Session 7: Staffing for programming & beyond
Roles during development
Selecting qualified programmers
Coping with weak programmers

Incompetent programmers
- We've known (SIGCPR, etc.) for decades that
  - There's a **20-to-1 range** in programmer productivity
  - Usually the most productive programmers also turn out the **highest quality** programs
- That means that one programmer may take **two weeks** to produce what another programmer produces in an **afternoon** (and it probably won't be as good).
- But many managers still don't believe it!
  - "A programmer is a programmer."
  - **What's their salary range?**

Building a staff
- If performance ratios are **20-to-1** and salary ranges are **3-to-1**:
  - Don't settle for "above average"
  - Go for the 85th percentile or better
- When recruiting a candidate always examine actual work samples
- How should we choose between:
  - a superb programmer who doesn't know the tools we need, and
  - an average programmer who can "hit the ground running"?

Measuring programmer productivity
- In the early days of programming, many organizations counted or estimated the **lines of code** (LOC) a programmer produces in order:
  - to evaluate the programmer's performance
  - to assess the relative power of different programming languages
  - to predict how long a programming task will take
- Some organizations still do that!
  - **What's wrong with that?**
The role of the programmer
- Is it
  - to write and test program code, or
  - to solve well-defined problems?

- We should often reward a programmer for not writing code:
  - For finding and using an existing program component
  - For avoiding unnecessary repetition
  - For discovering a reliable and efficient alternative solution, such as a spreadsheet model or a manual procedure.

Inheriting a team
- A project manager often takes over a project that's already underway.

- Even on a new project starting from Phase 1, we may be constrained to choose team members from I.T. staff who are not outstanding performers, but who
  - are available
  - know the tools we expect to use
  - have some knowledge of the application area or the user organization

What do we do with non-performers?
- Depends on the size of our organization.
  - In a large organization
    - there may be an alternative career path
    - you can sometimes arrange a trade with another department or another project.
  - In a small organization, you can't afford to retain them.

- In any case this is a delicate and often tragic situation.

Can modern methodologies hurt programmer performance?
- A manager may observe: "Ted was an excellent programmer when we were developing procedural programs in Cobol, but since we switched to test-driven object-oriented programs in C# with code inspections his work has been poor."

- What's to blame?
  - C#?
  - test-driven development?
  - object-oriented concepts and techniques?
  - code inspections
  - our training courses?
  - or . . .?
When should we consider an outside contractor?

- A local contract-development firm.
  - Will they work on our premises or at their office?
  - Do they have their own methodology?
  - Can we interview all of them and examine samples of their work?

- "Offshore" programmers
  - Are they as good? better?
  - Will they comply with our rules?
  - How will we monitor their performance?

- Which kind of contract?
  - Fixed price and delivery date
  - Time & materials

What's a "body shop"?