EECE 310

Software Engineering and

Software Processes

Last Week

We talked about:

- Syllabus and expectations

This Week

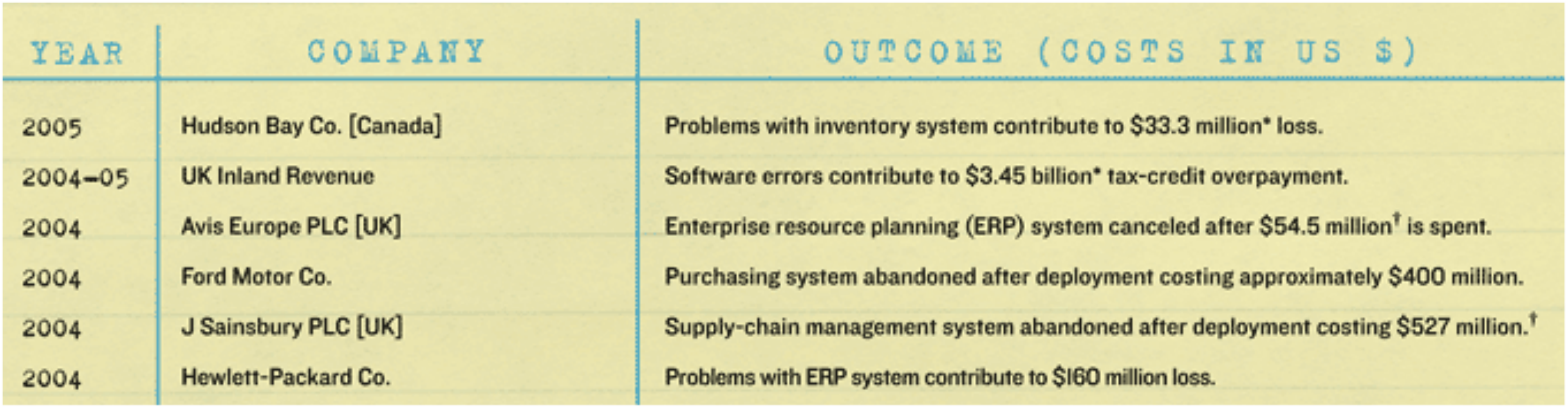
We talked about:

- Why software engineering is important
- Software process

Why is software engineering important?

Bad SE practices create…

- Failed projects
- Lost money
- Stressed employees
- Poor customer value



YEAR	COMPANY	OUTCOME (COSTS IN US $)
2005	Hudson Bay Co. [Canada]	Problems with inventory system contribute to $33.3 million* loss.
2004–05	UK Inland Revenue	Software errors contribute to $3.45 billion* tax-credit overpayment.
2004	Avis Europe PLC [UK]	Enterprise resource planning (ERP) system canceled after $54.5 million† is spent.
2004	Ford Motor Co.	Purchasing system abandoned after deployment costing approximately $400 million.
2004	J Sainsbury PLC [UK]	Supply-chain management system abandoned after deployment costing $527 million.†
2004	Hewlett-Packard Co.	Problems with ERP system contribute to $160 million loss.

Good SE practices create…

- Successful projects
- Happy customers
- Business value

Where we need SE

- Linux 2.6.34:
  - 9000 changesets
  - 1100 developers
  - (over 2 months or so)
  - Managing this ...?

Source: http://lwn.net/Articles/385949/

Question

- Have you ever developed a serious software system (in a company)?
  - If so, what SE practices did they use?

- Why are these practices useful?

What are good SE practices?

- Software process management
- Requirements engineering
- Version control system
- Coding standards
- Testing, testing, testing
- Continuous integration
- Clean code
- …

What should you expect?

EECE 310 is less about coding and more about software organization

- Although there is quite some coding involved as well

Previous classes

- Quantitative answers
  - What is the output of this code? *“27”*
  - What is the time complexity? *“O(n)”*
  - Which data structure should we use? *“BST”*

- There is a quantitative exact answer!

This class

- Qualitative answers to engineering problems
  - How good is this design?
  - How good is this elicitation question?
  - What problems could you encounter?
  - How could you deal with problem X?

- great, ok and bad answers possible

Midterm/final

- Multiple choice and True/False answers
  - Not necessarily all or nothing.
- Short answer or essay questions
  - Be concise to the point
  - Write clean!

How to succeed?

- Know the facts
  - Study the book, lecture notes, and readings
  - Listen actively in class
  - Take notes
  - Strive to identify factual information
- Practice applying the facts
  - In-class activities are good for this (cover many exam questions)
  - Be an active member of your team (peer evaluations)
  - Apply theory in practice (in the lab)

Continuous Evaluation (lab grade)

- Several bigger assignments
  - Learn the tools and toolkit
  - Elicit requirements / create user stories
  - Create initial design
  - …
- Weekly “Scrum Standup” meetings
  - Standup meeting in lab with TA
  - Discuss progress (accomplished tasks) and planned tasks
  - Discuss obstacles

Lab

- Labs start officially 3rd week of September and will be focused on the assignments.
- Lab attendance is **mandatory**
- You will have to work on your own laptops/ PCs since you need to install software.

Keys to success

- Attend lectures & lab
- Stay up-to-date on readings
- Pull your load in the group
- Work on your lab assignments/project **consistently** rather than leaving it until the **end**
- Be courteous to your teammates
- Understand & use the SE tools
- Ask if you are not sure, ask on a timely manner
  - (**questions asked a day before the final exam are not going to help you**)

After EECE 310 you should be able to…

- Explain the technical and interpersonal challenges of software engineering
- Communicate technical matters with programmers, managers, and clients effectively
- Perform the various activities/phases of software development effectively using modern methodologies and tools.

Lab: Sign up!

- Teams
  - Assignment 0: Create a GitHub account and add submit your username
    - (see Connect for the link)
  - If we don't have your GitHub username by **9th Sep @ 10: PM**, you will not be placed in a team!
- Deadlines are FIRM
    - points will be deducted for late submissions by TAs!

Social Coding

- We will use ***github*** for
  - Version control (source code)
  - Documentation (wiki)
  - Planning (issues and milestones)
  - Collaboration (notifications)
  - Monitoring progress and activities (graphs)
- Each team will get their own private repository

- You can start familiarizing yourself with Git/GitHub: http://teach.github.com/classnotes/2012-11-05-git-github-basics-online.html

Learning Goals (today)

- Explain what can go **wrong** when a software project is completed without using a software process
- Describe benefits of using a software process
- Introduce waterfall model: advantages, drawbacks
- Given a case study / project, choose an appropriate process model and justify your choice

Overview

- When things go wrong…
- Software process definitions
- Sequential process models
  - Waterfall, Spiral, V-model
- Next time … agile models

What is Process?

What is Process?

- Process helps us repeat systematically.

When things go wrong

- Denver airport
- HealthCare.gov
- UBC Connect!

Denver Airport Baggage System

FAIL

The new system

was supposed to

- reduce flight delays,
- shorten waiting times for luggage, and
- cut airline labor costs.

Automated, airport-wide baggage system

- 20 miles of track
- 6 miles of conveyor belts
- 56 laser arrays that read bar coded tags
- 3100 standard size baggage “telecars”
- Some 100 networked computers

The new system

The timeframe

- Tight schedule: started 17 months before scheduled opening (1993)

- *Meanwhile*: in Germany, engineers spent two years just testing a similar, but much smaller system!
  (With 24/7 operation for 6 months!)

More risks

- Most buildings already constructed
  - System must adapt: sharp turns, narrow corridors
- Little attention paid to German sister project
  - Devised system from scratch
- Internal communication problem
  - City, Airport management, consultants, airlines,
  - Several copies of everything
  - No proper synching! Restricted access, Other construction work, …

Consequences

- Airport opening **delayed** four times
  - Overall, 16 months delay
- Engineering firm went **bankrupt**
- New engineering firm
  - Split the system in 3 (one per terminal)
  - Manual backup system
- Overall damage: **1.3 billion $**
- Automatic system **abandoned** in 2005
  - Reported *savings* of 1M$ / month in maintenance costs

HealthCare.gov

- Federal health insurance exchange website
- Went live Oct. 1 2013
- So many problems plagued the site that it prompted *Congress* to hold hearings about it in late October!

HealthCare.gov

Problems:

- So many users on the first day that it crashed!
- Last-minute software changes
- Bad design decisions
  - Users had to register first before being able to browser the site
- Inadequate testing

UBC Connect

- The new course management tool at UBC
- Launched in September 2013
  - Vista (the old system) was abandoned abruptly

Problems

- Extremely slow,
- Many bugs and broken features,
- Too many unneeded features!

Class activity

- What do you think is the most important mistake that these projects made?
- How would you have done things differently?

Question:

- What is the biggest mistake?

A. The underestimation of complexity
B. Excessive schedule pressure
C. Communications breakdowns due to people working in isolation
D. Lack of due diligence
E. Use of Java

Question:

- What is the biggest mistake?

A. **The underestimation of complexity**

B. **Excessive schedule pressure**

C. **Communications breakdowns due to people working in isolation**

D. **Lack of due diligence**

E. Use of Java

Why do projects fail?

- Unrealistic project **goals**
- Inaccurate **estimates** of needed resources
- Badly defined system **requirements**
- Unmanaged **risks**
- Poor **communication**
- Poor project **management**
- Stakeholder politics / pressure
- Improper testing
- ….

What is missing?

- A software development process to ensure desirable characteristics
  - Accountability: who is doing what?
  - Timeliness: when is this done?
  - Observability: what is the status of the overall plan?
  - Efficiency: is there redundant work happening? Does everyone on the team have some task to do?

- Like any other engineering discipline

Questions?