FROM THE AUTHOR OF *THE MYTHICAL MAN-MONTH*

# THE DESIGN OF DESIGN

## ESSAYS FROM A COMPUTER SCIENTIST

FREDERICK P. BROOKS, JR.

Photo credit: © Jerry Markatos

## ABOUT THE AUTHOR

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

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

*This page intentionally left blank*

# *The Design of Design*

## *Essays from a Computer Scientist*

**Frederick P. Brooks, Jr.**

*University of North Carolina at Chapel Hill*

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

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.



*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.
ISBN 978-0-201-36298-5 (pbk. : alk. paper) 1. Engineering design. 2. Software engineering. 3. Design—Case studies. I. Title.
TA174.B752 2010
620'.0042—dc22

2009045215



ISBN-13: 978-0-201-36298-5
ISBN-10: 0-201-36298-8

Text printed in the United States on recycled paper at Courier in Stoughton, Massachusetts.
First printing, March 2010

*To all who have shared in my design adventures:*

*Family,*

*Colleagues,*

*Friends, and*

*Construction professionals*

*This page intentionally left blank*

# *Contents*


Preface ..... ix

**I Models of Designing ..... 1**

Chapter 1 The Design Question ..... 3

Chapter 2 How Engineers Think of Design—
The Rational Model ..... 13

Chapter 3 What's Wrong with This Model? ..... 21

Chapter 4 Requirements, Sin, and Contracts ..... 39

Chapter 5 What Are Better Design Process Models? ..... 51

**II Collaboration and Telecollaboration ..... 61**

Chapter 6 Collaboration in Design ..... 63

Chapter 7 Telecollaboration ..... 89

**III Design Perspectives ..... 103**

Chapter 8 Rationalism versus Empiricism in Design ..... 105

Chapter 9 User Models—Better Wrong than Vague ..... 113

Chapter 10 Inches, Ounces, Bits, Dollars—
The Budgeted Resource ..... 119

Chapter 11 Constraints Are Friends ..... 127

Chapter 12 Esthetics and Style in Technical Design ..... 139

Chapter 13 Exemplars in Design ..... 153

Chapter 14 How Expert Designers Go Wrong ..... 167

Chapter 15 The Divorce of Design ..... 175

Chapter 16 Representing Designs' Trajectories
and Rationales ..... 185

**IV A Computer Scientist's Dream System for Designing Houses . . . . . 201**

Chapter 17 A Computer Scientist's Dream System for Designing Houses—Mind to Machine . . . . . 203

Chapter 18 A Computer Scientist's Dream System for Designing Houses—Machine to Mind . . . . . 219

**V Great Designers . . . . . 229**

Chapter 19 Great Designs Come from Great Designers . . . . . 231

Chapter 20 Where Do Great Designers Come From? . . . . . 243

**VI Trips through Design Spaces: Case Studies . . . . . 257**

Chapter 21 Case Study: Beach House "View/360" . . . . . 259

Chapter 22 Case Study: House Wing Addition . . . . . 279

Chapter 23 Case Study: Kitchen Remodeling . . . . . 297

Chapter 24 Case Study: System/360 Architecture . . . . . 313

Chapter 25 Case Study: IBM Operating System/360 . . . . . 331

Chapter 26 Case Study: Book Design of *Computer Architecture: Concepts and Evolution* . . . . . 347

Chapter 27 Case Study: A Joint Computer Center Organization: Triangle Universities Computation Center . . . . . 355

Chapter 28 Recommended Reading . . . . . 367

Acknowledgments . . . . . 371

Bibliography . . . . . 375

People Index . . . . . 393

Subject Index . . . . . 401

# *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.[1]

## 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.[2] After all, "The cattle on a thousand hills are mine. … If I were hungry, I would not tell *you*."[3] 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.[4]

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.[5] A discipline of design studies arose, with dedicated conferences, journals, 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 understanding 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[6]

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.[7] I have long been interested in the design process; my 1956 dissertation was "The analytic design of automatic data processing systems."[8] 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.[9]

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."
2. Tolkien [1964], "On Fairy Stories," in *Tree and Leaf*, 54.
3. Psalm 50:10,12. Emphasis added.
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.
7. A table of the specific design experiences is included in the appendix materials on the Web site: http://www.cs.unc.edu/~brooks/DesignofDesign.
8. Brooks [1956], "The analytic design of automatic data processing systems," PhD dissertation, Harvard University.
9. I thus do not contribute to the design methodologists' goal as stated in http://en.wikipedia.org/wiki/Design_methods (accessed on January 5, 2010):

   > *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."
10. Glegg [1969], *The Design of Design*.

*This page intentionally left blank*

# I

# Models of Designing

# 1
# *The Design Question*

*[New ideas would come about] by a connexion and transferring of the observations of one Arte, to the uses of another, when the experience of several misteries shall fall under consideration of one mans minde.*

SIR FRANCIS BACON [1605], *THE TWO BOOKS OF THE PROFICIENCE AND ADVANCEMENT OF LEARNING*, BOOK 2, 10

*Few engineers and composers ... can carry on a mutually rewarding conversation about the content of the other's professional work. What I am suggesting is that they can carry on such a conversation about design, . . . [and then] begin to share their experiences of the creative professional design process.*

HERBERT SIMON [1969], *THE SCIENCES OF THE ARTIFICIAL*, 82

**Spiral staircase**
Corbis

## Is Bacon Right?

Sir Francis Bacon's hypothesis is our challenge. Are there invariant properties of the design process itself, properties that hold across a wide range of media of design? If so, it seems likely that designers in one medium would collectively grasp some of these principles more clearly than other designers, through struggles that are peculiarly difficult for that medium. Moreover, some media, such as buildings, have longer histories of both design and meta-design—"the design of design." If all this is true—and if Bacon's conclusion is true—designers in different media can expect to learn new things about their own several crafts by comparing their experiences and insights.

## What Is Design?

The *Oxford English Dictionary* defines the verb *design* as

> *To form a plan or scheme of, to arrange or conceive in the mind . . . for later execution.*

The essentials of this definition are *plan*, *in the mind*, and *later execution*. Thus, a design (noun) is a created object, preliminary to and related to the thing being designed, but distinct from it. Dorothy Sayers, the English writer and dramatist, in her magnificent and thought-provoking book *The Mind of the Maker*, breaks the creative process out further into three distinct aspects. She calls them the Idea, the Energy (or Implementation), and the Interaction,[1] that is,

1. The formulation of the conceptual constructs
2. Implementation in real media
3. Interactivity with users in real uses

A book, in this conception, or a computer, or a program, comes into existence first as an ideal construct, built outside time

and space, but complete in essence in the mind of the author. It is implemeted in time and space, by pen, ink, and paper; or by silicon and metal. The creation is complete when someone reads the book, uses the computer, or runs the program, thereby interacting with the mind of the maker.

In an earlier paper, I divided the tasks in building software into *essence* and *accident*.[2] (This Aristotelian language is not to denigrate the accidental parts of software construction. In modern language the terms would more understandably be *essential* and *incidental.*) The part of software building I called *essence* is the mental crafting of the conceptual construct; the part I called *accident* is its implementation process. *Interaction*, Sayers's third step, occurs when the software is used.

The design is thus the mental formulation, which Sayers calls "the Idea," and it can be complete before any realization is begun. Mozart's response to his father's inquiry about an opera due to the duke in three weeks both stuns us and clarifies the concept:

> *Everything has been composed, just not yet written down.*
>
> LETTER TO LEOPOLD MOZART [1780]

For most human makers of things, the incompletenesses and inconsistencies of our ideas become clear only during implementation. Thus it is that writing, experimentation, "working out," are essential disciplines for the theoretician.

The phases of Idea, Implementation, and Interaction operate recursively. Implementation creates a space in which another cycle of design must be done. Thus Mozart Implemented his opera Idea with pen on paper. The conductor, Interacting with Mozart's creation, conceived an Idea of an interpretation, Implemented it with orchestra and singers, and the Interaction with the audience completed the process.

A *design* is a created object; associated with it is a *design process*, which I shall call *design*, without any article. Then there is the verb *to design*. The three senses are intimately related; I believe context will resolve ambiguity.

## What's Real? The Design Concept

*If a number of individuals have a common name, we assume them to have also a corresponding idea or form:—do you understand me?*

*I do.*

*Let us take any common instance; there are beds and tables in the world—plenty of them, are there not?*

*Yes.*

*But there are only two ideas or forms of them—one the idea of a bed, the other of a table.*

*True.*

*And the maker of either of them makes a bed or he makes a table for our use, in accordance with the idea.*

PLATO, *THE REPUBLIC* [360 BC], BOOK X

At the 2008 Design Thinking Research Symposium 7, each of the speakers presented analyses of the same four design team meetings.[3] Videos and transcripts had been distributed well in advance.

Rachael Luck of the University of Reading identified in the architectural conversations an entity that none of us had remarked but all then recognized: *the Design Concept*.[4]

Sure enough, both architect and client referred from time to time to this shared invisible entity. Speakers usually gestured vaguely toward the drawings when they spoke thus, but it was clear they were not referring to the drawings or any particular thing therein. Always, the concern was for the conceptual integrity of the developing design.

Luck's insight made the *Design Concept* a thing in its own right. This resonated strongly with my experience. When the IBM System/360 "mainframe" computer family's single architecture

was being developed (1961–1963), such an entity was always present in the architecture group, although never named. Exploiting Gerry Blaauw's brilliant insight, we had explicitly separated the System/360 design activities into *architecture*, *implementation*, and *realization*.[5] The basic concept was a computer family with one face to the programmer—the architecture—and multiple concurrent implementations at various positions on the performance and price curves (Chapter 24).

The very simultaneity of multiple implementations, with their several engineering-manager champions, drove the common architecture toward generality and cleanliness and insulated it from small cost-saving compromises. These forces, however, were merely shields for the instincts and desires of the architects, who each wanted to make a clean machine.[6]

As the architecture design progressed, I observed what at first seemed quite strange. For the architecture team, the *real* System/360 was the Design Concept itself, a Platonic ideal computer. Those physical and electrical Model 50, Model 60, Model 70, and Model 90 things under construction out on the engineering floors were but Plato's shadows of the real System/360. The real System/360's most complete and faithful embodiment was not in silicon, copper, and steel, but in the prose and diagrams of *IBM System/360 Principles of Operation*, the programmer's machine-language manual.[7]

I had a similar experience with the View/360 beach house (Chapter 21). Its Design Concept came to be real long before any construction began. It persisted through many versions of drawings and cardboard models.

Interestingly enough, I never felt such a Design Concept entity of the Operating System/360 software family. Perhaps its architects did; perhaps I did not have an intimate enough acquaintance with its conceptual bones. Perhaps the Design Concept didn't emerge for me because OS/360 was in fact a fusion of four somewhat separate parts: a supervisor, a scheduler, an I/O control system, and a large package of compilers and utilities (Chapter 25).

### *What's the Value?*

Is there positive value to recognizing an invisible Design Concept as a real entity in design conversations? I think so.

First, great designs have conceptual integrity—unity, economy, clarity. They not only work, they *delight*, as Vitruvius first articulated.[8] We use terms such as *elegant*, *clean*, *beautiful* to talk about bridges, sonatas, circuits, bicycles, computers, and iPhones. Recognizing the Design Concept as an entity helps us to seek its integrity in our own solo designs, to work together for it in team designs, and to teach it to our youth.

Second, talking frequently about the Design Concept as such vastly aids communication within a design team. Unity of concept is the goal; it is achieved only by much conversation.

> *The conversation is much more direct if the Design Concept per se, rather than derivative representations or partial details, is the focus.*

Thus, moviemakers use storyboards to keep their design conversations focused on the Design Concept, rather than on implementation details.

Detailing will of course surface conflicting versions of the Concept and force resolution. For instance, System/360 architecture needed a decimal datatype, as a bridging aid for thousands of existing users of IBM's decimal machines. Our developing architecture already had several datatypes, including a 32-bit fixed-point twos-complement integer and a variable-length character string.

The decimal datatype could be made similar to either one. Which choice better fit the Design Concept of System/360? Strong arguments were made each way; the strength of each depends on one's version of the Design Concept. Some of the architects had implicit Design Concepts reflecting earlier scientific computers; others' implicit concepts reflected earlier business computers. System/360 was explicitly intended to serve both kinds of applications well.

We chose to model the decimal datatype after the character-string one, the one more familiar to the largest particular user

community of the decimal datatype, IBM 1401 users. I would decide that way again.

## Thinking about the Design Process

Thinking about designs has a long history, going back at least to Vitruvius (died ca. 15 BC). His *De Architectura* is the important book about design from the Classical period. Major milestones are the *Notebooks* of Leonardo da Vinci (1452–1529) and the *Four Books of Architecture* by Andrea Palladio (1508–1580).

Thinking about the design process itself is much more recent. Pahl and Beitz trace German thought from Redtenbacher in 1852, stimulated by the rise of mechanization.[9] For me, major milestones have been Christopher Alexander's *Notes on the Synthesis of Form* (1962), Herbert Simon's *The Sciences of the Artificial* (1969), Pahl and Beitz's *Konstructionslehre* (1977), and the founding of the Design Research Society and the starting of the journal *Design Studies* (1979).

Margolin and Buchanan [1995] is an edited collection of some 23 essays from the journal *Design Issues*, primarily design criticism and theory, with "occasional ventures into philosophical issues that bear on the understanding of design" (p. xi).

My *The Mythical Man-Month* [1975, 1995] reflects on the design process for IBM's Operating System/360, later evolved to MVS and beyond. It emphasizes the human, the team, the management aspects of that design and development project. Of particular relevance to the present work are Chapters 4–6 of those essays, which address how to achieve conceptual integrity in a team design.

Blaauw and Brooks [1997], *Computer Architecture: Concepts and Evolution*, includes extensive discussion of the design of the IBM System/360 (and System/370–390–z) architecture and the relationships of and rationales for dozens of design decisions. It doesn't treat the design process or human aspects of designing at all. But Section 1.4, which discusses criteria for goodness in computer architectural design, is indeed of particular relevance for this work.

## Kinds of Design

### *System Design versus Artistic Design*

This book is about the design of complex systems, and the viewpoint is that of the engineer, an engineer focused on utility and effectiveness but also on efficiency and elegance.

This contrasts with much of the design done by artists and writers, whose emphasis is on delight and the conveying of meaning. Architects and industrial designers, of course, fall into both camps.

### *Routine, Adaptive, Original Design*

We often think of bridge design as one of the high arts of engineering, one where breakthroughs in concept or of technology have dramatic and highly visible cost, function, and esthetic consequences.

Well, a high fraction of all highway bridges are short, so cranking out a design for a 50-foot concrete bridge is a routine and automatable process. For short bridges, civil engineers know, and long ago codified into handbooks, the design decision tree, the constraints, and the desiderata. The same situation prevails for the design of compilers for established languages on new platforms. There are many areas of routine, automatable design.

The emphasis in this book is on original design, as opposed to the routine redesign of object after object with changed parameters, or even adaptive design, which is essentially the modification of a preceding design or object to serve new purposes.

## Notes and References

1. Sayers [1941], *The Mind of the Maker*.
2. Brooks [1986], "No silver bullet."
3. McDonnell [2008], *About Designing*. This book is the edited Papers from the Design Thinking Research Symposium (DTRS7).
4. Luck [2009], "Does this compromise your design?" reprinted in McDonnell [2008], *About Designing*.

5. Blaauw and Brooks [1964], "Outline of the logical structure of System/360." Blaauw further divides Sayers's "Energy" into Implementation and Realization, a distinction I find immensely useful.
6. Janlert [1997], "The character of things," argues that designed things have character and discusses how one designs that character.
7. IBM Corp. [1964], *IBM System/360 Principles of Operation*.
8. Vitruvius [22 BC], *De Architectura.*
9. Pahl and Beitz [1984], *Engineering Design*.

- **Goal**
- **Desiderata**
- **Utility function**
- **Constraints, especially budget (perhaps not $ cost)**
- **Design tree of decisions**

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

# 2

# *How Engineers Think of Design—The Rational Model*

*… [F]or the theory of design is that general theory of search … through large combinatorial spaces.*

HERBERT SIMON [1969], *THE SCIENCES OF THE ARTIFICIAL,* 54

**A Rational Model of the design process**

## The Model

Engineers seem to have a clear, if usually implicit, model of the process of design. It is an orderly model of an orderly process as the engineer conceives it. I shall illustrate with an example of a beach house design (sketched in Chapter 21).

**Goal.** First one starts with a primary goal, or objective: "One wants to build a beach house to take advantage of wind and wave at an oceanfront lot."

**Desiderata.** Associated with the primary goal are a host of desiderata, or secondary objectives: "The beach house should be reinforced to withstand hurricane-force winds; it should sleep and seat at table at least 14 people; it should exploit the stunning views;" and so on.

**Utility Function.** One wants to optimize the design according to some utility or goodness function that weights the several desiderata as to their importance. So far as I can tell, most designers imagine the terms themselves to be linearly summed, but conceive of each goodness variable individually as not linear, but rather as curved asymptotically to saturation. For example, more window area is a desideratum, something desired in house design. But the utility added by each extra square foot of window diminishes. The same is true of electrical outlets. The utility of the windows and that of the outlets, however, seem simply to sum.

**Constraints.** Every design, and every optimization, is subject to constraints. Some of these are binary, either satisfied or not—"The house must be set back at least 10 feet from the lot's side lines." Others are more elastic, with steeply rising penalties as one approaches a limit, such as schedule constraints—one fiercely wants to have the beach house ready when warm weather comes.

Some constraints are simple, such as setback limits. Others blithely conceal terrifying complexity—"The house must satisfy all the building codes."

**Resource Allocations, Budgets, and Crucial Budgets.** Many constraints take the form of a fixed resource to be allocated

among design elements. The most common is a total cost budget. But this is by no means the only such constraint, nor is it necessarily the one that most controls the designer's attention in a particular project. In the beach house floor plan, for example, the controlling commodity to be rationed was the feet (even inches) of ocean frontage. In the design of a computer architecture, the critical budget may be the bits in a control register or an instruction format, or the uses of the total memory bandwidth. When people were solving Year 2000 problems in software, working days on the schedule were the crucial allocable resource.

**Design Trees.** Now, so the Rational Model goes, the designer makes a design decision. Then, within the design space narrowed by that decision, he makes another.[1] At each node he could have taken one or more other paths, so one can think of the process of design as the systematic exploration of a tree-structured design space.

In this model, design is conceptually (at least) very simple. One searches the tree-structured design space, testing each option against the constraints for feasibility and choosing so as to optimize the utility function. The search algorithms are well known and can be cleanly described.

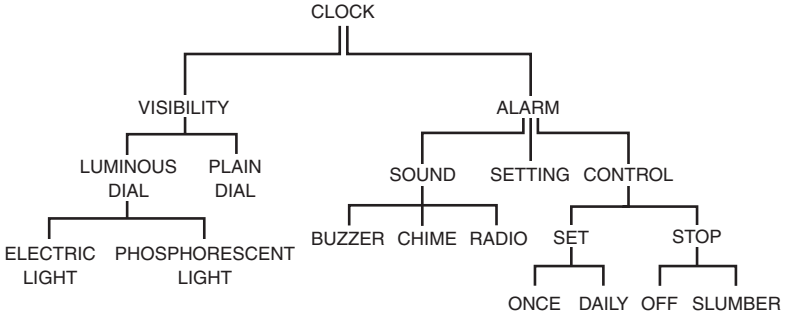



**Figure 2-1 Portion of a design tree for an alarm clock**
From Blaauw and Brooks [1997], *Computer Architecture*, Figures 1-12, 1-14.

That cleanliness holds only for an exhaustive search of all paths, seeking a truly optimal solution. Designers commonly satisfice by searching only until a "good enough" solution is found.[2] Many engineers seem to approximate some sort of depth-first search strategy, choosing at each node the most promising or attractive option and exploring it to the end. At dead ends, one backtracks and takes another path. Hunches, experience, consistency, and esthetic taste guide each option selection.[3]

## Whence Formulations of This Model?

The notion that the design process should be modeled as a systematic step-by-step process seems to have first developed in the German mechanical engineering community. Pahl and Beitz present the most widely used exposition in seven successive editions of their great work.[4] They observe the practice, but not the explicit statement, of systematic search of design alternatives in the *Notebooks* of Leonardo da Vinci (1452–1519).

Herbert Simon independently argues for design as a search process in *The Sciences of the Artificial* [1969, 1981, 1996]. His model and his discussion of it are much more sophisticated than those here. Simon, optimistic that the design process was a fit target for artificial intelligence (once adequate processing power became available), was motivated to lay out a strictly rational model of design precisely *because* such a model was a necessary precursor to automating design. His model remains influential even if today we recognize the "wicked problem"[5] of original design as one of the least promising candidates for AI.

In software engineering, Winston Royce, appalled at the failures of the "just write it" approach for large software systems, independently introduced a seven-step Waterfall Model to bring order to the process, as shown in the next chapter's frontispiece. In fact, Royce introduced his waterfall as a straw man that he then argued against, but many people have cited and followed the straw man rather than his more sophisticated models. I made that mistake myself in my younger days, and publicly repented of it later.[6] Even if ironically, Royce's seven-step model must be

considered one of the foundational statements of the Rational Model of design.

As Royce emphasizes, his seven steps are distinctly different from one another and must be planned and staffed differently. Iteration is provided for but carefully limited in scope:

> *The ordering of steps is based on the following concept: that as each step progresses and the design is further detailed, there is an iteration with the [immediately] preceding and succeeding steps but rarely with the more remote steps in the sequence. … What we have is an effective fallback position that tends to maximize the extent of early work that is salvageable and preserved.*[7]

The notion that a design space can be formulated as a tree is implied by Simon. It is described and illustrated by Gerry Blaauw and me in our *Computer Architecture*.[8] There we arrange the design choices for processor architecture strictly hierarchically in a giant tree, represented by 83 linked subtrees. A simple example of the design tree for the alarm of an alarm clock is shown in Figure 2-1. In it one observes two types of branches indicated by open and closed roots. The first, as shown for "Alarm," shows a subdivision; each branch is a different design attribute that must be specified. This is called an *attribute branch*. The *alternative branch*, shown for "Sound," enumerates alternatives of which one must be chosen.

## What's Right with This Model?

Any systematization of the design process is a great step forward compared to "Let's just start coding, or building." It provides clear steps for planning a design project. It furnishes clearly definable milestones for planning a schedule and for judging progress. It suggests project organization and staffing. It helps communication within the design team, giving everyone a single vocabulary for the activities. It wonderfully helps communication between the team and its manager, and between the manager and other stakeholders. It is readily teachable to novices. It tells the novice facing his first design assignment where to begin.

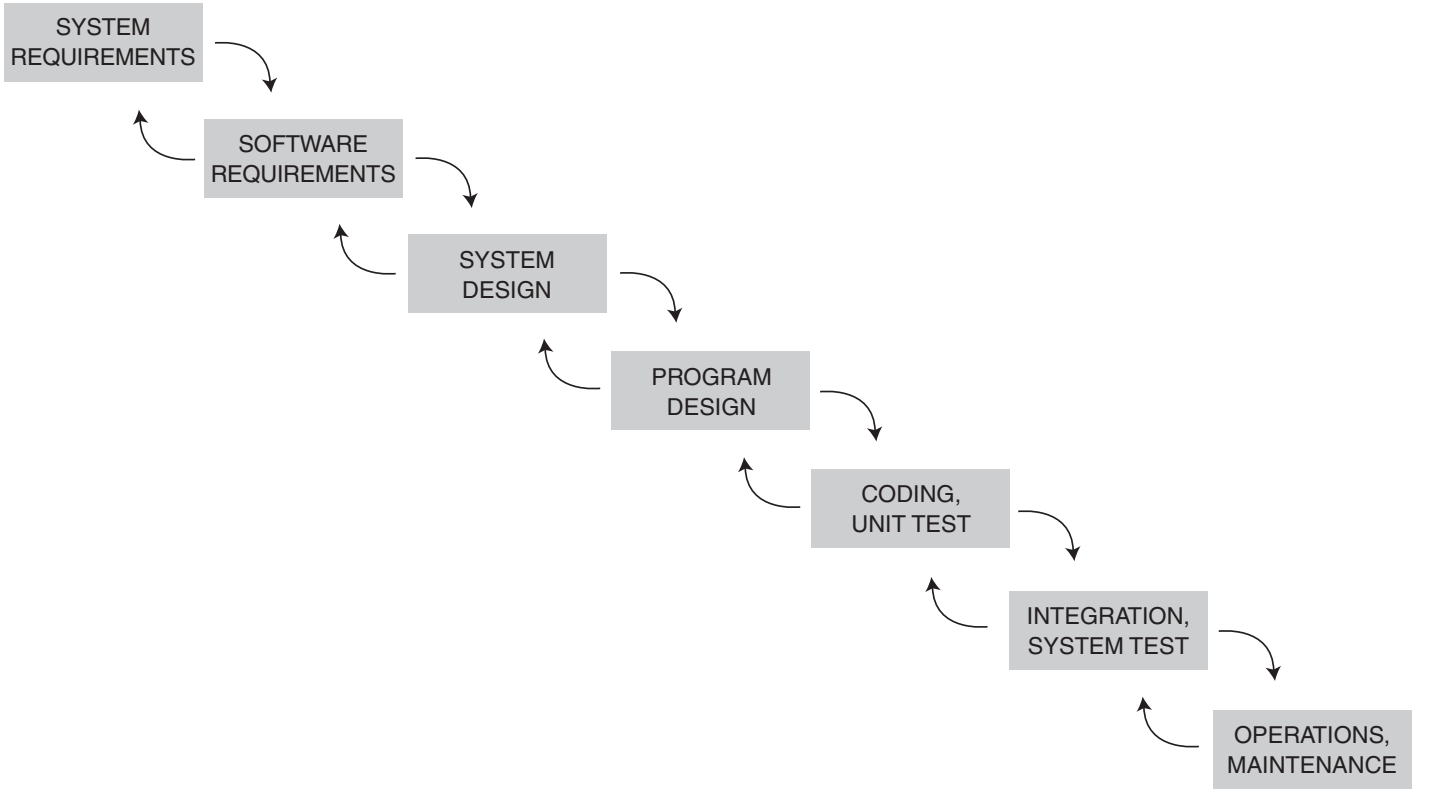
The Rational Model in particular brings yet more advantages. The early explicit statement of goals, secondary desiderata, and constraints helps a team avoid wandering, and it breeds team unification on purposes. Planning the whole design process before starting coding or formal drawings avoids many troubles and much wasted effort. Casting the process as a systematic search of a design space broadens the horizon of the individual designers and lifts their eyes far beyond their previous personal experiences.

But the Rational Model is much too simplistic, even in Simon's richly developed version. Hence we must examine its faults.

## Notes and References

1. Following Simon [1981], *The Sciences of the Artificial*, throughout this book I use *man* as a general noun, encompassing both genders, and *he*, *him*, and *his* as androgynous pronouns. I find it more gracious to continue the long tradition of including women and men equally in these general pronouns than to adopt more awkward, hence distracting, constructions.
2. To *satisfice* is to make good enough without necessarily optimizing (Simon [1969], *The Sciences of the Artificial*, 64).
3. But see Akin [2008], "Variants and invariants of design cognition," who finds evidence from the DTRS7 protocols that building architects tend to search laterally among several alternatives at every level, whereas engineering designers emphasize depth-first search based on an initial solution proposal.
4. Pahl and Beitz [1984ff.], *Engineering Design*.
5. Rittel and Webber [1973], "Dilemmas in a general theory of planning," define this term formally. It is well discussed in http://en.wikipedia.org/wiki/Wicked_problem.
6. Brooks [1995], *The Mythical Man-Month*, 265.
7. Royce [1970], *"Managing the development of large software systems,"* 329.
8. Blaauw and Brooks [1997], *Computer Architecture*.

*This page intentionally left blank*


SYSTEM REQUIREMENTS
SOFTWARE REQUIREMENTS
SYSTEM DESIGN
PROGRAM DESIGN
CODING, UNIT TEST
INTEGRATION, SYSTEM TEST
OPERATIONS, MAINTENANCE

# 3

# *What's Wrong with This Model?*

*Sometimes the problem is to discover what the problem is.*

GORDON GLEGG [1969], *THE DESIGN OF DESIGN*

*A designer makes things. … Typically his making process is complex. There are more variables—kinds of possible moves, norms, and interrelationships of these—than can be represented in a finite model.*

DONALD A. SCHÖN [1984],
*THE REFLECTIVE PRACTITIONER*

**Software's Waterfall Model**

After Royce [1970], "Managing the development of large software systems," and Boehm [1988], "A spiral model of software development and enhancement." Royce © 1970 IEEE. Boehm © 1988 IEEE.

In fact, every designer will recognize the Rational Model as only an ideal. It somehow describes how we think the design process ought to work, but not how it works in real life.

Indeed, not every engineer will even admit to harboring so naive and idealistic a model in his heart. But I think most of us really do; I did for quite a long time. Therefore, let us take a hard critical look at the Rational Model of design, to identify precisely where it most departs from reality.

## We Don't Really Know the Goal When We Start

The most serious model shortcoming is that the designer often has a vague, incompletely specified goal, or primary objective. In such cases:

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

As a student I spent one summer working at a large missile company, where I was once set to work designing and building a little database system for keeping track of the 10,000 drawings for a radar subsystem, and the updating status of each.

After a couple of weeks, I had a working version. I proudly presented a sample output report to my client.

"That's fine—it is what I asked for—but could you change it so that . . . ?"

Each morning for the next few weeks, I presented my client with the output report, revised yet again to accommodate the previous day's request. Each morning he studied the product report and asked for yet another system revision, using the same polite mantra.

It was a simple system (implemented on punched-card machines), and the revisions were conceptually simple. The most comprehensive change was to list the drawings sorted by, and indented to show, goes-into level, where level was represented by a single 0–9 digit in the card. Other refinements included multilevel subtotals, with exceptions of course, and the automatic marking of various noteworthy values with asterisks.

For a while, this frustrated me sorely: "Why can't he make up his mind as to what he wants? Why can't he tell me all at once, instead of one bit a day?"

Then, slowly, I came to realize that the most useful service I was performing for my client was helping him *decide* what he really wanted.

Well, today the software engineering discipline is much more sophisticated. We recognize that rapid prototyping is an essential tool for formulating precise requirements. Not only is the design process iterative; the design-goal-setting process is itself iterative.

This sophistication in software engineering does not forestall, or noticeably reduce, the numerous references in the literature to the "product requirements" as a normal given for a design process. But I will argue that knowing complete product requirements up front is a quite rare exception, not the norm:

> *A chief service of a designer is helping clients discover what they want designed.*

In software engineering, at least, the concept of rapid prototyping has a name and a recognized value, whereas it does not always have the same status in computer design and in building architecture. Nevertheless, I see the same goal iteration happening in these design fields. Increasingly, designers build simulators for computers and virtual-environment walk-throughs for buildings as rapid prototypes to drive goal convergence. Goal iteration must be considered an inherent part of the design process.

## We Usually Don't Know the Design Tree—We Discover It as We Go

For the original design of complex structures, such as computers, operating systems, spacecraft, and buildings, each major design effort has enough novelty in the

- Goal
- Desiderata, and the utility function
- Constraints
- Available fabrication technologies

that the designer can rarely sit down and a priori map out the design tree.

Moreover, in high-technology design, few designers can know enough to draw the basic decision tree for their domains. Design projects often last two years or more. And designers get promoted out of design. Consequently, few designers will work in any depth on as many as 100 projects over a working life. This means the individual designer has not begun to explore all the branches of the basic design tree for his discipline. For it is characteristic of engineering designers, as opposed to scientists, that they rarely explore alternatives that are not clearly on the way to a solution.[1]

Instead, designers discover the design tree as they work—making a decision, and then seeing the alternatives it opens and closes for the next consequent design decision.

## The Nodes Are Really Not Design Decisions, but Tentative Complete Designs

In fact, the very decision tree is itself only a simplistic model of the tree-search process. As Figure 2-1 illustrates, there are parallel attribute branches, as well as alternative branches. The choices in one branch are linked to those in others—by exclusion, affinity, or trade-off. Our massive design tree in *Computer Architecture* is much too simple; the entire "Computer Zoo" in that work is necessary to elucidate the decision linkings.[2]

This means that at each node of a design tree, one faces not a simple alternative choice among options for one design *decision*, but an alternative choice among multiple tentative complete *designs*.

Moreover, the ordering of decisions laying out a design tree matters greatly, as Parnas expounds in his classic paper "Designing software for ease of extension and contraction."[3]

The explosive combinatorics of these complications to the tree model boggle the mind. (This situation is like that of move trees in chess.) This difficulty is explored further in Chapter 16.

## The Goodness Function Cannot Be Evaluated Incrementally

The Rational Model assumes that design involves a search of the design tree, and that at every node, one can evaluate the goodness function of the several downward branches.

In fact, one cannot in general do this without exploring all the downward branches to all their leaves, for many goodness measures (for example, performance, cost) will depend heavily on the subsequent design detailing. So although the goodness evaluation is possible in principle, one encounters here again the combinatorial explosion of alternatives in practice.

So what is a designer to do? Estimate, of course, either formally or informally. One *must* trim the design tree as one goes down.

**Experience.** Many aids help intuition in this process. One is experience, both direct and surrogate: "The designers of OS/360 exposed detailed formats of system-wide-shared Control Blocks in Operating System/360, and it proved a maintenance nightmare. We will encapsulate them as objects." "The Burroughs B5000 family long ago explored the descriptor-based computer architecture. The performance hit was inherently too great, so we won't explore that subtree." Of course, the technological tradeoffs are no longer the same, but the experience lesson is illuminating anyway. The most potent reason to study design history is to learn what *doesn't work*, and *why*.

**Simple Estimators.** Designers routinely use simple estimators early in the design tree exploration. A building architect, given a budget goal, applies a rough square-foot cost estimate, derives a square-foot goal, and uses that for subsequent pruning of the design tree. Computer architects use instruction mixes to do rough-cut early estimates of computer performance.

A danger, of course, is that the rough estimator may lop off design branches that are in fact feasible but appear infeasible because of the very approximation involved in the estimator. I have watched an architect quote high costs for pushing out a wall under an already specified roof structure, based purely on a routine square-foot estimator. In fact, most of the cost of the added space was in the roof, and that was already committed, so the marginal cost was very low.

> *One can often get something for nothing, if one has previously bought nothing for something.*

## The Desiderata and Their Weightings Keep Changing

Donald Schön, the late MIT professor of urban studies and education, and a design theorist, said:

> *[As the designer] shapes the situation in accordance with his initial presentation of it, the situation "talks back" and he responds to the situation's back-talk.*
>
> *In a good process of design, this conversation with the situation is reflexive. In answer to the situation's back-talk, the designer reflects-in-action on the construction of the problem, the strategies of action, or the model of the phenomena, which have been implicit in his moves.*[4]

In short, as one ponders the trade-offs, there comes a new understanding of the whole design problem as an intricately interlocked interplay of factors. With it comes a change in the weightings of the desiderata. The same thing happens as the client, if there is one, grows his understanding of what he will get and develops his detailed vision of how he will use it.

In our house-remodeling design, for example (Chapter 22), a simple question, overlooked in the original program, arose well along in design, as my wife and I applied use scenarios to preliminary designs: "Where will guests at meetings put their coats?" This seemingly low-weight desideratum in fact tipped the big scales, and occasioned moving the Master Bedroom from one end of the house to the other.

Moreover, for designs that must be separately fabricated, such as buildings and computers, the designer learns from the builders a growing understanding of the interactions between design and fabrication. So many desiderata and constraints shift and refine. The fabrication technology may evolve as well, an especially common occurrence during computer design.

Since many desiderata (such as speed) are weighted on a value/cost ratio, yet another phenomenon occurs. As design proceeds, one finds opportunities to add some particular goodness at a very low marginal cost. So something that had not entered the original desiderata list at all comes in, and it often takes on a value that may demand preserving in later design changes.

Only after UNC's Sitterson Hall was designed, built, and in use, for example, did the Computer Science Department, as user, learn that the suite of spaces consisting of the Lower Lobby, Upper Lobby, Faculty Conference Room, Lecture Halls, and Vestibules combined beautifully into a facility well suited for hosting conferences of up to 125 people, with minimal impact on the work in the rest of the building. This was serendipitous—no such function was contemplated in the original architectural program. Yet it is a high-value feature: any future revision of Sitterson would surely aim to preserve this capability.

## The Constraints Keep Changing

Even if the goal were fixed and known, all the desiderata enumerated, the design tree known precisely, and the goodness function precisely defined, design would still be iterative, because the constraints keep changing.

Often the environment changes—the city council passes new shadow-casting setback requirements; the electrical code has an annual updating; a microchip one planned to use is withdrawn by the vendor. The world keeps changing around us, even while we design.

The constraints also change due to discovery during the design process, or during the fabrication—the builders hit solid rock; analysis shows that chip cooling has newly become a constraint.

Not all constraint changes are increases. Often constraints go away. When this is fortuitous instead of intentional, the skillful designer recognizes the new opportunity and, with his flexible design, leaps to exploit it.

Alas! Not all designs are flexible. More commonly, when we are deep into a design process, we do not recognize that a constraint has disappeared, nor do we remember which design alternatives it formerly foreclosed.

It is important to list the known constraints explicitly at the start of the design process, as part of what architects call the *design program*. The design program is a document, prepared with the client, that sets forth the goal, the desiderata, the constraints. An example is given in this book's Web site. The design program is *not* the same thing as a formal requirements statement, which usually has contractual force in defining acceptability of a design.

The explicit listing of constraints smokes them out early, avoiding unpleasant surprises. It also impresses them on the designer's mind, radically improving the chances that he will recognize when one goes away.

All of us have designed around constraints, a process that calls forth much invention and exploration of unconventional corners of the design space. This is part of the fun of design, and a big part of the challenge.

**Changing Constraints Outside the Design Space.** Sometimes, however, a design breakthrough is achieved by stepping completely outside the design space, and working there to remove the design constraint. In designing the house wing (Chapter 22), I wrestled a long time, unsuccessfully, with a shadow-casting setback requirement constraint and the Music Room's desiderata (hold two grand pianos, an organ, and a square space for a string octet plus a 1-foot teaching margin). Figure 3-1 shows one iteration of the design, and the constraints.

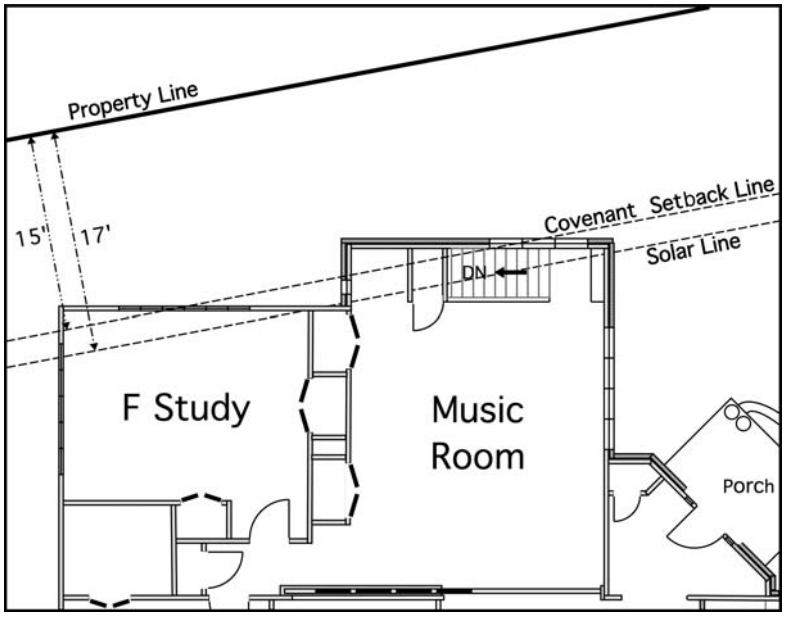



**Figure 3-1 Design up against constraints**