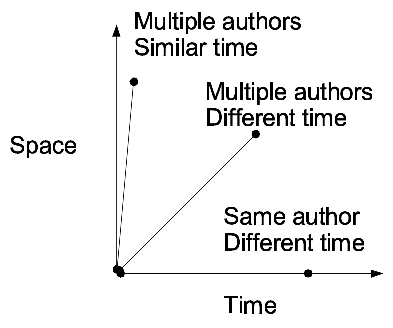
15 The Other Face

The Other Face

- What Documentation is Required?
- The Flow-Chart Curse
- Self-Documenting Programs

A program's two faces

- Programs
  - Instruct computers
  - Inform humans
- Space and time separate authors, maintainers and readers



Documentation failures

- Many program readers want better documentation
- Laziness, schedule pressure and incompetence discourage more thorough documentation
- Brooks says we have failed
  - Instructing programmers hasn't worked
  - Showing programmers should work

User documentation weaknesses

- Brooks says typical documents describe the bark the leaves but do not map the forest
- Program authors document features, not function
  - E.g., Automobile owners manuals outline the car's features, but not how to drive
- Most first-time users need documentation on how the program gets them to where they want to go

User documentation outline

1. Purpose
2. Environment
3. Domain & range
4. Functions realized (and algorithms used)
5. Input-output formats
6. (System) operating
7. Options
8. Running time
9. Accuracy and checking

- Three or four pages
- Drafted before the program is written
- I.e., embodying important design decisions

Test cases as documentation

- Users need to confirm the program's correct operation with known input/output conditions
- Maintainers need regression tests
  - Mainline cases for primary function
  - Barely legal cases to probe valid edge cases
  - Barely illegal cases to test error diagnostics
- Contemporary testing theory offers various rational for selecting test cases

Maintenance documentation

- Source comments provide details
- Introductory documentation summarizes structure
  1. Subprogram structure
  2. Algorithm descriptions or references
  3. File layouts
  4. Pass structure (in reading tape and disk)
  5. Suggested improvements in functionality and warnings about obscure design elements

Program structure chart

- Shows the caller-callee relation among
  - Subprocedures, or
  - Modules that contain calling subprocedures
- Useful for summarzing large programs
- Ideally fits on a single page

The flowchart curse

- Flowcharts show the control structure among predicate (if) and assignment (:=) statements
- Brooks calls (preparing) flowcharts
  - "Obsolete"
  - "A tedious, space-hogging drafting exercise"
  - "More preached than practised"
- Why?
- Parnas has a similar view of UML; Why?

Self-documenting programs

- Practise commonly attempts to maintain documents separately from the program
- Brooks says this causes poor documentation
  - "Changes to the program do not promptly, accurately and invariably appear in the paper"
- Merging the program and documentation
  - Improves the incentive to write documentation
  - Ensures accessiblity to the programmer

Brooks' approach

- Use labels, declaration statements and symbolic names to convey as much meaning as possible
- Use space and formatting to improve readability, and show structure and subordination
- Use prose comments, especially for overviews
- Norm suggests a hiearchy of overviews to assist in large-scale source navigation

Example techniques

1. Mnemonic job (program) name
2. Version number (in the source listing)
3. Prose descriptions for procedures
4. Standard literature references
5. Changes from the standard references, e.g., specialization and (data) representation
6. Variable declarations with line comments
7. Initialization label

Example techniques (cont.)

8. Statement group labels (summarizing purpose)
9. Indenting to show structure and grouping
10. Logical flow arrows
11. Line comments for non-obvious aspects
12. Multiple-assignment statements showing thought-grouping or algorithm correspondence