8
Calling the Shot

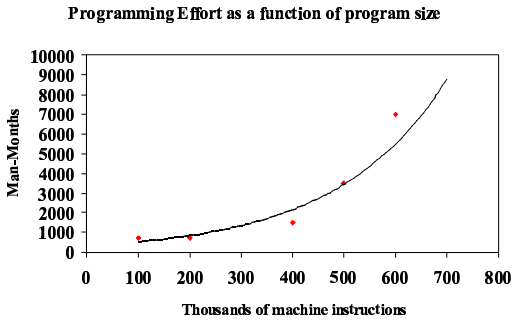
Calling the Shot

- Portman's data: Jobs take twice estimated
- Aron's data: Complexity dominates productivity
- Harr's data: Corroborates Aron's data
- OS/360 data: Corroborates Aron's and Harr's data
- Corbato's data
  - Productivity appears constant w.r.t. elements
  - High-level languages hold promise

Estimating Time for a System Programming Job

- Naive estimation
  - Estimate coding time, or
  - Estimate based on small-program experience
  - Extrapolate based on ratios
- E.g. Sackman, Erikson, and Grant report
  - For a certain size of program, average coding and debugging took 178 hours for a single programmer, i.e., annual productivity = 35,800 statements per year.
  - For a program of ½ size, time required was less than ¼, i.e., annual productivity = 80,000

Measuring effort



Estimating effort

- The above graph shows an exponent of 1.5
- Effort = (constant) x (number of instructions)$^{1.5}$
- Effort increases with a power of program size
- I.e., not linearly

Portman's Data

- Charles Portman, manager at ICL, Manchester
- Teams took ~ 2x estimates
- Estimates were done very carefully
- Each programmer to kept daily logs
- Logs showed that only 50% of the working week was spent in actual programming and debugging
- Rest of the time was spent in high priority, short, unrelated jobs, e.g., meetings, paperwork, personal time etc.

Aron's data

- Joel Aron, manager at IBM, Maryland studied programmer productivity of nine large systems
- Large: defined as more than 25 programmers and 30,000 deliverable instructions
- Related productivity to program complexity as follows (in instructions per person-year)
  - Very few interactions: 10,000 instructions
  - Some interactions: 5,000 instructions
  - Many interactions: 1,500 instructions
- Only for design and programming; Not test

Harr's data

	Prog. Units	No. of Programmers	Years	Man-Years	Program Words	Words/man-yr
Operational	50	83	4	101	52,000	515
Maintenance	36	60	4	81	51,000	630
Compiler	13	9	2.25	17	38,000	2230
Translator (Data assembler)	15	13	2.5	11	25,000	2270

Analyzing Harr's data

- First two jobs: Control Programs
- Second two jobs: language translators
- Productivity: Debugged words per person-year
- For control programs
  Productivity ~ 600 words per person-year
- For translators
  Productivity ~ 2200 words per person-year
- All four programs are of similar size

Analyzing Harr's data

- There were variations in size of work groups, length of time, and number of modules
- Open questions
  - Which is cause and which is effect?
  - Did control programs require more modules and more man-months because they were assigned more people?
  - Did they take longer because of complexity?
- Control programs were definitely more complex

OS/360 Data

- Productivities for control program ~ 600 – 800 debugged instructions per person-year
- Productivities for language translators ~ 2000 – 3000 debugged instructions per person-year
- Include planning, coding, component test, system test and some support activities

OS/360 data observations

- Supports Harr's Data
- Aron's, Harr's and OS/360 data all show differences in productivity related to complexity and difficulty of the task
- Brooks suggests the following related to the complexity
  - Compilers are 3 times as bad as normal batch application programs, and operating systems are 3 times as bad as compilers

Corbato's data

- System programming productivity data for high level language (Harr's data and OS/360 data were for assembly language programming)
- Corbato's data is for MIT's project MAC
- Productivity of 1200 *lines* of debugged PL/1 statements per man-yr on MULTICS system (1 – 2 million words)
- MULTICS includes control programs and language translators

Analyzing Corbato's Data

- Corbato's number is in *lines* per person-year i.e., not words (each statement corresponds to 3 to 5 words of handwritten code)
- Two important conclusions
  - Productivity is constant in terms of elementary statements; Reasonable in terms of thought a statement requires and errors it may have
  - Programming productivity may be increased by as much as 5 times when a suitable high-level language is used