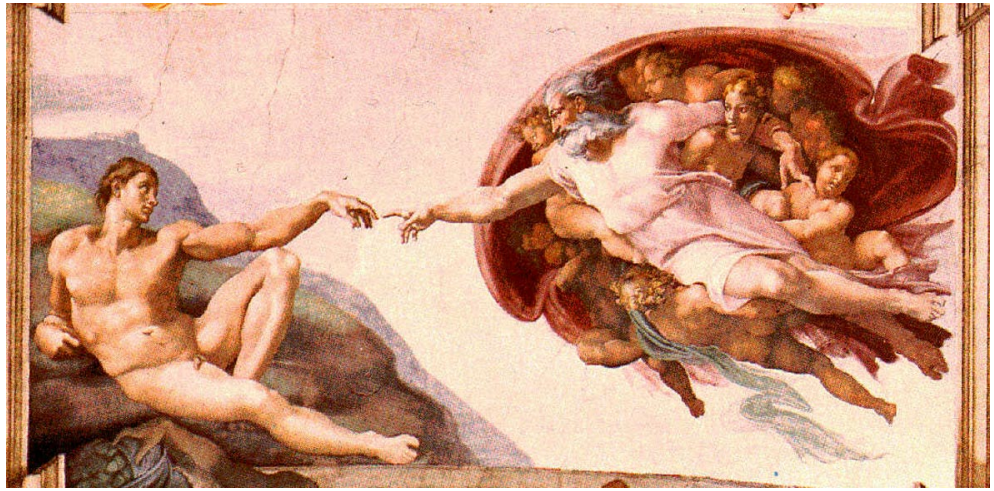
GOLDEN RULE #6:

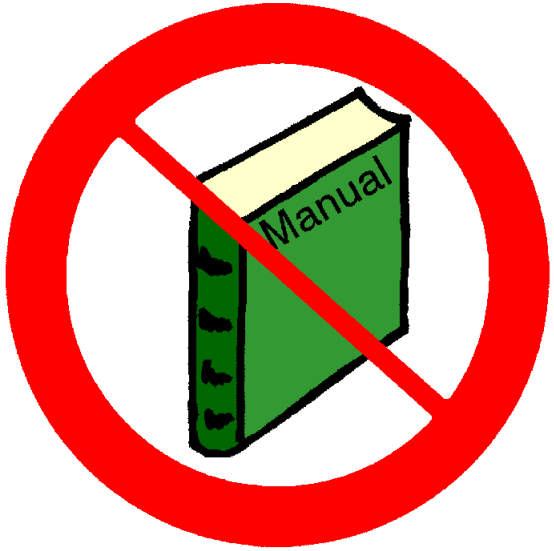
COMPUTER SCIENCE IS NOT SCIENCE

$\neq$



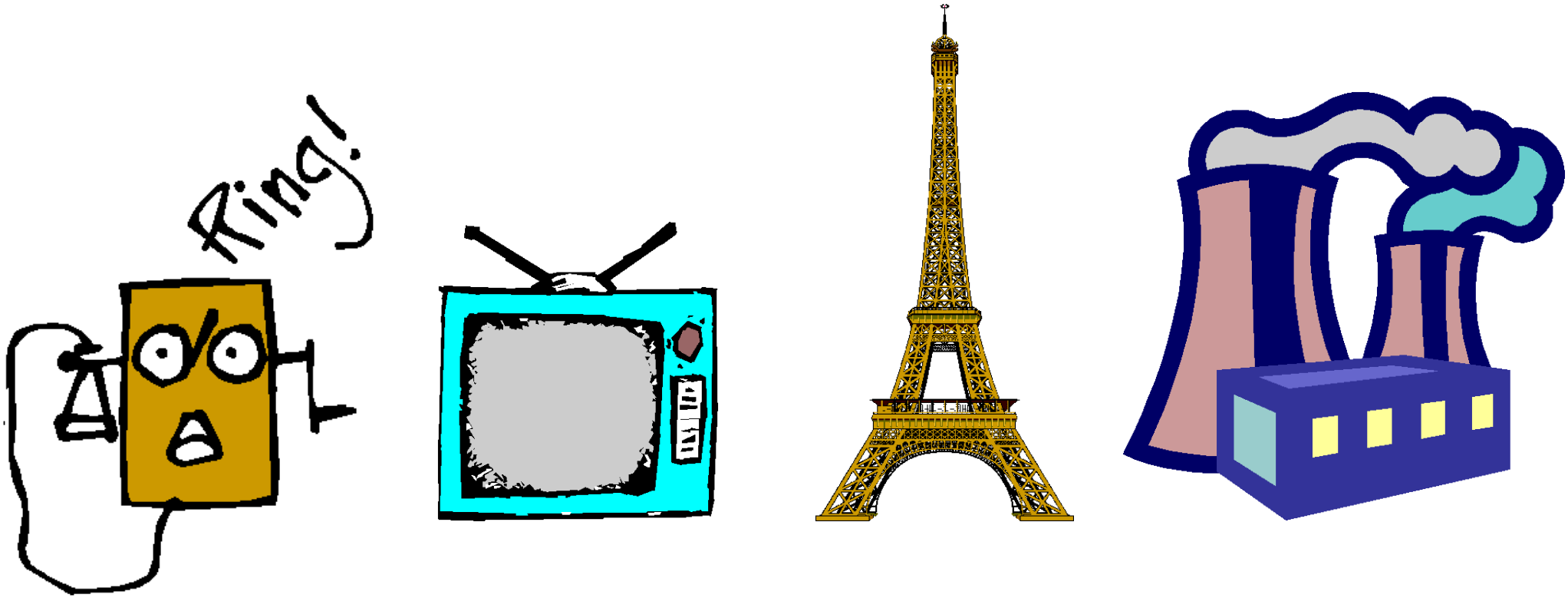
SCIENCE





When God created the universe, like many implementers who came later, he did not bother writing any documentation

ENGINEERING



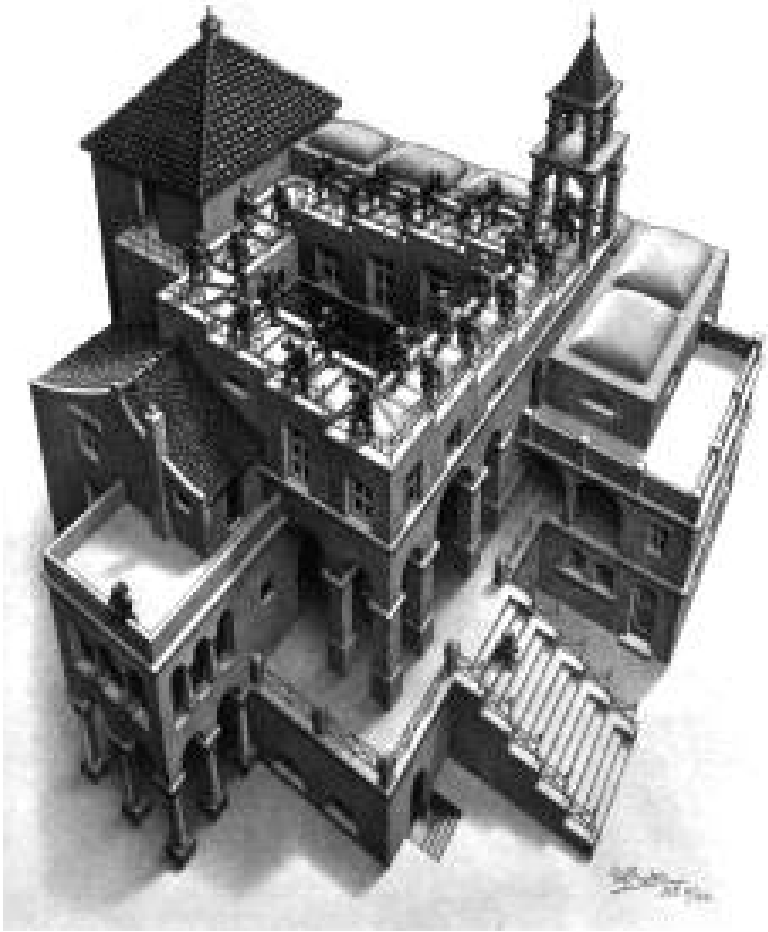
- Computer science is really the engineering of abstract objects
- An engineer is someone who can do for a dime what any fool can do for a dollar

ENGINEERING ISSUES

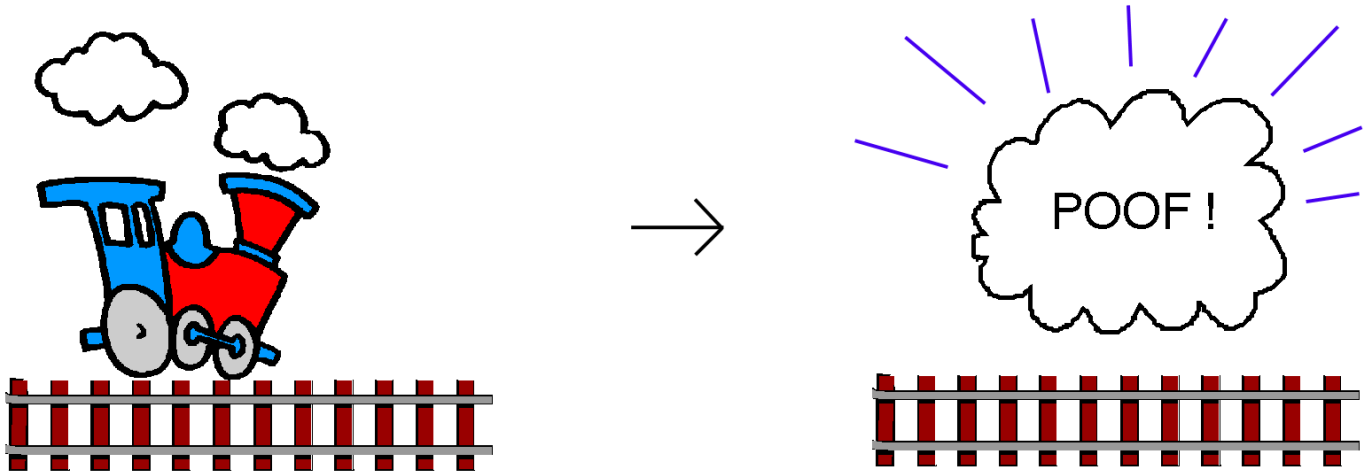
- The design process (use tools)
- Tradeoffs (space/time, cost/performance, design time/quality)
- Good heuristics (e.g. 10% of the code = 90% of the time)
- Prototyping and measurement (lab courses are essential)
- Standards (don’t reinvent the wheel)
- Maintainability (by someone other than the programmer)
- Working in teams (software hut)
- Quality control (design for testability)

GOLDEN RULE #7:

THINK IN TERMS OF SYSTEMS



AN EXAMPLE OF NOT THINKING ‘SYSTEMS’



- BART trains originally had drum brakes
- After the system was operational, they replaced all the drum brakes by disk brakes
- Trains suddenly began vanishing at random from the computers that controlled the system

ANOTHER EXAMPLE OF NOT THINKING ‘SYSTEMS’

- One of my students wrote the MINIX *mkfs* (make file system) program with elaborate block caching
- This program normally runs for about 30 sec per year
- How much time could the block caching save?
- The block caching was so hairy, debugging took 6 months

NEVER FORGET: THE USER IS PART OF THE SYSTEM

If the user hates your system, you have not done your job well

GOLDEN RULE #8:

KEEP THEORY UNDER CONTROL

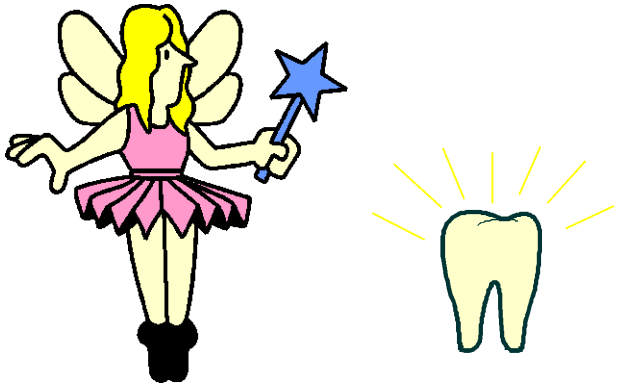
NONPROPOSAL TO NSF (PHYSICS DIVISION)

We propose to investigate the consequences to the entire universe of assuming:

$$E = mc^3$$

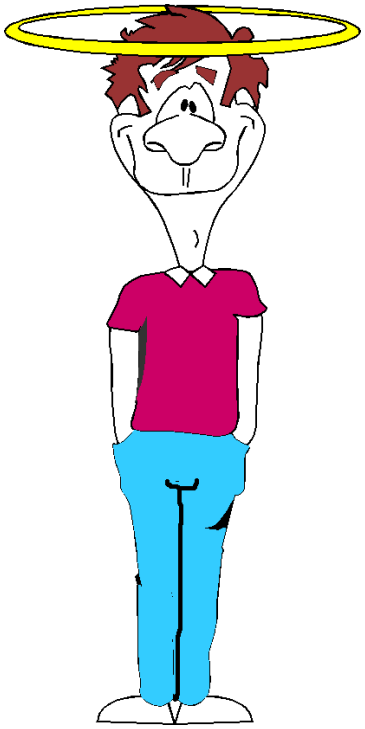
Budget required: $1,000,000 (first year)

CS THEORISTS OFTEN IGNORE REALITY



- The banker's algorithm for deadlock avoidance assumes that all resource demands are known in advance
- Most CPU scheduling algorithms ignore the time required to switch between processes
- Some distributed algorithms assume that sending *n* 1-byte messages takes the same time as 1 *n*-byte message

PROPER GOAL OF THEORY



The proper goal of theory in any field is to make models that accurately describe real systems. These models should help system builders do their jobs better

GOLDEN RULE #9:

IGNORE HYPE

EXAMPLES OF HIGHLY HYPED TOPICS

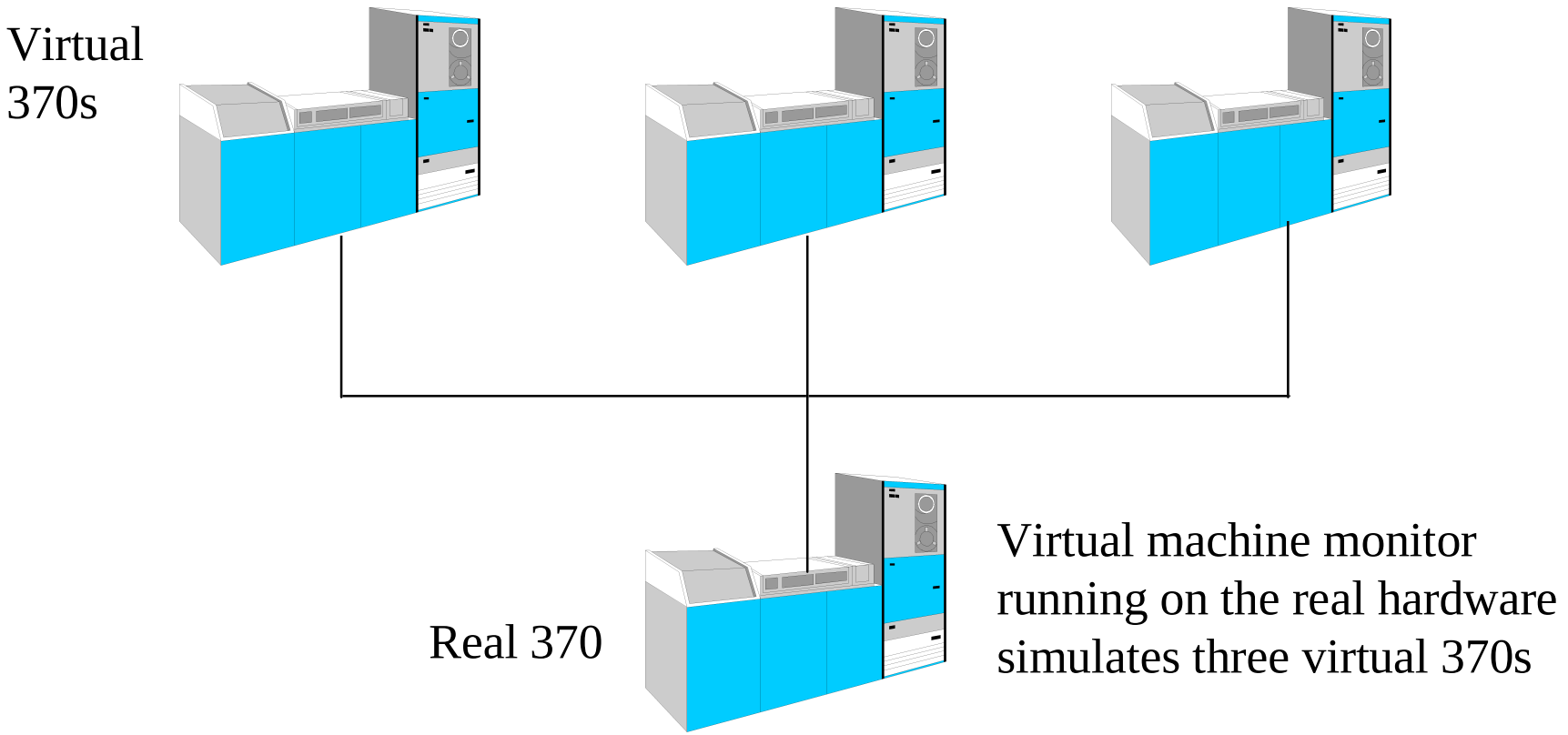
All of the following were once hyped as the solution to all your problems:

- PL/I
- Structured COBOL
- Ada
- Josephson junctions
- Program correctness proofs
- Fifth generation computers
- Automatic program generators
- The paperless office

GOLDEN RULE #10:

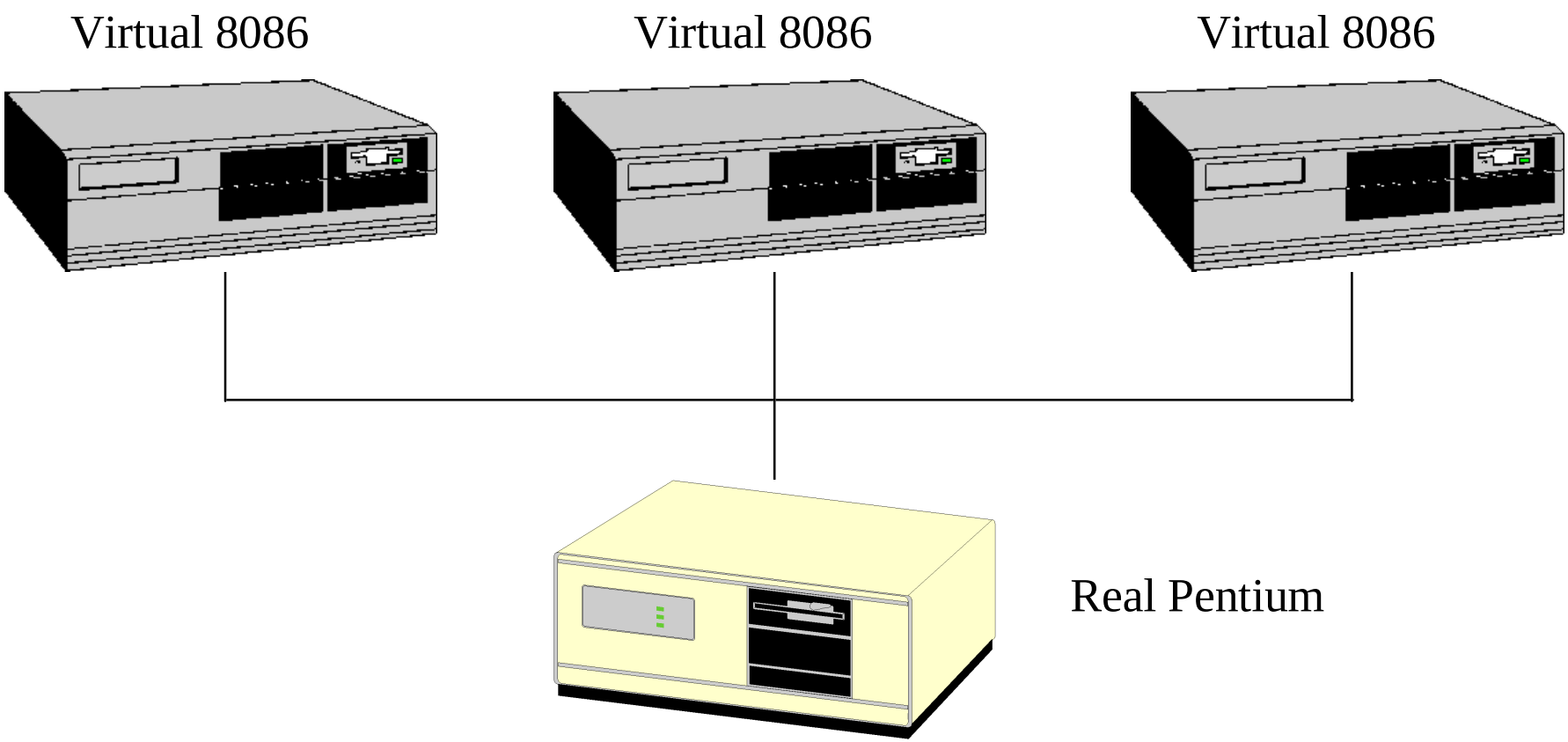
DON’T FORGET THE PAST

AN OLD IDEA: VIRTUAL MACHINES (VM/370)



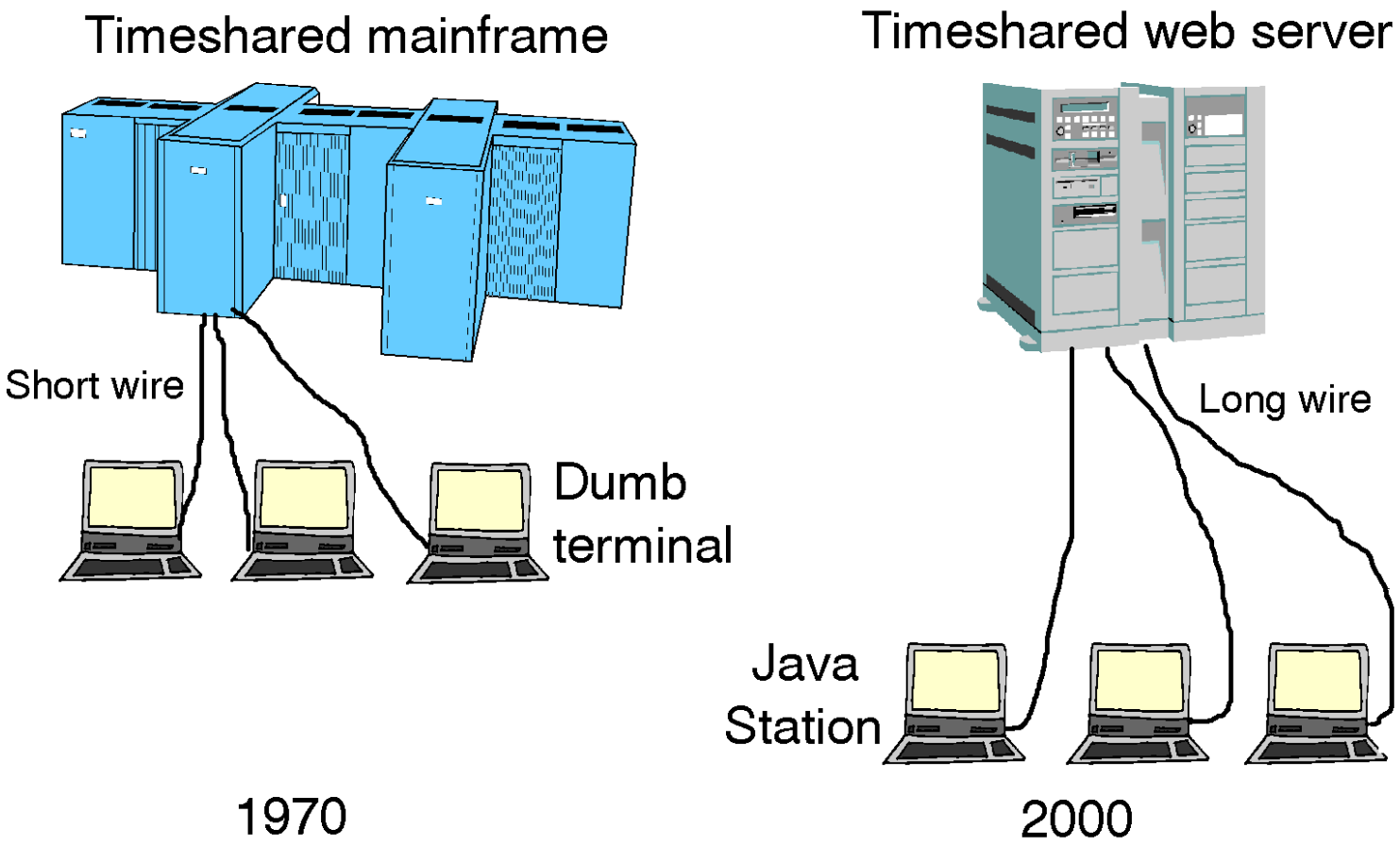
- VM/370 made it possible to separate multiprogramming from the user OS, provide excellent protection, etc.

A NEW OLD IDEA: PENTIUM VIRTUAL 8086 MODE



- Virtual 8086 mode on the Pentium makes it possible to run old 16-bit DOS applications on a virtual machine

TIMESHARING REVISITED



- Timesharing was killed by the PC, but it is coming back

ANOTHER OLD IDEA: INTERPRETATION

- In the days of yore, interpretive systems were common, for example, FORTH.
- As compilers got better, they passed out of fashion
- Now Java is making them popular again

SUMMARY

- Think long term
- Emphasize principles, not facts
- Expect paradigm shifts
- Explain how things work inside
- Show students how to master complexity
- Computer science is not science
- Think in terms of systems
- Keep theory under control
- Ignore hype
- Don’t forget the past