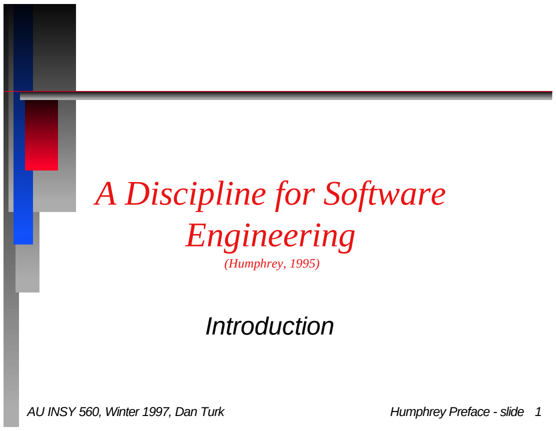
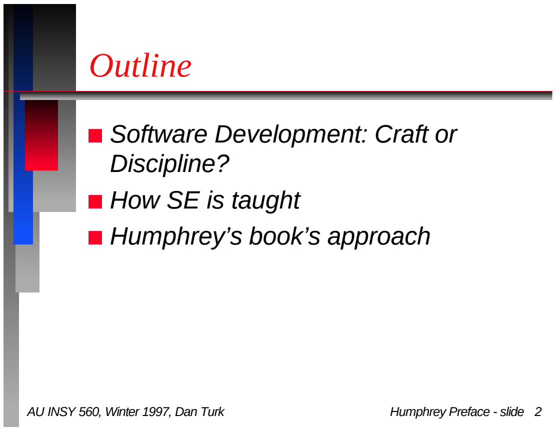


A Discipline for Software Engineering

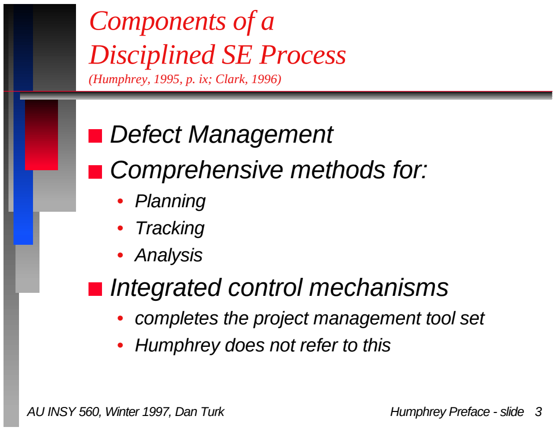
(Humphrey, 1995)

Introduction



Outline

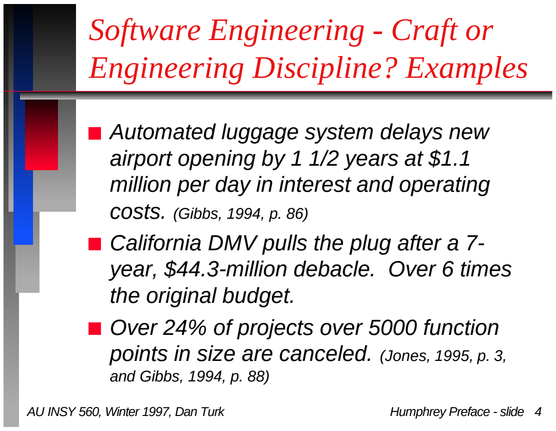
- Software Development: Craft or Discipline?
- How SE is taught
- Humphrey's book's approach



Components of a Disciplined SE Process

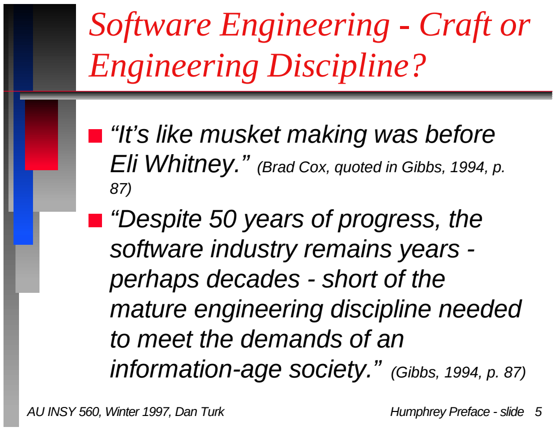
(Humphrey, 1995, p. ix; Clark, 1996)

- Defect Management
- Comprehensive methods for:
  - Planning
  - Tracking
  - Analysis
- Integrated control mechanisms
  - completes the project management tool set
  - Humphrey does not refer to this



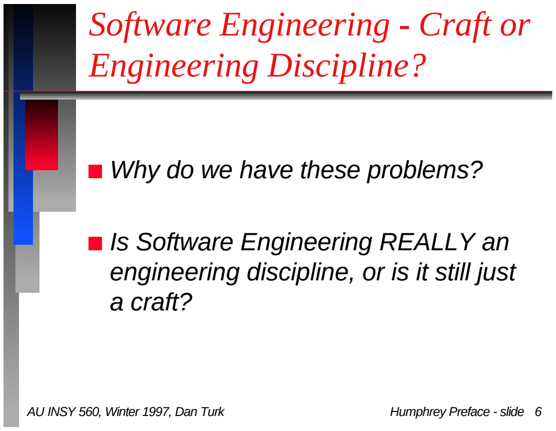
Software Engineering - Craft or Engineering Discipline? Examples

- Automated luggage system delays new airport opening by 1 1/2 years at $1.1 million per day in interest and operating costs. (Gibbs, 1994, p. 86)
- California DMV pulls the plug after a 7-year, $44.3-million debacle. Over 6 times the original budget.
- Over 24% of projects over 5000 function points in size are canceled. (Jones, 1995, p. 3, and Gibbs, 1994, p. 88)



Software Engineering - Craft or Engineering Discipline?

- "It's like musket making was before Eli Whitney." (Brad Cox, quoted in Gibbs, 1994, p. 87)
- "Despite 50 years of progress, the software industry remains years - perhaps decades - short of the mature engineering discipline needed to meet the demands of an information-age society." (Gibbs, 1994, p. 87)



Software Engineering - Craft or Engineering Discipline?

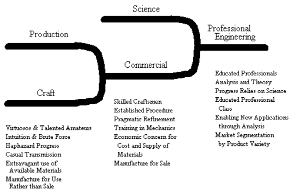
- Why do we have these problems?
- Is Software Engineering REALLY an engineering discipline, or is it still just a craft?

“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

- Origins of the Term “Software Engineering”
  - 1968 NATO workshop
- Def: Engineering (p. 15)
  - “a set of current best practices for development.”
  - “the disciplined application of scientific knowledge to resolving conflicting constraints and requirements for problems of immediate, practical significance.”

“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

- Evolution of an Engineering Discipline


Science
Production
Professional Engineering
Commercial
Craft

Virtuosos & Talented Amateurs
Intuition & Brute Force
Haphazard Progress
Casual Transmission
Extravagant use of Available Materials
Manufacture for Use Rather than Sale

Skilled Craftsmen
Established Procedure
Pragmatic Refinement
Training in Mechanics
Economic Concern for Cost and Supply of Materials
Manufacture for Sale

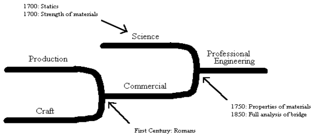
Educated Professionals
Analysis and Theory
Progress Relies on Science
Educated Professional Class
Enabling New Applications through Analysis
Market Segmentation by Product Variety

The lower lines track the technology, and the upper lines show how the entry of production skills and scientific knowledge contribute new capability to the engineering practice.

Shaw, 1994, "Prospects for an Engineering Discipline of Software", IEEE Software, p. 17.

“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

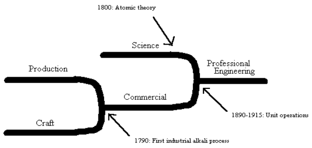
- Evolution of Civil Engineering


1700: Statics
1700: Strength of materials
Science
Production
Professional Engineering
Commercial
Craft
1750: Properties of materials
1850: Full analysis of bridge
First Century: Romans

Shaw, 1994, "Prospects for an Engineering Discipline of Software", IEEE Software, p. 17.

“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

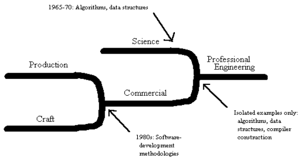
- Evolution of Chemical Engineering


1800: Atomic theory
Science
Production
Professional Engineering
Commercial
Craft
1890-1915: Unit operations
1790: First industrial alkali process

Shaw, 1994, "Prospects for an Engineering Discipline of Software", IEEE Software, p. 17.

“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

- Evolution of Software Engineering


1965-70: Algorithms, data structures
Science
Production
Professional Engineering
Commercial
Craft
Isolated examples only: algorithms, data structures, compiler construction
1980s: Software-development methodologies

Shaw, 1994, "Prospects for an Engineering Discipline of Software", IEEE Software, p. 17.

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

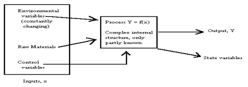
- “Information is much easier to store, describe, and manipulate than knowledge.” (p. 61)
- “The level of knowledge that a process has reached determines how a process should be controlled, whether and how it can be automated, the key tasks of the workforce, and other major aspects of its management.” (p. 61)

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- Examples
  - Chaparral Steel
    - doubled mill output
    - used original electric furnace and caster
  - Semiconductor manufacturers
    - routinely obtain 80% yield after only a few years, when starting with 40%
    - minimal capital investment
    - changed mfg. process (procedures, control adjustments, raw materials recipes, etc.)

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- Definition: Technological Knowledge
  - “Technological knowledge [is] understanding the effects of the input variables on the output.” (p. 62)
- Diagram of a Process


Environmental variables (constantly changing)
Raw Materials
Control variables
Inputs, x
Process Y = f(x)
Complex internal structure, only partly known.
Output, Y
State variables

Bohn, Roger. (1994). "Measuring and Managing Technological Knowledge", Sloan Management Review, 36:1 (Fall), 62.

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

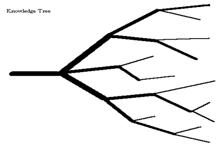
- Eight Stages of Knowledge (p. 63, 64)

	Name	Comment	Typical Knowledge Form
1	Complete Ignorance		Nowhere
2	Awareness	Pure Art	Tacit
3	Measure	Pretechnological	Written
4	Control of the mean	Scientific method feasible	Written & embodied in hardware
5	Process capability	Local recipe	Hardware and operating manual
6	Process characterization	Tradeoffs to reduce costs	Empirical equations (numerical)
7	Know why	Science	Scientific formulas & algorithms
8	Complete knowledge	Nirvana	

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- Evolution of Knowledge and Performance (p. 65)

“Knowledge about [a] process and how to run it is incomplete and develops gradually through various kinds of learning.” (p. 61)

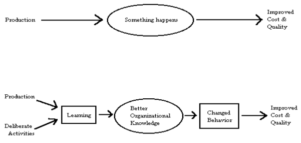


Each branch represents a variable affecting the process.
The thickness of each branch represents the importance of that variable

Bohn, Roger. (1994). "Measuring and Managing Technological Knowledge", Sloan Management Review, 36:1 (Fall), p. 65.

“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- Knowledge-based Learning and Organizational Improvement

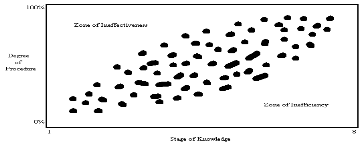


Bohn, Roger. (1994). "Measuring and Managing Technological Knowledge", Sloan Management Review, 36:1 (Fall), p. 66.

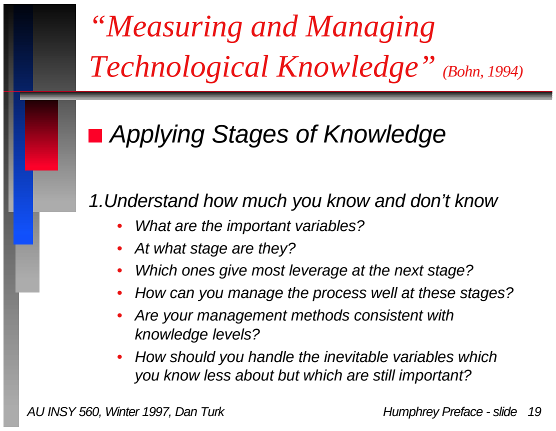
“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- Balance Between Degree of Procedure & Stage of Knowledge

“Automation of a large, complex, poorly understood, conventional manufacturing process leads to a large, complex, poorly understood, unreliable, expensive, and automated manufacturing process.” (p.



Bohn, Roger. (1994). "Measuring and Managing Technological Knowledge", Sloan Management Review, 36:1 (Fall), p. 67.

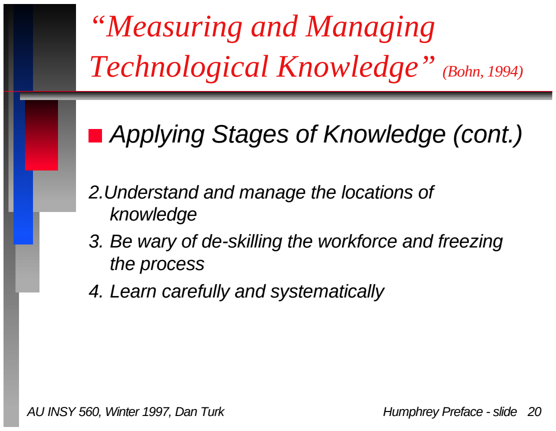


“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- *Applying Stages of Knowledge*

1.Understand how much you know and don’t know

- *What are the important variables?*
- *At what stage are they?*
- *Which ones give most leverage at the next stage?*
- *How can you manage the process well at these stages?*
- *Are your management methods consistent with knowledge levels?*
- *How should you handle the inevitable variables which you know less about but which are still important?*



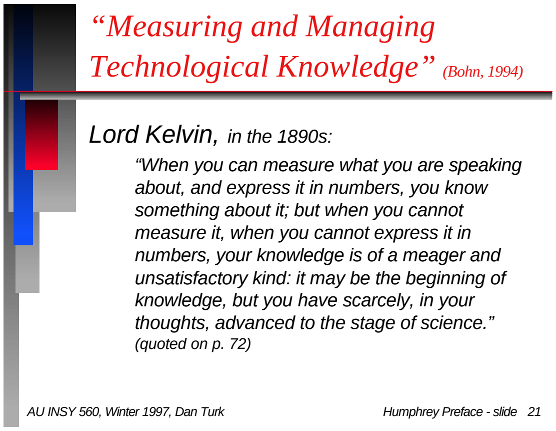
“Measuring and Managing Technological Knowledge” (Bohn, 1994)

- *Applying Stages of Knowledge (cont.)*

2.Understand and manage the locations of knowledge

3. Be wary of de-skilling the workforce and freezing the process

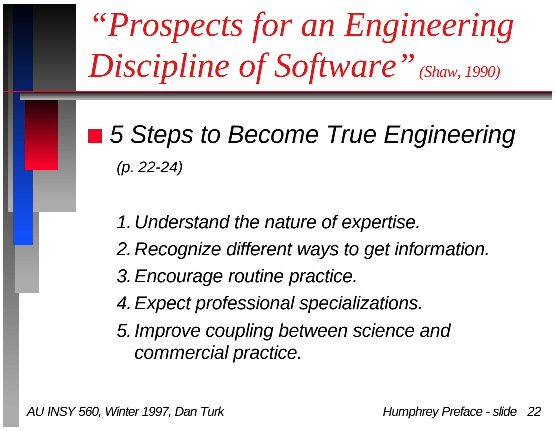
4. Learn carefully and systematically



“Measuring and Managing Technological Knowledge” (Bohn, 1994)

Lord Kelvin, in the 1890s:

> *“When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind: it may be the beginning of knowledge, but you have scarcely, in your thoughts, advanced to the stage of science.”* *(quoted on p. 72)*

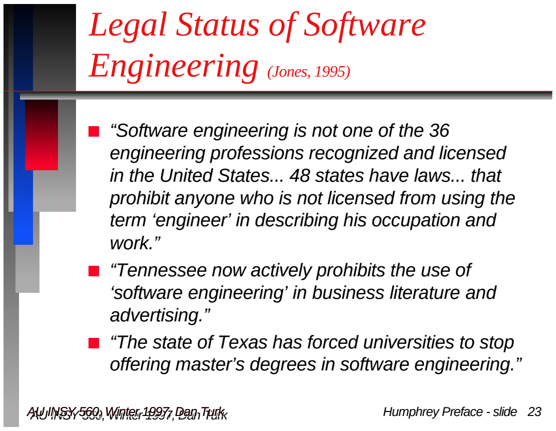


“Prospects for an Engineering Discipline of Software” (Shaw, 1990)

- *5 Steps to Become True Engineering*

(p. 22-24)

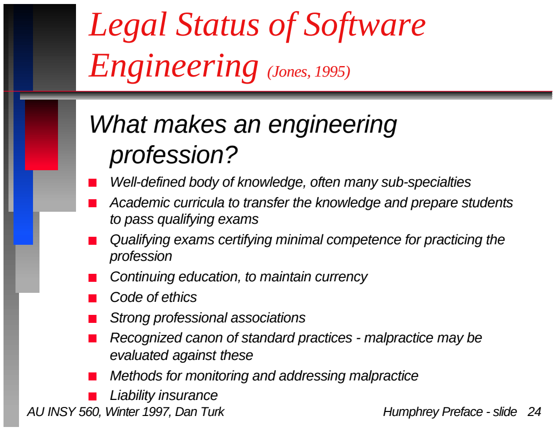
1. *Understand the nature of expertise.*
2. *Recognize different ways to get information.*
3. *Encourage routine practice.*
4. *Expect professional specializations.*
5. *Improve coupling between science and commercial practice.*



Legal Status of Software Engineering (Jones, 1995)

- *“Software engineering is not one of the 36 engineering professions recognized and licensed in the United States... 48 states have laws... that prohibit anyone who is not licensed from using the term ‘engineer’ in describing his occupation and work.”*
- *“Tennessee now actively prohibits the use of ‘software engineering’ in business literature and advertising.”*
- *“The state of Texas has forced universities to stop offering master’s degrees in software engineering.”*

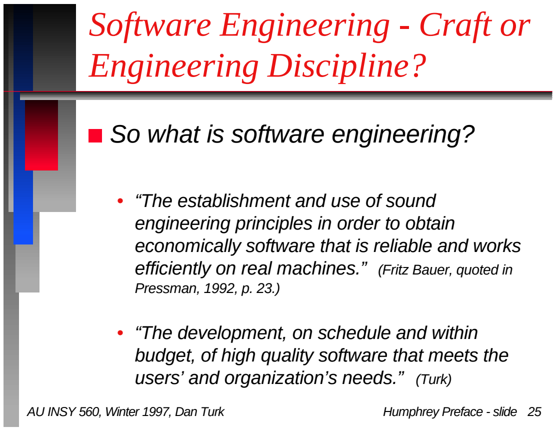
AU INSY 560, Winter 1997, Dan Turk



Legal Status of Software Engineering (Jones, 1995)

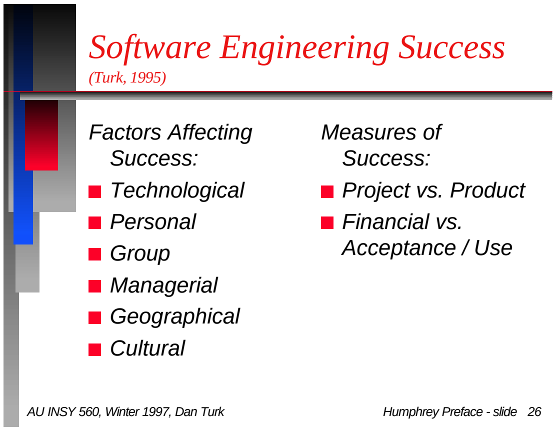
What makes an engineering profession?

- *Well-defined body of knowledge, often many sub-specialties*
- *Academic curricula to transfer the knowledge and prepare students to pass qualifying exams*
- *Qualifying exams certifying minimal competence for practicing the profession*
- *Continuing education, to maintain currency*
- *Code of ethics*
- *Strong professional associations*
- *Recognized canon of standard practices - malpractice may be evaluated against these*
- *Methods for monitoring and addressing malpractice*
- *Liability insurance*



Software Engineering - Craft or Engineering Discipline?

- So what is software engineering?
  - “The establishment and use of sound engineering principles in order to obtain economically software that is reliable and works efficiently on real machines.” (Fritz Bauer, quoted in Pressman, 1992, p. 23.)
  - “The development, on schedule and within budget, of high quality software that meets the users’ and organization’s needs.” (Turk)



Software Engineering Success

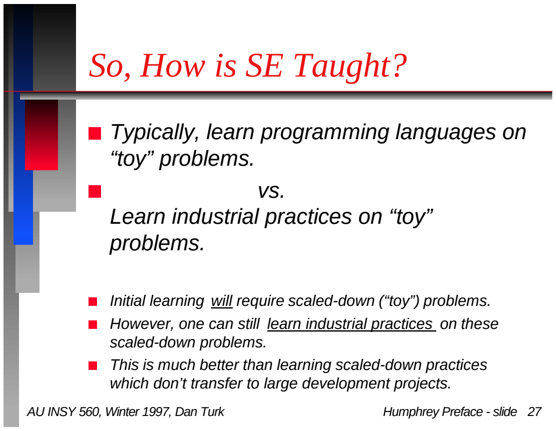
(Turk, 1995)

Factors Affecting Success:

- Technological
- Personal
- Group
- Managerial
- Geographical
- Cultural

Measures of Success:

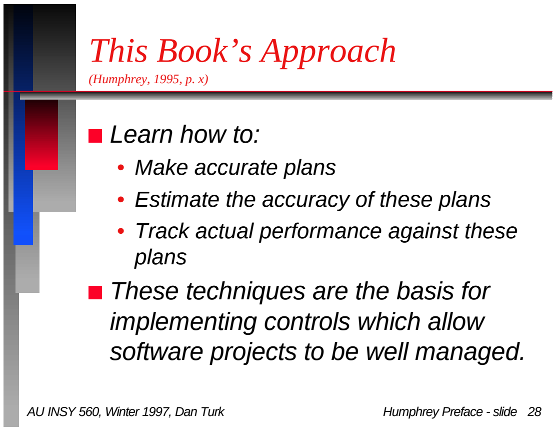
- Project vs. Product
- Financial vs. Acceptance / Use



So, How is SE Taught?

- Typically, learn programming languages on “toy” problems.
- vs.
  Learn industrial practices on “toy” problems.

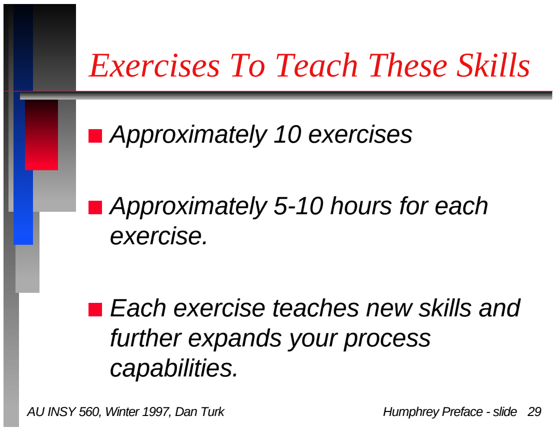
- Initial learning will require scaled-down (“toy”) problems.
- However, one can still learn industrial practices on these scaled-down problems.
- This is much better than learning scaled-down practices which don’t transfer to large development projects.



This Book’s Approach

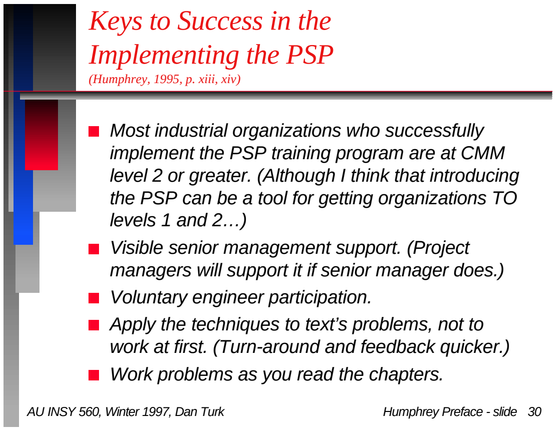
(Humphrey, 1995, p. x)

- Learn how to:
  - Make accurate plans
  - Estimate the accuracy of these plans
  - Track actual performance against these plans
- These techniques are the basis for implementing controls which allow software projects to be well managed.



Exercises To Teach These Skills

- Approximately 10 exercises
- Approximately 5-10 hours for each exercise.
- Each exercise teaches new skills and further expands your process capabilities.



Keys to Success in the Implementing the PSP

(Humphrey, 1995, p. xiii, xiv)

- Most industrial organizations who successfully implement the PSP training program are at CMM level 2 or greater. (Although I think that introducing the PSP can be a tool for getting organizations TO levels 1 and 2…)
- Visible senior management support. (Project managers will support it if senior manager does.)
- Voluntary engineer participation.
- Apply the techniques to text’s problems, not to work at first. (Turn-around and feedback quicker.)
- Work problems as you read the chapters.