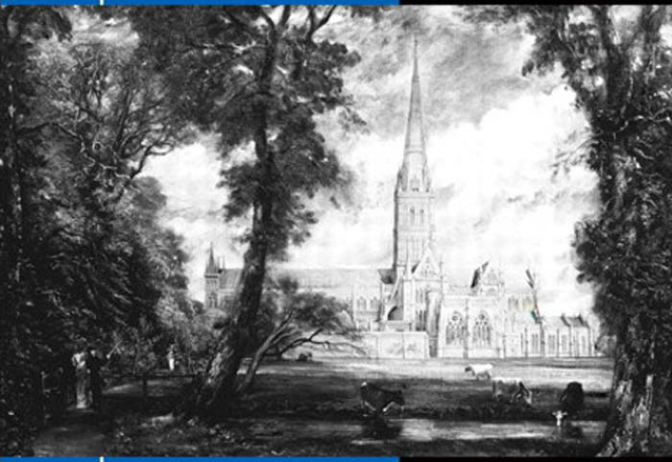
FROM THE AUTHOR OF *THE MYTHICAL MAN-MONTH*



THE DESIGN OF DESIGN

ESSAYS FROM A COMPUTER SCIENTIST

FREDERICK P. BROOKS, JR.

Photo credit: © Jerry Markatos

ABOUT THE AUTHOR

Frederick P. Brooks, Jr., is Kenan Professor of Computer Science at the University of North Carolina at Chapel Hill. He is best known as the "father of the IBM System/360," having served as project manager for its development and later as manager of the Operating System/360 software project during its design phase. For this work, he, Bob Evans, and Erich Bloch were awarded the National Medal of Technology in 1985. Earlier, he was an architect of the IBM Stretch and Harvest computers.

At Chapel Hill, Dr. Brooks founded the Department of Computer Science and chaired it from 1964 through 1984. He has served on the National Science Board and the Defense Science Board. His current teaching and research is in computer architecture, interactive computer graphics, and virtual environments.

To what degree are the sampled systems representative? Over the whole intended user set, what are the ranges of all those parameters? Their distributions?

What are the rates of change from scheduling period to scheduling period? What will the ranges be five years from now? Ten years?

As the questions get harder, the answers get vaguer. What is the designer to do if he has committed himself to making explicit use models?

Guess!

I am quite convinced that, once he has moved beyond questions that can be answered by reasonable inquiry, the designer should *guess* or, if you prefer, *postulate* a complete set of attributes and values, with guessed frequency distributions, in order to develop complete, explicit, and shared user and use models.

An articulated guess beats an unspoken assumption.

Many benefits flow from this "naive" procedure:

Guessing the values and frequencies forces the designer to think very carefully about the expected user set.

Writing down the values and frequencies exposes them to debate. It is easier to criticize something concrete than to create, so there will be more input from the whole team. The debate will inform all the participants and will surface the differences in user images that the several designers carry. It typically also will surface other unrecognized assumptions.[4]

Enumerating the values and frequencies explicitly helps everyone realize which decisions depend upon which user set properties.

More important, it raises these crucial questions: Which assumptions matter? How much? Even this sort of horseback sensitivity analysis is valuable. When it develops that important decisions are hinging on some particular guess, it is worth the cost to develop a better estimate.

In the end, however, many assumptions will remain debatable and unverifiable. The chief architect must own—and make known—the set the team goes with.

Better Wrong than Vague!

At this point the reader will object, "How can I know or even assume so much detail about uses and users?" The answer is, "You will in fact make those assumptions anyhow"; that is, each design decision will be guided, consciously or unconsciously, by the designer's assumptions about uses and users. What this often means in reality is that the vague designer substitutes *himself* for the user, designing for what he assumes he would want if he were the user. But he isn't.

Therefore, wrong explicit assumptions are much better than vague ones. Wrong ones will perhaps be questioned; vague ones won't.

Notes and References

1. A *use model* is a weighted collection of *use cases*. Robertson and Robertson [2005], *Requirements-Led Project Management*, treat use cases in considerable detail.
2. Cockburn [2000], *Writing Effective Use Cases*, is a thorough treatment.
3. Brooks [1995], *The Mythical Man-Month,* 56–57.
4. Students in my advanced computer architecture course have been required to do this for their term projects. When it has been done conscientiously, the effect on the designs has been very beneficial.

USA
USA

10

Inches, Ounces, Bits, Dollars—The Budgeted Resource

If a design, particularly a team design, is to have conceptual integrity, one should name the scarce resource explicitly, track it publicly, control it firmly.

***Apollo* rocket**
iStockphoto

What's the Budgeted Resource?

Within any design, there is at least one scarce resource to be rationed or budgeted. Sometimes there are two or more to be jointly optimized; but most commonly, one is dominant, and others appear chiefly as desiderata or constraints. Economists call this the *limiting resource*; I prefer to emphasize the necessary designer action: conscious budgeting.[1]

Although designers often talk as if cost or some performance/cost ratio were the resource to be optimized, that is often not how they act in practice. It follows that if a design, particularly a team design, is to have conceptual integrity, one wants to *name the scarce resource explicitly, track it publicly, control it firmly*.

Often Not Dollars

Ponder some budgeted critical resources that are not dollars:

- Inches of oceanfront, in a beach house
- Ounces of payload, in a spacecraft—or in a backpack
- Memory bandwidth, in any von Neumann computer architecture
- Nanoseconds of timing tolerance, in a GPS system
- Calendar days, on an asteroid-interception project
- Resident kernel memory space, in OS/360 design
- Program hours, at a conference
- Pages, on a grant proposal or a journal paper
- Power (and stored energy) on a communications satellite
- Heat, in a high-performance chip
- Water, on western farmland
- Student learning hours, in a degree curriculum
- Political power, in an organization's constitution
- Seconds, even frames, in a film or video
- Hours of access to the track per day, in London Underground engineering and maintenance
- Format bits, in a computer architecture
- Hours or minutes, in a military assault plan

Even Dollars Have Flavors, and Surrogates

Even when cost is in fact to be the budgeted commodity for a design project, cost varieties must be considered. For personal computers, made by the hundreds of millions, manufacturing cost is dominant. For supercomputers, made by the dozens, development cost dominates.

Quite often, designers adopt surrogates for dollars as their budgeted resources. Building architects will do program development, and often schematic design, using square feet as the rationed resource. Computer architects used to use bits of register and of various cache levels as surrogates for chip area.

Surrogates have several advantages: They are usually simpler. One can design with them long before one knows the surrogate/dollar ratio. They are more stable. Using the same surrogate harnesses one's previous design experiences, even though the surrogate/dollar ratio may be different, or even varying. One knows how many square feet per occupant an auditorium needs.

Surrogates can also lead one astray; the temptation is to use them after they cease being appropriate. Chip designers often thought in terms of area well after wiring length or pin count became the important critical resource.

The Budgeted Resource Can Change

Shifts in technology sometimes change which resource is critical. Therein lie snares for the unwary. As chip densities went up, I/O pins displaced chip area as the limiter on function, hence became the rationed commodity. But power dissipation has now displaced pins as the rationed commodity on many chip designs. Seymour Cray once famously said, "Refrigeration is the key to supercomputer design."[2] Gene Amdahl told me, about the same time, that off-chip capacitance was the speed-limiting commodity in his designs.[3]

Some think that number of classes has replaced function points as the estimator for software complexity, hence for probable size. Experienced consultants Suzanne and James Robertson, however, say that they find function points still to be the