In 2000, total sales of software reached approximately $180 billion, supported by a large workforce encompassing 697,000 software engineers and 585,000 computer programmers.

The global software market had total revenues of $292.9 billion in 2011. -- MarketLine

The Software Problem

- Scale
- The cost of change
- Users as bugs
- Evolution yields complexity and bugs
- Software engineering matters
The Cost of Change

Steve McConnell, Software Quality at Top Speed, Software Development, August 1996
Evolution yields Complexity/Bugs

Figure 4 Serial and average growth trends of a particular attribute

Figure 7 Complexity growth during the interval prior to each release

Belady & Lehman, A Model of Large Program Development, IBM Systems Journal, (15)3, 1976
Users as Bugs

![Bar chart showing percent of software maintenance effort by category.]

- Emergency Program Fixes: 12.4%
- Routine Debugging: 9.3%
- Accom Changes to Input Data, Files: 17.4%
- Accom Changes to Hardware, OS: 6.2%
- Enhancements for Users: 41.8%
- Improve Documentation: 5.5%
- Improve Code Efficiency: 4.0%
- Other: 3.4%
Scale, Bugs, Evolution

It’s a wonder software works at all.
And it’s so cheap, too.
What’s up with that?
RAYTHEON HAS SAVED $17.2 million in software costs since 1988, when its equipment division began using rigorous development processes that doubled its programmers' productivity and helped them to avoid making expensive mistakes.
The Changing Face of Software

• Applications
  - Web 2.0, Mobile 2.0, …
  - Ubiquitous computing
  - Developing world
  - Big data, AI, …

• Methodologies
  - Open Source
  - Agile (XP, Scrum)

• Technologies
  - Web services, javascript, AJAX, JQuery, …
  - Programming environments (Eclipse), AOP
  - Component-based, Model-driven software development

Do we rewrite the rules, or just reinterpret them?
Technical Themes of the Course

Scale

All of computer science, especially CS research, is about managing scale. So is SE.

Risk

SE is all about managing risk. Doing something important requires taking risks. SE seeks to increase upside risk (great products), while decreasing downside risks (late, buggy, etc.)
Goals of the Course

• Learn **foundational concepts** of SE
• Exposure to the foundational literature
• Improve reading papers critically
• Improve discussing technical ideas
• Take ideas and skills into your own practice
• Ultimately, **software engineering literacy**
  • *Conversant in issues – think and talk like a software engineer(ing researcher)*
OUR CONTRACT
My Promise

Authentic practice

A minimum of busy work
Your Promise

Come prepared every day
Rest of Today

• Structure of course
• Grading
• How to read and discuss papers
• Project
• Questions (at any time)
First Week: Intro to Agile Process

• You come from many backgrounds
• Seen different variants of software process
• I’m going to introduce a generic Agile Process
• Will be point of contrast for much of course
• Also will be used in project
• Looks like a lot of reading, but actually not many words, and goes fast
  • Don’t skip the side bars pictures!
  • Great examples, great exercises
  • Don’t have to do the crosswords
The Computer for the 21st Century

Mark Weiser
Palo Alto Research Center, Xerox, CA, USA

Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence.

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it. Consider writing, perhaps the first information technology. The ability to represent spoken language symbolically for long-term storage and information from the limits of individual memory. Today, this technology is ubiquitous in industrialized countries. Not only do books, magazines and newspapers convey written information, but we do street signs, billboards, shop signs and even graffiti. Candy wrappers are covered in writing. The constant background presence of these products of "literacy technology" does not require active attention, but the information to be transmitted is ready at a glance. It is difficult to imagine modern life otherwise.

Silicon-based information technology, in contrast, is far from having become part of the environment. More than 50 million personal computers have been sold, and the computer nonetheless remains largely in a world of its own. It is approachable only through complex jargon that has nothing to do with the tasks for which people use computers. The state of the art is perhaps analogous to the period when scribes had to know as much about making ink or baking clay as they did about writing. The arcane aura that surrounds personal computers is not just a "user interface" problem. My colleagues and I at the Xerox Palo Alto Research Center think that the idea of a "personal" computer itself is misplaced and that the vision of laptop machines,dial-up and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computers an integral, invisible part of people's lives. We are therefore trying to conceive a new way of thinking about computers, one that takes into account the human world and allows the computers themselves to vanish into the background.

Such a disappearance is a fundamental consequence not of human psychology, but of computer technology. Wherever people learned something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading. Computer scientist, economist and Nobel laureate Herbert A. Simon calls this phenomenon "compiling": philosopher Michael Polanyi calls it the " tacit dimension"; psychologist J. J. Gibson calls it "a visual invariant"; philosophers Hans Georg Gadamer and Martin Heidegger call it the "horizon" and the "ready-to-hand"; John Seely Brown of PARC calls it the "perpetuity." All say, essentially, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond  

*Reprinted with permission. Copyright © 1991 by Scientific American. Inc. All rights reserved. This article first appeared in Scientific American, Vol 265, No. 3 (September 1991), pp. 94-104.

Mobile Computing and Communications Review, Volume 3, Number 3
In-Class Discussion

“Socratic Circles” round-table discussion
More dynamic, less controlled, more open-ended
“Peer learning”
Alternative Formats Throughout

A half-dozen or more class sessions will use alternative formats

- “Workshop”
  - working or problem-solving in small groups
- Lecture (talk) – me or visitor
- *No class on Tuesday before Thanksgiving* (week 9 – work on your project)
First Few In-Class Discussions

Directed by me. Repeat:
1. I will ask a question (see next slide)
2. I will select one of you to answer
3. An assessment of the answer will be made
4. If appropriate, I will open the question for you to discuss with neighbors
5. I will select someone to elaborate the first answer [iterate as necessary]

(I will experiment with variants of the above)
In-Class Discussion – First $K$ Weeks

Questions will come from a few places:

• Paper-reading rubric
  – people problem, technical problem…

• One of your questions from the paper

• One of my questions

(The more and better your questions are, the fewer of my questions I’ll ask. :)