The Design of Design: Essays from a Computer Scientist

Frederick P. Brooks, Jr.
Credits and permissions appear on page 421, which is a continuation of this copyright page.

Front Cover: John Constable’s design (painting) for his view of Salisbury Cathedral, the design of Elias de Dereham and Nicholas of Ely. © Geoffrey Clements/CORBIS. All rights reserved.

This material is based upon work supported by the National Science Foundation under Grant No. 0608665.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
corpsales@pearsoned.com

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Brooks, Frederick P. (Frederick Phillips)
The design of design: essays from a computer scientist / Frederick P. Brooks, Jr.
p. cm.
Includes bibliographical references and indexes.
TA174.B752 2010
628/.042—dc22
2009045215

Copyright © 2010 Pearson Education, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.
Rights and Contracts Department
501 Boylston Street, Suite 900
Boston, MA 02116

Fax: (617) 671-3447

ISBN-10: 0-201-36298-8

Text printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.

First printing, March 2010
Preface

I write to prod designers and design project managers into thinking hard about the process of designing things, especially complex systems. The viewpoint is that of an engineer, focused on utility and effectiveness but also on efficiency and elegance.

Who Should Read This Book?

In *The Mythical Man-Month* I aimed at “professional programmers, professional managers, and especially professional managers of programmers.” I argued the necessity, difficulty, and methods of achieving conceptual integrity when software is built by teams.

This book widens the scope considerably and adds lessons from 35 more years. Design experiences convince me that there are constants across design processes in a diverse range of design domains. Hence the target readers are:

1. Designers of many kinds. Systematic design excluding intuition yields pedestrian follow-ons and knock-offs; intuitive design without system yields flawed fancies. How to weld intuition and systematic approach? How to grow as a designer? How to function in a design team?

   Whereas I aim for relevance to many domains, I expect an audience weighted toward computer software and hardware designers—to whom I am best positioned to speak concretely. Thus some of my examples in these areas will involve technical detail. Others should feel comfortable skipping them.

2. Design project managers. To avoid disaster, the project manager must blend both theory and lessons from hands-on experience as he designs his design process, rather than just replicating
some oversimplified academic model, or jury-rigging a process without reference to either theory or the experience of others.

3. Design researchers. The study of design processes has matured; good, but not all good. Published studies increasingly address narrower and narrower topics, and the large issues are less often discussed. The desire for rigor and for “a science of design” perhaps discourages publication of anything other than scientific studies. I challenge design thinkers and researchers to address again the larger questions, even when social science methodology is of little help. I trust they will also challenge the generality of my observations and the validity of my opinions. I hope to serve their discipline by bringing some of their results to practitioners.

Why Another Book on Design?

Making things is a joy—immensely satisfying. J. R. R. Tolkien suggests that God gave us the gift of subcreation, as a gift, just for our joy. After all, “The cattle on a thousand hills are mine. … If I were hungry, I would not tell you.” Designing per se is fun.

The design process is not well understood either psychologically or practically. This is not for lack of study. Many designers have reflected on their own processes. One motivation for study is the wide gaps, in every design discipline, between best practice and average practice, and between average practice and semi-competent practice. Much of design cost, often as much as a third, is rework, the correction of mistakes. Mediocre design provably wastes the world’s resources, corrupts the environment, affects international competitiveness. Design is important; teaching design is important.

So, it was reasoned, systematizing the design process would raise the level of average practice, and it has. German mechanical engineering designers were apparently the first to undertake this program. The study of the design process was immensely stimulated by the coming of computers and then of artificial intelligence. The initial hope, long delayed in realization and I think impossible,
was that AI techniques could not only take over much of the
drudgery of routine design but even produce brilliant designs
lying outside the domains usually explored by humans. A dis-
cipline of design studies arose, with dedicated conferences, jour-
nals, and many studies.

With so much careful study and systematic treatment already
done, why another book?

First, the design process has evolved very rapidly since
World War II, and the set of changes has rarely been discussed.
Team design is increasingly the norm for complex artifacts. Teams
are often geographically dispersed. Designers are increasingly
divorced from both use and implementation—typically they no
longer can build with their own hands the things they design. All
kinds of designs are now captured in computer models instead of
drawings. Formal design processes are increasingly taught, and
they are often mandated by employers.

Second, much mystery remains. The gaps in our understand-
ing become evident when we try to teach students how to design
well. Nigel Cross, a pioneer in design research, traces four stages
in the evolution of design process studies:

1. Prescription of an ideal design process
2. Description of the intrinsic nature of design problems
3. Observation of the reality of design activity
4. Reflection on the fundamental concepts of design

I have designed in five media across six decades: computer
architecture, software, houses, books, and organizations. In each
I have had some roles as principal designer and some roles as
collaborator in a team. I have long been interested in the design
process; my 1956 dissertation was “The analytic design of auto-
matic data processing systems.” Perhaps now is the time for
mature reflection.

What Kind of Book?

I am struck by how alike these processes have been! The mental
processes, the human interactions, the iterations, the constraints,
the labor—all have a great similarity. These essays reflect on what seems to be the underlying invariant process.

Whereas computer architecture and software architecture each have short histories and modest reflections about their design processes, building architecture and mechanical design have long and honorable traditions. In these fields design theories and design theorists abound.

I am a professional designer in those fields that have had only modest reflection, and an amateur designer in some long and deep fields. So I shall attempt to extract some lessons from the older design theories and to apply them to computers and software.

I believe “a science of design” to be an impossible and indeed misleading goal. This liberating skepticism gives license to speak from intuition and experience— including the experience of other designers who have graciously shared their insights with me.

Thus I offer neither a text nor a monograph with a coherent argument, but a few opinionated essays. Even though I have tried to furnish helpful references and notes that explore intriguing side alleys, I recommend that one read each essay through, ignoring the notes and references, and then perhaps go back and explore the byways. So I have sequestered them at the end of each chapter.

Some case studies provide concrete examples to which the essays can refer. These are chosen not because of their importance, but because they sketch some of the experience base from which I conclude and opine. I have favored especially those about the functional design of houses—designers in any medium can relate to them.

I have done functional (detailed floor plan, lighting, electrical, and plumbing) design for three house projects as principal architect. Comparing and contrasting that process with the process of designing complex computer hardware and software has helped me postulate “essentials” of the design process, so I use these as some of my cases, describing those processes in some detail.

In retrospect, many of the case studies have a striking common attribute: the boldest design decisions, whoever made them, have accounted for a high fraction of the goodness of the outcome. These bold decisions were made due sometimes to vision, sometimes to
desperation. They were always gambles, requiring extra investment in hopes of getting a much better result.

**Acknowledgments**

I have borrowed my title from a work of a generation ago by Gordon Glegg, an ingenious mechanical designer, a charming person, and a spellbinding Cambridge lecturer. It was my privilege to lunch with him in 1975 and to catch some of his passion for design. His title perfectly captures what I am attempting, so I reuse it with gratitude and respect.\[10\]

I appreciate the encouragement of Ivan Sutherland, who in 1997 suggested that I grow a lecture into a book and who more than a decade later sharply critiqued the draft, to its great improvement. My resulting intellectual journey has been very rewarding.

This work has been possible only because of three research leaves granted by UNC-Chapel Hill and my department chairmen, Stephen Weiss and Jan Prins. I was most graciously welcomed by Peter Robinson at Cambridge, Mel Slater at University College London, their department chairmen, and their colleagues.

The NSF Computer and Information Science and Engineering Directorate’s Science of Design program, initiated by Assistant Director Peter A. Freeman, provided a most helpful grant for the completion of this book and the preparation of the associated Web site. That funding has enabled me to interview many designers and to concentrate my principal efforts for the past few years on these essays.

I am deeply indebted to the many real designers who have shared their insights with me. An acknowledgments table listing interviewees and referees is an end piece. Several books have been especially informative and influential; I list them in Chapter 28, “Recommended Reading.”

My wife, Nancy, co-designer of some of the work herein, has been a constant source of support and encouragement, as have my children, Kenneth P. Brooks, Roger E. Brooks, and Barbara B. La Dine. Roger did an exceptional review of the manuscript, providing dozens of suggestions per chapter, from concepts to commas.
I've been blessed by strong administrative support at UNC from Timothy Quigg, Whitney Vaughan, Darlene Freedman, Audrey Rabelais, and David Lines. Peter Gordon, Publishing Partner at Addison-Wesley, has provided unusual encouragement. Julie Nahil, Full-Service Production Manager at Addison-Wesley, and Barbara Wood, Copy Editor, have provided exceptional professional skills and patience.

John H. Van Vleck, Nobel-laureate physicist, was Dean of Harvard’s Division of Engineering and Applied Science when I was a graduate student there, in Aiken’s lab. Van Vleck was very concerned that the practice of engineering be put on a firmer scientific basis. He led a vigorous shift of American engineering education away from design toward applied science. The pendulum swung too far; reaction set in; and the teaching of design has been contentious ever since. I am grateful that three of my Harvard teachers never lost sight of the importance of design and taught it: Philippe E. Le Corbeiller, Harry R. Mimno, and Howard H. Aiken, my adviser.

Thanks and praise to The Great Designer, who graciously grants us the means, the daily sustaining, and the joys of subcreation.

Chapel Hill, NC
November 2009

Endnotes

1. The caption for the book cover is based on Smethurst [1967], The Pictorial History of Salisbury Cathedral, who adds, “… Salisbury is thus the only English cathedral, except St. Paul’s, of which the whole interior structure was built to the design of one man [or one two-person team] and completed without a break.”


4. Pahl and Beitz [1984], in Section 1.2.2, trace this history, starting in 1928. Their own book, Konstructionslehre, through seven editions, is perhaps the most important systematization. I distinguish study of the design process from rules for design in any particular medium. These are millennia older.
5. The major monograph, tremendously influential, was Herbert Simon’s *The Sciences of the Artificial* [1969, 1981, 1996].

6. Cross [1983], *Developments in Design Methodology*, x.


   *The challenge is to transform individual experiences, frameworks and perspectives into a shared, understandable, and, most importantly, a transmittable area of knowledge. Victor Margolin states three reasons why this will prove difficult, [one of which is]:

   ‘… Individual explorations of design discourse focus too much on individual narratives, leading to personal point-of-view rather than a critical mass of shared values.’

   To this I must plead, “Guilty as charged.”

6
Collaboration in Design

A meeting is a refuge from “the dreariness of labor and the loneliness of thought.”

BERNARD BARUCH, IN RISEN [1979], “A THEORY ON MEETINGS”
Is Collaboration Good Per Se?

Two big changes in design have taken place since 1900:

- Design is now done mostly by teams, rather than individuals.
- Design teams now often collaborate by using telecommunications, rather than by being collocated.

As a consequence of these big shifts, the design community is abuzz with hot topics:

- Telecollaboration
- “Virtual teams” of designers
- “Virtual design studios”

All of these are enabled by telephony, networking, computers, graphic displays, and videoconferencing.

If we are to understand telecollaboration, we must first understand the role of collaboration in modern professional design.

It is generally assumed that collaboration is, in and of itself, a “good thing.” “Plays well with others” is high praise from kindergarten onward. “All of us are smarter than any of us.” “The more participation in design, the better.” Now, these attractive propositions are far from self-evident. I will argue that they surely are not universally true.

Most great works of the human mind have been made by one mind, or two working closely. This is true of most of the great engineering feats of the 19th and early 20th centuries. But now, team design has become the modern standard, for good reasons. The danger is the loss of conceptual integrity in the product, a very grave loss indeed. So the challenge is how to achieve conceptual integrity while doing team design, and at the same time to achieve the very real benefits of collaboration.

Team Design as the Modern Standard

Team design is standard for modern products, both those mass-produced and one-offs such as buildings or software. This is indeed a big change since the nineteenth century. We know the
names of the leading 18th- and 19th-century engineering designers: Cartwright, Watt, Stephenson, Brunel, Edison, Ford, the Wright Brothers. Consider, on the other hand, the Nautilus nuclear submarine (Figure 6-1). We know Rickover as the champion, the Will who made it happen, but which of us can name the chief designer? It is the product of a skilled team.

Consider great designers, and think of their works:

- Homer, Dante, Shakespeare
- Bach, Mozart, Gilbert and Sullivan
- Brunelleschi, Michelangelo
- Leonardo, Rembrandt, Velázquez
- Phidias, Rodin

Most great works have been made by one mind. The exceptions have been made by two minds. And two is indeed a magic number for collaborations; marriage was a brilliant invention and has a lot to be said for it.
Why Has Engineering Design Shifted from Solo to Teams?

Technological Sophistication. The most obvious driver toward team design is the increasing sophistication of every aspect of engineering. Contrast the first iron bridge (Figure 6-2) with its splendid descendant (chapter frontispiece).

The first had to be wrought very conservatively, that is, heavily and wastefully, even though elegantly. Both the properties of the iron and the distribution of static and dynamic stresses were understood imperfectly (though remarkably well!).

Menn’s bridge, on the other hand, soars incredibly but confidently, the fruit of years of analysis and modeling.

I am impressed that there are no naive technologies left in modern practice. It was my privilege to tour Unilever’s research laboratory at Port Sunlight, Merseyside, UK. I was astonished to find a PhD applied mathematician doing computational fluid dynamics (CFD) on a supercomputer, so as to get the mixing of shampoo right! He explained that the shampoo is a three-layer emulsion of aqueous and oily components, and mixing without tearing is crucial.
The designers of a John Deere cotton-picking machine used CFD to structure the airflow carrying the cotton bolls. A modern farmer spends not only hours on the tractor, but also hours on the computer, matching fertilizer, protective chemicals, seed variety, soil analysis, and crop rotation history. The master cook at Sara Lee adjusts the cake recipe continually to match the chemical properties of the flour coming in; the boss in the paper mill similarly adjusts for the varying pulpwood properties.

Mastering explosive sophistication in any branch of engineering forces specialization. When I went to graduate school in 1953, one could keep up with all of computer science. There were two annual conferences and two quarterly journals. My whole intellectual life has been one of throwing passionate subfield interests overboard as they have exploded beyond my ability to follow them: mathematical linguistics, databases, operating systems, scientific computing, software engineering, even computer architecture—my first love. This sort of splintering has happened in all the creative sciences, so the designer of today’s state-of-the-art artifact needs help from masters of various crafts.

The explosion in the need for detailed know-how of many technologies has been partially offset by the stunning explosion in the ready availability of such detailed know-how—in documents, in skilled people, in analysis software, and in search engines that find the documents and plausible candidates for collaborators.

**Hurry to Market.** A second major force driving design to teams is hurry to get a new design, a new product, to market. A rule of thumb is that the first to market a new kind of product can reasonably expect a long-run market share of 40 percent, with the remainder split among multiple smaller competitors. Moreover, the pioneer can harvest a profit bubble while the competition builds up. In the biggest wins, the pioneer continues to dominate. These realities press design schedules hard. Team design becomes a necessity when it can accelerate delivery of a new product in a competitive environment.

Why is this competitive time pressure more intense than before? Global communications and global markets mean that any great idea anywhere propagates more quickly now.
**Costs of Collaboration**

“Many hands make light work”—Often  
But many hands make more work—Always  

We all know the first adage. And it is true for tasks that are partitionable. The burden on each worker is lighter, hence the time to completion is shorter. But no design tasks are perfectly partitionable, and few are highly partitionable. So collaboration brings extra costs.

**Partitioning Cost.** Partitioning a design task is itself an added task. The crisp and precise definition of the interfaces between subtasks is a lot of work, slighted at peril. As the design proceeds, the interfaces will need continually to be interpreted, no matter how precisely delineated. There will be gaps. There will be inconsistencies in definition and conflicts in interpretation; these must be reconciled.

To simplify manufacture, there must be standardization of common elements across all the components; some commonality of design style must be established.

And then the separate pieces must be integrated—the ultimate test of interface consistency. It is not just in shipyards where the reality of integration is “Cut to plan; bang to fit.”

**Learning/Teaching Cost.** If \( n \) people collaborate on a design, each must come up to speed on the goals, desiderata, constraints, utility function. The group must share a common vision of all of these things—of what is to be designed. To a first approximation, if a one-person design job consists of two parts—learning \( l \) and designing \( d \)—the total work when the job is shared out \( n \) ways is no longer

\[
\text{work} = l + d
\]

but now at least

\[
\text{work} = n \cdot l + d
\]

Moreover, someone with the vision and knowledge must do the teaching, hence will not be designing. One hopes that the efficiencies of specialization will buy back some of these costs.

**Communication Cost during Design.** During the design process, the collaborating designers must be sure their pieces will fit together. This requires structured communication among them.
The Challenge Is Conceptual Integrity!

Change Control. A mechanism for change control must be put into place so that each designer makes only those changes that (1) affect only his part or (2) have been negotiated with the designers of the affected parts. Since much of the cost of design is indeed change and rework, the cost of change control is substantial. The cost of not having formal change control is much greater.

The Challenge Is Conceptual Integrity!

Much of what we consider elegance in a design is the integrity, the consistency of its concepts. Consider Wren’s masterpiece, St. Paul’s Cathedral (Figure 6-3).
Such design coherence in a tool not only delights, it also yields ease of learning and ease of use. The tool does what one expects it to do. I argued in *The Mythical Man-Month* that conceptual integrity is the most important consideration in system design. Sometimes this virtue is called coherence, sometimes consistency, sometimes uniformity of style. Blaauw and I have elsewhere discussed conceptual integrity at some length, identifying as component principles orthogonality, propriety, and generality. The solo designer or artist usually produces works with this integrity subconsciously; he tends to make each microdecision the same way each time he encounters it (barring strong reasons). If he fails to produce such integrity, we consider the work flawed, not great.

Many great engineering designs are still today principally the work of one mind, or two. Consider Menn’s bridges. Consider the computers of Seymour Cray. The genius of his designs flowed from his total personal mastery over the whole design, ranging from architecture to circuits, packaging, and cooling, and his consequent freedom in making trades across all design domains. He took the time to do designs he could master, even though he used and supervised a team. Cray exerted a powerful counterforce against those corporate and external pressures that would have steered his own attention away from design to other matters. He repeatedly took his design team away from the laboratories created by his earlier successes, considering solitude more valuable than interaction. He was proud of having developed the CDC 6600 with a team of 35, "including the janitor."

One sees this pattern—physical isolation, small teams, intense concentration, and leadership by one mind—repeated again and again in the design of truly innovative, as opposed to follow-on, products: for example, the Spitfire team under Joe Mitchell, off at Hursley House, a stately home in Hampshire, UK; Lockheed’s Skunk Works under Kelly Johnson, from which the U-2 spy plane and F-117 stealth fighter came; IBM’s closed laboratory in Boca Raton, Florida, home of IBM’s successful effort to catch up with Apple on the PC.

**Dissent**

Not everyone agrees with the thesis I have been arguing. Some argue the social justice of participatory design—that it is right
for users to have a significant role in the design of objects for their use.12 Whereas this participation is feasible (and prudent as well as fair) for buildings, user participation in the design of mass-market products is inherently limited to a small sample of prospective users. Such a voice must be conditioned by the representativeness of the sampling, and the vision of the designer.

Others argue that my facts are wrong, that team design has in fact always been the norm.13 The reader will have to judge for himself.

How to Get Conceptual Integrity with Team Design?

Any product so big, so technically complex, or so urgent as to require the design effort of many minds must nevertheless be conceptually coherent to the single mind of the user.14 Whereas such coherence is usually a natural consequence of solo design, achieving it in collaborative design is a management feat, requiring a great deal of attention. So, how does one organize design efforts to achieve conceptual integrity?

Modern Design as an Interdisciplinary Negotiation?

Many (mostly academic) writers conclude from the high degree of today’s specialization that the nature of design has changed: design today must be done as an “interdisciplinary negotiation” (among the team). The clear implication, though not explicit, is that the team members are peers, and each must be satisfied. NO! If conceptual integrity is the final goal, negotiation among peers is the classic recipe for bloated products! The result is design by committee, where none dare say “No” to another’s suggestion.15

A System Architect

The most important single way to ensure conceptual integrity in a team design is to empower a single system architect. This person must be competent in the relevant technologies, of course. He must be experienced in the sort of system being designed. Most of all, he must have a clear vision of and for the system and must really care about its conceptual integrity.
The architect serves during the entire design process as the agent, approver, and advocate for the user, as well as for all the other stakeholders. The real user is often not the purchaser. This is evidently true with military acquisitions, where the purchaser (and even the specifier) is far removed from the user. Indeed, the same system may have multiple users, wielding it at strategic, battalion, and personal levels. The purchaser is represented at the design table by marketers. The engineers are represented. The manufacturers are represented. Only the architect represents the users. And, for complex systems as well as for simple residences, it is the architect who must bring professional technology mastery to bear for the users’ overall, long-run interest. The role is challenging. I have discussed it in considerable detail in Chapters 4–7 of *The Mythical Man-Month*.

**One User-Interface Designer**

A major system will require not only a chief architect, but indeed an architectural team. So the conceptual-integrity challenge recurses. Even architecture work must be partitioned, controlled, and hence reintegrated. Here again, conceptual integrity requires special effort.

The user interface, the user’s crucial system component, must be tightly controlled by one mind. In some teams, the chief architect can do this detailed work. Consider MacDraw and MacPaint, early Mac tools that were in fact built by their designers. In large architecture teams, the chief architect’s scope is too large for him to do the interface himself. Nevertheless, one person must do it. If one architect can’t master it, one user can’t either. At Google, for example, one vice president, Marissa Mayer, maintains personal control over the page format and the home page.

Such an interface designer not only needs lots of using experience and listening skills, he above all needs taste. I once asked Kenneth Iverson, Turing Award winner and inventor of the APL programming language, “Why is APL so easy to use?” His answer spoke volumes: “It does what you expect it to do.” APL epitomizes consistency, illustrating in detail orthogonality, propriety, and generality. It also epitomizes parsimony, providing many functions with few concepts.
I once was engaged to review the architecture of a very ambitious new computer family, the Future Series (FS) intended by IBM’s developers to be a successor to the S/360 family. The architectural team was brilliant, experienced, and inventive. I listened with delight as the grand vision unfolded. So many fine ideas! For an hour, one of the architects explained the powerful addressing and indexing facilities. Another hour, another architect set forth the instruction sequencing, looping, branching capabilities. Another described the rich operations set, including powerful new operators for data structures. Another told of the comprehensive I/O system.

Finally, swamped, I asked, “Can you please let me talk to the architect who understands it all, so I can get an overview?”

“There isn’t one. No one person understands it all.”

I knew then that the project was doomed—the system would collapse of its own weight. Being handed the 800-page user manual confirmed in my mind the system’s fate. How could any user master such a programming interface?

When Collaboration Helps

In some aspects of design the very plurality of designers per se adds value.

Determining Needs and Desiderata from Stakeholders

If deciding what to design is the hardest part of the design task, is this a part where collaboration helps? Indeed so! A small team is much better than an individual at studying either an unmet need or an existing system to be replaced. Typically, several minds think of many different questions and kinds of questions. Many questions mean many unexpected answers. The collaborating team must ensure that each member gets full opportunity to explore his trains of inquisitiveness.

Establishing Objectives. Under any design process, the designer begins by conversing with the several stakeholders. These conversations are about the objectives and constraints for the design. The hard task is to flush out the implicit objectives and constraints, the ones the stakeholders don’t even recognize.
that they have. Indeed, from these conversations—what is said, how it is said, what is unsaid—comes the designer’s first estimate of the utility function.

A crucial part of this phase is observation of how the user does the job today, with today’s tools and circumstances. It often helps to videotape these observations, and to view them over and over.

Having collaborating designers participate is extremely useful for this phase. Extra minds

- Ask different questions
- Pick up different things that are not said
- Have independent and perhaps contradictory opinions of how things are said
- Observe different aspects of working
- Stimulate the discussion of the videotapes

**Conceptual Exploration—Radical Alternatives**

Early in the design process, designers begin exploring solutions—the earlier the better (as long as no one gets wedded to any solution), for the concreteness of postulated solutions usually elicits hitherto unspoken user desiderata or constraints.

**Brainstorming.** This is the time for brainstorming. Severally, each member of the design team sketches multiple individual schemes. Collectively, the team members prod each other into radical, even wild, ideas. The standard rules for this stage include “Focus on quantity,” “No criticism,” “Encourage wild ideas,” “Combine and improve ideas,” and “Sketch all of them where all can see.” More minds mean more ideas. More minds stimulating each other yield lots more ideas.

The ideas are not necessarily better. Dornburg [2007] reports a controlled industrial-scale experiment at Sandia Labs:

*Individuals perform at least as well as groups in producing quantity of electronic ideas, regardless of brainstorming duration. However, when judged with respect to quality along three dimensions (originality, feasibility, and effectiveness), the individuals significantly (p<0.05) out performed the group working together.*
Competition as an Alternative to Collaboration. In the conceptual exploration phase, one can alternatively harness and stimulate the creative powers of multiple designers by holding design competitions. These work best when the known constraints and objectives are concretely stated and shared, and when unnecessary constraints are carefully excised.

In architecture this practice has been routine for centuries. Brunelleschi established himself by winning the design competition for the dome of the Santa Maria del Fiore cathedral in Florence in 1419 (Figure 6-4). His radical concept, its feasibility made plausible by a scale model, opened new vistas, seen today in St. Paul’s and the U.S. Capitol.

Figure 6-4  Brunelleschi’s Dome, Santa Maria el Fiore
Anonymous, “View of Florence from the Boboli Gardens,”
19th Century, Watercolor, Museo di Firenze com’era, Florence, Italy/
Scala/Art Resource, New York.
In architecture and some major civil engineering works, there is a single client and multiple designers hoping to get the job. So a competition naturally suggests itself.

The situation is quite different in the normal product-development environment of a computer or software developer. There it is customary for a single team to be assigned to develop a particular product. There will always be competing ideas inside the team about different design decisions, and debates are routine. But only rarely does a management set up multiple teams to pursue a single objective competitively.

Occasionally, however, there will be a formal design competition within a corporate product-development setting. During System/360 architectural design we worked on a stack architecture for six months. Then came the first cost-estimating cycle. The results showed the approach to be valid for mid-range machines and up, but a poor cost-performer at the low end of the seven-model family.

So we had a design competition. The architecture team self-selected into some 13 little (one- to three-person) teamlets, and each did an architectural sketch, against a fixed set of rules and deadlines. Two of the 13 designs were best in my opinion as judge. They were surprisingly alike, more surprising because the teams were rather cool toward each other and had not communicated.

The confluence of those designs set the pattern for the project. (Their big difference, 6-bit-byte versus 8-bit-byte, occasioned the sharpest, deepest, and longest debate of the whole design process.)

I reckon the design competition, originally suggested by Gene Amdahl, to have been immensely invigorating and fruitful. It put everyone hard to work again after a demoralizing cost estimate. It got each person deeply involved in all aspects of the design, which greatly helped morale and proved valuable in the later design development. It produced a consensus on many design decisions. And it produced a good design.20

Unplanned Design Competitions: Product Fights. Not infrequently, it happens that design team B will so evolve its design that it begins to overlap the market objective of design team A. Then one has an ad hoc design competition, a product fight.
When Collaboration Helps

I’ve seen many product fights. They follow a standard script in five acts:

1. The two teams, who may not already know the details of each other’s work, meet, compare products and intended markets, and conclude unanimously that there is no real overlap between their products. Both should proceed full speed.

2. Reality appears, in the form of a market forecast or a skeptical boss.

3. Each team changes the design of its product to encompass all of the other product’s market, not just the overlapping part.

4. Each team begins wooing supporters among customers, marketing groups, and product forecasters.

5. There comes a shootout before some executive with the power to decide.

Scripts diverge at this point: team A wins; team B wins; both survive; neither survives the intense scrutiny engendered by the competition.

This scenario can and usually should be shortened by early action by a skeptical boss. Sometimes, however, it may be the best way to get a thorough (and impassioned) exploration of two quite different design approaches.

Design Review

The phase of design where collaboration is most valuable, even necessary, is design review. Multiple disciplines must review: other designers, users and/or surrogates, implementers, purchasers, manufacturers, maintainers, reliability experts, safety and environment watchdogs.

Each disciplinary specialist must review the design documents alone, for careful review takes time, reflection, and perhaps the study of references, archives, and other designs. Each will bring a unique point of view; each will raise different issues and find different flaws. But joint, group review is also imperative.

Demand Multidisciplinary Group Review. Group review has the power of numbers, but special power comes from the viewpoints of multiple disciplines. The review team should be much
larger than the design team. Those who will build the design, those who will maintain it, sample users, those who will market it—all must be included. Consider the review for a new submarine design. The supply officer sees a shortcoming; his spoken concern triggers a similar concern for the damage control specialist. The manufacturing tooling expert sees something hard to build; his suggested solution sets off alarms in the acoustic expert’s mind.

Designers at the Electric Boat Division of General Dynamics told me of a review in which the shipyard foreman took one look at a semicylindrical storage tank and quickly suggested rolling a one-piece cylinder, cutting it in half, and roofing it with a flat plate. This was in place of some 20 pieces the engineer had specified. Said the foreman, “We submarine builders are good at rolling cylinders.”

Similarly, a designer at Brown & Root in Leatherhead, England, told me of a design review for a deep-sea oil-drilling platform. The maintenance foreman pointed to a particular unit and said, “Better make that one out of heavy-gauge steel.”

“Why?”

“Well, we can paint it in the workshop before it’s installed, but where it goes, we’ll never be able to paint it again.”

The engineers redesigned the whole vicinity of the platform so the unit could be reached.

Use Graphical Representations. For design review, the most important aid is a common model of the product—a drawing, a full-scale wooden mock-up or virtual-reality simulation of a submarine, a prototype of a mechanical part, perhaps an architectural diagram of a computer.

A multidisciplinary design review often demands a richer variety of graphical representations of the design than the designers themselves have been using. Not everyone in the review will be able to visualize the end product from the engineering/architectural drawings. My observation from visiting various facilities is that such design reviews are probably the most fruitful applications of virtual-environment visualization technology.22
Sharing the product model and sharing each other’s comments are both vital to effective design review; tools for simulating such sharing are the sine qua non of group design reviews where all the players cannot be physically present. Here telecollaboration comes into its own.

When Collaboration Doesn’t Work—for Design Itself

The Fantasy Concept of Design Collaboration. The computer-supported-collaborative-work literature is peppered with a fantasy version of collaborative design. This would be harmless, except that the fallacious concept focuses ever more elaborate academic research on ever less useful technological tools for collaboration.

In this fantasy, a design team really or virtually sees a model of the design object—whether a house, a mechanical part, a submarine, a whiteboard diagram of software, or a shared text. Any team member proposes changes, usually by effecting the change directly in the model. Others propose amendments, discussion proceeds, and bit by bit the design takes form.

Not How Collaborators Design. But the fantasy concept doesn’t fit how collaborators really do design, as opposed to design review.

In all the multi-person design teams I’ve seen, each part of a design has at any time one owner. That one person works alone preparing a proposal for the design of his part. Then he meets with his collaborators for what is in effect a micro-session of design review. Then he normally retires and works out the detailed consequences of the decisions and directions discussed collaboratively.

If alternate proposals are made in the session, and not accepted by the owner, the proposer will often withdraw and develop an alternate design. Then the session will convene again, to choose, fuse, or strike off in some third direction.

Where’s Design Control? The fantasy concept has no function for originating designs, only refining them. The fantasy concept is
flawed as a model for collaborative design change, too. Schedule gain from collaboration implies concurrent activity; and concurrent activity requires synchronization, a step totally missing from solo design. Designer Jack owns the air ducts in an oceangoing tanker; Jill owns the steam pipes. As each fleshes out his design, and at every subsequent change, some mechanism of design control must monitor that they don’t both use the same space. Some resolution procedure must be in place for settling conflicts. Some version control must be established so that each designs against a single time-stamped version of all the earlier design work.

In one instance of the fantasy concept I have actually seen proposed, the client admiral views the design model for a nuclear submarine, and he moves a bulkhead to give equipment repairers better access. (Making this possible is a technically challenging task in a virtual-reality interface to a CAD system. Many techniques for real-time visualization depend upon the static nature of most of the world-model.)

But the challenge is not worth accepting! The admiral may want to move the bulkhead to see how the space will look and feel, and he may be allowed to do that in a playpen version of the model. But before any such move becomes part of the standard design version, someone or some program must check the effects on the space on the other side of the bulkhead, the structural consequences, the acoustic consequences, the effects on piping and wiring. Imagine the horror of the responsible engineers to find that the bulkhead has been moved by the admiral, who cannot possibly have known the constraints and design compromises it embodied. By the time there is a design for the admiral to walk through virtually, it is far enough along to require formal change control.

The fantasy model of collaborative design reflects a monumental unconcern about conceptual integrity. Jill pats the design here; Jim nudges it there; Jack patches it yonder. It is spontaneous; it is collaborative; and it produces poor designs. Indeed, we know the process so well that we have a scornful name for it—committee design. If collaboration tools are designed so they encourage committee design, they will do more harm than good.
Two-Person Teams Are Magical

Conceptual Design, Especially, Must Not Be Collaborative

Once the exploratory stage is past and a basic theme is selected, it’s time for conceptual integrity to rule. A design flows from a chief designer, supported by a design team, not partitioned among one.23

To be sure, the conceptual design thus pursued may run into a blind alley. Then a different basic scheme must be selected, and collaborative exploration is again in order until that new basic scheme is selected.

Two-Person Teams Are Magical

The foregoing discussion of design collaboration dealt with teams of more than two people. Two-person teams are a special case. Even in the conceptual design stage, when conceptual integrity is most imperiled, pairs of designers acting uno animo can be more fruitful than solo designers. The literature on pair programming shows this to be true during detailed design. Typical initial productivity runs less than two working separately, but error rates are radically reduced.24 Since perhaps 40 percent of the effort on many designs is rework, net productivity is higher and products are more robust.

The world is full of two-person jobs. The carpenter needs someone to hold the other end of the beam. The electrician needs help when feeding wire through studs. Child raising is best done by two actively collaborating parents. “It is not good for man to be alone,” while spoken in its truest sense about marriage,25 might usefully be preached to lone-ranger designers.

The typical dynamics of two-person design collaboration seem different from those of multi-person design and solo design. Two people will interchange ideas rapidly and informally, with neither a protocol as to who has the floor nor domination by one partner. Each holds the floor for short bursts. The process switches rapidly among micro-sessions of proposal, review and critique, counterproposal, synthesis, and resolution. There is typically a single thread of idea development, without the maintenance of
separate individual threads of thought as in multi-person discussions. Two pencils may move over the same paper with neither collision nor contradiction.

“As iron sharpens iron,” each stimulates the other to more active thought than might occur in solo design. Perhaps the very need to articulate one’s thinking—to state why as well as what—causes quicker perception of one’s own fallacies and quicker recognition of other viable design alternatives.

A classic 1970 paper by Torrance showed that dyadic interaction produced twice as many original ideas, produced ideas of twice as much originality, increased enjoyment, and led subjects to attempt more difficult tasks.26

Pair-wise design sessions still need to be interspersed with solo ones—to detail, to document the creative fruit, and to prepare proposals for the next joint session.

So What, for Computer Scientists?

Much effort by academic computer scientists has gone into the design of tools for computer-assisted collaboration by workers in their own and other disciplines. Distressingly few of these ideas and tools have made it into everyday use. (Important tools that have succeeded are code control systems and “Track Changes” in Word.) Perhaps this is because it is especially easy for academic tool builders to overlook some crucial properties of real-world team design:

- Real design is always more complex than we tend to imagine.27 This is especially true since we often start with textbook examples, which have perforce been oversimplified. Real design has more complex goals, more complex constraints to be satisfied, more complex measures of goodness to be satisfied. Real design always explodes into countless details.
- Real team design always requires a design-change control process, lest the left hand corrupt what the right hand has wrought.
- No amount of collaboration eliminates the need for the “dreariness of labor and the loneliness of thought.”
For these reasons, I think we should be very leery about assigning graduate students with little or no real-world design experience dissertation topics in the field of collaborative design tools. Moreover, our journals should be very slow to accept such papers that are not based on real-world experience and/or real design applications.

Notes and References

1. This marvelous phrase was quoted by Bernard Baruch, who said his attorney said it to him.
2. *Economist* [2009], “Harvest moon.”
3. The wise manager of a multi-project organization early launches a solo designer, or a pair, to start exploring designing for a technology foreseeable, but not yet buildable.
5. Shipyard foreman at Electric Boat, Groton, CT (personal communication).
6. The most complete scientific study I have seen comparing solo and individual designers is Cross [1996a], *Analysing Design Activity.*

The Delft protocols included a solo designer and a three-person team attacking the same problem, with both observed by video and the solo designer encouraged to think aloud. Twenty different chapters, each using its own analytical method, analyze the Delft video protocols. Most apply their authors’ own predefined categories of activity to one or both of the protocols. Many of the chapters either compare the activities and performance of the two alternatives or else analyze the social behavior of the team. The most specific conclusion is that by Gabriela Goldschmidt [1995], “The designer as a team of one”: “Detailed analysis leads to the conclusion that there are almost no differences between the individual and the team in the way they bring their work to fruition.”

Together the studies offer a rich set of perspectives that allow a reader to understand both the fertileness and the idiosyncrasy of design processes. The video transcription obviously captured a rich characterization of design behavior, . . . The limitation of current methods of protocol analysis, however, are made readily apparent. Each study by itself provides only a small peephole into the overall design process. Only through the cumulative breadth of multiple studies does the sense of the full process emerge.

This book clearly presents the current state of design protocol studies after thirty years of effort and relates them more generally to various theories of design.

7. Brooks [1995], The Mythical Man-Month, Chapter 4, 42ff.
13. Weisberg [1986], Creativity: Genius and Other Myths; Stillinger [1991], Multiple Authorship and the Myth of Solitary Genius.
14. R. Joseph Mitchell, the designer of the Spitfire, warned one of his test pilots (the user!) about engineers: “If anybody ever tells you anything about an aeroplane which is so bloody complicated you can’t understand it, take it from me: it’s all balls.”
15. Eoin Woods of Artechra says,

I'm not as pessimistic as you about joint design. I've worked in teams where we had spirited discussion to drive our designs and then agreed the solution among us (albeit sometimes with a benign dictator making final decisions). The designs remained coherent because it was one or two strong concepts of the design that won out and then drove all of the other decisions; we didn’t design by committee and “horse-trade” the detailed decisions (personal communication [2009]).
16. Brad Parkinson, now at Stanford, one of the two system architects/contracting officers for the GPS system, pointed out that the challenges of that task were substantially increased by having multiple contractors for the several system pieces (personal communication [2007]).

17. Holson [2009], “Putting a bolder face on Google.”

18. Mary Shaw of Carnegie-Mellon asks, “What does this say about modern software development environments and their APIs?”


20. Design competitions in organization design are yet different: the task is inherently political. The various competing forces usually do not even share the objective of getting the organization that works best. How well the organization will work is subordinated to who will have which levers of power.

21. Margaret Thatcher: “One wants documents [as opposed to viewgraph foils] so one can think through beforehand, and consult colleagues” (personal communication via Sir John Fairclough). American business all too often does reviews via PowerPoint presentations. Those vague bullets enable each participant to interpret the information as he pleases; they also facilitate the suppression of embarrassing but crucial details.

Lou Gerstner, turnaround CEO of IBM, startled the whole culture early on ([2002], Who Says Elephants Can’t Dance?, 43): “Nick was on his second foil when I stepped to the table and as politely as I could in front of his team, switched off the projector . . . it had a terribly powerful ripple effect . . . Talk about consternation. It was as if the President of the United States had banned the use of English at White House meetings.”

22. Brooks [1999], “What’s real about virtual reality?”

23. Harlan Mills’s concept of a supported-chief-designer team, a “surgical” team, is detailed in Brooks [1995], The Mythical Man-Month, Chapter 3.

24. Williams [2000], “Strengthening the case for pair-programming”; Cockburn [2001], “The costs and benefits of pair programming.”


27. See, for example, the impressive PhD dissertations by Hales [1991], “An analysis of the engineering design process in an industrial context” (Cambridge), or Salton [1958], “An automatic data processing system for public utility revenue accounting” (Harvard), for detailed documentation of what is involved in an actual design.
Subject Index

Note: Boldface page references indicate definitions or the beginning point of a substantial treatment. Page numbers in those scopes are not separately listed under that term.

2-D
  context view, 221
  drawing, 220
  specification, 210
  24/7 operation, 338, 359
3-D
  display, 221
  model, 214
  perception, 216
  specification, 211
6-bit byte, 76, 92
8-bit byte, 76, 92, 314, 319, 322
abstract data type, 334
access method, 332, 340, 342
ACM Turing Award
to John Cocke, 249
to Ted Codd for relational database concept, 248
acoustic analysis, 109
acoustic simulation, 225
Ada programming language, 179
Advanced Computer Architecture, Brooks’s course, 134
adverbs, specifying, 212
gale method, 180
Air Force (U.S.), 210
Air Force Studies Board, 42
air-traffic control, 57, 179
Airbus 380, 91, 99
Airbus UK, Division of BAE Systems, 251
airline reservation system, 318
Algol, 232, 335
alphabet, lowercase, 8-bit byte, 319
alternative branch, 17, 190, 192
amateur designer, 168
ambassador, 91
Amdahl Corporation, 323, 341
American Telephone & Telegraph (AT&T), 233
analysis software, 66
Ancient Airs and Dances (music), Respighi, 149
annotation, 197
AFL programming language, 72, 124, 141, 348
Apollo space program, 325
appeal procedure, 362, 364
Apple (Computer) Inc., 70
iPhone, 177, 233
iPod, 177
Macintosh interface, 142
Apple I computer, 150
Apple II computer, 232
Appletalk, 232
application set, 134, 335
apprentice, 178
architect, 41, 43, 45, 204, 223
building, 153
chief, 115, 344
chief, giving full authority to, 344
computer, 121, 168
naval, 176
system, 130
architectural
control, 343
diagram, 78
program, 254
sketch, 76
style, 253
architecture
building, 135, 140, 176, 204
computer (see also computer architecture), 120, 141, 142, 153, 313
computer, general-purpose, 133
computer, special-purpose, 134
definition of, 348
doctor of a design, 4, 347
multiple concurrent, 326
operating system, 161
team, 76
arithmetic unit, 157
Army (U.S.), 210
Art Institute of Chicago, 151
Artecha Ltd., 84
artifact
general-purpose, 133
special-purpose, 133
artificial
constraint, 128
intelligence, 16
artistic
concept, 128
design, 10
as-built, 193
assembler software, macro, 333, 335
assessment of design case, 271, 293, 306, 323, 341, 351, 362
assumption, 168, 173, 179
assumption, implicit or explicit, 114
Atlantic Systems Guild, 240
atomic bomb, 233
attribute branch, 17, 189
audio display, 224
authority, 94
autostereoscopic display, 222
Baby (Manchester early computer), 158
backtracking, 16
backtracking, automatic, 223
BAE Systems (UK), 91, 92, 198, 199, 250
ballistic missile, 232
barriers
cultural, 93
space, 93
time-zone, 93
batch operation of a computer, 115, 169, 172, 332, 335
Bauhaus architecture, 163
Bazaar Model of designing, Raymond’s, 54
beach house, 120, 259
binding
name, 339
too-early, 44
bloat, feature, 42, 48, 115
blueprint, 149
Board of Directors, Triangle Universities Computation Center, 357, 360
Boehm’s Spiral Model, 51
Boeing 777 airplane, 94
Boeing-Sikorsky RAH-66
Comanche Helicopter, 39, 40
bold decisions, in design case, 257, 260, 280, 298, 314, 332, 348, 356
bold leader, 238
book, design of, 347
boss, skeptical, 77
box, organization chart, 238
Box Structure Method, 111
brainstorming, 74
branch
  alternative, 190
attribute, 189
branching, 171, 172
branching thought-trail, 224
breakthrough, 188, 192
bridge collapse, 168
brilliance, design, 245
Brooklyn Bridge, 244
Brooks house, 1990s remodeling of, 188, 279, 297
Brown & Root (now Kellogg, Brown & Root), 78
budgeted resource, 14, 119, 224, 254
budgeted resource, in design case, 260, 266, 275, 285, 289, 302, 304, 350, 382, 356
bug, program, 55
building, 155, 204
design, 135, 140
design disciplines, 45
bureaucracy, 234, 250
Burroughs B5000 family, 25
business data processing, 316
By-Laws of Triangle Universities Computation Center, 364
byte
  6-bit, 76, 92
  8-bit, 76, 92, 314, 319, 323
calendar, mental spatial model of, 33
Cambridge University, 158, 198, 322
Capability Maturity Model (CMM), 236
career, academic, 256
case studies, Part VI
cathedral, 54, 57, 59
CATIA CAD software, 91
CAVE virtual environment installation, 222
CDC 6600 supercomputer, 70, 244, 252, 325
cell phone, 129, 232
change
  configuration, 360
  control, formal, 69, 79
design, 94
  order, 44, 45
changes, design, in design case, 268, 269, 292, 319, 342
character of a designed object, 11
charter, project, 47
checking, hardware, 320
Chicago Manual of (English prose) Style, The, 148, 149
chief architect, 71, 115, 344
chief architect, giving full authority to, 344
chief designer, 81, 239
Chief Executive Officer (CEO), 364
  of owning university, 360, 362
  of TUCC, 361
churches, London (Wren’s), 128
Civil War (U.S.), 109
clear architecture, 94, 140, 142
cleanroom software technique, 108, 111
collaboration, Part II
  aids, 91
  co-located, 98
  computer-assisted, 82
design, 80
  manners, 95
  pattern, 98
  remote, 95
co-evolution model of designing, 44, 83
doctrine, 69
COBOL, 170, 232, 321, 333, 335
code control system, 82
code generation, 143
coherence, a component of design goodness, 70
collaboration, Part II
  aids, 91
  co-located, 98
  computer-assisted, 82
design, 80
  manners, 95
  pattern, 98
  remote, 95
collaborative design tools, 83
collaborative working, computer-mediated remote, 98
collaborator, design, 67, 95
collaboratory, scientific, 100
collection of exemplars, 160, 161
Comanche Helicopter, Boeing-Sikorsky RAH-66, 40, 46
command, to a CAD system, 208
committee design, 80
Committee on Pre-Milestone A System Engineering, 39, 42
Common Lisp programming language, 141
communication
channels, informal, 93
cost, 68
technology, 90, 95, 315
comparative analysis, 160, 161
compatibility
program, 147, 339, 348
upward and downward, binary, 314, 318, 325, 333, 334
Compendium software, 185, 195, 196
compensation, 247
competence, assuming, 253
competition, 324
competition, design, 78, 85, 248, 319, 328
compile time, 335, 339
compiler, 332, 333, 335
multiple, 335, 337
optimizing, 249, 255
completeness, 144
complexity
design, 115
software, 121
composability, 141, 144
comprehensibility, 145
computation
center, 355
engineering, 134
computational fluid dynamics (CFD), 66
computer architect, 121, 168, 253
computer architecture, 67, 70, 73, 94, 120, 133, 141, 155, 253, 254, 313, 334, 347
general-purpose, 133
special-purpose, 134
Computer Architecture: A Quantitative Approach, Hennessy and Patterson, 351
Computer Architecture: Concepts and Evolution, Blaauw and Brooks, 149, 347
computer center, scientific, 178
descriptions, 351
design, 161
tamily, 333
first-generation, 157, 358
graphics, 204
graphics model of a design, 45
graphics simulation, 223
science, 67
science building, 135
scientist, 82, 203
second-generation, 158, 315, 324, 333
stored-program, 157
third-generation, 158
"Computer Zoo," 24, 352
computer-assisted
collaboration, 82
design (CAD), 80, 197, 224, 254, 298, 307
computing
academic, 356
administrative, 356
center, campus, 363
interactive, 356
purpose of, 358
scientific, 134
Concepts and Facilities, OS/360, Witt, 344
conceptual design, 45, 74, 81 integrity, 9, 41, 46, 56, 64, 69, 80, 115, 119, 120, 124, 145, 151, 205, 239 concurrent engineering, 180 conferencing, face-to-face, 97 configuration change, 340, 360 of a computer, 189 I/O, 130, 324 topology, 131 consensus, 233, 234 consistency, 69, 70, 72, 142, 146, 148 console, operator’s, 337 constitution, 120 Constitution of the United States, 109 constraint, 14, 27, 41, 68, 82, 109, 120, 123, 127, 131, 133, 250, 254, 341 artificial, 128, 135 misperceived, 130 constraints, for design case, 262, 286, 300, 318, 351, 360 construction drawing, 45, 223 context in design case, 260, 280, 298, 314, 333, 335, 349, 357 view, 221, 225 contract, 39, 44, 52, 54, 57, 58, 132 contract, fixed-price, 45, 57, 59, 85 contracting point, 57 contractor, 39 contractors, multiple, 85 control block, in a software system, 25, 342 control card, for a program scheduler, 171, 172 Control Data CDC 6600, 158 Control Data Corporation (CDC), 252, 325, 337 convergence of book writing, 347, 352 of computer architectures, 349 conversion from older computers, 319 coop education program, 245 corporate processor manager (IBM), 316 Corporate Product Procedure (IBM), 234 correctness-proving, 106, 107 cost as budgeted commodity in design, 121 development, 121 estimate, 224, 236 hardware, 158, 318 lifetime, 132 of living, 90 manufacturing, 121 software, 321 surrogates, 327 varieties, 121 cost-plus contract, 44 cotton-picking machine, 67 Cray 1 supercomputer, 244, 252 Cray Computer Corporation, 252 Cray Research Corporation, 252 Cray supercomputers, 158 creativity, 51, 82 criteria for goodness in computer architecture, 9 critical-path project scheduling, 196 criticism of design, 252 of exemplar designs, 160 critique of the Rational Model, 29 critiqued practice, as pedagogy, 244, 252 cryptanalysis, 158 cultural barrier, 93, 97 curricula, design, 180 “Cut to plan; bang to fit,” 68 cutaway view, 223
Data Declaration (DD) statement, 171
data
  format, 143
    management, operating
      system component, 335, 338, 341
    processing, 318, 320, 336, 356
      structure, 335
    type, abstract, 334
  data-streaming co-processor, 158
database, 67, 323
David, Michelangelo, 127, 128
De Architectura, Vitruvius, 9, 139
debug, 107, 115, 336, 341
DEC (Digital Equipment Corporation), 159, 323
DEC PDP-11, 324
DEC PDP-8, 159
decimal datatype, 8
decision
  bold (see bold decision)
  design, 187
    design, for a computer
      architecture, 348
    design, for Triangle
      Universities Computation
        Center, 360
  tree, 24, 187, 348
    tree versus design tree, 193
decision-making
  power, 356
    burden of, 146
decisions, decision, in design case, 262, 286, 305, 319, 338, 351, 360
decomposition of design
  problem, 30, 144
  Defense Science Board, 40
  Delft protocols, 83
delight
  in design case, 271, 306, 325, 341, 351
  Vitruvius’s design criterion, 8, 139, 145
DeltaSphere Inc., 223
Department of Defense (U.S.)
  acquisitions process, 42
depth perception, 216
depth-first search, 16, 187
desideratum, for a design, 14, 23, 26, 57, 68, 73, 109, 120, 134, 254
design
  3-D, 213
    adaptive, 10
    alternate, 79
    alternative, 224, 254
    artistic, 10
    building, 204
    by committee, 71, 84
    committee, 80
    competition, 78, 85, 319, 328
    concept, 6
    conceptual, 45
    decision, for a computer
      architecture, 348
      decision tree, 187
      development, 45
      disciplines, technical, 155, 176
      exemplar-based, 160
      functional, 204
      house, 204
      innovative, 232
      integrated, 194
      language, spatial, 207
      mechanical, 204
      meeting, 197, 198
      methodology, 106
      modular, 194
      organization, 355
      original, 10
      paradigm, solo, 244
      paradigm, team, 244
      problem, intractable, 280
      process, 5, 17, 123, 157, 244
      process, action-centered, Denning and Dargan’s, 57
      process, speed and ease of, 195
      program, architectural, 27
      rationale, 186
      reasoning, appositional nature of, 30
review, 77
routine, 10
schematic, 121
software, 106, 135, 161
space, 95, 187, Part VI
space, working outside of the,
28
spatial, 135
style, 95, 246
team, 17, 114, 119, 148
theory, 30
theory of, 153
trajectory, 254
tree, 15, 24, 280
tree versus decision tree, 193
verification, 108
Design of Design, The, Glegg, xv
Design Research Society, 9
Design Studies journal, 9
Design Thinking Research Society
Symposium 7, 6, 10
design-build process, 44, 46
designer
airplane, 176
chef, 239
formal education of, 244
great, Part V, 231, 249
lone-ranger, 81
designer-computer interface, 204
designer-implementer link, 177
designers, in design case, 261,
281, 298, 314, 343, 349, 357
"Designing Software for
Ease of Extension and
Contraction," Parnas, 195
Desktop (Macintosh), 142
development
cleanroom, 111
distributed, 92
incremental, 179
device-independent input-output,
340
diagnosis of faults, 199
diagramming tool, generic, 197
Digital Equipment Corporation
(DEC), 155, 159, 323
Digitek Fortran compiler, 124
director
CEO of TUCC, 361
of Triangle Universities
Computation Center,
selection of, 261
university computation center,
363
discipline, for a design team,
122
disciplines, multiple, 77
disk, 324, 335, 356
accesses, 122
residence of an operating
system, 337
Disney World, 151, 164
display, 220
audio, 224
context, 225
design, 225
haptic, 225
test cases, 225
workbook, 223
distributed development, 92
divergence of computer
architectures, 349
divorce, of designers from users
and implementers, 175
documentation, 55, 82, 148
maintenance, 156
shared, 95
DoD Standard 2167A (DoD-STD-2167A), 32, 36
drawing
construction, 46, 223
house, 225
view, 220
dream system for designing
houses, 219, Part IV
"dreariness of labor and the
loneliness of thought," 82
DRed (design rationale capture
software), 197
dual ladder of promotion, 247
Duke University, 151, 179, 387
Dutch Golden Age, 145
ease of extension, 145
ease of learning, 144, 145
ease of maintenance, 145
ease of recollection, 145
ease of use, 144, 145
economy of scale, 359
EDSAC (Cambridge early computer), 158
education
architectural, 244
formal, 180, 244
medical, 244
technical, 248
Electric Boat Division, General Dynamics, 78, 83
elegance, 142
empirical measurement, 182
empiricism, 105
emulation, 321, 323, 334
“Energy” (Sayer’s term for component of creating, same as “Implementation”), 4
engineer
airplane, 91
manufacturing, 198
mechanical, 176
engineering
computation, 134, 316, 324
concurrent, 180
drawing, 180
product, 199
software (see software engineering)
Engineering and Physical Sciences Research Council (UK), 198
Engineering Design Centre (Cambridge University), 198
Engineering Research Associates (ERA), early computer manufacturer, 252
entropy, 240
Epcot (Disney World), 151
epistemology, 109
epistemology of practice, 31
error rate, 81
escape hatch, to protect vital interest, 362
essence (Aristotle’s term), 5
esthetics, 139, 204, 216
estimate, cost, 237
estimator (metric used for estimating), 25
ETA 10 supercomputer, 337
Etruscans, 253
evolution
biological, 54
computer architecture, 347
doctorate, 192
evolutionary selection, 53, 55
exemplar, 153, 207, 253, 350
extensibility, 141
exterior view, 222
extraneousness, 143
eye height, 213, 214
EyeBall viewpoint specification device, 213, 222, 225
FAA (Federal Aviation Administration), 130
face-to-face time, 93, 97
facial expression, 97
failure, 167, 173
fallen human, 44, 52
fallibility, 106, 107
fan club, 232
FBI (computer) system, 41
Federalist Papers, The, 109, 147
Fetchmail software, 54
firmness, in design case, 274, 323, 341, 351, 357, 362
firmness, Virtuvious’s design criterion, 139, 140
first to market, 67
first-generation computer, 358
fixed-price contract, 45
flow, uninterrupted concentration, 250
follow-on product, 235
forecast, market, 236, 237, 327
“forget the budget,” 289
“Form is liberating,” 127
formal education, 244, 247
formal method, 108
formal model, 331
formal proof, 108
formal specification, 111
formal synthesis method, 181
Fortran programming language, 136, 169, 170, 232, 252, 333, 334, 335, 337
Fortune magazine, 316
free software, 335
Fujiitsu, 324, 341
function
point, 121, 122
set, 144
too rich, 341
functional design, of a building, 204
functional space, in a building, 205
General Dynamics, 78
General Electric (GE), 159
general-purpose design, 127, 133
generality, 70, 72, 135, 144, 173
generation process for operating system (“sysgen”), 332
genius, 231, 243, 249, 250
geometric model, 57
Georgian house architecture, 205
gIBIS (Conklin’s graphical version of Issue-Based Information System), 196, 197, 198
gift-prestige incentive and reward, 55
glass house (mainframe computing center), 233
global communications, 67
global market, 67
Global Positioning System (GPS), 85, 120
global strategy, 99
global village, 89
goal, 14, 68
goal iteration, 22
goal-defining document, 114
goal-setting process, 23
good practice, rules of, 161
Google, 72, 85, 124
Gothic architecture, 148
grant proposal, 120, 123
graph, non-planar, 186
graphical representation, 78
great design, 231, 244, Part V
great designer, 231, 243, Part V
greed, 44
Greeks, 253
GRIP system (UNC molecular graphics system), 179, 203, 216
group review, multidisciplinary, 77, 108
growing yourself as a designer, 252
guess, 116
Handbook of Software Architectures, Booch, 161
hands-free operation, 223
haptic delight, 140
haptic display, 220, 225
haptic display, passive, 310
hardware
error, 338
computer, 109
Harvard Mark IV, 107
Harvard University, 107
head-mounted display, 96, 179
heir, project, 224
Hewlett-Packard, 159
hierarchical order, 186, 190, 206
high-level language, 158, 333, 335
highbrow style, 150
highlights and peculiarities of design case, 260, 280, 298, 314, 332, 348, 356
Hitachi, 324, 341
house
beach, 259
design, 120, 133, 168, 203, 219, 226, 239, 279
remodeling, 279
virtual, 225
wing addition, 279
humble access to supervisor program, 339
humility, 144, 253
hurry to market, 67, 351
I/O (input-output)
attachment tree, System/360, 342
channel, 322, 326
configuration, 326, 332
control, 339
device, 232, 339
device, 8-bit, 99, 324
device, random-access, 211, 337
device-independent, 332, 340
interface, 320, 324, 325, 339
IBIS (Issue-Based Information System), 196
IBM (International Business Machines Corp.), 70, 85, 92, 108, 159, 234, 237, 238, 248, 249, 313, 350, 358
1401, 155, 323
1401S (never delivered), 323
1410/7010 operating system, 171
2311 disk, 337
704, 155, 170
7074, 321
7080, 321
709, 178
7090, 321
8000 series (never delivered), 315, 322
801 RISC computer, 160
9020 System for FAA air traffic control, 130
corporate product lines, 315
corporate processor manager, 316
Corporate Product Procedure, 234
Data Systems Division (DSD), 315, 323
Disk Operating System/360 (DOS/360), 334
Future Series (FS, never built), 73
General Products Division (GPD), 315, 323
MVS (Multiple Virtual System) Operating System, 25
Operating System/360 (OS/360), 25, 120, 164, 169, 178, 240, 331
OS/360 Job Control Language (JCL), 169, 339, 340, 342
Research Division, 250
Stretch multiprogramming operating system, 178
Stretch supercomputer, 48, 158, 178, 249, 315, 320, 337
System z/90, 323
System/360 (“mainframe” computer family), 6, 76, 92, 123, 130, 155, 158, 168, 234, 237, 251, 310, 313, 333, 349, 356
System/360 Model 20, 234
System/360 Model 30, 323
System/360 Model 75, 325
System/360 Model 91, 325
System/360 Models 30, 40, 50, 65, 75, 90, 323
System/360 name origin, 336
System/370 computer family (descendant of System/360), 323
System/370 computer family (descendant of System/360), 323
z/90 computer family
(descendant of System/360), 136
z/OS operating system, 115, 232, 169
“IBM’s $5 billion gamble,” 316, 324
IBM Laboratory
Böblingen, German, 92, 234, 240
Boca Raton, FL, 70, 251
Boulder, CO, 92
Endicott, NY, 92
Hursley House, UK, 70, 92
La Gaude, France, 92
Lexington, KY, 92
Lidingö, Sweden, 92
Poughkeepsie, NY, 92
San Jose, CA, 92
Uithoorn, Netherlands, 92
IBM System/360 Principles of Operation (programmer’s manual), 7
“Ideal” (Sayers’s term for component of creating, same as “Architecture”), 4, 5
Implementation (component of creating), 4, 5, 134, 143, 325, 348
implementation incremental, 107, 111
technology, 134, 177
microprogrammed, 321, 326
multiple concurrent, 327
over-specified, 177, 327
implementer, 42, 77
incidental (or “accidental” task component), 5
incremental building of software, 226
incremental development, 179
incremental implementation, 107, 111
index book, 187
register, 141
industrial design, 140
industrial-strength operating system, 338
informal communication channel, 93
informal culture, 156
innovation, 70, 92, 162, 238
insight, 355
instruction cache, 255
format, 123, 143
integrated circuit, 158, 333
integration, system, 68
Intel 8080A, 144
microprocessor style, 156
intellectual property, 55
“Interaction” (Sayers’s term for component of creating), 4
interaction with users, 179
interactive computing, 356, 359
interactive debugging, 178
interactive graphics, 179, 204
interdisciplinary negotiation, 71
interface between system components, 68, 344
clean, 94
definition of, 94
designer-computer, 204
standard I/O, 92, 320
two-handed, 207
user, 143
interior view in architecture, 213, 222
Internal Revenue Service (computer) system, 41
international engineering group, 97
international venture, 91
interruption, program, 320, 335
introvert, 246
investment, financial, 359
Iron Bridge (Shropshire, UK), Pritchard and Darby’s, 66
“iron sharpens iron,” 82
Subject Index

“Issue of Fundamental Importance,” 361
iteration, 17, 111, 171, 172
iteration between problem and solution space, 53
iterative design, 179, 182
job
computation unit, 335
concurrent execution of in scheduler, 339
two-person, 81
Job Control Language (JCL), 169, 339, 340, 342
John Deere, 67
joint computer center, 355
journal, 254
journal reviews of exemplars, 160
joy of ownership, 95
of work, 95
Kenwood House, UK, Adam’s, 136
kernel, formal proof of operating system, 110
keyboard, 212
keyboard equivalent of menu commands, 208
keypad, numeric, 213
kinetic depth effect, 97, 216
kitchen design, 178, 297
La Sagrada Familia cathedral (Barcelona), Gaudi’s, 151
language
concept, distinct, in programming language, 124
high-level, 170
imperative, 207
scheduling, 170
layering of drawings, 221, 308
laziness, 162
lead time, long, 46
learning/teaching cost, 68
Leatherman (multipurpose tool), 163
lessons learned, 173
lessons learned from design case, 276, 294, 310, 327, 344, 352, 363
library, program and declaration, 343
library of exemplars, 154, 155, 206, 226
lifetime of a computer architecture, 136
cost, 132
product, 134
Linac word-processing software, 226
limiting resource, 120
line authority, 316
linearization of general graph, 186
linker software, 339
Linux operating system, 54, 55, 56, 164, 177
Lisp programming language, 141
locality, 212
Lockheed F-117 (Nighthawk stealth fighter), 70
log of design trajectory, 186, 223, 280, 308
London churches (Wren’s), 128
lone-ranger designer, 81
look and feel, of an interface, 80
Lotus software, 142
lowbrow style, 150
lower-case alphabet, 319
Lufthansa Flight 2904 disaster, 110
MacDraw software, 72
MacPaint software, 72
macro assembler, 333, 335
macro assembler, OS/360, 170
macro-operation, 335
mainframe computer (see also IBM System/360 computer family), 117, 313, 333
maintenance, 25, 77, 78, 120, 132, 156, 186, 262, 275, 276, 317
majority vote, in organization, 361
management style, 250, 252
manager, project, 123, 239
Manchester University (UK),
early computer successes, 158
Manchester Atlas, 159
manipulation, of virtual objects, 207
manual, user, 149
manufacturing, 72, 77, 91, 176, 198
manufacturing cost, 121
“Many hands make light work—
Often, but many hands
make more work—Always,” 68
Marine Corps (USMC), 40
market
forecast, 236, 315, 333
mechanism, 55
marketer, 42, 77
marriage, 65, 81
mass market, 71, 179
massing, in architectural design, 204
mathematical linguistics, 67
matrix organization, 348
McGraw-Hill Construction, 224
mechanical engineering, 16, 204
meeting
as refuge from labor and
thought, 63
face-to-face, 93
whole-team, 93
Memex (Vannevar Bush’s
information system), 186
memory
addressing capacity, 317
bandwidth, 120, 123
dump, 336
magnetic core, 318
management, automatic, 168
size, 122, 129
size configurations for
OS/360, 332, 334
mentor, 244, 245, 246
menu, 208, 209
menu, customizable, 212
merger sort, 232
meta-design, 4
metaphor, 142
microcomputer, 156, 157
microdecision, 146
microprogrammed
implementation, 321
Microsoft, 246
Excel spreadsheet, 142
PowerPoint, 85
Project software, 196
Visio, 197
Visual Basic programming
language, 141
Windows, 232
Word, 156
Word document, 96
MIL-STD-498, U.S. Department of
Defense, 36
milestone, 39, 43, 47, 54
military
assault plan, 120
weapon system acquisition,
42, 72
mind, 203, 219
Minicad software, 308
minicomputer, 157, 233
minicomputer revolution, 357
Minneapolis I-35W bridge
collapse (2007), 168
miscommunication, 177
misperceived constraints, 130
mistake, 167
MIT (Massachusetts Institute of
Technology), 159, 322, 338
MIT Whirlwind, 155
MITRE Corporation, 130
mock-up, 45, 78, 208, 307, 310
model
sound, 225
starting, 224, 226
modeling, computer, 66
models, library of, 210
Models of Designing, Part I
Modern English Usage, Fowler, 148
modular design, 194
Monticello (Jefferson’s home), 139
Morse code, 145
mouse input device, 220
multi-person discussions, 81
Multics, 164, 338
multidisciplinary review, 77
multiple designers, 226
multiprocessing, 321, 338
multiprogramming, 332, 337
naïve technologies, 66
name-space, 159
National Medal of Science, awarded to Cocke and Gomory, 250
National Medal of Technology (awarded to Capability Maturity Model in 2005), 240
National Research Council, 42
National Science Foundation, xiii, 358
natural language, 142
natural selection evolutionary process, 54
Nautilus, U.S.S., submarine, 65
Navy (U.S.), 210
network management, 173
North Carolina, 356
Computer Orientation Project (NCCOP), 356, 361
Museum of Art, 151
State University (NCSU), 357
State University, School of Design, 254
notebook, 252
Notes on the Synthesis of Form, Alexander, 194
noun specification in computer interface, 210
noun-verb rhythm, 207
numeric keypad, 213

Oak Park Church, Frank Lloyd Wright’s, 146
object-oriented programming, 179, 342
objectives, 42, 73, 109, 123, 133, 160, 254
in design case, 261, 283, 285, 299, 316, 336, 350, 358
discovered, 284
Office of the Future, Fuchs’s, 89
olfactory display, 220
OmniPlan software, 196
one-liner (APL program), 141, 150
open-source design, 54, 177, 226
operating system, 67, 122, 156, 161, 317, 320
batch, 169
evolution, 338
first-generation, 332
in control, 337
industrial-strength, 332
multiprogramming, 178
second-generation, 333
secure, 108
tape-based, 172
time-sharing, 159, 164, 168, 178, 240, 331
Operating System/360 (OS/360), 7, 42, 115, 120, 122, 141
operation set, 143
operator, computer, 332
operator’s console for computer, 178
opportunities, in design case, 262, 292, 300, 317, 350, 359
optimization, 120, 255
organization
design of, 108, 355
multi-project, 83
originality, 82, 162
ornamentation, 146
orthogonality, 70, 72, 143
overview chart, 199
owner, of a design, 79
ownership
  of a design, joy of, 95
  of a design, sequential, 95
paging, 159
pair programming, 81, 85
Panama Canal, 244
paradigm
  shift, 171, 173
  solo design, 244
  team design, 244
parsimony, 72, 135, 140
participatory design, 70
partitioning of a task, 68, 91, 92
partitioning of a task, cost of, 68
Pascal programming language, 232
pattern, system-structure, 156, 252
Pavilions (University of Virginia), Jefferson's, 151
perception, 3-D, 216
performance, 141
  parameter, 43
  range, 335
  simulator, 122
performance/cost curve, quadratic, 356, 359
ratio, 120, 141, 318, 359
peripheral processor, 325
personal computer (PC), 70, 232, 323
philosopher of technology, 54
pipe, in UNIX and Linux, 55
pipelined data path, 144
pipelining, instruction, 249
pitch axis, 213
PL/I programming language, 170, 335, 337, 343
Platonic ideal, 6, 7
pointing, 208
politics, 91
postulating unknown user and use characteristics, 116
power
decision-making in an organization, 356, 362
dissipation, 121
Praeludium and Allegro in the style of Pugnani (music), Kreisler, 150
precedent, design, 154, 253, 301, 350
presentation, 97, 181
prestige incentive, 226
pricing, 132, 236, 237, 315
pride, 44, 162
problem, separable, 190, 192
process
  house design, 187
  improvement, 237
  processes and procedures, standardized, 43
  processor, peripheral, 325
product
definition, 236
development environment, 76
engineering, 199
fight, 76, 155, 323
follow-on, 327
line, 234, 315
procedure, 232
software, 111
special-purpose, 127
professional responsibility, 320
program
  architectural, 27, 45, 121, 254
development, 121
Program Evaluation and Review Technique (PERT), 196, 200
programmer, 176
programming language, 124, 135, 140, 142, 161, 169, 170, 171, 172, 335
programming language, specialized, 336
progressive discovery and evolution of requirements, 52, 54, 57
progressive refinement, 205, 217, 224
progressive truthfulness, 204, 224
project
course, in education, 181
management tool, 196
manager, 123, 239
projection, 3-D to 2-D, 211
proof, formal, 107, 108, 140
propriety, property of a design, 70, 72, 143
protection
of great designers, 250
of operating system, 338
prototype, 48, 57, 78, 107, 182, 205, 344
punched card, 170, 171, 172
purchaser, 72, 77
Python programming language, 232
quality control, statistical, 111
radiation-treatment, design of, 180
random generation, evolutionary process, 54
Rational Model of designing, 13, 52, 54, 58, 187
Rational Model of designing, critique of, 21
rationale-capture culture, 198
rationale for design decision, 156, 185, 223, 308, 314
rationalism, 105
rationed resource (see budgeted resource)
RCA (Radio Corporation of America), 323
real-world experience, 82
real-world team design, 96
realization of a design, 5, 325, 347
recruiting great designers, 245
Reduced Instruction Set Computer (RISC), 157, 159, 249, 255
redundancy of human language, 142
regression testing of software, 107
relational database, 248
reliability, 77, 111, 130, 317
remodeling, house, 279
remote access, 332
remote job entry, 356
Report Program Generator (RPG) software, 333, 335
representation of design, 186
requirement
creep, 42
design, 33, 39, 54, 57, 190
discovery of, 289
statement, formal, 27
system-level, 39
top-level, 42
Requirements Traceability Matrix, 43
requirements-setters, 42
research
monograph, 349
problem, 226
Research Triangle (region of central North Carolina), 359
Research Triangle Park, NC, 357
resolution procedure, 80, 81
review, design, 77, 80, 81, 181, 198
revolution, 235
microcomputer, 159
minicomputer, 159
RISC, 159
technology, 157
rework, 94
rhythm, noun-verb, 207
RISC (see Reduced Instruction Set Computer)
RISC I (Berkeley computer), 160
risk, 48, 57, 59
robustness, 41, 338
rocking about yaw axis, to aid depth perception, 216
roll axis, motion about, 213
Rolls-Royce plc; turbine engines provider, 198, 199
Roman architecture, 253
rotation of assignments, 246
Royal Academy of Engineering (UK), 240
rules
of good practice, 161
protection from, 250
sabbatical leave, 249
Salisbury Cathedral, iv, xiv
sampling, 116
sandwich education program, 245
Santa Maria del Fiore cathedral (Florence), 75
satisfice, 16, 18, 34, 82
scale model, 45
scenario (see use case)
schedule
project, 94, 123
urgency, 42
scheduler, operating system component, 171, 335, 338, 342, 169
scheduling
language, 169
time, between compilation and execution, 170, 171, 339
schematic design, 120
scientific computing, 67, 134, 178, 316, 318, 320, 336, 356
scope of object selection, 210
screen size, 220
scripting language, 169, 171
search engines, 87
search of design space, 15, 53, 128, 153
Second Life virtual world, 101
selection
menu, 212
object, 209
self-expression, 162
semantics, 94, 210
sensitivity analysis, 116
separable problems, 192
separation of policy and operations, 360
sequence, writing, 351
shampoo, 66
shared whiteboard, 97
shipyard, 68, 180
short course, education, 247
Siemens AG, 324
Silicon Valley CA, 90
simulation, 107, 108, 213
simulation, computer graphics, 225
simulator, 334
executable computer, 348
performance, 122
sin, 39, 43
Sitterson Hall (campus building, University of North Carolina at Chapel Hill), 165
situation awareness, 221
size (of work surface), 220
sketch, 211, 252, 254, 308
Sketch Graphics Acts software, 99
skill, specialized, 90
skunk works, 70
Slinky toy, 163
sloth, 44
Small Homes Council, 305
SmartDraw software, 196, 197
social justice, 70
sociological advance of minicomputer, microcomputer revolutions, 159
sociological status, in dual ladder, 247
software
custom application, 156
ingeering, 16, 22, 32, 67, 92, 106, 122, 135, 155, 176, 204, 225, 231, 236, 244, 252
engineering laboratory course, 181, 200
failure, 338
mass-product, 156
process, 111, 231, 331
product, 111, 231
support package, 333
Software Engineering Institute (SEI), 231, 236
Solid Logic Technology (SLT), 318
sort program generator, 336
sound intensity plot, 225
Soviet (USSR), 324
space barrier, 93
spacecraft, 120
Spanish Architecture Museum (Barcelona), 151
spatial design, 135
special-purpose artifacts, 127, 133
specialization, 67, 90
specialization, technological, 93
specialization, 3-D, 211
architectural, 223
cosy, 148
formal, 111
hierarchical, 148
software design document, 48
view, 223
Spiral Model, Boehm’s, 44, 51, 57
Spitfire (World War II aircraft), 70, 84, 232, 244
SPool (simultaneous peripheral operation on-line), 335
SPREAD Report, of IBM committee, 316, 321
St. Paul’s Cathedral (London), Wren’s, 69, 164
Staatsbibliothek of Berlin, 153
stability, financial, 358
stack architecture, computer, 76, 319, 322
staff authority, 316
stakeholder, 47, 73
standard industry, 59, 161
industry software development, 59
of living, 90
quality, 111
standardization, 43, 68
statistical quality control, 111
stealth airplane, 232
stress analysis, 109
structural engineering, 204
style, 139, 153, 162, 205, 245, 248
style, corporate, 156
submarine, 78, 80, 232
subroutine, 149, 171, 172, 336
Sunni Bridge, Menn’s, 63
supercomputer, 121, 154, 158, 252, 315, 325, 357
supervisor, component of operating system, 335, 338
surrogate for cost, 121
Sweets File and Network, 224
synchronization of tasks, 80
synonym dictionary, 210
syntactic analysis, 210
synthesis rules, 161
system architect, 73, 94, 130, 131
generation process, 342
integration, 94
residence, operating, 337
Tacoma Narrows Bridge collapse (1940), 167
tailoring processes as necessary, 43
talent, 93, 238
tape, magnetic, 178, 335, 356
task, sequential execution of in scheduler, 339
Task Architect software, 195
taste and instinct, 70
taxonomy, 206
team design paradigm, 64, 71, 82, 114, 119, 148, 244, 320
design, real-world, 111
two-person, 81
Technical Rationality (Schein’s term), 31, 35, 244
technological sophistication, 66
technology, telecollaboration, 98
telecollaboration, 64, 79, 89, 380

Part II
telecommunication, 64, 91, 92, 93
telephone, for collaboration, 96
teleprocessing, 173, 316, 318, 332, 333, 341, 356, 359
terminal, 169, 171, 338, 341
test cases display, 225
testing
dynamic stress, 109
regression, 107
software, 55, 111
user, 107, 179
text, specifying, 212
third-generation computer, 333, 359
thought-stuff, 108, 207
thought-trail, branching, 224
time
compile, 339
design, plenty, 280, 293, 294, 307, 310, 318, 327, 344, 350, 351
development, 42
design, run, or execution time, 339
scheduling, in compile, schedule, execute sequence, 339
specification, 210
time-sharing, 115, 332, 338
toolsmith, 98
Toothpick (viewpoint specification device), 215
top-down design, 204
topology, configuration, 131
Tower of Babel, 163
Track Changes (feature of Microsoft Word), 96
tracking (of budgeted resource), 119, 120, 123
traffic pattern, 215, 298, 299, 306
trajectory of a design, 185
trans-Atlantic interaction, 92, 95
transcription scheme, 189, 196, 197
transistor-diode logic, 159
translation software
media, 336
format, 336
transparency
transparency controlled for layers, 221
property of a design, 144
“Tree and Leaf,” Tolkien, seeing tree
of decisions, 189
of decisions versus tree of designs, 193
of designs, 15, 189
representation, hierarchical, 186, 221
search, 34, 303
Triangle Universities
Computation Center
(TUCC), 355
two-dimensional access, 96
two-handed interface, 207, 211
two-person interaction, 82
two-person jobs, 81
U-2 (spy plane), 70
unanimous consent, 361, 362
unbundling of software and hardware pricing, 344
UNC Effective Virtual Environments Research Project (EVE), 297
Unisys plc, 66
UNIVAC I, 320
University of Michigan, 159, 248, 322
University of North Carolina at Chapel Hill (UNC-CH), 200, 297, 357
University of Pennsylvania, 157
University of Toronto, 207
University of Utah, 96
University of Virginia, 151
UNIX, 56, 164, 177, 232, 244
uno animal (with one mind), 81, 239
use case, 117, 135, 179, 205, 289, 295, 301, 310, 311
usefulness
in design case, 272, 293, 306, 324, 341, 351, 363
Virtuvious’s design criterion, 139
user
  analysis and profile, 178, 181
  association, 335, 337
  outside, 182
  representative, 176
  set, 116
  testing, 107
  and use model, 113, 134, 335
  user-designer link, 177
utility
  function, 10, 68
  software, 332, 333
value, added, 298
value/cost ratio, 44
venture, international, 91
verb, 171, 173
verb specification, 208
Verein Deutscher Ingenieure
  Standard VDI-2221, 30, 32
verification, design, 108, 109, 111, 181
version control, 80, 223
veto, in an organization, 234
video teleconferencing, 93, 96, 97
view
  2-D context, 220
  3-D, 221
  context, 221
detailed, 221
direction specification, 214
drawing, 220
exterior, 215, 222
interior, 213, 221
  of library of objects, 221
ocean, 260, 266, 273
  of specifications, 223
workbook, 223
View/360 beach house, 7, 15, 259
View-Graph slides, 85
viewing parameters, 213
virtual design studio, 64
virtual environment (VE), 23, 178, 180, 298, 305, 307, 310
virtual environment (VE) model, 78, 297
virtual memory, 157, 159, 322, 323, 325, 333, 341, 342
virtual team, 64
virtual worlds (networked), 101
VisiCalc spreadsheet, 142
visual representation of design, 78
  of model of design process, 52, 54
vital interest, 359, 363
vocabulary, common, 179
voice
  command, 208, 209
  recognition, 209, 212
voting, in an organization, 360
walkthrough, virtual environment, 23, 80, 109, 213
Waterfall Model
  of designing, 16, 30, 34, 41, 44, 52, 196
  Royce’s critique of, 31
weakness in OS/360 design and design process, 342
web of knowledge, 186
whirligig model of designing, 54
whiteboard, 33
why, 156, 185, 223, 253
wicked problem, 16
WIMP interface (Windows-Icons-Menus-Pointing), 154, 208, 210
windows, multiple concurrent, 220
Women’s Reserve Naval Service (WRENS) (UK), 145
workbook display, 223
workstation, house design, 219
yaw, 213, 222
You Are Here, 214
“you bet your company,” 316
ZEBRA computer, 150
zoo, computer, 348, 351
zoom viewing parameter, 221


