THE DESIGN OF DESIGN

F P Brooks

Rapporteur: Jonathan Halliday

THE DESIGN OF DESIGN

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill 27599-3175, USA

The Design of Design

Fred Brooks

University of North Carolina
at Chapel Hill

brooks@cs.unc.edu

Design

"To form a plan or scheme of,

to arrange or conceive in the mind...

for later execution."
OED

Design and Designers

1. My quest – What is the process of design?
2. How engineers think of design
3. What's wrong with this model?
4. Rationalism and Empiricism in design
5. Exemplars and their role
6. Esthetics in technical design
7. Solo and team design
8. Great designs come from great designers
9. Where do great designers come from?

Why Study Design?

- My design experiences:

A principal	A participant
Payroll machine	8000 series architecture
Stretch supercomputer	
Harvest cryptanalytic co-processor	
System/360 architecture	System/360 Assembler
APL	PL/I
GRIP, GRINCH, GRO PE	VIEW, SCULPT
Walkthrough	Several VR systems
Beach house	Computer Science bldg
Home remodeling	Church fellowship hall

- Quick net – these experiences are more alike than different!
- Can we design better by looking at design as a process?
- Teach others better to design better?

How Engineers Think of Design

- Goal
- Desiderata
- Non-linear utility function
- Constraints, especially budget (not necessarily $ cost)
- Design tree of decisions

```
UNTIL (design is "good enough") or (time has run out)
  DO another design (to improve utility function)
    UNTIL design is complete
      WHILE design remains feasible,
        make another design decision
      END WHILE
      Backtrack up design tree
      Explore a path not searched before
    END UNTIL
  END DO
  Take best design
END UNTIL
```

What's Wrong with This Model?—I

- We don't really know the goal at first –

  The hardest part of design is deciding *what* to design.

  Often the most important function of the designer is helping the client decide what he wants.

- Where experts go wrong:
  - Miss fresh vision – e.g., minicomputer, microcomputer
  - Vision not high enough – e.g., OS/360 JCL

THE DESIGN OF DESIGN

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill 27599-3175, USA

OS/360 JCL – the Worst Language

- One job language for all programming languages
- Like Assembler language, rather than PL/I, etc.
- But not exactly like Assembler
- Card-column dependent
- Too few verbs
- Declarations do verbish things, in the guise of parameters
- Awkward branching
- No clean iteration
- No clean subroutine call

- Done under my management

- Basic problem was pedestrian vision
  - We did not see it as a schedule-time programming language at all, but as a "few control cards"
  - It was not *designed,* it just grew as needs appeared.

What's Wrong with This Model?—II

- The desiderata and their weightings keep changing.
  - Donald Schön – "One wrestles with the problem."
  - As one in fact *makes* the trade-offs, the weights change.
  - Sometimes one hits new opportunities.
- We usually don't know the design tree.
- The constraints keep changing.
  - Often by the ever-changing world.
  - Sometimes by inspiration! Genius is finding the third way!

(Drawing of House)

Design Models

- Simon-Newell vs Lawson-Schön
- Simon is rational but wrong—
  doesn't describe what really goes on
  - Chris Alexander: "Any problem one can solve that way is not really design."
  - But still the "consensus model" in engineering literature.
  - See for example, G.Pahl and W. Beitz, 1984 *Engineering Design: A Systematic Approach*
- Schön is right but not elaborated—
  doesn't usefully prescribe actions

AI and Design Research

- Much of the AI work has been chasing the wrong goal.
- The human mind beats AI hands down on three tasks:
  - Pattern recognition
  - Evaluation
  - Associative search of vast database for context match
- The design problem, as shown by observation and protocol analysis, is rich in all three:
  => a poor candidate for AI
- Go for intelligence amplification, not artificial intelligence!
- IA>>AI

Rationalism vs. Empiricism in Design

- Aristotle vs. Galileo; France vs. Britain; Descartes vs. Hume; Roman law vs. Anglo-Saxon law
- Wegner: Prolog vs. Lisp; Algol vs. Pascal; Dijkstra vs. Knuth; proving programs vs. testing programs
- I am a thoroughgoing, dyed-in-the-wool empiricist:

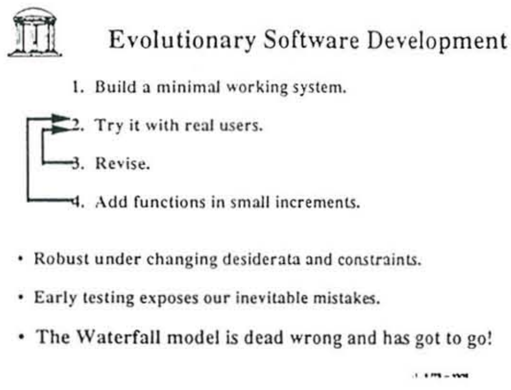
  Our designs are so complex there is no hope of getting them right first time by pure thought.

  To expect to is arrogant.

  So, we must adopt design-build processes that incorporate evolutionary growth
  vs. waterfall: specify-design-build
  vs. "Plan to throw one away."

THE DESIGN OF DESIGN

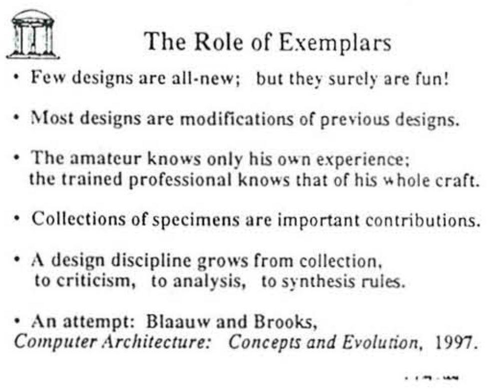
Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill 27599-3175, USA



Evolutionary Software Development

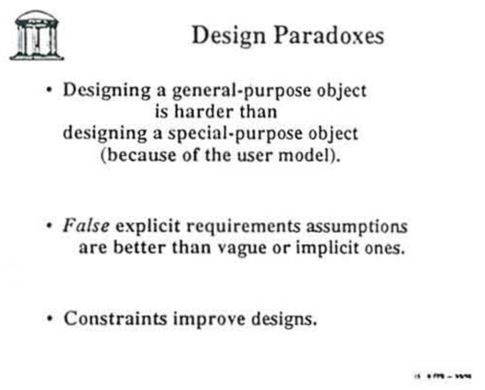
1. Build a minimal working system.
2. Try it with real users.
3. Revise.
4. Add functions in small increments.

- Robust under changing desiderata and constraints.
- Early testing exposes our inevitable mistakes.
- The Waterfall model is dead wrong and has got to go!



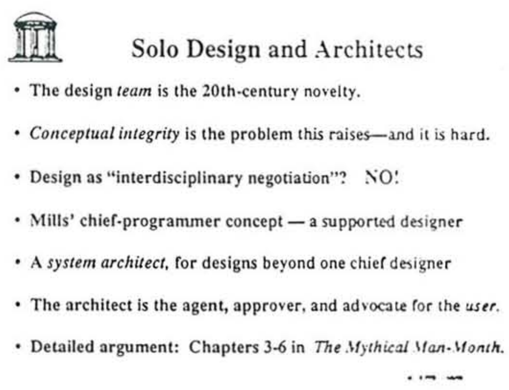
The Role of Exemplars

- Few designs are all-new; but they surely are fun!
- Most designs are modifications of previous designs.
- The amateur knows only his own experience; the trained professional knows that of his whole craft.
- Collections of specimens are important contributions.
- A design discipline grows from collection, to criticism, to analysis, to synthesis rules.
- An attempt: Blaauw and Brooks, *Computer Architecture: Concepts and Evolution,* 1997.



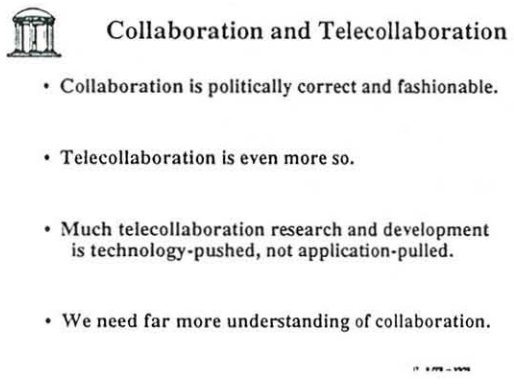
Design Paradoxes

- Designing a general-purpose object is harder than designing a special-purpose object (because of the user model).
- *False* explicit requirements assumptions are better than vague or implicit ones.
- Constraints improve designs.



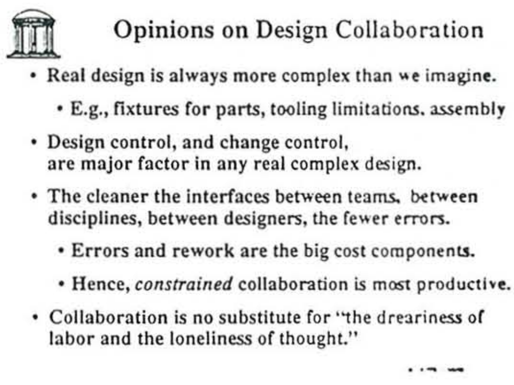
Solo Design and Architects

- The design *team* is the 20th-century novelty.
- *Conceptual integrity* is the problem this raises—and it is hard.
- Design as "interdisciplinary negotiation"? NO!
- Mills' chief-programmer concept — a supported designer
- A *system architect,* for designs beyond one chief designer
- The architect is the agent, approver, and advocate for the *user.*
- Detailed argument: Chapters 3-6 in *The Mythical Man-Month.*



Collaboration and Telecollaboration

- Collaboration is politically correct and fashionable.
- Telecollaboration is even more so.
- Much telecollaboration research and development is technology-pushed, not application-pulled.
- We need far more understanding of collaboration.



Opinions on Design Collaboration

- Real design is always more complex than we imagine.
  - E.g., fixtures for parts, tooling limitations, assembly
- Design control, and change control, are major factor in any real complex design.
- The cleaner the interfaces between teams, between disciplines, between designers, the fewer errors.
  - Errors and rework are the big cost components.
  - Hence, *constrained* collaboration is most productive.
- Collaboration is no substitute for "the dreariness of labor and the loneliness of thought."

THE DESIGN OF DESIGN

Frederick P. Brooks, Jr.
University of North Carolina at Chapel Hill 27599-3175, USA

When Does Collaboration Help?

- Determining needs from users
  - More minds —> more diverse questions
- Conceptual exploration
  - Brainstorming and synthesis among radical alternatives
- *Not* conceptual design nor detailed design
  - Observe the great works of the human mind
  - Conceptual integrity
- Design reviews
  - Especially with different disciplines

Designs with Fan Clubs

Yes	No
Fortran	COBOL
VM	OS/360
Unix	DEC VMS
Pascal	Algol
C	PL/I
Macintosh	PC/DOS
APL	

Esthetics in Technical Design

- Even designs we cannot see have an esthetic dimension:
  - A "clean" machine
  - An "elegant" language
- Elegance: "Accomplishing many things with few elements."
- But that's not enough. Straightforwardness is required, too.
  - van der Poel's one-instruction computer.
  - APL one-liner, idiomatic programming style
- *Consistency* – few concepts
  - Orthogonality
  - Propriety: parsimony and transparency
  - Generality: completeness and open-endedness
- Thesis: *Good esthetics yields good economics.*

Great Designs Come From Great Designers

- Conceptual integrity
- Solo vs. committee design
- Where elitism is proper
- Within-product-process vs. outside-product-process;
  What are product procedures for?
- The LHX light attack helicopter
  "and ferry itself across the Atlantic."
  Requirements fusion without either pilots nor engineers.
- How does one do great designs *within* a product process?
- How does one make a product process than encourages, rather than inhibits, great designs?

Where Do Great Designers Come From?

- We have to grow them *deliberately.*
  - Recruit for design brilliance, not just meeting, talk skills
  - Make the dual ladder real and honorable
  - Career planning and training, just as for managers
  - Mentors
  - Planned experience and rotation: up, down, and sideways
  - Planned mid-career educational experiences, sabbaticals
- We have to manage them *imaginatively.*
  - The Steinmetz story
  - The John Cocke story
- We have to protect them *fiercely.*
  - From managers
  - From managing

Where is Design Research Going?

- Important that researchers also do design themselves, in order not to go wildly astray.
- Easy to get swallowed up in methodological purity.
- I think it important to publish, and to *distinguish:*
  - Facts established by controlled experiment, however narrow.
  - Rules of thumb derived from multiple uncontrolled observations—
    "Piping designers seem to do X generally."
  - Observations from cases— "I saw a designer do X."
- We want to help both practice and education.

DISCUSSION

Rapporteur: Jonathan Halliday

Lecture One

The majority of discussion for this session was in the form of comments and exchanges interjected during the presentation. Only a small number of points were made at the end of the session.

Professor Brooks postulated that whilst amateur designers made mistakes in the technical detail of designs, it was the 'experts' who typically produced designs which were comprehensively 'wrong'. It was felt that this was due to the accumulated experience of a particular way of thinking making it difficult to shift to new paradigms. This was illustrated by the experienced staff of companies such as IBM missing the boat on shifts in the industry (e.g. mainframe to mini), instead continuing to produce designs which were essentially just 'bigger and better' versions of the same product.

Professor Shaw enquired, during discussion of the design disaster which constituted JCL360, if the rumour that the designers had approached the PL1 design team for assistance and been shunned was true. Professor Brooks maintained that, as the (then) manager of the JCL360 design team, he had no knowledge of this. He felt that the problems with JCL360 grew from the designer's attempting to apply experience of old job control languages to the new and different model employed for the O/S360 architecture. Professor Randell made the point that JCL360 was a classic example of designers needlessly and inappropriately reinventing the wheel with regard to basic programming language elements such as conditional branching.

The power of annotated design documentation as a teaching tool for designers was discussed. It was felt that examples in the form of annotations to reference documents were a valuable asset. Professor Randell made reference to his experience in documenting an Algol compiler he had co-authored. He had been well advised by Professor Dijkstra that production of such design documentation was worthwhile only if all design decisions were documented and explained, and clearly described as arbitrary if this was the case.

It was noted that the user model for a given piece of software formed part of its design, but was frequently undocumented. This produced a contradiction, in that only written specifications could usually be considered as part of a design process. Professor Brooks maintained that the user model always formed part of the design, and hence an effort should always be made to document it. Whilst a large amount of guesswork was often needed in specifying a user model, this should always be undertaken and documented in order that the specification be precise, even if wrong. The point was made that for highly generalised software products; the user model was so general that it could not easily exist, other than in the heads of the designers.

Professor Brooks made the point that great designs typically came from a single individual working alone (e.g. painters, composers), whereas the prevalent model in computing was for the use of collaborative design teams. Members of the audience pointed out that much work considered to be that of a single individual was often the product of a strongly led team. e.g. 16-17th century painters who only did the faces of their characters, leaving students to fill in the landscape, etc. It was felt the key was strong, unequivocal leadership of the group.

Professor Brooks stated that in the area of (tele)collaboration, the model of group working was often powered by 'technology-push' rather than 'application-pull', leading to poor working practices. The point was made that one person's technology was another person's application. However, the speaker maintained that, in general, collaborative working was no substitute for an individual sitting down and thinking through the problem.

In the brief discussion that concluded the session, the design of the Java language was discussed. Whilst clearly a group effort, this project was felt to be a clear case of a single, highly experienced individual (James Gosling) providing strong leadership to a team. The ability of a single designer to draw not only on his own experience, but also on that of other group members was felt to be a key factor. The point was made that the strong leadership of the project made it one of the few cases where the product had successfully avoided the 'feature creep' which often affected group designed systems. Indeed, it was noted that features had been added to Java and then removed when it was found that they did not fit with the strongly defined goals of the project.