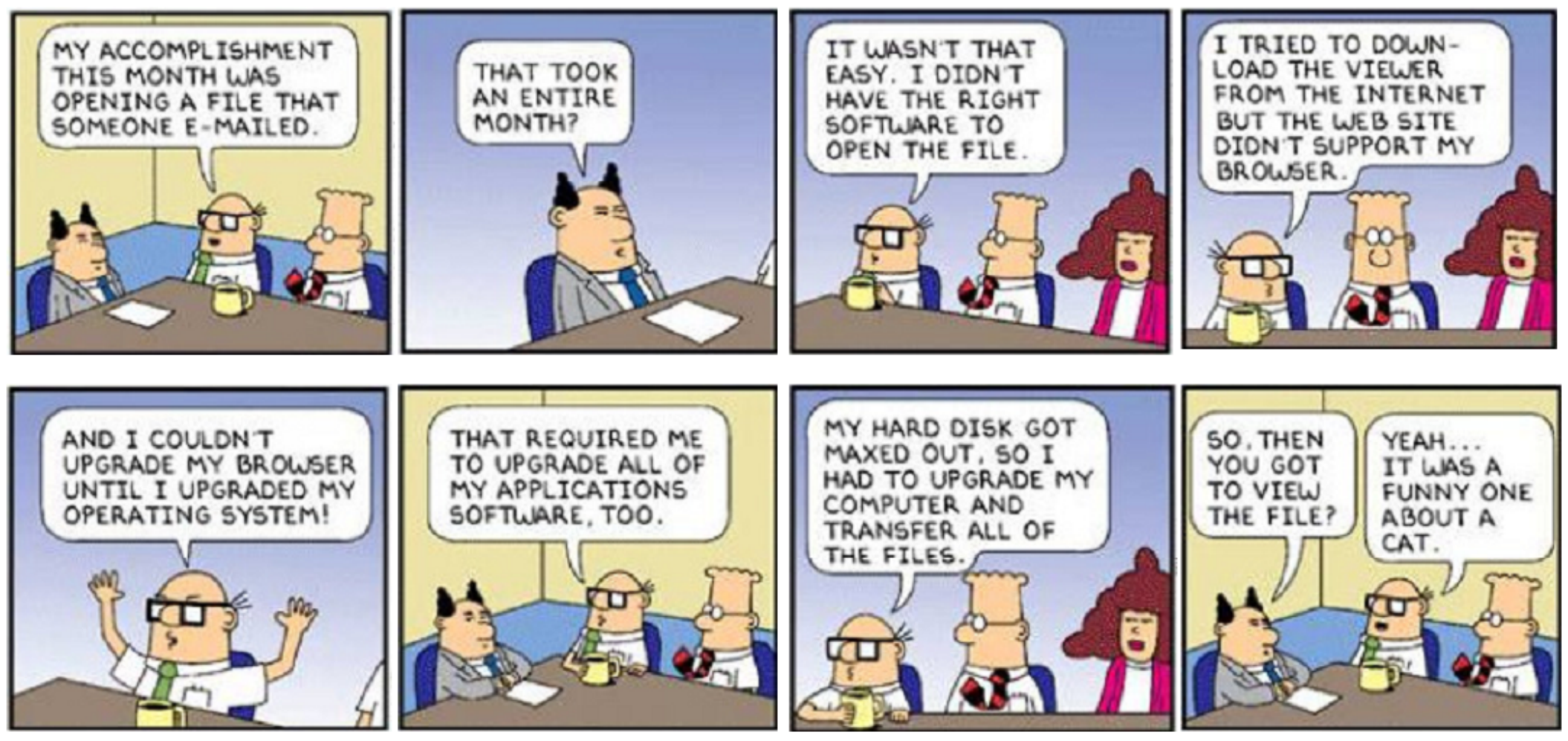
IT 314 SOFTWARE ENGINEERING

GETTING STARTED – ABOUT THE COURSE

MOTIVATION?

- Why Software Engineering?
  - Get good grades?
  - Become a good Software Engineer?

SOFTWARE IS ALL ABOUT FUN!



www.dilbert.com

GOOD AT IT: WRITING PROGRAMS !

- Find roots of a quadratic equation!
- Sorting algorithm!
- Implementing a data structure!
- Graph problem/algorithms!
- Writing database queries!
- Programming LEX, YACC
- Writing programs for Chat, Library, Reservation, etc.!

WRITING PROGRAMS – DO'S !

- You choose a programming language
- You write everything from scratch
- For you, writing program means mainly coding
- You do some testing to show that it works (ad hoc)
- You decide to stop coding (ad hoc)
- You show/submit it to your teacher
- That's All !

TRIVIAL EXAMPLE

- Try to move a file from folder A to another folder B
- What are the possible scenarios?

POSSIBLE SOLUTIONS

- Trying to move the file when it is open
- You do not have the security rights to paste the file in folder B
- Folder B is on a shared drive and storage capacity is full
- Folder B already has a file with the same name

ANOTHER SCENARIO

- Suppose you have a 15 input fields each one having 5 possible values.
- How many combinations to be tested?

5^15 = 30517578125!!!

A SELF ASSESSMENT TEST

The function triangle takes three integer parameters that are interpreted as the lengths of the sides of a triangle. It returns whether the triangle is equilateral (three lengths equal), isosceles (two lengths equal), scalene (no lengths equal), or invalid (impossible lengths).

```
final int EQUILATERAL = 0;
final int ISOSCELES = 1;
final int SCALENE = 2;
final int INVALID = 3;
int triangle(int a, int b, int c)
{
   if (a >= b+c || b >= a+c || c >= a+b)
         return(INVALID);
   if (a == b && b == c)
         return(EQUILATERAL);
   if (a == b || a == c || b == c)
         return(ISOSCELES);
   return(SCALENE);
}
```

WRITING PROGRAMS – DON'TS !

- You don’t do Requirement Engineering
- You don’t do Formal Designing
- You don’t follow standard coding practices
- You don’t do Formal Verification and Validation
- You don’t do Deployment
- You don’t check-in your code
- That’s NOT All !

WRITING PROGRAMS – DON'TS !

Above All

- Project Management Activities
  - Planning, Estimation, Scheduling, Tracking
- Configuration Management
  - Versioning, Change management, Bug tracking
- Quality Management
  - Reviews, Audits
- Software Evolution

DEVELOPING SOFTWARE

- You ~~choose~~ a programming language
- You ~~write everything~~ from scratch
- For you, writing program means ~~mainly coding~~
- You do testing to show that it ~~works (ad hoc)~~
- You decide to stop coding ~~(ad hoc)~~
- You ~~show/submit it to your teacher~~
- That's All !

DEVELOPING MAINTAINING SOFTWARE (Industrial Practice)

- Multi-million LOC software project
- Organization-specific process–centric approach
- People-oriented (team) effort
- Some unfamiliar language
- Different development environment
- Support services

COURSE OBJECTIVES

- Main objective: Give an idea of how industrial-strength software gets developed

- Understand and Appreciate Software Engineering:
  - How to build high quality complex software systems within time while dealing with complexity and change

- Acquire technical knowledge (main emphasis)
- Acquire managerial knowledge
- At the end you should have the ability to plan, execute, and manage small software projects
- Lectures will discuss how to perform different tasks in a project
- In the project, the techniques will be applied

ACQUIRE TECHNICAL KNOW-HOW

- Factual SE information
  - Learn how to undertake an SE activity individually as well as in groups
  - Learn how to design and develop software systems
  - RUP, Agile, SA/SD
- Understand Software Modeling and Design
- Learn different modeling techniques using UML models:
  - Use Case modeling
  - Object Modeling
  - Functional Modeling
  - Dynamic Modeling
- Learn how to Validate Software
- Learn how to use CASE (Computer Aided Software Engineering) Tools:
  - Rational Rose, JUnit

ACQUIRE MANAGERIAL KNOW-HOW

- Understand the Software Lifecycle (SDLC)
  - Process, Product, Project and People
  - Learn about different software development lifecycles
  - Best Practices
  - Process Improvement, CMM
- Estimates, Planning and Executions
  - Cost, Effort, Resources, Risk
  - Deliverables
  - Scheduling and tracking
- Communicate and critically evaluate yours results
  - key to your future success

THIS COURSE IS ABOUT …

- Understanding and learning about SDLC
- Methodologies, techniques and tools that can be used for better S/W development
- Technical and not so technical issues
- Learning on real life projects

DISCLAIMER!

- This course will not teach you about
  - Any specific software development technologies like .NET, Java
  - Any specific software process that is followed in an industry (not a case study)

COURSE ORGANIZATION

- Technical and Managerial Know-how for SDLC
- Teaching
  - Lectures (with slides)
  - Course Project
  - Hands-on (Modeling, TDD, Unit Testing, Code Inspection)
  - Presentations

TOPICS TO BE COVERED

- SDLC, Software products, processes, people, and projects
- Requirements
  - Gathering, Analysis Modeling and Specifications
  - Use case modeling, case studies
- Object-Oriented and Function-Oriented Analysis and Design
  - Software Architecture
  - Static and Dynamic Modeling, Scenarios, System-subsystem, Class diagrams, DFDs, SASD
- Software Development Methodologies
  - UP and Agile (RUP, XP, Agile, TDD, Scrum)
- Software Testing and Quality Assurance
  - Unit Testing, Integration Testing, System Testing, Regression
- Testing
  - Reviews, Walkthroughs, Code Inspections

EVALUATION CRITERIA

30%	Course Project
	- a poor project cannot get a good grade

Project - group grade; marks equally divided unless the team specifies a diff distribution (based on contribution)

25%	End-Semester
15%	First-In Semester
15%	Second-In Semester
15%	Lab Sessions/Experiments/Quiz

- **ZERO marks for copying or allowing someone to copy**

There will be bonus marks (max 10%) for undertaking additional work on course objectives

BEWARE OF !

- Plagiarism
- Copyrights

SE AS A SEPARATE DISCIPLINE

- Several Universities Offers Undergraduate program in SE [SWEBOK'04]
  - University of New South Wales (Australia),
  - McMaster University (Canada),
  - the Rochester Institute of Technology (US),
  - the University of Sheffield (UK), and others.

MAIN BOOK REFERENCES

- RB-1: Pankaj Jalote. "An Integrated Approach to Software Engineering", 3rd Edition, Narosa, 2005
- RB-2: Bernd Bruegge, Allen Dutoit: “Object-Oriented Software Engineering: Using UML, Patterns, and Java”, Prentice Hall, 2003.
- RB-3: Blaha and Rumbaugh. "Object-Oriented Analysis and Modeling using UML, 2nd Edition, TMH 2005.
- RB-4: "Head First Design Pattern" Book

Questions??