Software Project Management

Lecture 1

Why Software Project Fail?

- Management problems were more frequently dominant cause than technical problems
- Schedule overruns were more common (89%) than cost overruns (62%)

KPGM's Survey in UK

- Primary causes of software runaway
  - Project Objectives not fully specified
  - Bad planning and estimating
  - Technology new to the organization
  - Inadequate/No project management methodology
  - Insufficient senior staff on the team
  - Poor performance by Supplier of hardware/software

What makes a software project success?

- User involvement – 20 points
- Executive Support – 15 points
- Clear Business Objectives – 15 points
- Experienced Project Manager – 15 points
- Small milestones – 10 points
- Firm basic requirements – 5 points
- Competent staff – 5 points
- Proper planning – 5 points
- Ownership – 5 points
- Others – 5 points

Chaos ten – Standish Group Report

Software Project Management FIRST

- “Organizations that attempt to put software engineering discipline in place before putting project management discipline in place are domed to fail”

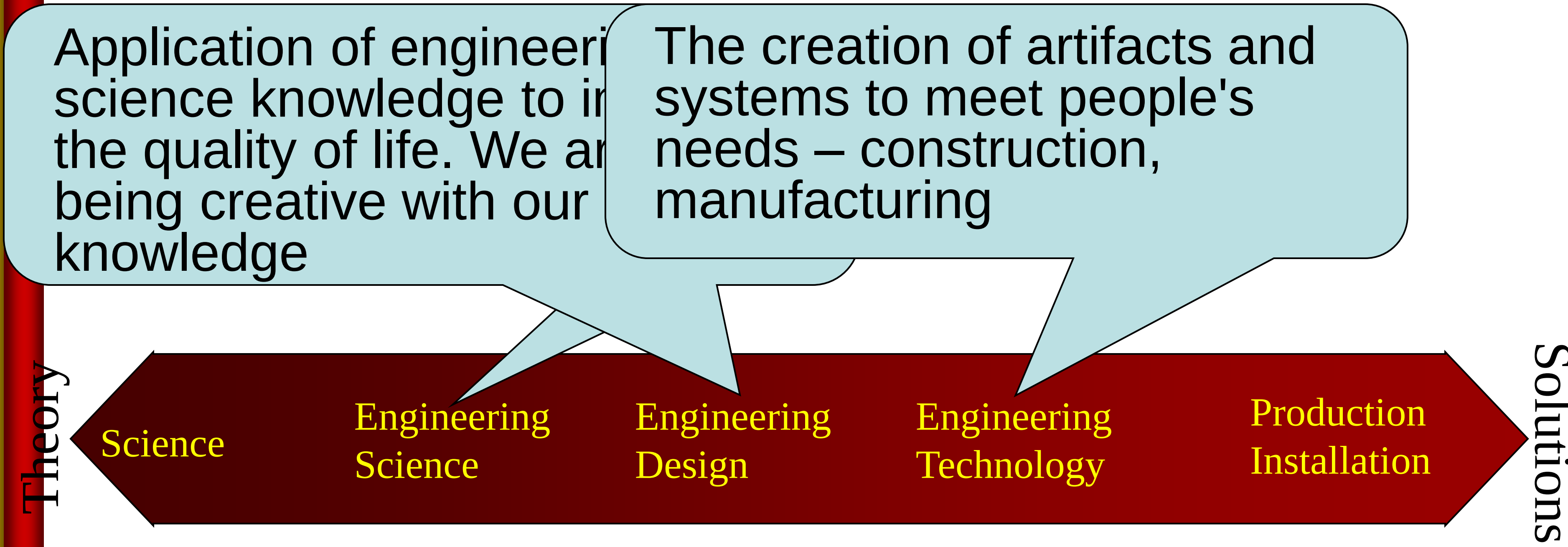
  Software Engineering Institute

What is Software Engineering?

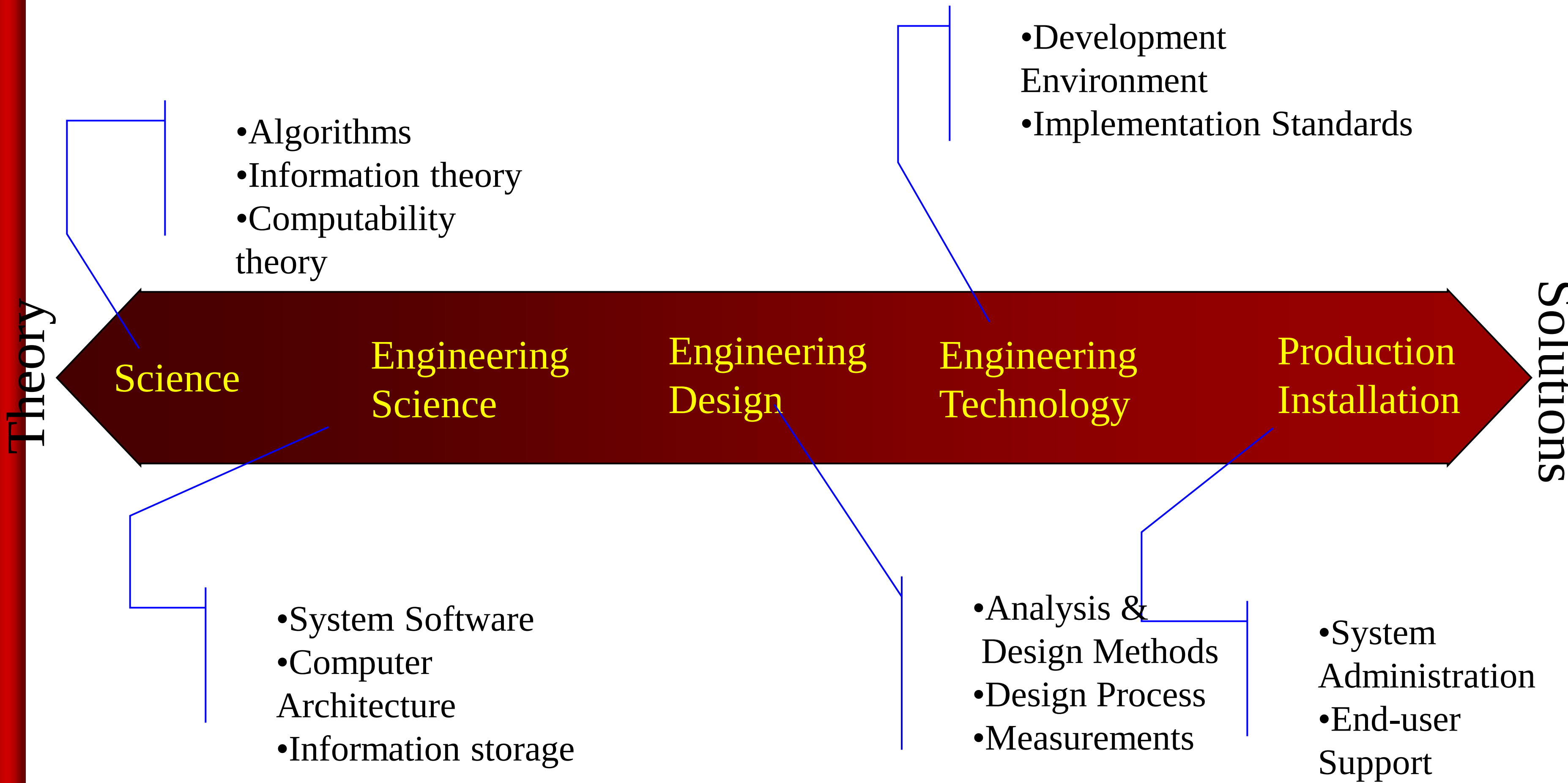
- Software Engineering is an engineering discipline
- Concerns with the development of software by applying Engineering Principles
- Goal is the cost-effective development of software systems
- Software Engineering [IEEE-93]*:
  - The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
  - The study of approaches as in 1

- *IEEE-93: IEEE Standard Glossary & Software Engineering Terminology

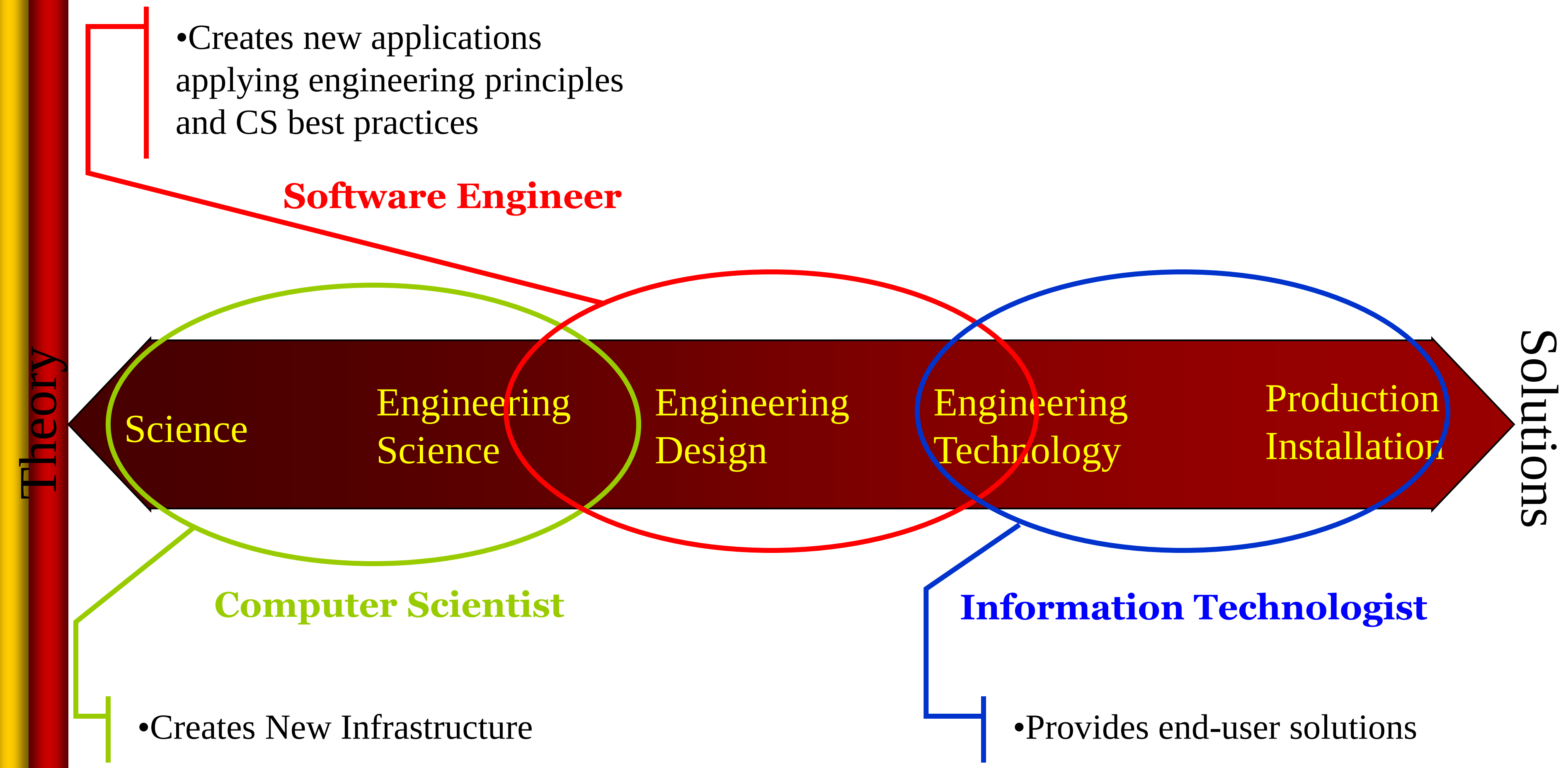
Knowledge Spectrum



A Computing Spectrum



Computing Professions



"Software Engineering"
Barry Boehm ICSE 2006 Keynote Address May 25, 2006

- SE definition (Based on Webster definition of "engineering")
  The application of science and mathematics by which the properties of software are <u>made useful to people</u>

- Includes computer science and the sciences of making things useful to people
  - Behavioral sciences, economics, management sciences

What is Software Engineering?

- What is Software?
- What is Engineering?

The Nature of Software

F.P. Brooks, “No Silver bullet: Essence and Accident of Software Engineering”, IEEE Computer Magazine, April 1987

No Silver bullet by F. P. Brook

- Motivation of Paper
  - Missed schedule, blown budgets and flawed products
  - Something to make software cost drop as rapidly as computer hardware cost do
  - “We look to the horizon of a decade hence we see **no silver bullet.** There is no single development, in either technology or in management technique that by itself promises even one order-of-magnitude improvement in productivity, in reliability, in simplicity”

Nature of software

- Not only are there no silver bullets now in view, the very nature of software makes it unlikely that there will be any
- **Essence**
  - The difficulties inherent in the nature of software
- **Accidents**
  - Those difficulties that today attend its production but are not inherent

- The essence of a software entity is a construct of interlocking concepts (conceptual construct)
- I believe that hard part of building software to be the specification, design, and testing of this conceptual construct, not the labor of representing it and testing the fidelity of the representation. We still make syntax errors, to be sure; but they are fuzz compared with the conceptual errors in most systems

Essence and Accident of SE

- **Irreducible essence of modern software**
  - Complexity
  - Conformity
  - Changeability, and
  - Invisibility.

Complexity

- No two parts are alike
- Software systems have many times more states even than computer hardware.
- Scaling-up of software entity is not merely a repetition of the same elements in larger sizes. It is necessarily an increase in the number of different elements. In most cases, the elements interact with each other in some non-linear fashion, and the complexity of the whole increases much more than linearly.

Complexity

- The complexity of software in an essential property, not an accidental one.
- Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence.
- Models are abstractions – simplification of reality

Complexity

- From complexity comes the difficulty of
  - Communication among team members which leads to product flaws, cost overrun, schedule delays.
  - Enumerating
  - Less understanding of all the possible states of the program

Conformity

- Natural Science – God VS Software – People
- Much of the complexity that he (developer) must master is arbitrary complexity, forced without rhyme or reason by many human institutions and systems to which his interfaces must conform.
- These (standards) differ from interface to interface, and from time to time, not because of necessity but only because they were designed by different people, rather than by God.

Changeability

- The software entity is constantly subject to pressures for change. Of course, so are buildings, cars, and computers. But manufactured things are infrequently changed.
- Software of a system embodies its function, and the function is the part that most feels the pressure of change
- In part it is because software can be changed more easily – it is pure thought stuff, infinitely malleable

- Building do in fact get change but the high cost of change understood by all serve to dampen the whims of the changes
- The software product is embodied in a cultural matrix of applications, users, laws, and machine vehicles. These change continually, and their changes inevitably force change upon the software product.

Invisibility

- Software is invisible and un-visualizable.
- A geometric reality is captured in a geometric abstraction
- The reality of software is not inherently embedded in space
- As soon as we attempt to diagram software structure, we find it to constitute not one, but several, general directed graphs superimposed one upon another.
- This lack not only obstruct the process of design within one mind, it severely hinders communication among minds.

Engineering

- Derived from old French engin = skill, which stems from Latin ingenium = ability to invent, brilliance, genius
- The word was created in the 16th century and originally described a profession that we would probably call an artistic inventor *("Encyclopedia" by The Software Toolworks, 1991)*
- Engineering is applied science
- Engineering is the application of science for practical purposes
- Engineers put theory into practice.

Engineering

- ABET, USA
  - The profession in which a knowledge of the mathematical and natural sciences gained by study, experience, and practice is applied with judgment to develop ways to utilize, economically, the material and forces of nature for the benefit of mankind
- The profession of or work performed by an engineer. Engineering involves the knowledge of the mathematical and natural sciences (biological and physical) gained by study, experience, and practice that are applied with judgment and creativity to develop ways to utilize the materials and forces of nature for the benefit of mankind (International Technology Education Association).

Engineering Software

- Engineering methods put a lot of emphasis on planning before you build
- Design, Plan and then Construct
- Construction is much bigger in both cost and time than design and planning
  - 80-90%
- What is the proportion in Software Engineering?
  - 10-30%
  - Software construction is so cheap as to be free
  - All effort is of design

Engineering Software

- Design requires creative and talented people
- Creative processes are not easily planned, and so predictability may well be an impossible target
  - Everything else in software development depends on the requirements. If you cannot get stable requirements you cannot get a predictable plan
- This raises an important question about the nature of design in software compared to its role in other branches of engineering
- “We should be very wary of the traditional engineering metaphor for building software. It's a different kind of activity and requires a different process” (Martin Fowler, The New methodology)

Reading Material

- “Software's Chronic Crisis”, W. Wayt Gibbs, Scientific American, Vol. 271, No. 3, September 1994

  OR
- “Who killed the Virtual Case File”, Harry Goldstien, IEEE Spectrum, September 2005

Q&A