The Software Problem

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

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


Address http://www.nist.gov/public_affairs/releases/n02-10.htm Go software perc
Google software percentage economy NIST Search Web 119 blocked AutoFill

NIST News Release NIST Hom

A-Z subject index | Search NIST webspace | Contact NIST | Home

oftware Errors Cost U.S. Economy $59.5 Billion Annually
ST Assesses Technical Needs of Industry to Improve Software-Testing

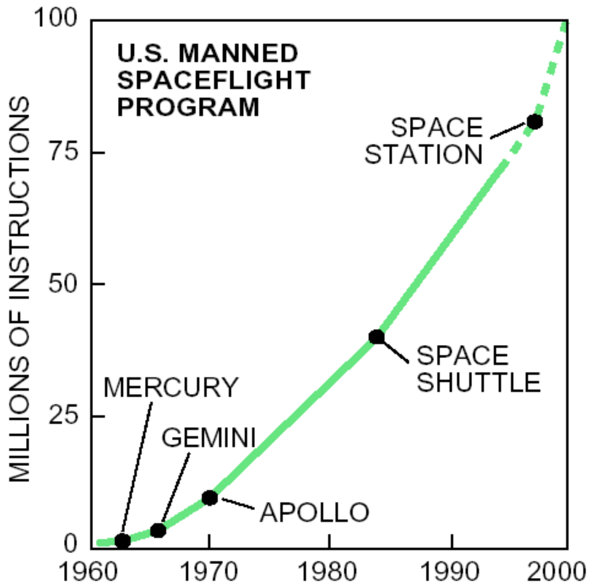
R IMMEDIATE RELEASE: Contacts: Michael Newman
e 28, 2002 (301) 975-3025

NIST 2002-10

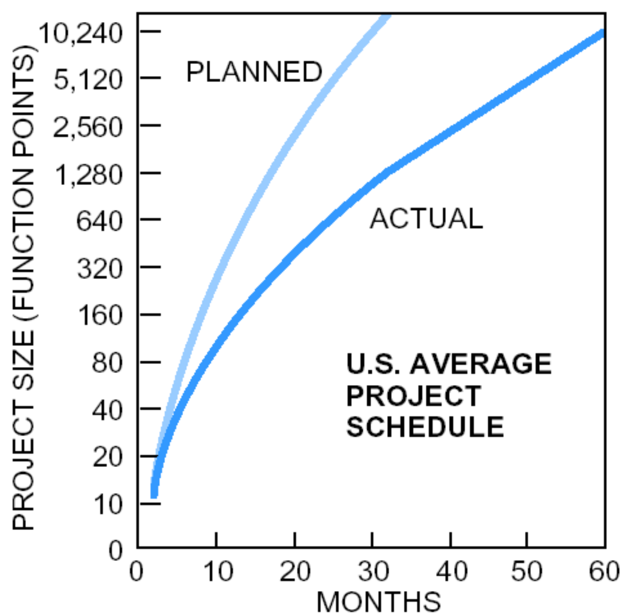
ware bugs, or errors, are so prevalent and so detrimental that they cost the
. economy an estimated $59.5 billion annually, or about 0.6 percent of the
gross domestic product, according to a newly released study commissioned by the Department of Commerce's National Institute of Standards and Technology (NIST). At the national level, over half of the costs are borne by software users and the remainder by software developers/vendors.
The study also found that, although all errors cannot be removed, more than a third of these costs, or an estimated $22.2 billion, could be eliminated by an improved testing infrastructure that enables earlier and more effective

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

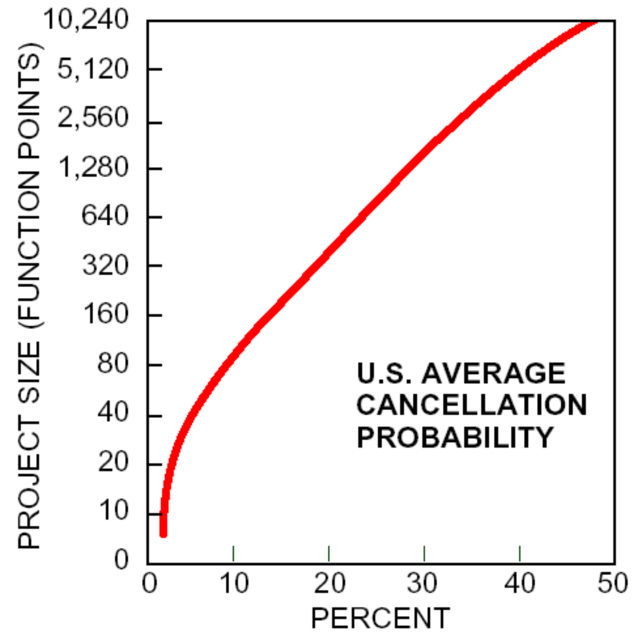
Scale



SOURCE: Barry W. Boehm



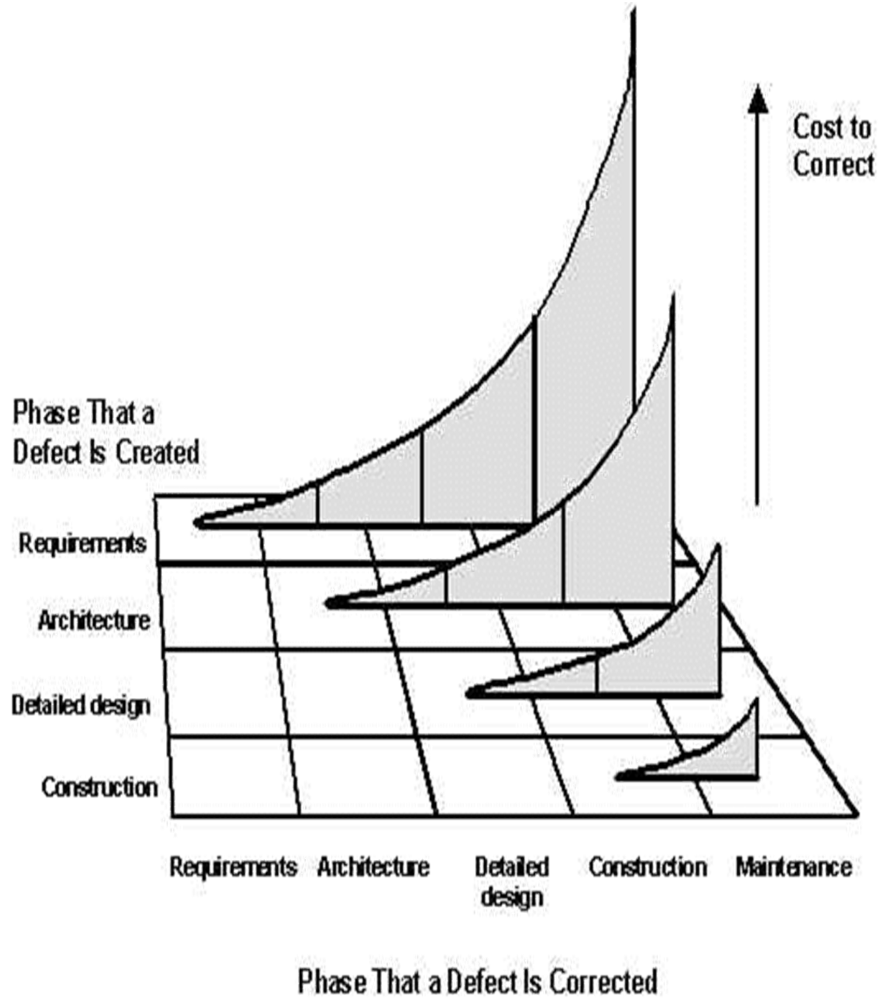
SOURCE: Software Productivity Research



SOURCE: Software Productivity Research

Gibbs, Software's Chronic Crisis, *Sci. Am.*, Sept. 1994

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

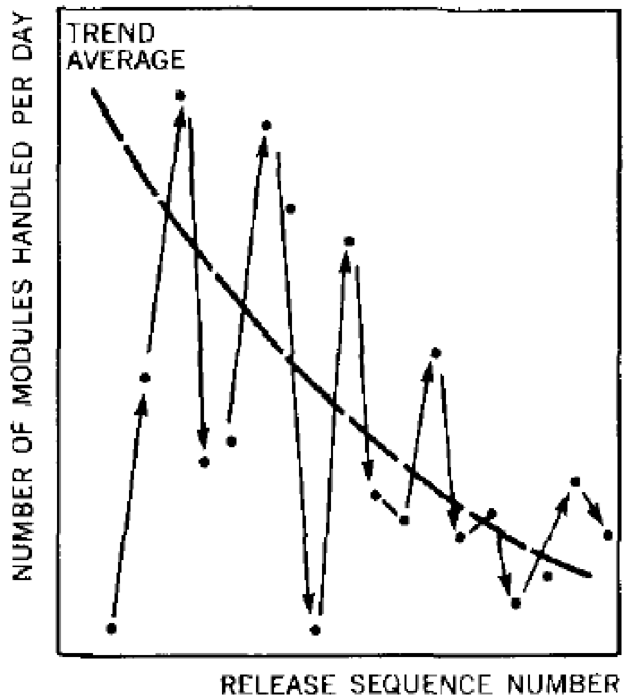
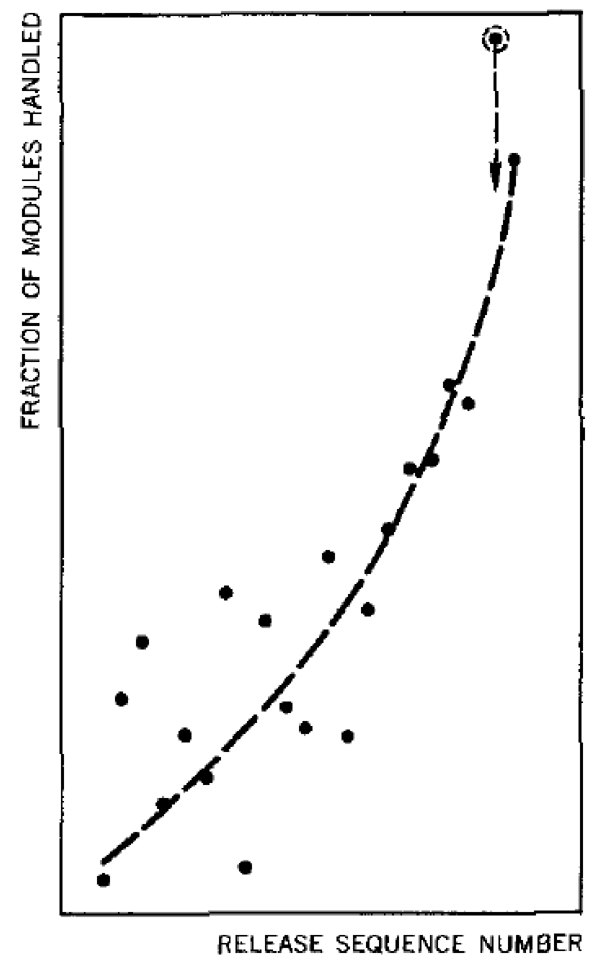
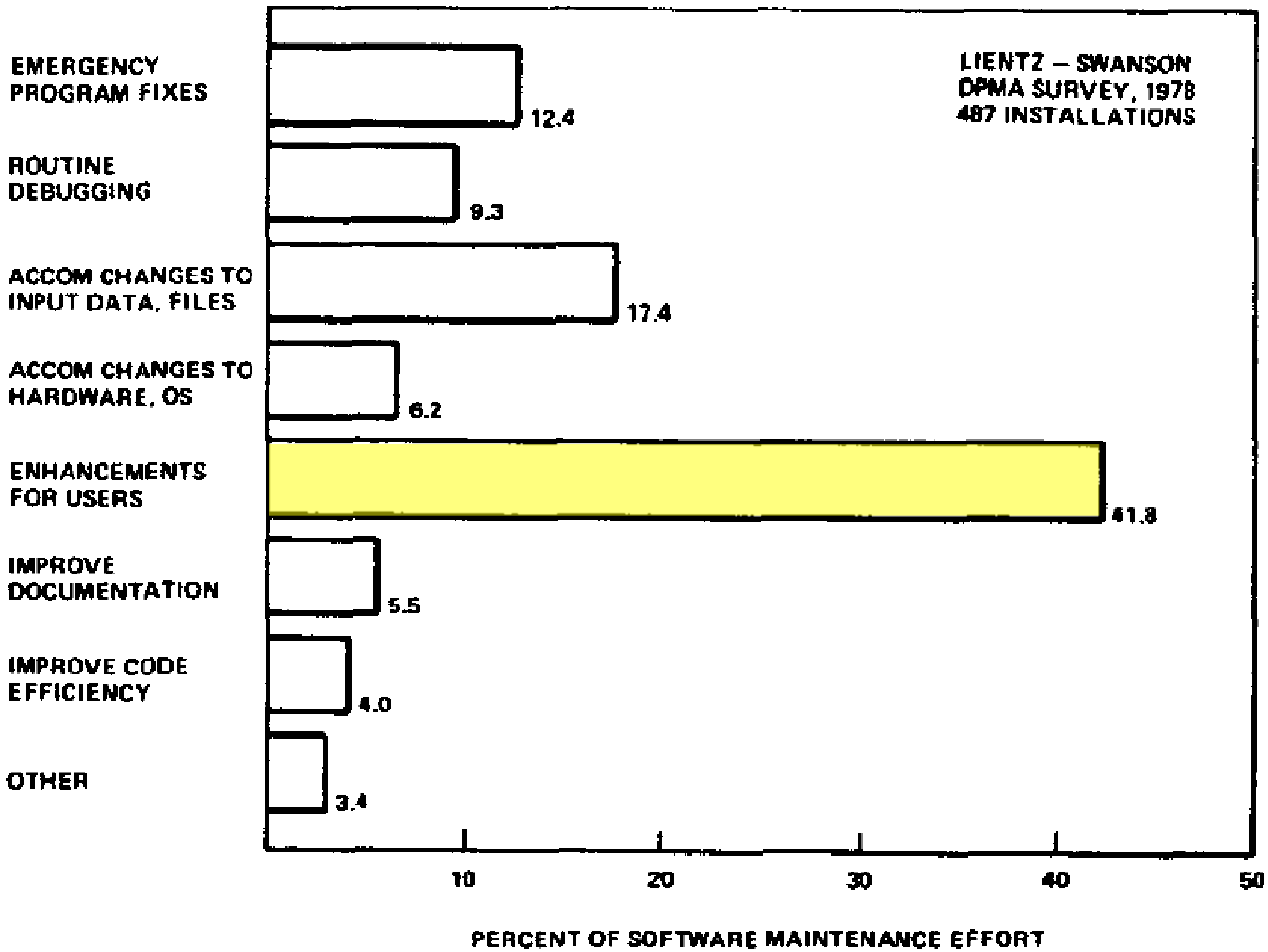


Figure 7 Complexity growth during the interval prior to each release



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

Users as Bugs



Boehm, SE as it is, ICSE'79

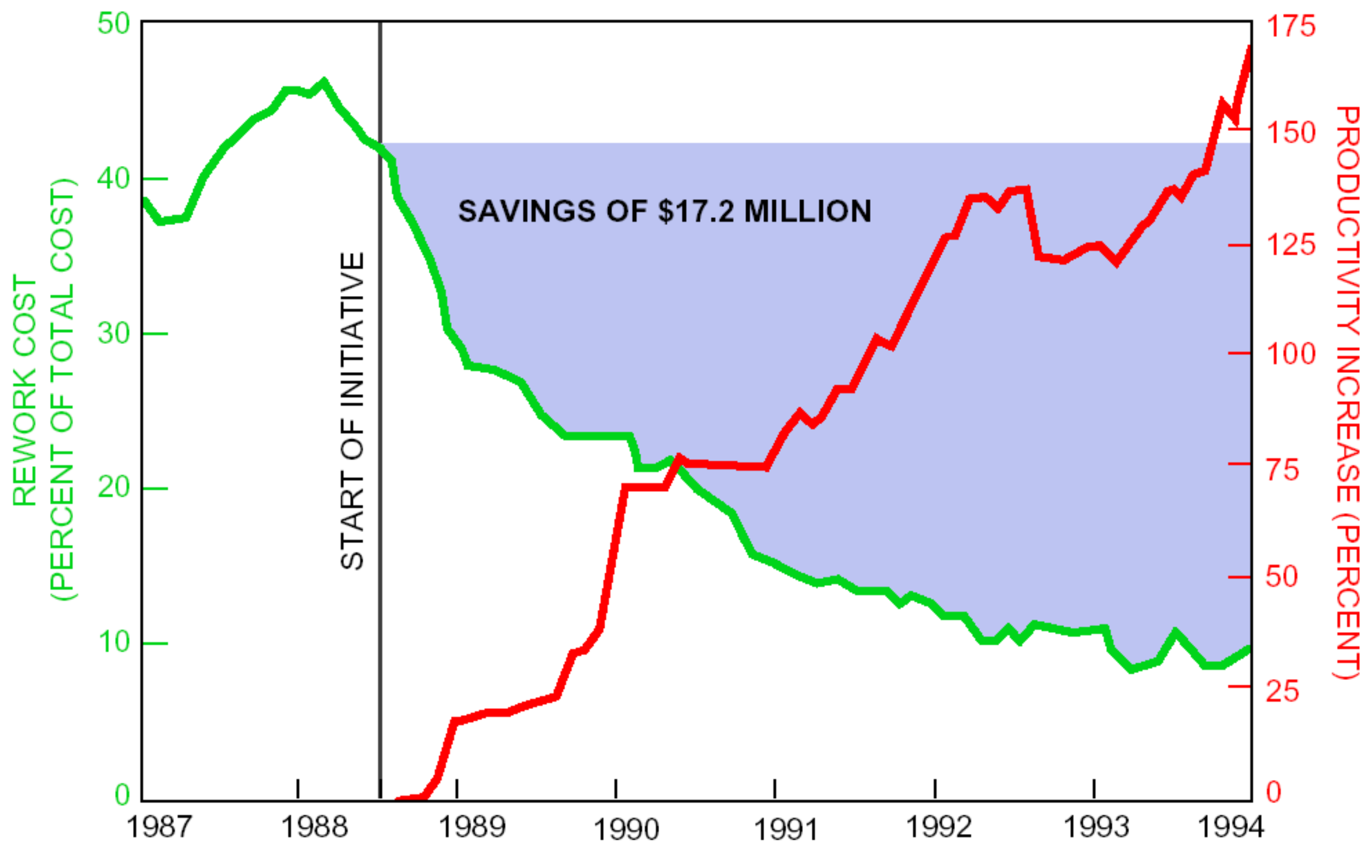
Scale, Bugs, Evolution

It’s a wonder software works at all.

And it’s so cheap, too.

What’s up with that?

Software Engineering Works



SOURCE: Raytheon

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.

SCIENTIFIC AMERICAN *September 1994* 91

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 longterm storage freed 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 so 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 for use 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, dynabooks and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computing 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 technology but of human psychology. Whenever people learn 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 Nobelist Herbert A. Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist J. J. Gibson calls it "visual invariants"; philosophers Hans Georg Gadamer and Martin Heidegger call it the "horizon" and the "ready-to-hand"; John Seely Brown of PARC calls it the "periphery." All say, in essence, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals.

The idea of integrating computers seamlessly into the world at large runs counter to a number of present-day trends. "Ubiquitous computing" in this context does not mean just computers that can be carried to the beach, jungle or airport. Even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box. By analogy with writing, carrying a super-lap-top is like owning just one very important book. Customizing this book, even writing millions of other books, does not begin to capture the real power of literacy.

Furthermore, although ubiquitous computers may use sound and video in addition to text and graphics, that does not make them "multimedia computers." Today's multimedia machine makes the computer screen into a demanding focus of attention rather than allowing it to fade into the background.

Perhaps most diametrically opposed to our vision is the notion of virtual reality, which attempts to make a world inside the computer. Users don special goggles that project an artificial scene onto their eyes; they wear gloves or even bodysuits that sense their motions and gestures so that they can move about and manipulate virtual objects. Although it may have its purpose in allowing people to explore realms otherwise inaccessible — the insides of cells, the surfaces of distant planets, the information web of data bases — virtual reality is only a map, not a territory. It excludes desks, offices, other people not wearing goggles and bodysuits, weather, trees, walks, chance encounters and, in general, the infinite richness of the universe. Virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists.

Indeed, the opposition between the notion of virtual reality and ubiquitous, invisible computing is so strong that some of us use the term "embodied virtuality" to refer to the process of drawing computers out of their electronic shells. The "virtuality" of computer-readable data — all the different ways in which they can be altered, processed and analyzed — is brought into the physical world.

How do technologies disappear into the background? The vanishing of electric motors may serve as an instructive example. At the turn of the century, a typical workshop or factory contained a single engine that drove dozens or hundreds of different machines through a system of shafts and pulleys. Cheap, small, efficient electric motors made it possible first to give each tool its own source of motive force, then to put many motors into a single machine.

A glance through the shop manual of a typical automobile, for example, reveals 22 motors and 25 solenoids. They start the engine, clean the windshield, lock and unlock the doors, and so on. By paying careful attention, the driver might be

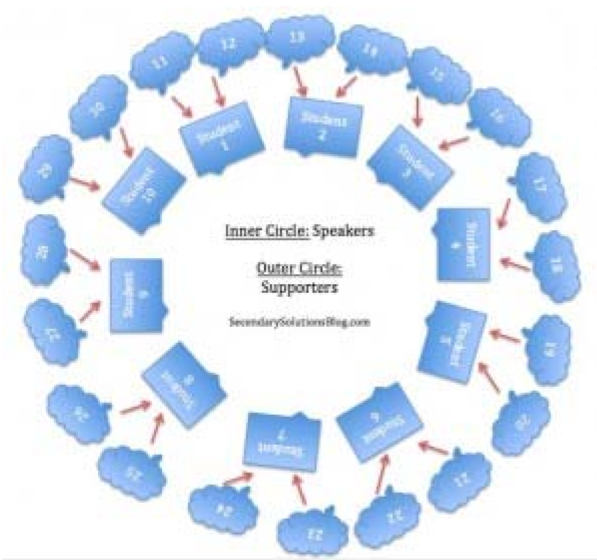
*Reprinted with permission. Copyright (c) 1991 by Scientific American, Inc. All rights reserved. This article first appeared in Scientific American, Vol. 265, No. 3 (September 1991), pp. 94-104

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. :)