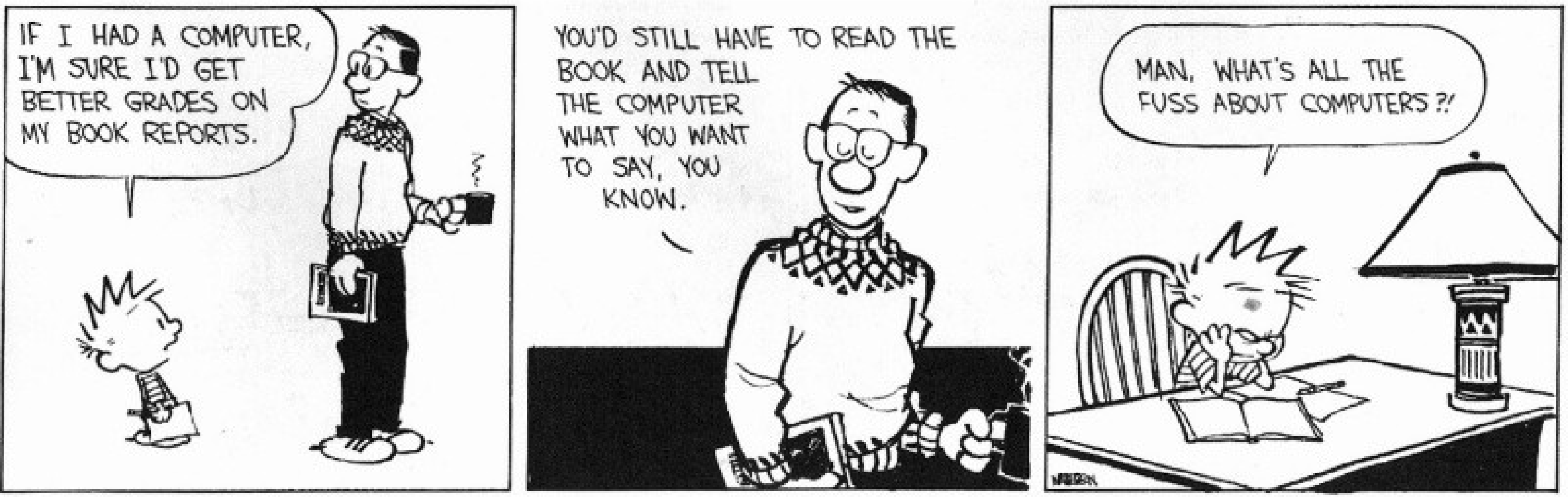
Programming Language Design and Implementation

Wes Weimer

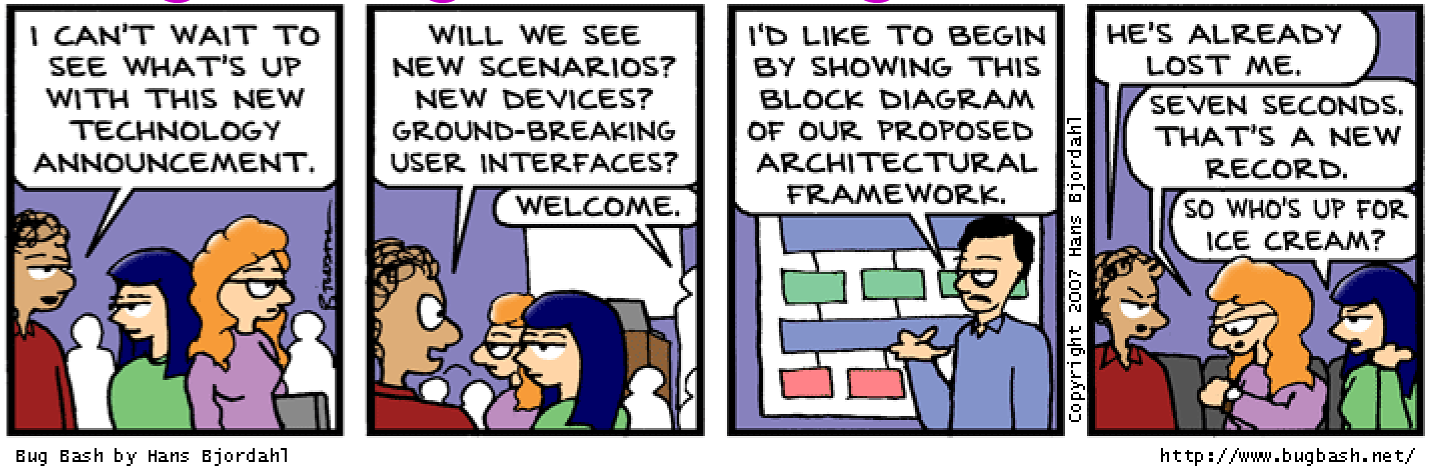
MW 3:30 – 4:45

OLS 120


IF I HAD A COMPUTER, I'M SURE I'D GET BETTER GRADES ON MY BOOK REPORTS.
YOU'D STILL HAVE TO READ THE BOOK AND TELL THE COMPUTER WHAT YOU WANT TO SAY, YOU KNOW.
MAN, WHAT'S ALL THE FUSS ABOUT COMPUTERS ?!

Cunning Plan

- Who Are We?
  - Wes, Adam Brady
- Administrivia
- What Is This Class About?
- Brief History Lesson
- Understanding a Program in Stages



I CAN'T WAIT TO SEE WHAT'S UP WITH THIS NEW TECHNOLOGY ANNOUNCEMENT.
WILL WE SEE NEW SCENARIOS? NEW DEVICES? GROUND-BREAKING USER INTERFACES?
WELCOME.
I'D LIKE TO BEGIN BY SHOWING THIS BLOCK DIAGRAM OF OUR PROPOSED ARCHITECTURAL FRAMEWORK.
HE'S ALREADY LOST ME.
SEVEN SECONDS. THAT'S A NEW RECORD.
SO WHO'S UP FOR ICE CREAM?
Copyright 2007 Hans Bjordahl
Bug Bash by Hans Bjordahl
http://www.bugbash.net/

Course Home Page

- **google: virginia cs 4610**
- **http://www.cs.virginia.edu/~weimer/4610**
- Lectures slides are available before class
  - You should still take notes!
- Assignments are listed
  - also grading breakdown, regrade policies, etc.

- Use the class Collab forum for all public questions

Course Options

- Three options are available:
  - Take ***only*** CS 4610 (Programming Languages) for 3 credits
  - Take CS 4610 (Progamming Languages) for 3 credits ***and also*** take CS 4501 (Compilers Practium) for 2 credits
  - Take neither
- CS 4501 Compilers Practicum
  - Must attend CS 4610 PL lectures
  - Has additional lectures, homeworks, tests

Should I Take This Course?

- **Beg for more mid-assignment checkpoints. Also, beg that the optional two-credit class only be opened just before the withdrawal deadline so you don't have to decide early on before you understand what you've decided to do to yourself. Finally, beg for mercy. :)**
- Professor Weimer is very engaging in his lectures, which kept me awake and interested even through the most boring or difficult sections of the class. **And some of the class material WAS boring. And it was most DEFINITELY difficult.** While **the assignments were VERY time-consuming and difficult (sometimes ridiculously so),** I felt great accomplishment after completing them, and I feel that I may have learned more from this class than any I have ever taken before.

Should I Take This Course?

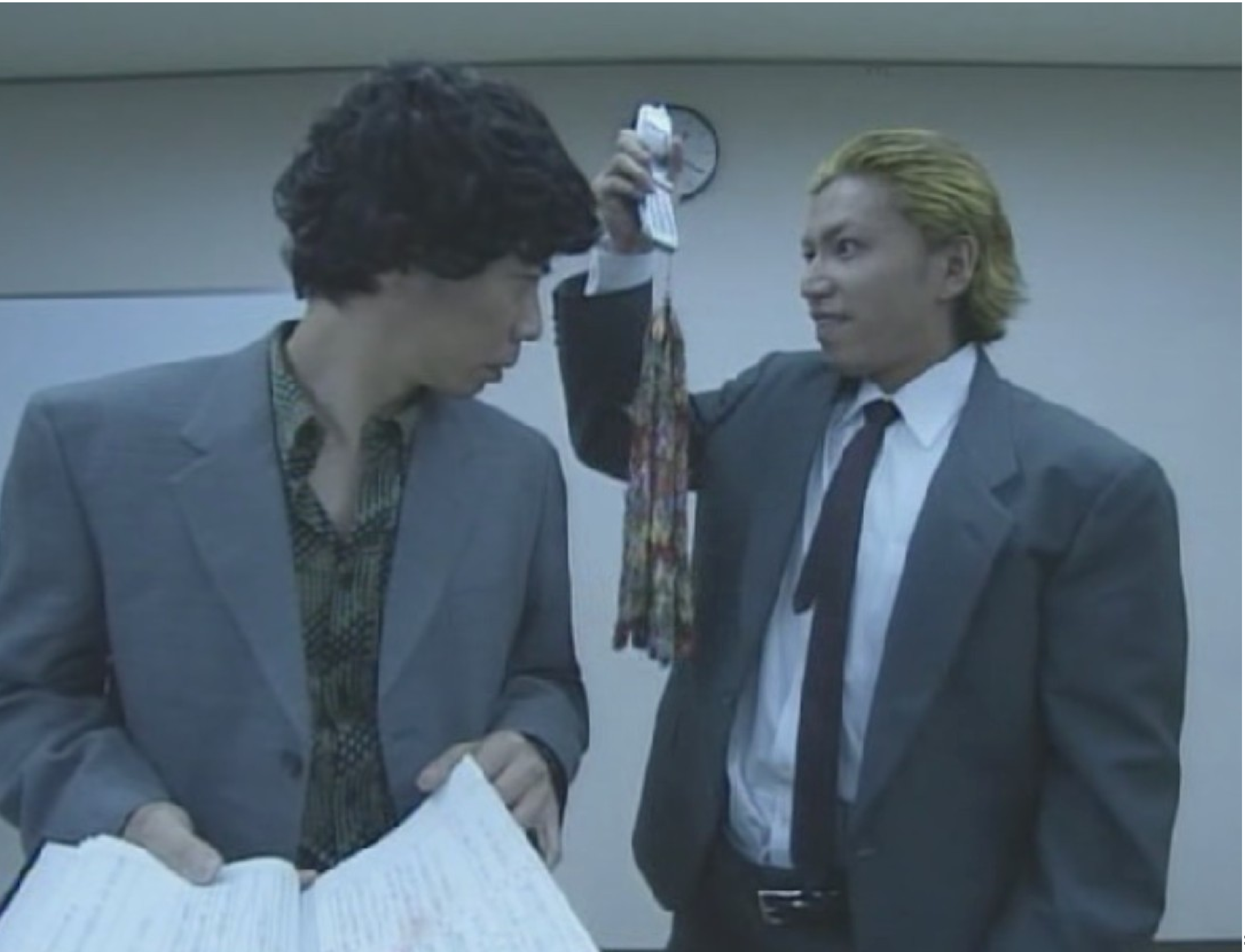
- **Easily the most time-consuming class that I have taken at UVA. There were times that I got very little sleep at all. In addition, the assignments require a deep understanding of the material, in order to do well on them.** Nevertheless, I learned a lot from this class.
- On top of this, Professor Weimer is extremely knowledgeable and teaches in such a way that even very difficult concepts become much easier to grasp. This being said, **do not take Programming Languages, because you will put in over 9,000 hours into the class.**
- **The hardest class that I've taken at UVA. Although it's worth 5 credits, the extra 2 credits are essentially worthless, since all of the credit requirements are for classes worth 3 credits each.**

Should I Take This Course?

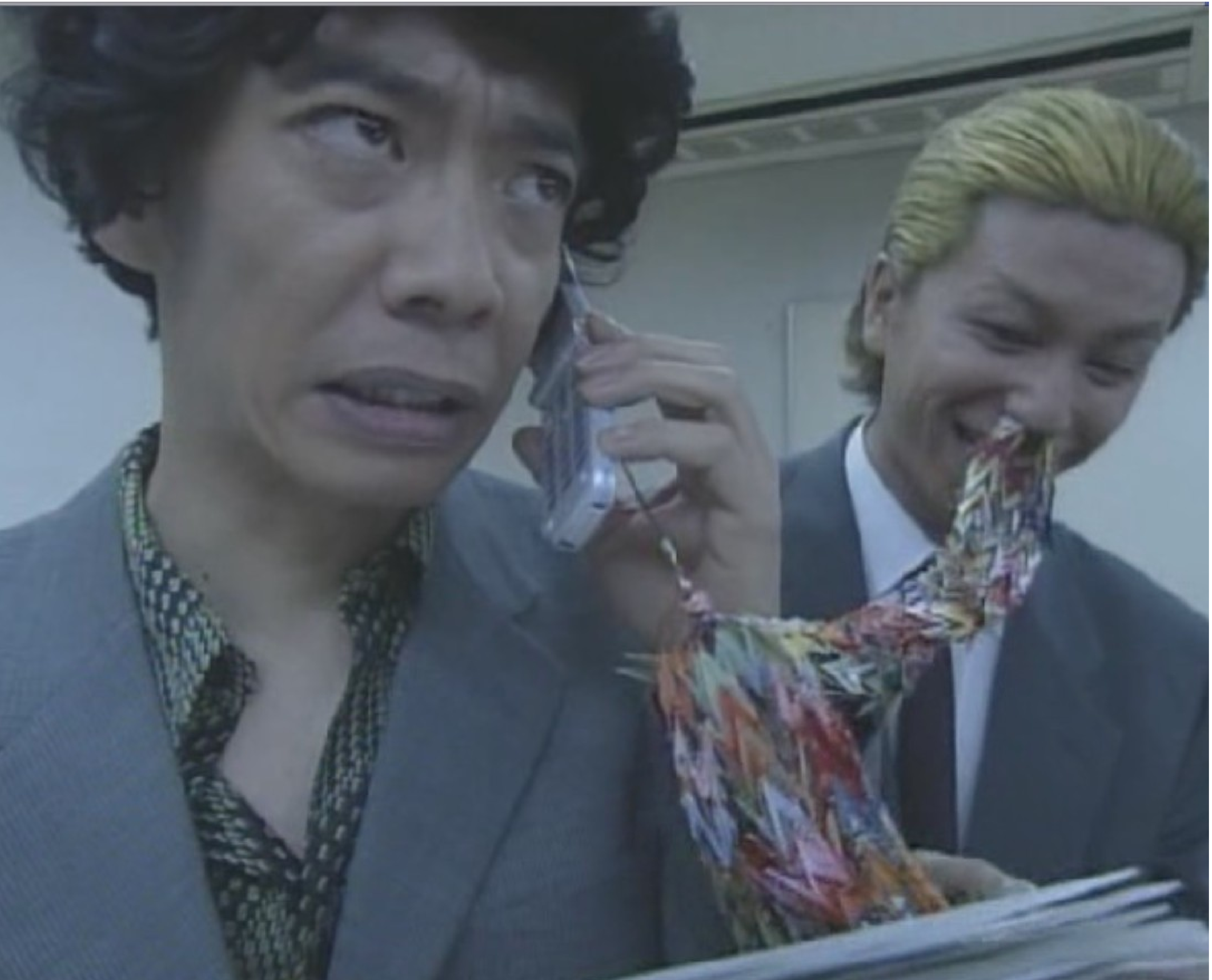
- **This course was the hardest course I have taken thus far,** but Professor Weimer made it very enjoyable. **The amount of work at times was a little overwhelming,** but I definitely learned a lot from it. Weimer was one of the best lecturers I have had in the Engineering school and he truly wanted the students to understand the material.
- This was a great course. **A ridiculous amount of work** but that was known information from the first day of the course. The instructor was enthusiastic and taught very well.
- **I don't know what you should do, but this class had ruined my life. I have spent more time on this class than the rest of my classes combined this semester.** However, knowing what I know now I would take this class again!

Compilers Practicum Sections

- There will be one sixty-minute Compilers Section each week
  - Hosted by Adam Brady, the TA
- We will vote for times that everyone in Compilers can make
  - If you cannot make the Section, you must drop Compilers. Sorry.
  - Notes posted on web.
- Next Week: pass around time signup sheet



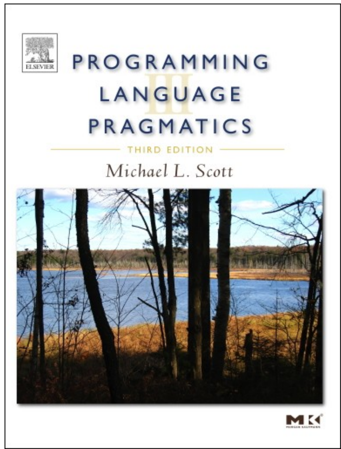
#9



#10

Course Structure

- Course has theoretical and practical aspects
  - Best of both worlds!
- Need both in programming languages!
- Reading = both
  - Many external and optional readings
- Written assignments = theory
  - Class hand-in, right before lecture, 0-5 points
- Programming assignments = practice
  - Electronic hand-in
- Strict deadlines
  - Ask me why ...

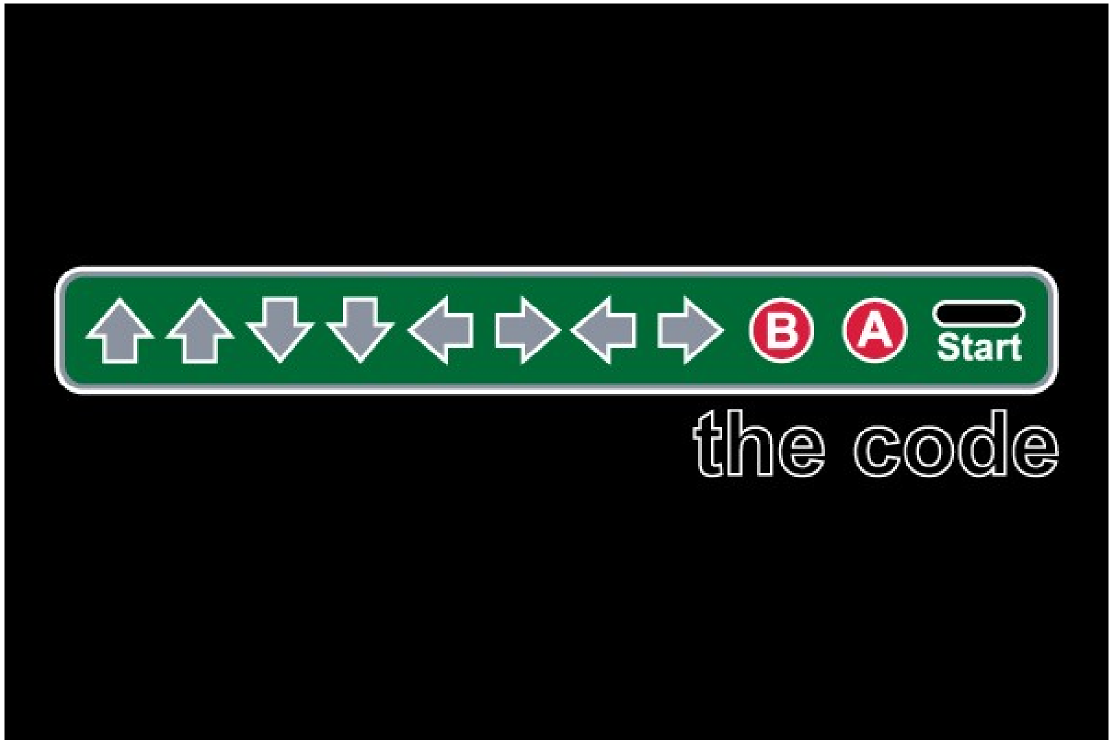


Academic Honesty

- Don't use work from uncited sources
  - Including old code
- We often use plagiarism detection software

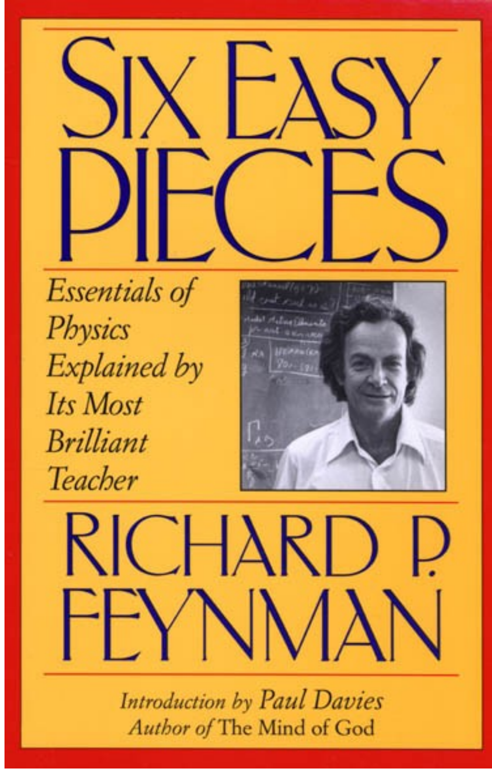
Academic Honesty

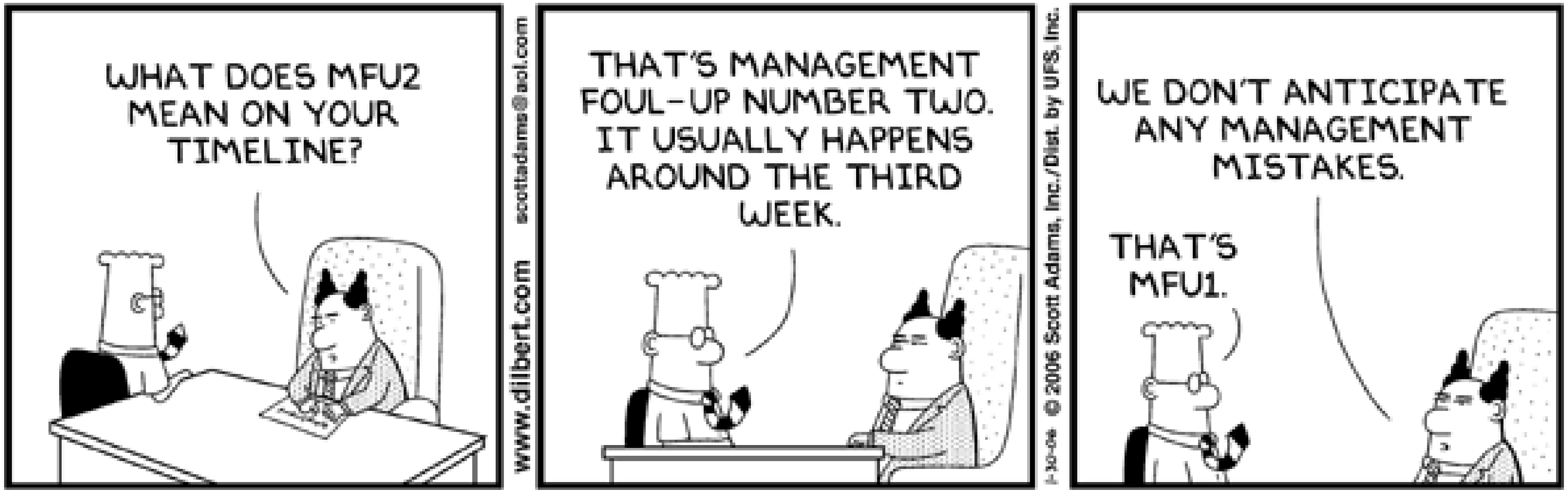
- Don’t use work from uncited sources
  - Including old code
- We often use plagiarism detection software



PL Course Project

- A big project: an Interpreter!
- … in four easy parts
- Start early!


SIX EASY PIECES
Essentials of Physics Explained by Its Most Brilliant Teacher
RICHARD P. FEYNMAN
Introduction by Paul Davies Author of The Mind of God


WHAT DOES MFU2 MEAN ON YOUR TIMELINE?
THAT'S MANAGEMENT FOUL-UP NUMBER TWO. IT USUALLY HAPPENS AROUND THE THIRD WEEK.
WE DON'T ANTICIPATE ANY MANAGEMENT MISTAKES.
THAT'S MFU1.
scottadams@aol.com
www.dilbert.com
1-30-06 © 2006 Scott Adams, Inc./Dist. by UFS, Inc.

© Scott Adams, Inc./Dist. by UFS, Inc.

PL Course Goals

- At the end of this course, you will be acquainted with the fundamental concepts in the **design** and **implementation** of high-level programming **languages**. In particular, you will understand the **theory** and **practice** of **lexing**, **parsing**, **semantic** analysis, and **code** interpretation. You will also have gained practical experience programming in multiple **different** languages.

Compilers Project & Course Goals

- The final project for the Compilers Practicum is a working **optimizing compiler** (code generator) that produces either bytecode or x86-64 assembly.
- At the end of this course, you will be acquainted with the fundamental concepts in code generation and optimization. In particular, you will understand the **theory** and **practice** of **code generation**, **stack layouts**, **calling conventions**, **dynamic dispatch**, **control flow graphs**, and **dataflow analyses**.

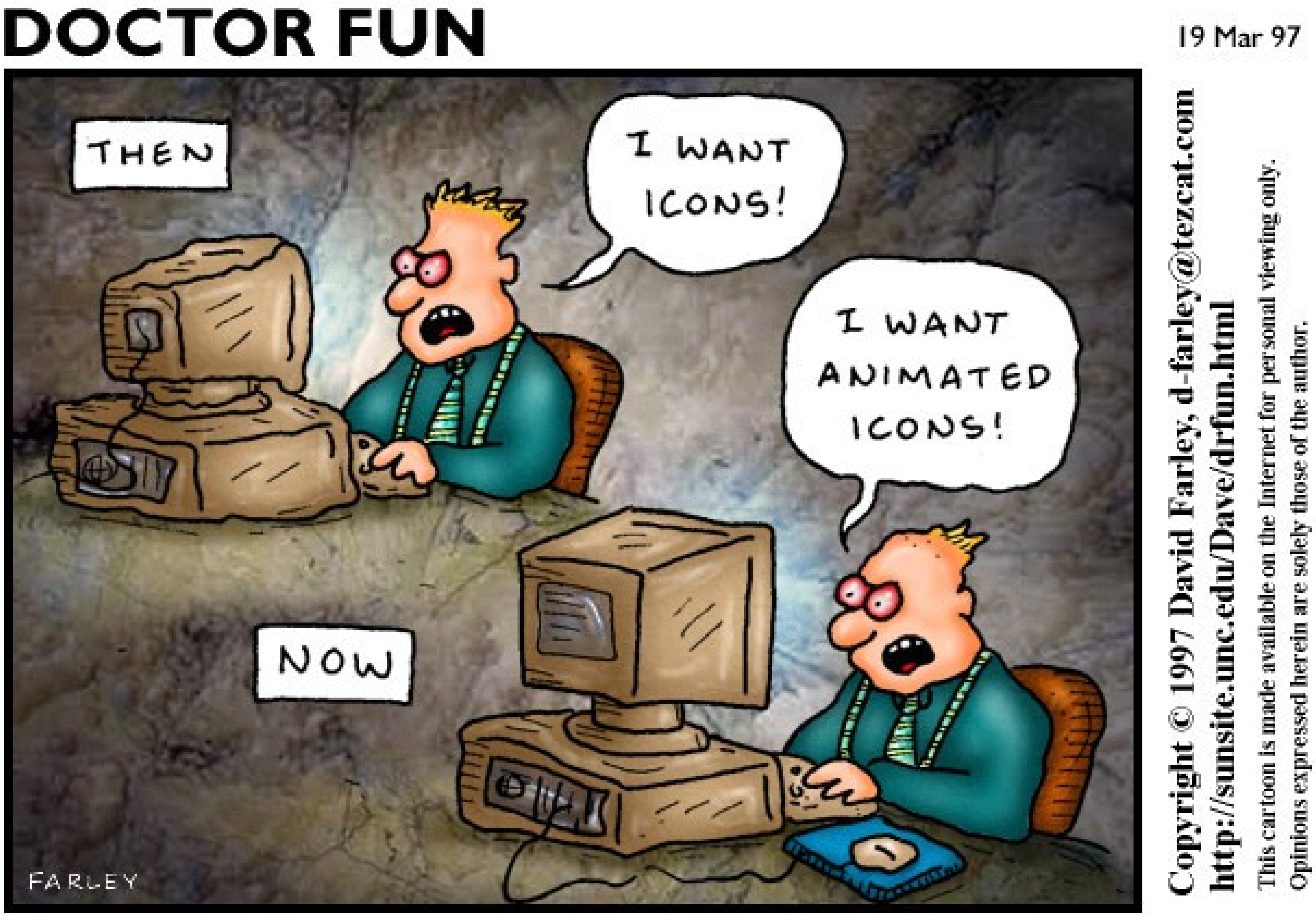
How are Languages Implemented?

- Two major strategies:
  - Interpreters (take source code and run it)
  - Compilers (translate source code, run result)
  - Distinctions blurring (e.g., just-in-time compiler)

- Interpreters run programs “as is”
  - Little or no preprocessing

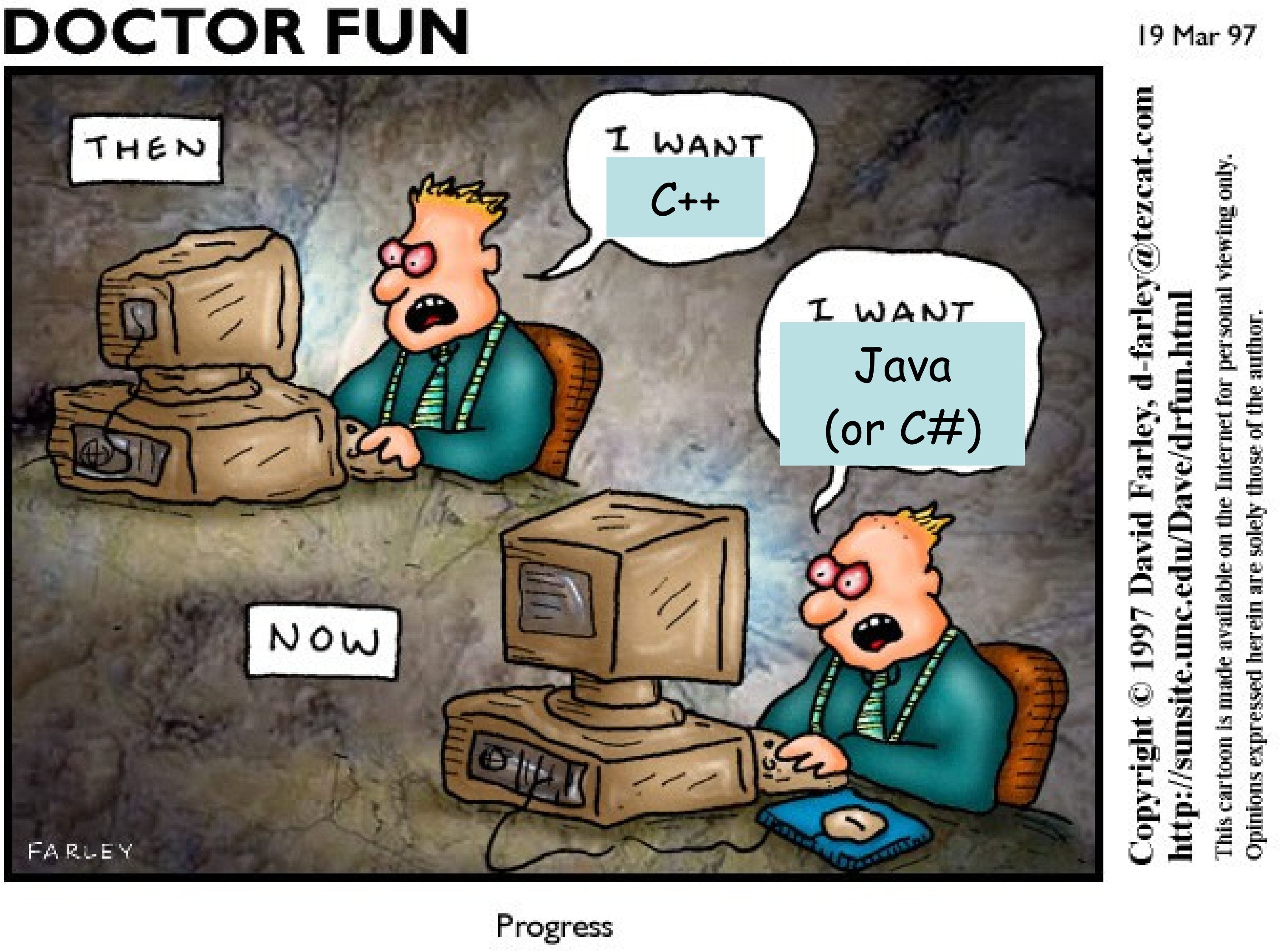
- Compilers do extensive preprocessing
  - Most implementations use compilers

Don't We Already Have Compilers?


DOCTOR FUN
19 Mar 97
THEN
I WANT ICONS!
NOW
I WANT ANIMATED ICONS!
FARLEY
Copyright © 1997 David Farley, d-farley@tezcat.com
http://sunsite.unc.edu/Dave/drfun.html
This cartoon is made available on the Internet for personal viewing only.
Opinions expressed herein are solely those of the author.

Progress

Dismal View Of Prog Languages



Copyright © 1997 David Farley, d-farley@tezcat.com
http://sunsite.unc.edu/Dave/drfun.html
This cartoon is made available on the Internet for personal viewing only.
Opinions expressed herein are solely those of the author.

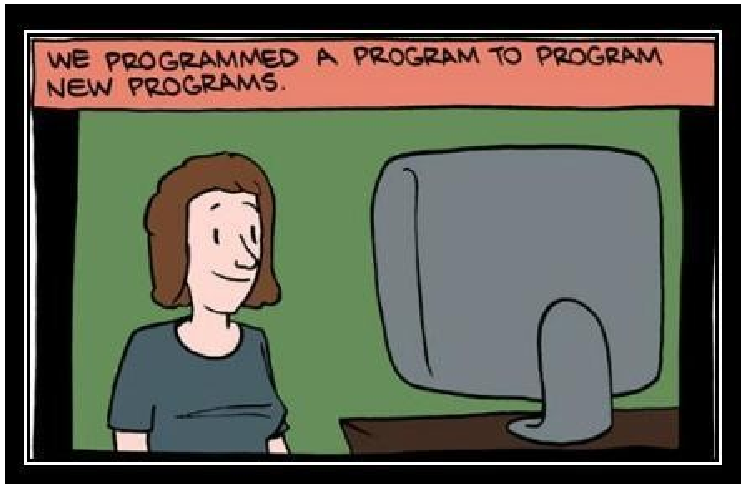
(Short) History of High-Level Languages

- 1953 IBM develops the 701 "Defense Calculator"
  - 1952, US formally ends occupation of Japan
  - 1954, Brown v. Board of Education of Topeka, Kansas

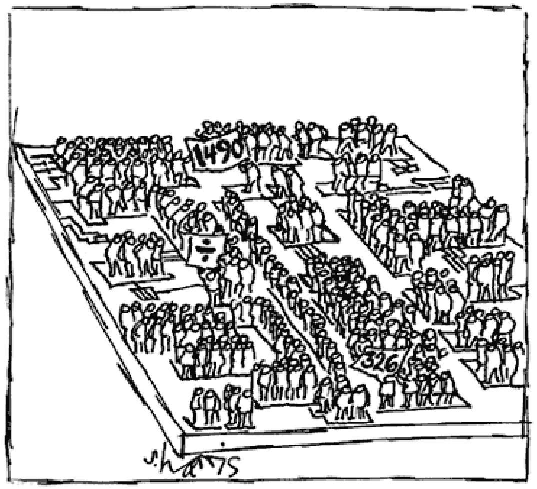
- **All programming done in assembly**

- **Problem: Software costs exceeded hardware costs!**

- John Backus: "Speedcoding"
  - An interpreter
  - Ran 10-20 times slower than hand-written assembly



FORTRAN I



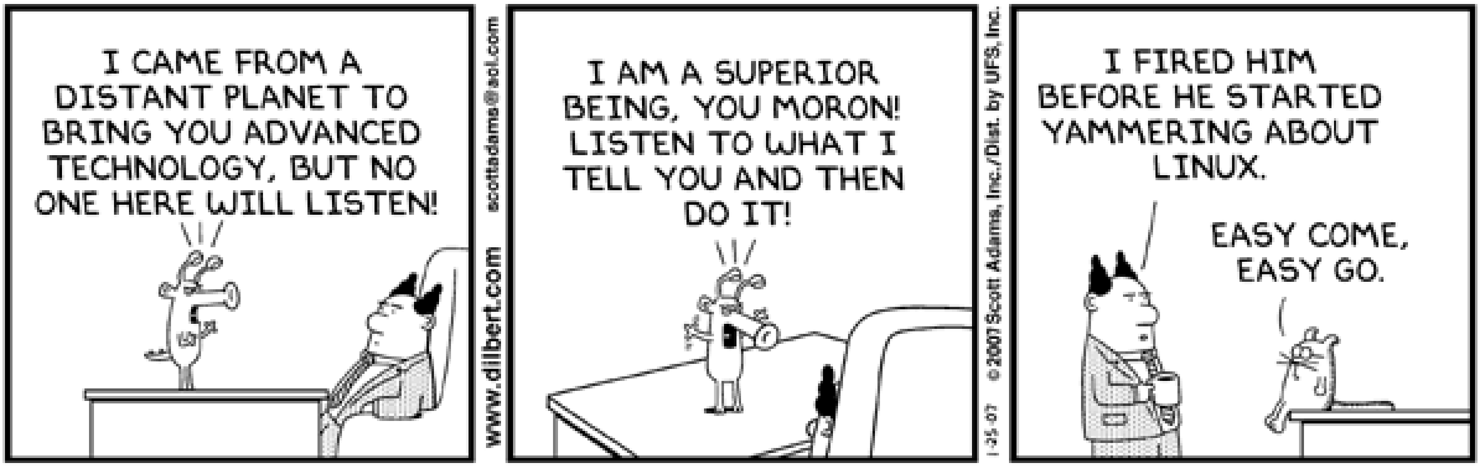
HUMAN SILICON CHIP: CAPABLE OF 6 COMPUTATIONS PER HOUR

- 1954 IBM develops the 704
- John Backus
  - Idea: translate high-level code to assembly
  - Many thought this impossible

- 1954-7 FORTRAN I project
- By 1958, >50% of all software is in FORTRAN
- Cut development time dramatically
  - (2 weeks → 2 hours)

FORTRAN I

- The first **compiler**
  - Produced code almost as good as hand-written
  - Huge impact on computer science
- Led to an enormous body of theoretical work
- Modern compilers keep the outlines of FORTRAN I


I CAME FROM A DISTANT PLANET TO BRING YOU ADVANCED TECHNOLOGY, BUT NO ONE HERE WILL LISTEN!
I AM A SUPERIOR BEING, YOU MORON! LISTEN TO WHAT I TELL YOU AND THEN DO IT!
I FIRED HIM BEFORE HE STARTED YAMMERING ABOUT LINUX.
EASY COME, EASY GO.
www.dilbert.com
scottadams@aol.com
© 2007 Scott Adams, Inc./Dist. by UFS, Inc.
1-25-07

© Scott Adams, Inc./Dist. by UFS, Inc.

Real-World Languages

- This Indo-European language is associated with South Asian Muslims and is the lingua franca of Pakistan. It developed from Persian, Arabic and Turkic influences over about 900 years. Poetry in this language is particularly famed, as is a favorite of US President Barack Obama.
- Example: السلام علیکم

Interpreters	Compilers
Lexical Analysis	Lexical Analysis
Parsing	Parsing
Semantic Analysis	Semantic Analysis
Optimization (optional)	Optimization (optional)
Interpret The Program	Generate Machine Code
	Run that Machine Code

The first 3, at least, can be understood by analogy to how humans comprehend English.

Lexical Analysis

- First step: recognize words.
  - Smallest unit above letters

This is a sentence.

- Note the
  - Capital “T” (start of sentence symbol)
  - Blank “ ” (word separator)
  - Period “.” (end of sentence symbol)

More Lexical Analysis

- Lexical analysis is not trivial. Consider:

  How d'you break "this" up?

- Plus, programming languages are typically more cryptic than English:

  ```
  *p->f += -.12345e-6
  ```

Bink Converter

QuickTime couldn't open this file, probably because it was a long filename or used unusual characters.

You might have to rename or move this file if you want QT to read it.

Yes, this is ridiculous.

And More Lexical Analysis

- Lexical analyzer divides program text into "words" or tokens

  if x == y then z = 1; else z = 2;

- Broken up:

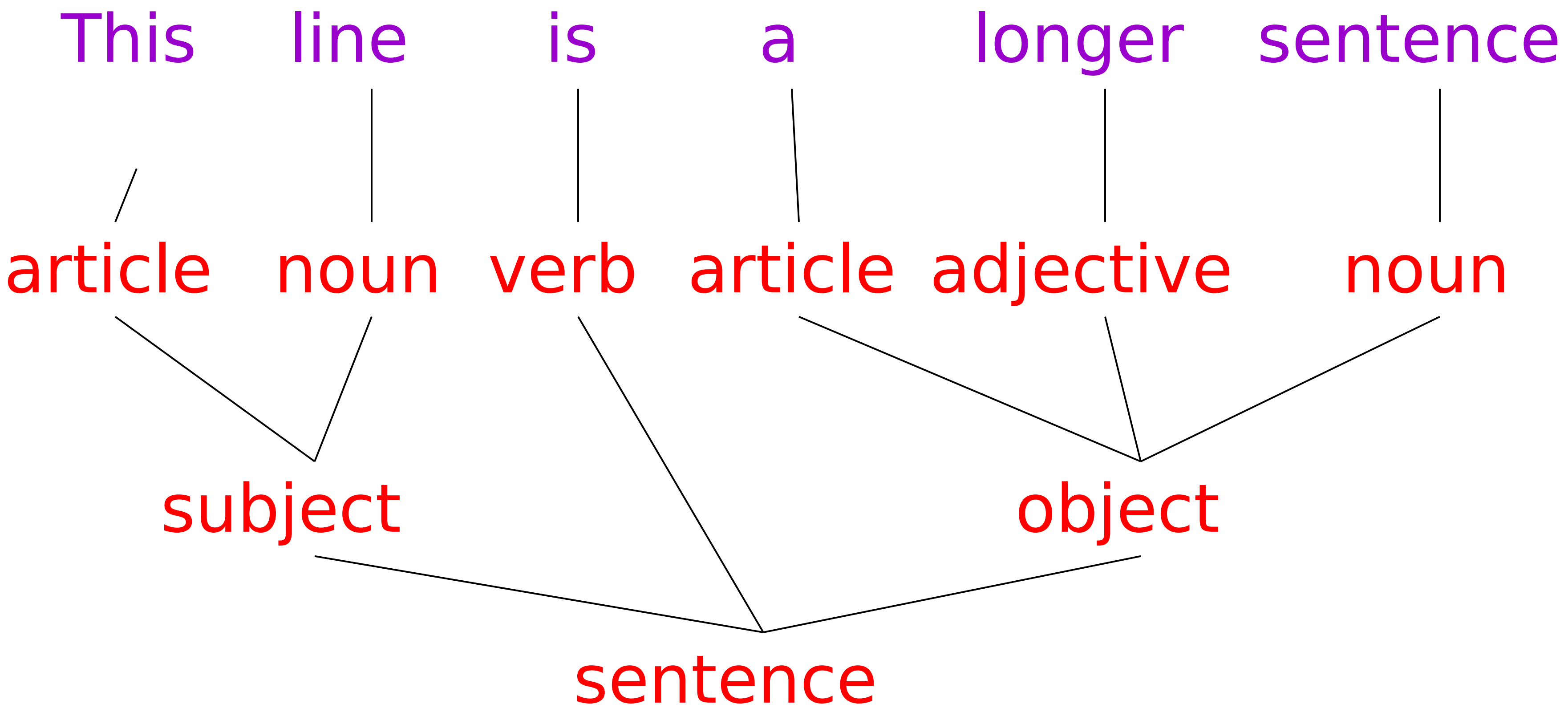
  if, x, ==, y, then, z, =, 1, ;, else, z, =, 2, ;

Parsing

- Once words are understood, the next step is to understand sentence structure

- Parsing = Diagramming Sentences
  - The diagram is a tree
  - Often annotated with additional information

Diagramming a Sentence

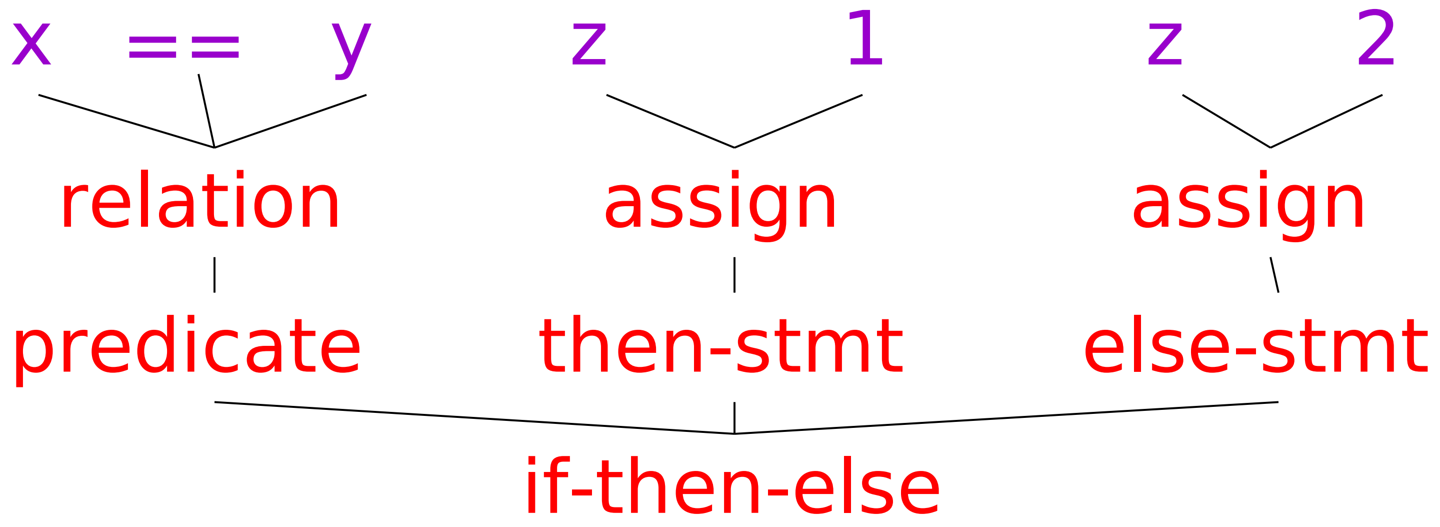


Parsing Programs

- Parsing program expressions is the same
- Consider:

  if x == y then z = 1; else z = 2;

- Diagrammed:



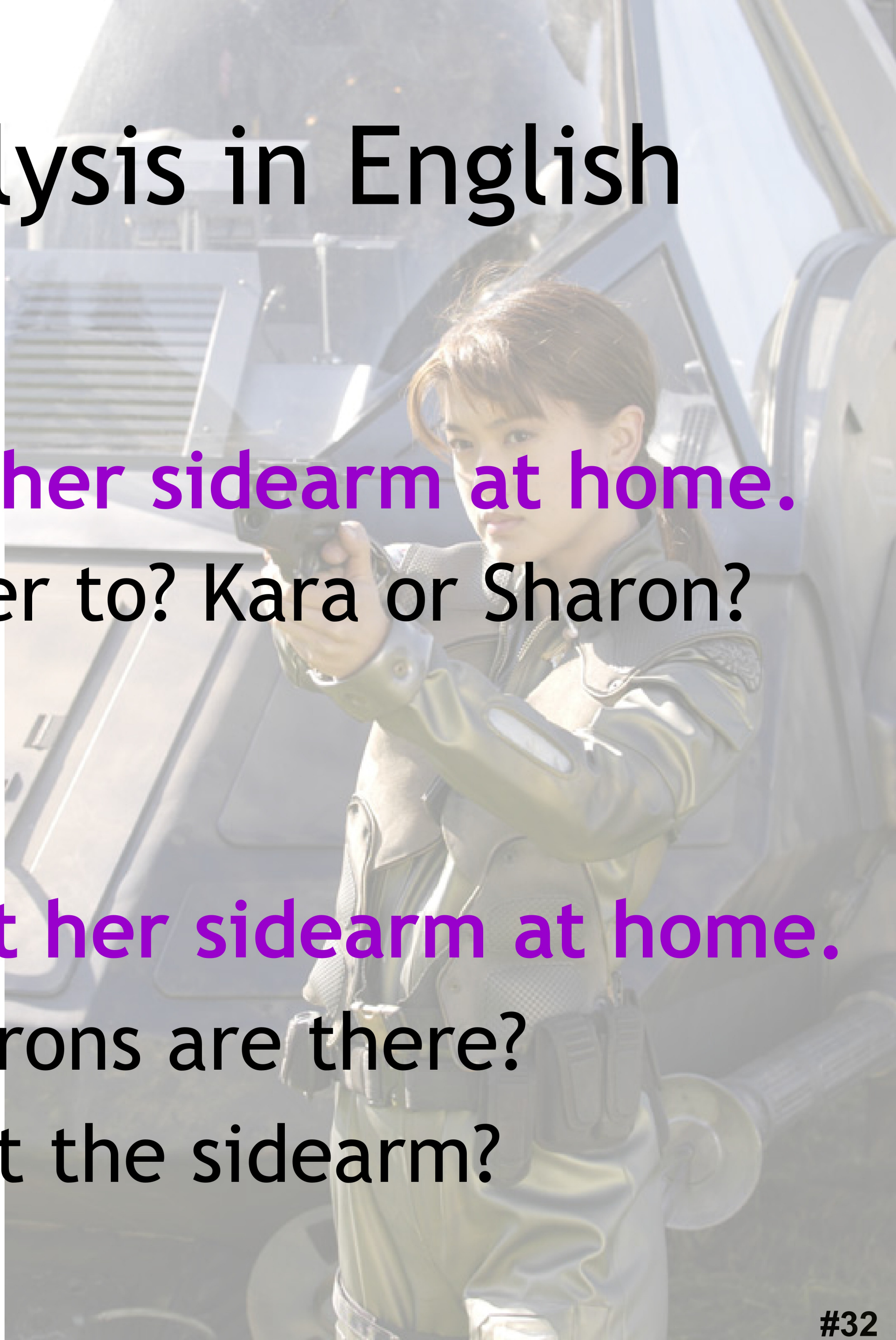
Semantic Analysis

- Once sentence structure is understood, we can try to understand “meaning”
  - But meaning is too hard for compilers

- Compilers perform limited analysis to catch inconsistencies: reject bad programs early!

- Some do more analysis to improve the performance of the program

Semantic Analysis in English



- Example:

Kara said Sharon left her sidearm at home.

What does “her” refer to? Kara or Sharon?

- Even worse:

Sharon said Sharon left her sidearm at home.

How many Sharons are there?

Which one left the sidearm?

It's context-sensitive!

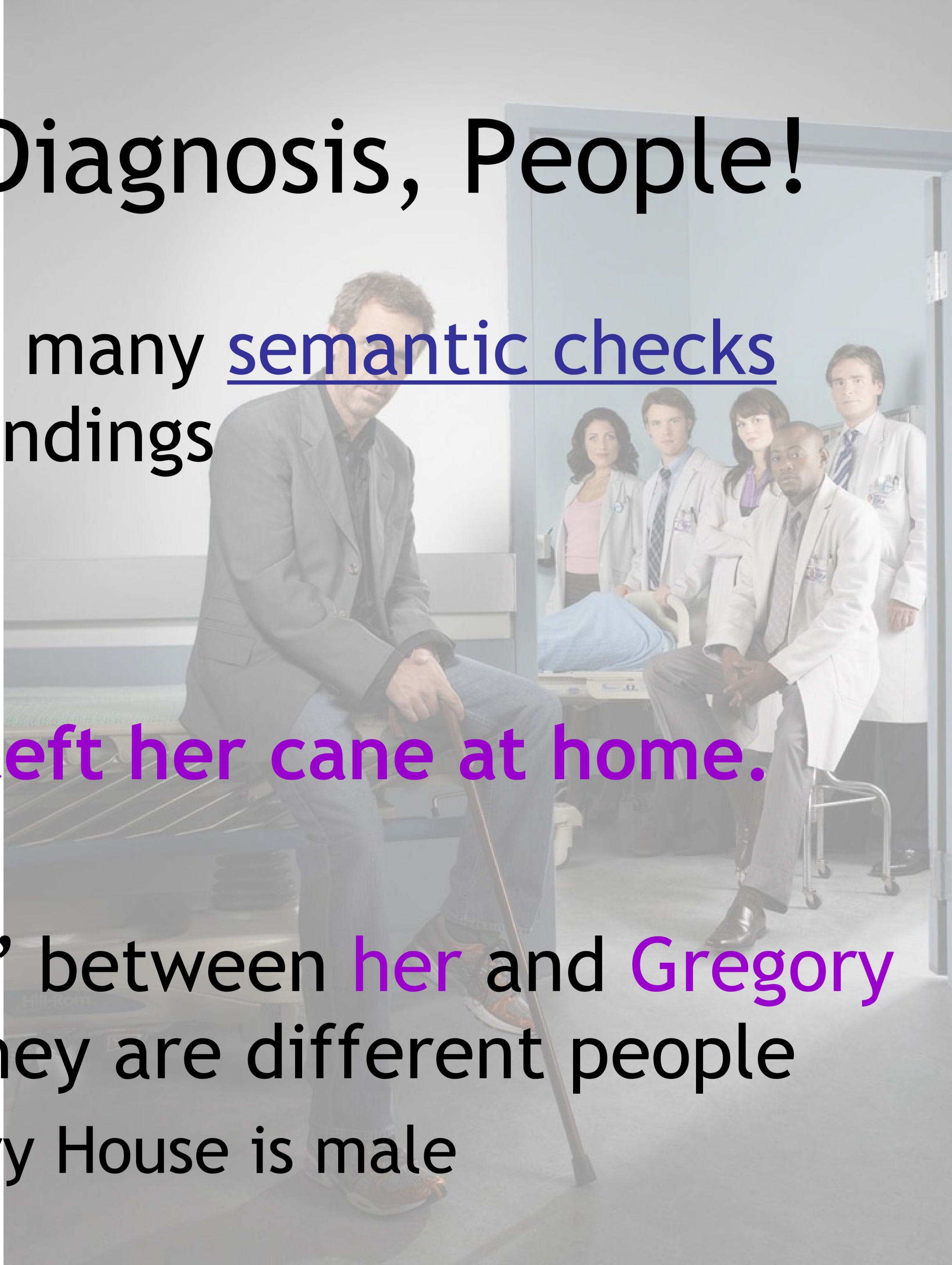
Semantic Analysis in Programming

- Programming languages define strict rules to avoid such ambiguities
- This C++ code prints "4"; the inner definition is used

```
{
  int Sydney = 3;
  {
     int Sydney = 4;
     cout << Sydney;
  }
}
```

Scoping or *aliasing* problem.

Differential Diagnosis, People!



- Compilers perform many semantic checks besides variable bindings

- Example:

  Gregory House left her cane at home.

- A "type mismatch" between her and Gregory House; we know they are different people
  - Presumably Gregory House is male

Optimization

- No strong counterpart in English, but akin to editing
- Automatically modify programs so that they
  - Run faster
  - Use less memory
  - In general, conserve some resource
- CS 4501 has an Optimization assignment

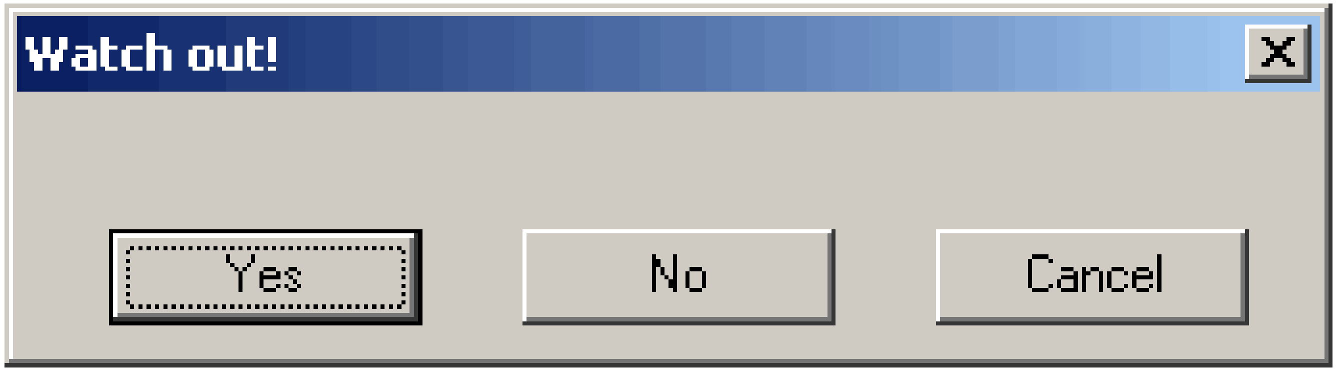
Code Generation

- Produces assembly code (usually)
  - which is then assembled into executables by an assembler
- A translation into another language
  - Analogous to human translation
- CS 4501: produce machine code
  - Either “Java Bytecode” or x86-64 assembly

Issues

- Compiling and interpreting are almost this simple, but there are many pitfalls.

- Example: How are bad programs handled?
- Language design has big impact on compiler
  - Determines what is easy and hard to compile
  - Course theme: **trade-offs in language design**



Languages Today

- The overall structure of almost every compiler & interpreter follows our outline

- The proportions have changed since FORTRAN
  - Early: lexing, parsing most complex, expensive
  - Today: optimization dominates all other phases, lexing and parsing are cheap
  - ... but still matter, ramble ramble ...

Trends in Languages

- Optimization for speed is less interesting. But:
  - scientific programs
  - advanced processors (Digital Signal Processors, advanced speculative architectures)
  - Small devices where speed = longer battery life
- Ideas we'll discuss are used for improving code reliability:
  - memory safety
  - detecting concurrency errors (data races)
  - type safety
  - automatic memory management
  - …

Why Study Prog. Languages?

- Increase capacity of expression
  - See what is possible
- Improve understanding of program behavior
  - Know how things work “under the hood”
- Increase ability to learn new languages
- Learn to build a large and reliable system
- See many basic CS concepts at work

What Will You Do In This Class?

- Reading (textbook, outside sources)
- Learn about different kinds of languages
  - Imperative vs. Functional vs. Object-Oriented
  - Static typing vs. Dynamic typing
  - etc.
- Learn to program in different languages
  - Python, Ruby, ML, “Cool” (= micro-Java)
- Complete homework assignments
- Write an interpreter!

What Is This?

A lungo il mio cuore di tali ricordi ha voluto colmarsi!

Come un vaso in cui le rose sono state dissetate:

Puoi romperlo, puoi distruggere il vaso se lo vuoi,

Ma il porfumo delle rose sarà sempre tutt'intorno.

Długo, długo moje serce przepełnione było takimi wspomnieniami!

Były jak waza, w której kiedyś róże destylowały:

Możesz sprawić by pekła, możesz gruchotać waze jeśli chcesz,

Ale zapach róż bedzie wciaż czuć dookoła.

Mon coeur est brûlant rempli de tels souvenirs

Comme un vase dans lequel des roses ont été distillées:

Tu peux le briser, tu peux détruire le vase si tu le désires,

Mais la senteur des roses sera toujours là.

Lang, lang soll die Erinnerung in meinem Herzen klingen!

Gleich einer Vase, drin Rosen sich einst tränkten:

Lass sie zerbrechen, lass sie zerspringen,

Der Duft der Rose bleibt immer hängen.

Muito, muito tempo seja meu coração preenchido com tais lembranças!

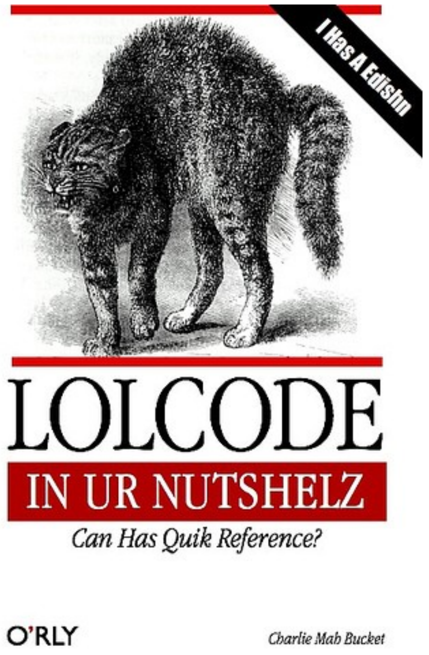
Tal qual o vaso onde rosas foram uma vez destiladas:

Pode quebrar, pode estilhaçar o vaso se o desejas,

Mas perdurará para sempre o aroma das rosas perfumadas.

The Rosetta Stone

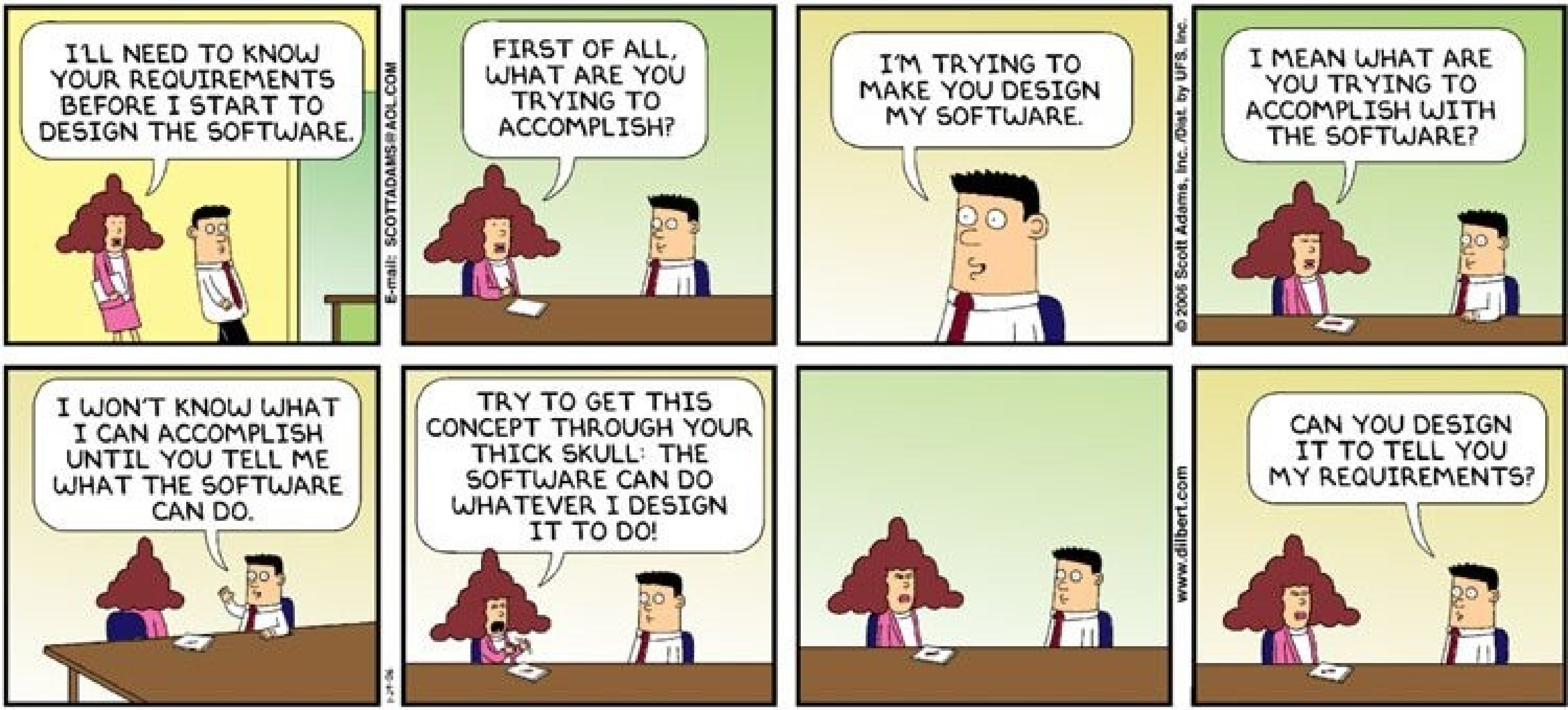
- The first programming assignment involves writing the same simple 75 line) program in:
  - **Ruby, Python, OCaml, Cool and C**
- PA1c, **due Wed Jan 25**, requires you to write the program in two languages (you pick)
- PA1, due one week later, requires all five



Long, long be my heart with such memories fill'd!
Like the vase in which roses have once been distill'd:
You may break, you may shatter the vase if you will,
But the scent of the roses will hang round it still.
- Thomas Moore (Irish poet, 1779-1852)

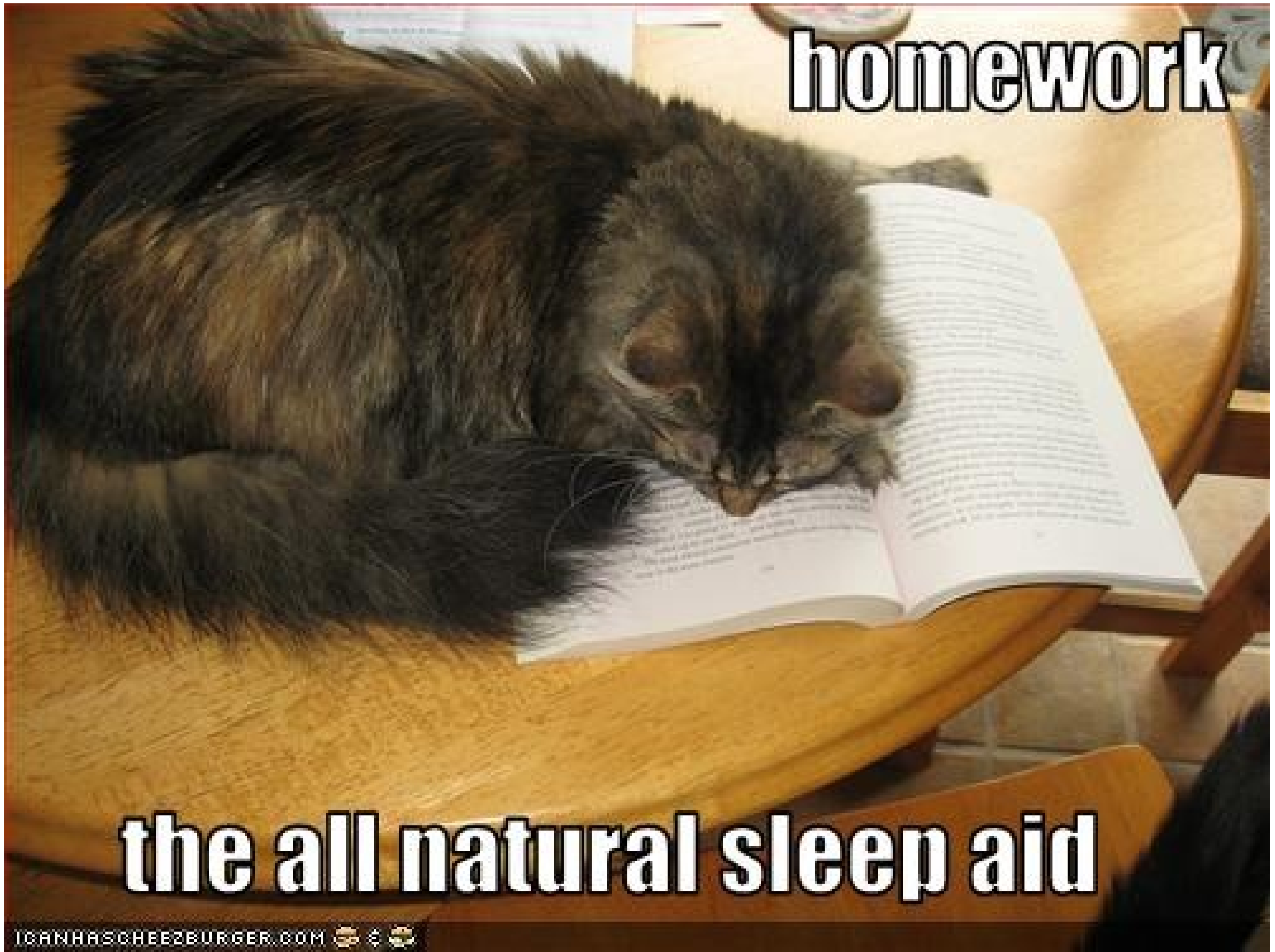
Live Submission Demo

- Let's visit the automated submission website


I'LL NEED TO KNOW YOUR REQUIREMENTS BEFORE I START TO DESIGN THE SOFTWARE.
E-mail: SCOTTADAMS@AOL.COM
FIRST OF ALL, WHAT ARE YOU TRYING TO ACCOMPLISH?
I'M TRYING TO MAKE YOU DESIGN MY SOFTWARE.
© 2006 Scott Adams, Inc./Dist. by UFS, Inc.
I MEAN WHAT ARE YOU TRYING TO ACCOMPLISH WITH THE SOFTWARE?
I WON'T KNOW WHAT I CAN ACCOMPLISH UNTIL YOU TELL ME WHAT THE SOFTWARE CAN DO.
TRY TO GET THIS CONCEPT THROUGH YOUR THICK SKULL: THE SOFTWARE CAN DO WHATEVER I DESIGN IT TO DO!
www.dilbert.com
CAN YOU DESIGN IT TO TELL YOU MY REQUIREMENTS?

Start The Homework Now

- It may help you decide whether to stay in this course



Homework

- Scott Book reading (for Monday)
- Get started on PA1c (due in 7 days)

Questions?

HEY, YOU WANNA PLAY THE QUESTION GAME?
WHAT'S THAT?
YOU'VE NEVER HEARD OF THE QUESTION GAME?
HOW DO YOU PLAY?
DO I REALLY HAVE TO EXPLAIN IT?
HOW CAN I PLAY IF I DON'T KNOW THE RULES?
ARE YOU PUTTING ME ON?
WHAT DO YOU MEAN?