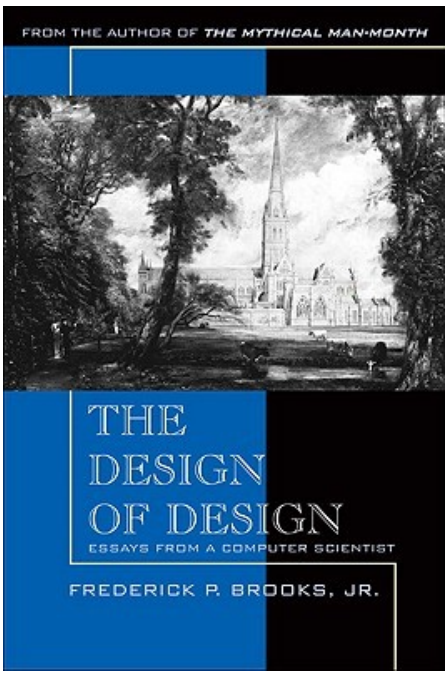

The Design of Design: Essays from a Computer Scientist

Frederick P. Brooks Jr.

Download now

Read Online

The Design of Design: Essays from a Computer Scientist

Frederick P. Brooks Jr.

The Design of Design: Essays from a Computer Scientist Frederick P. Brooks Jr.
Making Sense of Design Effective design is at the heart of everything from software development to engineering to architecture. But what do we really know about the design process? What leads to effective, elegant designs? The Design of Design addresses these questions. These new essays by Fred Brooks contain extraordinary insights for designers in every discipline. Brooks pinpoints constants inherent in all design projects and uncovers processes and patterns likely to lead to excellence. Drawing on conversations with dozens of exceptional designers, as well as his own experiences in several design domains, Brooks observes that bold design decisions lead to better outcomes. The author tracks the evolution of the design process, treats collaborative and distributed design, and illuminates what makes a truly great designer. He examines the nuts and bolts of design processes, including budget constraints of many kinds, aesthetics, design empiricism, and tools, and grounds this discussion in his own real-world examples--case studies ranging from home construction to IBM's Operating System/360. Throughout, Brooks reveals keys to success that every designer, design project manager, and design researcher should know.

The Design of Design: Essays from a Computer Scientist Details

Date	:	Published March 26th 2010 by Addison-Wesley Professional (first published March 10th 2010)
ISBN	:	9780201362985
Author	:	Frederick P. Brooks Jr.
Format	:	Paperback 421 pages
Genre	:	Design, Computer Science, Programming, Science, Nonfiction, Software, Technology

Download The Design of Design: Essays from a Computer Scientist ...pdf

Read Online The Design of Design: Essays from a Computer Scientis ...pdf

Download and Read Free Online The Design of Design: Essays from a Computer Scientist Frederick P. Brooks Jr.

From Reader Review The Design of Design: Essays from a Computer Scientist for online ebook

Margaret Heller says

Thinking about how design works and models for design is important, even if the models are imperfect. Brooks describes the importance of models, flaws in the commonly used models, and some ways to think about better models and the types of practical problems that arise in design. I think given the structure it's harder to pick out a coherent argument, but there are some important ideas in here. I admit I totally skimmed all the case studies, since I didn't think they added all that much other than generally to see the practical considerations in epistemology of design,

---

Ushan says

In 1961-1965, in his early 30s, Frederick Brooks was the project manager of the IBM System/360 and OS/360 project, one of the most successful engineering projects in history. Called "I.B.M.'s $5,000,000,000 gamble", this project succeeded enormously, and transformed not only the company and the industry, but the entire civilization: the computer architecture and it successors, and the operating system and its successors have been the de facto standard for mainframe computers ever since (of course, mainframe computers have been becoming ever less important in the world of computers, though they are still a multibillion-dollar business). Though Brooks has done other things later in life (working on interactive computer graphics, serving on the Defense Science Board, writing a textbook on computer architecture together with one of the architects of System/360, writing the famous book on software engineering The Mythical Man-Month), working on System/360 and OS/360 was the experience of his lifetime. He goes deeply into the design decisions made in these two projects. Having 8 bits instead of 6 bits in a byte, which allowed lowercase letters in software applications, was a terrific idea. JCL, on the other hand, was "the worst computer programming language ever" because its designers did not realize that they were creating a programming language (I would add templates in C++ as a much later example of an inadvertent programming language). A design goal was for the architecture to have 24 bits of address space, to be eventually extended to 32 bits; unfortunately, one instruction used the top 8 bits for its own purposes, which the architect overlooked, so when the architecture was extended to 31 bits (not 32 because of other design flaws) in System/370, programs compiled for System/360 had to run in a special compatibility mode. Emulation of earlier IBM computers was a great idea; adding decimal arithmetic for monetary values probably wasn't; applications or compilers should have stored the number of cents as an integer. Implementing OS/360 in assembly instead of PL/I made it less reliable, less clean, and slower to be delivered. Brooks gives more examples of good and bad design from his career, although System/360 and OS/360 are clearly where his passion lies. As a civilian expert on the Defense Science Board, he was invited to the review of the RAH-66 Comanche helicopter program, which was later canceled after a $6.9 billion investment. The colonel who briefed them said that one of the requirements for the helicopter was to be able to ferry itself across the Atlantic. Brooks asked, why it needed to do that: surely, this capability had to come at the expense of other capabilities more relevant to an attack helicopter. The colonel answered that this was because they did not have enough C-5 transport planes. Brooks asked, why they couldn't take some money from the helicopter program and use it to buy more planes. The colonel replied, "That's not possible," without explanation. There was no helicopter pilot or aircraft engineer on the committee formulating the requirements; just bureaucrats protecting their own turf. How could good design ever have emerged from that?

Dainius Jocas says

Žmogus, kuris buvo paskirtas vadovauti IBM's System/360 projektui, kur? tuo metu pasaulyje vadino "5000000000 dollar gamble", o tai buvo didžiausias projektas žmonijos istorijoje, negali b?t? toks visai šiaip sau. "Mythical Man-Month" irgi yra to paties autoriaus darbas. Kadangi traukiniu ried?jau ? "Theory of Computing" egzamin? ir tur?jau kr?v? gražaus laiko, pasiryžau perversti šit? id?jomis tiršt?/tank? veikal?.

Ši knyga yra palyginti nesenas darbas -- 2010 metai. Reikia pasteb?ti, kad autorius jau kopia ? devint? dešimt?. Tikriausiai tod?l skaitant jaut?si toks savotiškas žvilgsnis ? praeit?. Pvz., auorius pripaž?sta buv?s atsakingas už absoliu?iai blogiausi? dizain? turin?ios programavimo kalbos, JCL (Job Control Language), k?rim?. Tik savižudis arba solidus žmogus tokius dalykus apie savo karjer? atvirai pasakot?. ?ia pastarasis variantas.

Knyga yra apie dizain?. Dizain? bendr?ja prasme, ne tik kompiuteri? ar programin?s ?rangos dizain?. Sakysit užsiman? informatikas papasakoti, kaip reikia daryti dizain?. Bet ponas Brooks ne vien kompiuterius k?r?, jis dar ir pastat? architektas. Jis iš esm?s pats suprojektavo savo šeimos nam?. Ši namo statymo istorija yra detaliai išnagrin?ta. Gan ?domus atvejis, nes savo nam? židin? vienaip ar kitaip teks kurti bene kiekvienam.

Knygoje autorius grindžia mint?, kad dizainas yra bene svarbiausia bet kokios veiklos dalis. Negalima gail?ti, nei laiko, nei kit? ištekli? šitam procesui. Charakteringi pavyzdžiai iš turtingos autoriaus profesin?s ir akademin?s karjer?, vietos abejon?ms palieka nedaug.

Frederick Brooks kovoja su gan ?sigal?jusiais m?stymo šablonais. Pvz.: "visi kartu esame protingesni nei kiekvienas iš m?s?". Autorius ?ia sako "ne". Pavieniai žmon?s ir daugiau, ir geresnes id?jas sugeneruoja. Taip, kad šiandien populiar?s kooperacin?s k?rybos (angl. collaboration) ?rankiai yra daugiau trendas, o ne sidabrin? kulka. Jie tiesiog negali pakeisti juodo darbo ir sukoncentruotos vienišos minties. Ta?iau, anot autoriaus, bendra dizaino perži?ra (angl. review) yra tiesiog b?tina.

Draugas kart? man d?st? mint?, kad žmon?s tampa intelektualiai turtingesni, kai materializuoja abstrak?ias id?jas ir gali jomis operuoti (iš esm?s, abstrak?iais terminais turtingas žodynas byloja apie sukaupt? išmint?. Ta?iau žodynas, turtingas ?vairiais to paties daikto pavadinimais, ?ia neb?tinai reiškia išminties gaus?).

Knygoje buvo diskutuojama apie formalaus proceso reikšm? dizaino kokybei (program? sistem? inžineriai žino apie k? ?ia eina kalba). Akivaizdu, kad formalus procesas su kr?va dokumentacijos veikia kaip saugiklis nuo klaid? ar papras?iausi? neapsiži?r?jim?, ta?iau daro vis? dizaino proces? sunk? ir nuobod?. Mano nuomone, autorius ?ia padar? genial? skyrim? tarp inovatyvaus dizaino ir dizaino, kuris turi b?ti padarytas laiko ir biudžeto r?muose.

Vietoje pabaigos, jau?iantys poreik? pasimokyti iš tikrai patyrusio žmogaus patirties, šioje knygoje tikrai atras perl?.

Michael Scott says

I bumped into The Design of Design accidentally, while browsing the Kindle store for Peopleware and The

Mythical Man-Month. I'm just glad I did: from the moment I picked up this book, I regretted every second I could not spend on it.

In The Design of Design, Frederick P. Brooks Jr. starts from the premise that the process of designing anything---computers, software, houses, books, and organizations are the prime examples used in this book---follows very similar processes, when the outcome needs to be top-class. This book tries to decipher the structure and contents of a process that can lead to excellent design.

The book includes 27 chapters in 6 parts. Brooks looks at models of design process (part I) and elements of design (part III), the growth of team design into remote design and tellecolaboration (part II), the relationship between great designs and great designers (part V), and seven use cases in design (part VI). There is also an incursion into architecting houses via advanced computer visualization and interfaces (part IV). Interspersed in this book are summaries of Brooks' wisdom, such as "A chief service of a designer is helping clients discover what they want designed"---this lack of initial understanding of the real objectives of a design is a major obstacle in the theory of systematic design exploration, which is much favored by engineers as a better alternative to purely creative processes. The writing is robust, often from the perspective of absolute truth, which helps with starting designers but may not be so much fun for experienced professionals.

There are many technical elements in this book that may raise the attention of the wannabe designer. I found so many of the things I was looking for: notes on the design of a system architecture, notes on the design of an academic book, anecdotes and practical advice, an incursion into the history of design, useful literature references (up to 2010), etc. I also took notes, oh, so many notes (over 150 of them)!

Among the "trivial" treatment of some subjects, I absolutely loved the anecdote about rigid procedure following, among the German IBM team (Chapter 19)---this approach turned a talented team with strong leadership into a production pipeline rather than a creative powerhouse. There is plenty more in this book.

On the negative side, I was unable to relate to the architectural design. I found it very difficult to follow the process, perhaps because there are so many different aspects; space, the management of which I could easily follow, has to give room to considerations about materials, weather, etc. I enjoyed very much in the Chapters related to this topic, though, the discussion about patterns of house use.

Overall, a brilliant book about design. Highly recommended for any computer scientist, and perhaps for anyone in the world of design.

Tony says

Yes, it took me about a year and a half to read this. Parts of it can be dry. Parts of it are quite engaging.

What is design? What needs to be designed? How does one go about this? For each of these questions, there is no one, single answer. And different answers may be better than others, depending on the context.

I originally tumbled to this book while reading another one. It mentions the Comanche, a prototype scout/attack helicopter which I find quite interesting. The poor thing was doomed from the start. Dr Brooks was in one of the early meetings where the requirements for the chopper were laid out. It should be stealthy. It should be able to use/deploy these weapons. It should have this combat radius.

It should have a tremendous "ferry range," enough that it could deploy overseas with the benefit of loading it into another aircraft or ship.

These are mutually exclusive goals. The engines required to handle the weight were going to be pretty thirsty. The fuel required for that distance would mean having some extremely large fuel capacity. Which would add too much weight, even if the long-range tanks were removable. It simply wasn't do-able. And yet, the requirements had been developed by a committee, with different people bringing different goals to the table. Why did need that ferry range? Because the Army, which would be operating it, didn't want to have to turn to the Air Force or Navy to provide a means for deployment. Inter-service rivalry sank this bird before it made it off the drawing board. They needed a designer, sitting in on the meetings, who could keep them in line with reality.

And that is the essence of design. What is easy to use? What is efficient in operation? What is useful? What is possible? A good designer sits at the intersection of these questions, with enough domain knowledge to answer them (or find answers) and help synthesize all the competing requirements into something which works and works well.

The latter part of the books dives into case studies. I found these dry and boring; it took me over a year to slog through them. What's designed? A beach house. Additions to am existing house. A book. A computer. An operating system. A computer center and the associated governance/management of same. Each one talks about how principles of design were used, what worked, what didn't. If you want a frank discussion of how to do this, backed up with real-world examples of it, this book is for you.

Sten Vesterli says

Interesting thoughts from one of the grand old men of the young field of Information Technology, Fred Brooks (of "Mythical Man-Month" fame). This book is in six parts of varying relevance.

In part I, Brooks reflects on design methods, the problems with waterfall and how to improve it, and in part II, he discusses collaboration and distributed teams. These parts show the age of the book (it was written in 2010) and is of limited use in todays where we have agile and good tool support for virtual teams.

Part III discusses the design process and is still relevant with interesting observations on how to determine the main design criteria (which Brooks calls the "Budgeted Resource") and how to build decision trees to make design designings explicit and visible.

Part IV is about how Brooks would like building design software to work and can be skipped.

Part V is about great designers and posits that you need one chief designer to ensure conceptual integrity. However, finding, growing and nurturing great designers, and making use of their gift in real-life software development remains an unsolved problem.

In part VI, Brooks shares various designs he has been involved with, from his own beach house to the IBM System/360 he is famous for. There are a few interesting anecdotes, but they do not support the points of the book in any major way.

If you are an IT professional, you should read this book to feed your mind with ideas that could improve

your own practice in the field.

Dave Peticolas says

Another book on design, but this one emphasizes the actual practice of design and how to make it better. The first two-thirds or so is just excellent. Brooks's contention is that the best designs always come from either a single designer or (at most) two designers working very closely together. The reason is that the quality most important to a great design is conceptual integrity, an attribute that "design by committee" can never achieve. But design reviews are best done by multitudes, bringing many different perspectives to bear.

The last third of the book is a compendium of case studies. A few, like the design of IBM's S/360 mainframe computer, are interesting to a computer geek like myself for their historical value. But others, like the design of the Brooks family beach house, are quite frankly boring. But no matter, the good parts are well worth the price.

Kev says

Nice meta read

Vladyslav says

Every time I start a new book it is excitement: imagination grasps everything from the reviews to the cover to draw new universe of thoughts.
What one can expect from Frederick Brooks? Author of one of the classical books in software development The Mythical Man-Month: Essays on Software Engineering?

I must say I expected more or rather I expected something else.. The book is over bloated and written in highly academical manner difficult to read. It is full of references to "The Mythical Man-Month" in every chapter in almost every paragraph. Reader might get a feeling that it worth skip this one and go to "The Mythical Man-Month" directly.

Around one third of the book is long going story about design and construction of the house at the beach for himself and his family. And a lot of thoughts how design the house is similar to designing software. With lists of decisions made, plans presented, sketches, opinions. I can not stand this metaphor. And because of that I do not see any point to trying to find any similarities. Many years ago it felt natural to think that building software is like building a house until I actually participated in commercial software development myself. I believe another metaphor purposed by Andrew Hunt and Dave Thomas in The Pragmatic Programmer: From Journeyman to Master reflects true nature of software development much better - programming is gardening.

"With a garden, there's a constant assumption of maintenance. A garden is something that you're always interacting with to improve. We want people to view software as being far more organic, far more malleable, and something that you have to be prepared to interact with to improve all the time."

Another part of the book covers the revelation of Mr. Brooks regarding waterfall model. Mr. Brooks claims that earlier he was complete ally to it but understood how wrong he was and there are references to scientific report regarding its inefficiency that helped him to change his mind. Can not comment much about that.

The only thing I actually liked the most is the list of recommended reading because I found there the book I would strongly recommend myself Peopleware: Productive Projects and Teams

But after all it is only my personal opinion and it is up to you to decide is this journey worth it or not.

John says

Interesting stuff! Brooks examines not of what makes designs good or bad, but instead various issues around *how* designers come up with designs - the process of design. He looks at the subject from several different perspectives, since he himself has designed computer hardware (architectures), software, houses (his own), and books. Along the way he looks at questions including how designers can be enabled to work creatively, what "style" is and how it fits into design, collaboration versus solo design, and more.

The first 20 chapters are more closely connected than one might expect from a collection of essays; they definitely demand to be read serially, rather than standing alone. The next seven chapters are case studies in design, of various projects the author participated over a 40-year period, in the four areas mentioned above. Finally, there is a short chapter listing some recommended reading, including a number of seminal books and papers in the computing field.

Unlike a traditional book in either computing or design theory, this is a very personal book. The author carefully states his conclusions as opinions, backed up his own experiences as well as by referenced literature. This is both good and bad; not all of Brooks's points seem well supported by evidence he provides; it's hard to tell in those cases whether his position is actually weak, or whether he's simply failed - intentionally or otherwise - to provide all the support he actually has for it.

Regardless, Brooks is an unquestionably skilled and influential computer scientist, and anyone who is interested in the craft of computing should read this book. However, while there might be some points of interest for people in other fields, I'm not certain that it would be of as much value to, say, an architect or a graphic designer.

Yevgeniy Brikman says

I was excited to read this, as I was hoping that studying the design process would help me become a better designer. Unfortunately, the book wasn't particularly insightful, and I don't think I took away any lessons that will impact my design process or abilities.

In part, this is because the book tries to focus on all types of design—how to design a house, how to design software, how to design a computer, how to design a tool for designing, etc—rather than focusing on one discipline. As a result, it seems like a somewhat random collection of observations about various aspects of design, with a few interesting essays, and a lot of boring ones. The writing is stiff ("academic") and while I love the idea of the "case studies" at the end, those case studies solely show the _result_ of the design

process, rather than the design process itself.

In short, the book will get you to think about design as its own discipline, but it probably won't do much to make you a better designer.

As always, I've saved a few of my favorite quotes from the book:

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

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

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

"In software engineering practice, another kind of harm can readily be spotted—the Rational Model, in any of its forms, leads us to demand up-front statements of design requirements. It leads us to believe that such can be formulated. It leads us to make contracts with one another on the basis of this enshrined ignorance."

"A design flows from a chief designer, supported by a design team, not partitioned among one."

"Is a computer program a mathematical object to be fashioned in abstraction and made correct by proof? So the rationalists would contend, led by Edsger Dijkstra. It is all a matter of proper careful thinking. One can, and should, design software to be correct and then prove the design is correct. And that will suffice. Now, granted, a program is a pure mathematical object and in principle can be designed perfectly by correct thought. The difficulty is not with the design medium but with designers. Empiricists believe that humans will inevitably make mistakes: in defining objectives, in software architecture, in implementation in objects (algorithms and data structures), and in realization in code itself. This firm faith in fallibility prescribes a design methodology that includes design, early prototypes, early user testing, iterative incremental implementation, testing on a rich bank of test cases, and regression testing after changes."

"An articulated guess beats an unspoken assumption."

"I believe that consistency underlies all principles of quality. A good architecture is consistent in the sense that, given a partial knowledge of the system, one can predict the remainder."

"By its very nature, a product process “fights the last war,” encouraging tactics that have worked in the past and discouraging those that have failed. Hence for the product addressing a new war—a totally new need or mode of operating—both kinds of tactics may be irrelevant."

"Consensus stifles great design in several ways. First, each expert watchdog is paid to avoid mistakes, not to make great things happen. So each is separately biased toward finding reasons not to proceed. Even when a really new product is not vetoed, consensus mechanisms often take off the sharp edges by forcing compromises. But the sharp edges are the cutting edges!"

"Product design and release processes cannot turn good designers into great ones. They rarely produce great designs without a great designer. But the disciplines imposed can bring up the low end of the design curve and improve the average performance of the art. That’s nothing to sneeze at.
The software engineering community has given much attention to its development processes. It has needed

to, for I know of few design communities where average practice is so far behind best practice, and where worst practice is so far behind average practice."

William Blair says

No words I might add to the still-growing tide of reviews of Dr. Brooks' latest book would ever convince anyone to buy and read it. You either already know who Frederick P. Brooks, Ph.D is, or you don't. The sad thing is, like his other popular book, The Mythical Man-Month: Essays on Software Engineering, 20th Anniversary Edition (TMMM), most who will comment on it will never have read it completely, cover-to-cover, and understood much, if anything, in it. In the online world we find ourselves in today, what passes for research (never mind genuine, original research) sometimes only consists of finding some text (quoted elsewhere) using Google, and accepting it as fact. Hence, for example, the frequent misquotation (not to mention, the misrepresentation and misinterpretation) of "Brooks' Law." Sad to say, but *this* gem -- with an even larger load than TMMM of what in the broadcast world is termed "sound bites" -- will be equally misquoted.

For computer programmers, there is much, but not as much, I think, as the horde seems to have expected. I have already encountered online grumbling that the book is not what they were told to expect. Brooks discloses (as well as can be expected) that the book is about design -- and the care and feeding of what he calls *great designers*. Only some of his examples are taken from software engineering, and I am afraid the rest are likely to be deemed "too boring" for the impatient, increasingly-web-oriented audience to slog through and thereby learn what Dr. Brooks has to teach.

I think Dr. Brooks anticipated this (as he has many things) because he cautions the reader to treat the book as a series of related, but disconnected essays on the topic at hand. But I think most that would miss the point of many such essays would also ignore that advice to begin with (and then go on to complain).

I recently recommended this book to a well-known blogger who (tongue-in-cheek) complained that he would have to add it to his "already-too-long reading list." I responded that he should move it to the *top* and that he would be glad he did. I also offered the suggestion to heed Dr. Brooks' advice that it is not necessary to read every chapter. Dr. Brooks even suggests some chapters that can be (initially) skipped. For many, the chapters on the design of the NC beach house and the remodeling and expansion of his Chapel Hill, NC home can be initially omitted. The case studies of System/360 and OS/360 are interesting, but I think they would be obscure to anyone who was not once a programmer on those (or the early System/370) systems.

The bottom line is that you can skip almost half the pages, and not just initially. Brooks' prose is so engaging that I think most will go back and read the rest, anyway, but that's not necessary (nor even particularly valuable) the very first time. Like all of Brooks' written works, more and more value can be gleaned from rereading and reflective study. Except for the fact that it's been this way for 40 years, it would amaze me that, when I need to explain some things to those that are still wet behind their young programmers' ears, I can still find inspiration relevant to certain issues by reference to a book of his which is even older than The Mythical Man-Month.

And so I told the blogger to move it to the *top* of his reading list – and then put it back in the *middle* of the stack.

There is much here of interest, regarding both Dr. Brooks' personal and professional life and research, tidbits

about the writing of TMMM and his equally great work, Computer Architecture: Concepts and Evolution (with Gerrit A. Blaauw), and his well-known Christian convictions (which do not intrude). Unlike TMMM, this is a good book to give to rising stars in any field of endeavor to encourage them to learn more about shining in their own field, how to go about it, and how to develop humility about their own talents and that of others on whom their efforts will necessarily depend.

Ole Laursen says

Fred Brooks discusses the flaws of the waterfall model and the importance of all parties in a project that involves design (in a wide sense, including design of systems, of software, of houses) to learn about the problem domain and to use the knowledge gained to improve the design, incrementally.

The book contains a lot of discussion and back and forth which is probably less interesting to many readers, although I personally enjoyed the glimpse into the world of a former IBM chief, but it also makes a couple of crystal-clear and well-articulated insights, for which the entire book is worth reading, just like The Mythical Man-Month.

Erika RS says

This is a book of essays from Fred Brooks, author of *The Mythical Man-Month* . A number of the essays are excellent. A couple were positively dull. The rest were in the middle.

Brooks' aim in this book is to talk about the process of design in a way that is reasonably general. In that respect, I think he fails. I think if you're not designing computer systems, the essays on design will not provide a good overlap with the reader's experience. But if you are, you'll likely find much that is valuable.

What I found most valuable about this book was revisiting ideas about design and collaboration from a perspective that isn't as heavy with the modern buzz words and paradigms. By shaking me out of my normal mindset, I was able to look at some topics in a different way. Part III: Design Perspectives, does the best job of this.

I don't think this is likely to become as much of a classic as *The Mythical Man-Month*, but it's definitely food for thought if you care about the design of computer systems.

David Workman says

This is quite an interesting book, delving in quite some depth into what makes a good design, a good designer, and asking (and answering) many important questions related to these topics. It stays quite process-agnostic, instead expounding the view that the important things are:
- To have some form of design process
- To have good designers
- To give your designers the power to actually design
- To ignore the formal design process as required to make a good design

The structure of the book will be familiar to anyone who has read Brooks' previous work 'The Mythical Man Month', with each chapter of the book actually being a stand-alone essay written by Brooks (or sometimes a guest contribution). This means there is little cross-linking between chapters, and also that sometimes chapters will repeat themselves in places to remain self-contained. This is a minor niggle though, and doesn't detract seriously from the book as a whole. There isn't really a 'narrative' over the entire book, so the book seems to lack a bit of cohesiveness from this, but the ordering of the essays is well chosen to provide a somewhat sensible linear ordering to the book. In a somewhat 'strange-loop-esque' (Godel-Escher-Back) self-reflective way, there is even a description of the process of 'flattening' a design graph into a hierarchical tree description that can be made into a linear ordering for a book, meaning this book of design talks about the design of a book in a limited fashion, which appeals to my sense of style.

The book ends with several case-studies on several designs that Brooks has been involved in over the years. Two of these are construction projects (one a from-scratch, one an addition to an existing structure), one is the architecture of the 360 series computers that Brooks was in charge of designing at IBM, and so on. Each case-study is presented in a consistent and easily reviewed style, bringing out important decisions that were made in the design process, budgeted resources and so on. They provide a good summation of the ideas presented in the rest of the book in a more practical and useful context. They also provide exemplars that can be studied to give a budding (or experienced) designer some insight into the thought processes of a peer, which Brooks does cite as a key part of a designers education in several essays.

Overall, I would say that this book is interesting and a good read, but it isn't as important a book as Brooks' previous work. Someone familiar with design will likely not pick up too much new information here as Brooks does mainly cover familiar ground. However, if you are new to the world of design, or looking for something to tie together disparate ideas in their understanding, this book is good as it provides a well-rounded, cohesive view of the subject and contains a lot of references to other works that can be used to further expand your knowledge.
